POLSKO-JAPOŃSKA
WYŻSZA SZKOŁA
TECHNIK KOMPUTEROWYCH

Andrzej Polański

Podstawy Bioinformatyki

WYDAWNICTWO
PJWSTK

Notka biograficzna

Andrzej Polański jest inżynierem, bioinformatykiem. Legitymuje się kilkunastoletnim stażem współpracy z biologami i klinicystami w realizacjach wspólnych projektów i badań naukowych. Kierował lub brał udział w projektach badawczych, w których istotną część stanowiły zagadnienia związane z bioinformatyką, takie jak analiza danych z mikromacierzy DNA, analiza spektrów proteomicznych, problemy adnotacji sygnatur molekularnych. Od ponad dziesięciu lat organizuje interdyscyplinarne seminarium „Matematyka w biologii i medycynie". Jest autorem lub współautorem ponad 150 publikacji, w tym monografii: A. Polański, M. Kimmel, „Bioinformatics", Springer, 2007. Odbywał staże i wizyty w uczelniach zagranicznych (m.in., University of Texas, Medical Center, Rice University, Houston, USA).

Streszczenie

Niniejsza książka ma na celu zwięzłe omówienie najważniejszych zagadnień należących do bioinformatyki, nowej interdyscyplinarnej dziedziny wiedzy obejmującej zarówno elementy dziedzin należących do nauk ścisłych i technicznych, matematyki, informatyki, jak też nauk biomedycznych. W książce zamieszczono krótki przegląd wyników z zakresu biologii molekularnej, genomiki, proteomiki i transkryptomiki, który ma na celu przedstawienie najważniejszych motywacji dla wprowadzania metod modelowania matematycznego oraz metod współczesnej informatyki do tych dziedzin. W dalszej części przedstawiono najczęściej wykorzystywane metody zaliczane obecnie do bioinformatyki, przeszukiwanie sekwencji molekularnych, odtwarzanie topologii i metryki drzew filogenetycznych, uliniowienie sekwencji molekularnych oraz sekwencjonowanie DNA. Książka jest przeznaczona przede wszystkim dla studentów lub pracowników naukowych kierunków technicznych, informatyki, automatyki, biocybernetyki. Książka może być pomocna dla inżynierów rożnych specjalności, którzy chcieliby zapoznać się ze współczesnymi problemami oraz metodami bioinformatyki.

Seria: Podręczniki akademickie

Edytor serii: Leonard Bolc
Tom serii: 44

POLSKO-JAPOŃSKA
WYŻSZA SZKOŁA
TECHNIK KOMPUTEROWYCH

Andrzej Polański

Podstawy Bioinformatyki

WYDAWNICTWO
PJWSTK

© Copyright by Andrzej Polański
Warszawa 2010

© Copyright by Wydawnictwo PJWSTK
Warszawa 2010

Wszystkie nazwy produktów są zastrzeżonymi nazwami handlowymi lub znakami towarowymi odpowiednich firm.

Książki w całości lub w części nie wolno powielać ani przekazywać w żaden sposób, nawet za pomocą nośników mechanicznych i elektronicznych (np. zapis magnetyczny) bez uzyskania pisemnej zgody Wydawnictwa.

Edytor
Leonard Bolc

Redaktor techniczny
Ada Jedlińska

Korekta
Anna Bittner

Komputerowy skład tekstu
Grażyna Domańska-Żurek

Projekt okładki
Andrzej Pilich

Wydawnictwo Polsko-Japońskiej Wyższej Szkoły Technik Komputerowych
ul. Koszykowa 86, 02-008 Warszawa
tel. 022 58 44 526, fax 022 58 44 503

Oprawa miękka
ISBN 978-83-89244-91-8

Wersja elektroniczna
ISBN 978-83-63103-52-1

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt „Nowoczesna kadra dla e-gospodarki" – program rozwoju Wydziału Zamiejscowego Informatyki w Bytomiu Polsko-Japońskiej Wyższej Szkoły Technik Komputerowych współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego w ramach Podziałania 4.1.1 „Wzmocnienie potencjału dydaktycznego uczelni" Programu Operacyjnego Kapitał Ludzki

This book should be cited as:

Polański, A., 2010. Podstawy bioinformatyki. Warszawa: Wydawnictwo PJWSTK.

Spis treści

1

Wstęp

Bioinformatyka jest niedawno powstałą interdyscyplinarną dziedziną wiedzy leżącą na przecięciu takich nauk jak biologia molekularna, genomika, proteomika, transkryptomika z jednej strony, a matematyka, informatyka, nauki o komputerach czy teoria algorytmów z drugiej strony. Do rozwoju bioinformatyki przyczyniły się przede wszystkim postępy w technikach eksperymentalnych biologii molekularnej. Zsekwencjonowanie kompletnych genomów organizmów doprowadziło do powstania bardzo obszernych zbiorów danych przechowywanych w elektronicznych bazach danych. Zasoby te ponadto stale rosną, gdyż wykonuje się sekwencjonowanie coraz nowych genomów organizmów. Informacje zawarte w odczytanych sekwencjach genomowych stanowią podstawę do badań we wszystkich dziedzinach biologii molekularnej: w studiach genomiki porównawczej, genomiki strukturalnej, proteomiki, transkryptomiki itd. Obok danych o sekwencjach genomowych rozwijają się także bardzo obszerne repozytoria takich danych bioinformatycznych jak dane o sekwencjach aminokwasów w pierwszorzędowych strukturach białek, dane o przynależności taksonomicznej oraz aspektach funkcjonalnych białek, dane o strukturach przestrzennych białek, dane o strukturach oraz aspektach funkcjonalnych sekwencji RNA i wiele innych. Istnieją bardzo silne powiązania pomiędzy zawartością różnych bioinformatycznych baz danych. Dlatego też przechowywanie oraz odpowiednie udostępnianie tych danych wymaga wykorzystywania najnowszych technik informatycznych.

Obok wiedzy na temat struktury i zawartości bionformatycznych baz danych ważną częścią bioinformatyki jest zastosowanie różnorodnych technik modelowania oraz eksploracji danych bioionformatycznych. W bioinformatyce stosuje się bardzo różnorodne techniki modelowania matematycznego oraz techniki eksploracji danych. Bardzo duża część z tych metod i technik nie jest typowa czy też dedykowana specjalnie do zastosowań bioinformatycznych, lecz jest także szeroko wykorzystywana w innych dziedzinach informatyki czy też matematyki stosowanej. Jednak istnieją pewne techniki bardzo charakterystyczne dla bioinformatyki. Należą do nich przede wszystkim metody odtwarza-

nia topologii i metryki drzew filogenetycznych, metody uliniowienia sekwencji molekularnych, a także metody asemblacji sekwencji DNA.

W niniejszym skrypcie przedstawiono podstawowe metody i techniki stosowane w bioinformatyce. Przy tym przyjęto układ przedstawianego materiału związany z interdyscyplinarnym charakterem bioinformatyki. Dla badań w dziedzinie bioinformatyki bardzo istotna jest współpraca pomiędzy informatykami, bioinformatykami i biologami. Wagę jak również użyteczność wyników studiów w dziedzinie bioinformatyki najlepiej mogą ocenić biolodzy. Dlatego obok strony obliczeniowej i algorytmicznej przedstawianych metod omówiono także w skrócie podstawowe pojęcia z biologii molekularnej dotyczące genomiki, proteomiki oraz transkryptomiki. Omówienie tych pojęć miało na celu szersze naświetlenie motywacji dla podejmowania badań w dziedzinie bioinformatyki.

2

Podstawowe pojęcia z dziedziny biologii molekularnej

W rozdziale tym przedstawione zostaną podstawowe fakty z dziedziny biologii molekularnej. Ich znajomość pozwala na zrozumienie motywacji w konstrukcji przedstawianych dalej algorytmów. Rozdział niniejszy składa się z trzech punktów odpowiadających podstawowym działom biologii molekularnej: genomiki, proteomiki oraz transkryptomiki.

2.1 Genomika

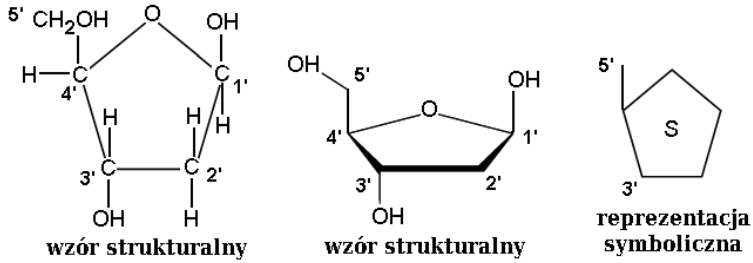
Genomika jest dziedziną biologii molekularnej, której przedmiotem badań są genomy żywych organizmów, tzn. całość dziedziczonej informacji genetycznej w tych organizmach. Zakres badań genomiki obejmuje analizę struktury genomów, informację zapisaną w sekwencjach genomowych, porównania genomów różnych organizmów, aspekty funkcjonalne, kod genetyczny, a także ewolucję genomów. Genomika jest jedną z najważniejszych dziedzin w naukach biologicznych, dlatego, że wszystkie aspekty związane z funkcjonowaniem i replikacją żywych organizmów znajdują swoje odbicie w informacji (sekwencjach DNA) zawartej w ich genomach. Nośnikiem informacji genetycznej w genomach organizmów jest kwas deoksyrybonukleinowy, DNA, polimer organiczny, który stanowi bazę dla dwóch podstawowych funkcji realizowanych przez komórki żywych organizmów, samoreplikacji oraz przechowywania informacji o liniowej (pierwszorzędowej) strukturze białek. Genomy organizmów zawierają zasoby informacji, których zrozumienie i analiza, m.in. z uwagi na wielką objętość, stanowi wyzwanie dla współczesnej nauki oraz wymaga stosowania najbardziej zaawansowanych technik informatycznych. Inne, podstawowe molekuły organiczne, które także odgrywają podstawowe role w funkcjonowaniu organizmów żywych, polimery RNA (kwas rybonukleinowy) oraz białka mają znacznie bardziej, od kwasu DNA, złożone struktury przestrzenne, ważne dla ich funkcjonalności. Jednak ich struktury pierwszorzędowe, sekwencje rybonukleotydów lub sekwencje aminokwasów są zawsze kopiami krótkich fragmentów DNA.

Metody eksperymentalne stosowane w badaniach genomów, ich struktur i funkcji przede wszystkim bazują na technikach sekwencjonowania, tzn. odczytywania sekwencji zasad w DNA. Obejmują jednak także bardzo wiele technik eksperymentalnych, które planowane są tak, aby odkrywać związki pomiędzy sekwencjami w DNA a ich aspektami funkcjonalnymi. Badania eksperymentalne w genomice muszą być powiązane z wdrożeniem i zastosowaniem odpowiednich metod bioinformatyki. Celem stosowania tych metod jest przechowywanie, organizacja, prezentacja wyników eksperymentalnych, na przykład w bazach danych sekwencji genomowych, a także badanie struktury zarówno istniejących jak i nowo odkrywanych genomów organizmów czy też rozwijanie nowych metod analizy sekwencji genomowych. Wynikiem badań modelowych i obliczeniowych struktury i funkcji genomów jest czasem wysuwanie nowych hipotez naukowych, które są bazą dla nowych modeli eksperymentalnych. Przeprowadzane eksperymenty biologiczne prowadzą do wytwarzania nowych danych, które powiększają i wzbogacają istniejące zasoby danych bioinformatycznych.

W niniejszym punkcie najpierw zostaną w skrócie przedstawione podstawowe fakty z zakresu struktury oraz budowy genomu i genów. Następnie omówione będą najważniejsze techniki eksperymentalne w dziedzinie genomiki. Po tych wstępnych informacjach zostaną omówione metody bioinformatyki stosowane w genomice, sekwencjonowania DNA, statystyki pokrycia DNA, adnotacji genomów.

2.1.1 Kwas DNA i centralny dogmat biologii molekularnej

Polimer DNA jest zbudowany z sekwencji powtarzających się podstawowych jednostek, które nazywane są nukleotydami. Każdy z nukleotydów z kolei składa się z cukru - deoksyrybozy, grupy fosforanowej oraz jednej z czterech możliwych zasad fosforanowych, adeniny (oznaczanej symbolem A), cytozyny (C), guaniny (G) lub tyminy (T). Struktura chemiczna cukru - deoksyrybozy jest przedstawiona na rysunku 2.1. Cząstka deoksyrybozy zawiera płaski pierścień zbudowany z pięciu atomów: czterech atomów węgla oraz jednego atomu tlenu. Atomy węgla w tym pierścieniu, są zgodnie ze stosowaną powszechnie konwencją numerowane jako $1' - 4'$. Dodatkowo, cząstka deoksyrybozy zawiera jeszcze jeden atom węgla oznaczony numerem $5'$, znajdujący się poza płaszczyzną pierścienia. Wiązanie pomiędzy atomami węgla $4'$ i $5'$ jest prostopadłe do pierścienia zawierającego atomy węgla $1' - 4'$. Jest ono wykorzystywane do określania orientacji przestrzennej cząstki deoksyrybozy. Dla określenia kierunku mierzonego wzdłuż łańcucha DNA używa się określenia kierunek $5'$ lub kierunek $3'$. Przez kierunek $5'$ rozumie się kierunek wyznaczony przez wektor środków atomów węgla $4'$ i $5'$. Z kolei kierunek $3'$ jest kierunkiem przeciwnym do $5'$, jest on wyznaczony przez wektor, którego jednym końcem jest atom węgla $3'$, a drugim związana z tym atomem wiązaniem kowalencyjnym, grupa hydroksylowa OH. Przedrostek „deoksy" wynika z porównania struktur dwóch

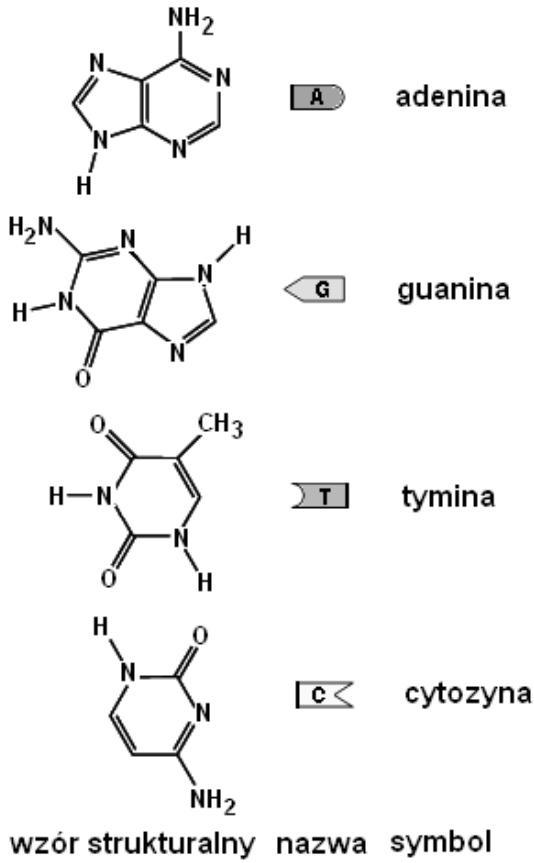


Rysunek 2.1. Cukier deoksyryboza. Pięc atomów węgla w cząsteczce cukru - deoksyrybozy numeruje się zgodnie z konwencją $1'$ - $5'$. Z lewej strony oraz w środku rysunku strukturalne wzory chemiczne cząstki cukru - deoksyrybozy. Z prawej strony - reprezentacja symboliczna wykorzystywana dalej do przedstawienia struktury polimeru DNA

cukrów, rybozy (omówionej dalej w tym rozdziale, o schemacie strukturalnym przedstawionym na rys. 2.9) oraz deoksyrybozy. Cząstka deoksyrybozy zawiera o jeden atom tlenu mniej niż cząsteczka rybozy. Na rysunku 2.1 przedstawione są różne konwencje reprezentowania struktury deoksyrybozy w postaci schematu strukturalnego, a także jej reprezentacja w postaci symbolu, które stosuje się później dla wytłumaczenia budowy przestrzennej DNA.

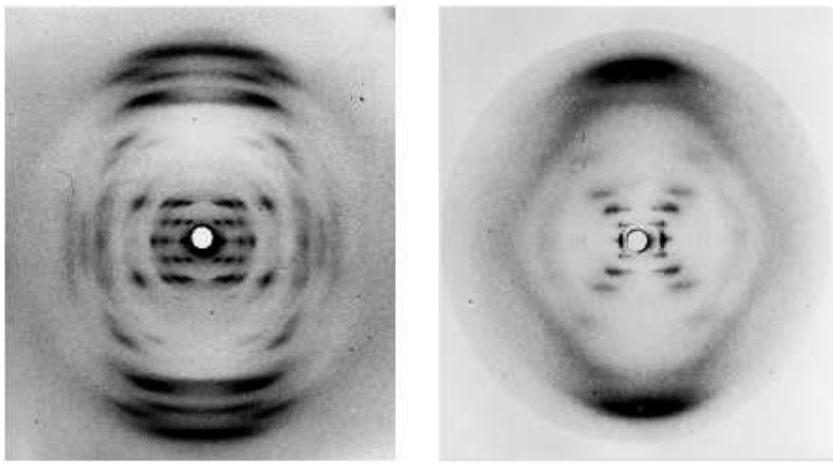
Na rysunku 2.2 pokazane są struktury chemiczne czterech zasad azotowych, które mogą występować w polimerze DNA: adeniny, cytozyny, guaniny, oraz tyminy. Wszystkie zasady azotowe przedstawione na rysunku 2.2 mają budowę planarną, tzn. ich atomy, w przybliżeniu, leżą na jednej płaszczyźnie. Adenina oraz guanina są związkami dwupierścieniowymi, nazywane są purynami. Cząsteczki cytozyny i tyminy mają po jednym pierścieniu, nazywane są pirymidynami. Obok strukturalnych wzorów chemicznych na rysunku 2.2 reprezentuje się zasady azotowe także w postaci symboli, które dalej stosuje się dla wytłumaczenia budowy przestrzennej polimeru DNA.

W 1953 roku James Watson i Francis Crick opublikowali [60] odkryty przez siebie model struktury przestrzennej polimeru DNA. Dla badania struktury przestrzennej DNA Watson i Crick wykorzystywali obrazy dyfrakcji rentgenowskiej skrystalizowanych struktur DNA otrzymane przez innych badaczy, Rosalyn Franklin, Maurice Wilkinsa i R. Goslinga. Obrazy dyfrakcyjne kryształów DNA wykorzystywane w opracowaniu modelu polimeru DNA przedstawione są na rysunku 2.3. Model został opracowany przez powiązanie obserwacji obrazów z rysunku 2.3 z wiedzą na temat stereochemii związków wchodzących w skład polimeru DNA. Kluczowym elementem w odgadnięciu struktury DNA była obserwacja zilustrowana na rysunku 2.4. Pary zasad azotowych pasują do siebie, to znaczy mogą tworzyć kompleksy utrzymywane w całości przez wiązania wodorowe, w taki sposób, że adenina tworzy kompleks AT z tyminą, a cytozyna kompleks CG z guaniną. Kompleks AT jest utrzymywany w całości przez dwa, a kompleks CG przez trzy wiązania wodorowe. Oba kompleksy są płaskie i mają w przybliżeniu taki sam kształt. Kompleksy

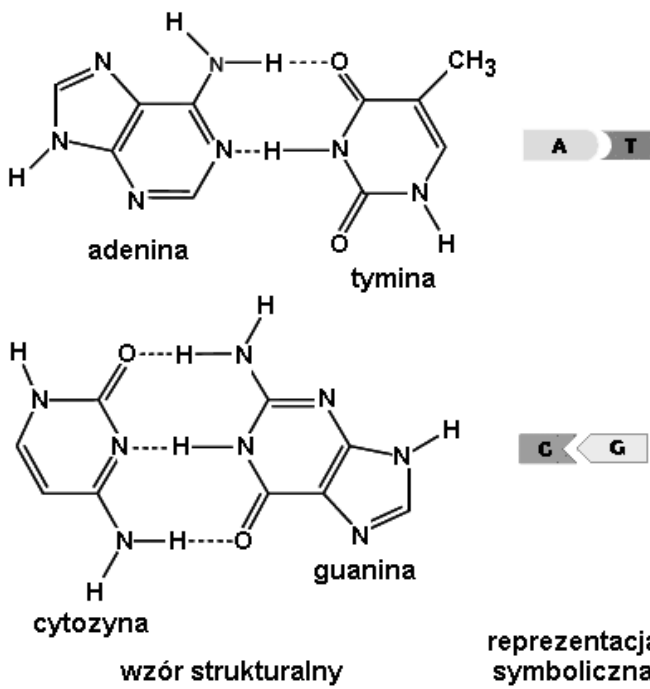


Rysunek 2.2. Zasady azotowe: adenina (A), guanina (G), cytozyna (C) i tymina (T)

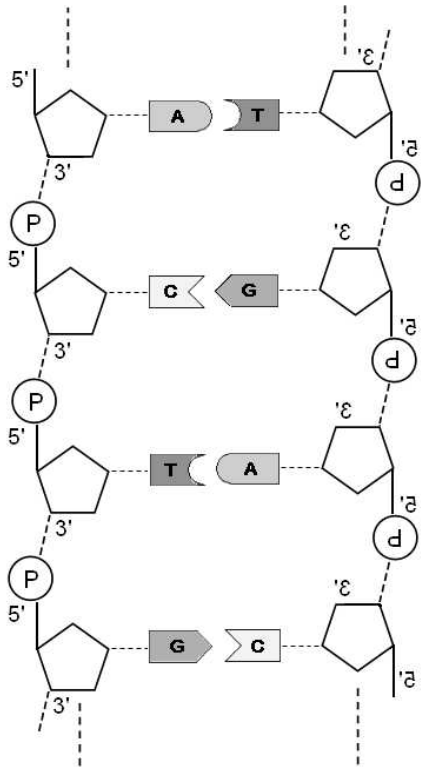
tworzone przez pary zasad AT lub CG tworzą wewnętrzną część dwuniciowej helisy DNA. Pary zasad AT lub CG nazywa się parami komplementarnymi albo parami Watsona-Cricka. Część zewnętrzną molekuły DNA tworzą dwie antyrównoległe nici zbudowane z ułożonych na przemian cukrów - deoksyroboz oraz jonów fosforanowych, połączonych ze sobą wiązaniami diestrowymi. Na rysunku 2.5 przedstawiono schematycznie przestrzenne ułożenie cząstek tworzących polimer DNA. Użyto przy tym symboli wprowadzonych wcześniej na rysunkach 2.1, 2.2 oraz 2.4. Dodatkowo dla reprezentacji jonu fosforanowego (grupy fosforanowej) użyto dodatkowego symbolu, kółka z literą P w środku. Jak już wspomniano, powtarzającym się elementem w polimerze DNA jest nukleotyd - kompleks cząstek, deoksyrybozy, grupy fosforanowej oraz zasady azotowej. Dwie przeciwległe nici fosforanowo - cukrowe mają orientację antyrównoległą. Gdy patrzymy od dołu rysunku do góry, widzimy, że jedna z nich ma orientację $3' \rightarrow 5'$ a druga $5' \rightarrow 3'$.



Rysunek 2.3. Diagramy dyfrakcyjne form A i B kryształów DNA.



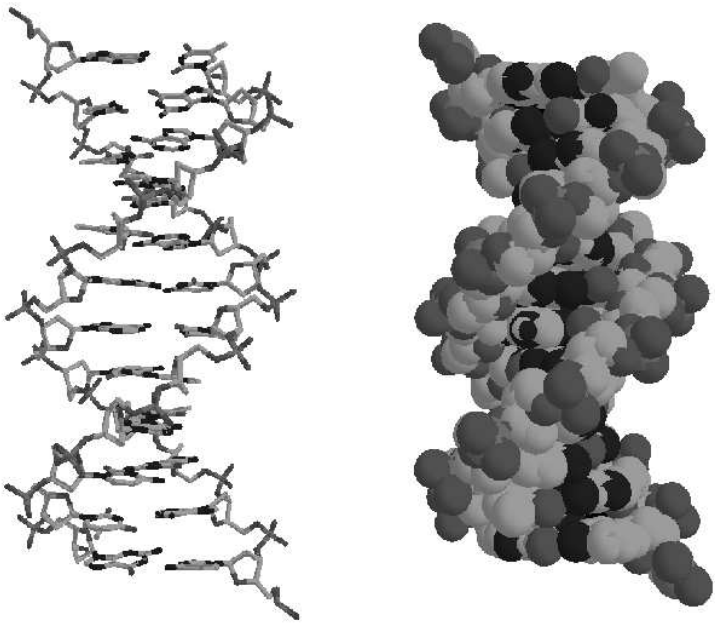
Rysunek 2.4. Kompleksy tworzone przez pary zasad AT i CG



Rysunek 2.5. Schematyczne przedstawienie budowy polimeru DNA

Schematyczne przedstawienie z rysunku 2.5 może być uzupełnione przez bardziej realistyczne modele, gdzie wyliczone są dokładne pozycje przestrzenne wszystkich atomów wchodzących w skład polimeru DNA. Istnieje szereg publicznie dostępnych programów komputerowych umożliwiających wykonywanie rysunków i widoków molekuły DNA. Program 3DNA [63] jest narzędziem wyspecjalizowanym dla wyliczania położeń atomów w polimerze DNA, program RasMol [66] to z kolei uniwersalne narzędzie do rysowania widoków dowolnych molekuł. Wykorzystując oba te programy, uzyskano obraz fragmentu polimeru DNA przedstawiony na rysunku 2.6.

Kwas DNA przechowuje informację reprezentowaną przez kolejność nukleotydów. Informację tę można najłatwiej zapisać w postaci tekstowej jako ciąg (listę) symboli oznaczających nukleotydy, w kolejności, w jakiej występują one w polimerze DNA. Na przykład dla nici DNA po lewej stronie rysunku 2.5 reprezentacja tekstowa jest następująca: $5'-GACTG-3'$, a dla nici po prawej stronie, komplementarnej do niej, reprezentacja tekstowa to $3'-CTGAC-5'$. Cały, dwuniciowy polimer DNA z rysunku 2.5 może być przedstawiony jako dwie linie tekstu:

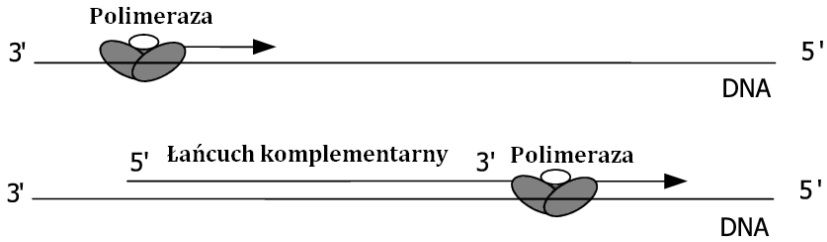


Rysunek 2.6. Reprezentacja budowy polimeru DNA uzyskana z zastosowaniem programu grafiki molekujlarnej RasMol

$$5' - GACTG - 3'$$
$$3' - CTGAC - 5'$$

przy czym linia na górze odpowiada lewej nici DNA z rysunku 2.5, a linia na dole odpowiada prawej nici DNA z tego rysunku. Przy zapisie kolejności nukleotydów w DNA w zbiorach tekstowych, w genomowych bazach danych, powyższy zapis nie jest stosowany. Powszechnie opuszcza się symbole $5'$ i $3'$, zapisuje się tylko jedną z dwóch komplementarnych nici, przy czym kolejność wypisywanych nukleotydów ustala się jako od $5'$ do $3'$. Jest to kolejność, w jakiej nukleotydy są umieszczane w nowo budowanej nici DNA w procesie replikacji. Zarówno w procesie replikacji (dobudowywania komplementarnej kopii do istniejącej nici DNA), jak też w procesie transkrypcji (tworzenia sekwencji RNA na podstawie fragmentu łańcucha DNA) enzymy białkowe należące do rodziny polimeraz „ślizgają się" wzdłuż nitki DNA, tak jak to jest przedstawione na rysunku 2.7. Kierunek przesuwania się polimerazy jest od $3'$ do $5'$, zatem nowe sekwencje dobudowują się wzdłuż kierunku od $5'$ do $3'$.

Pojedyncza nić DNA jest zbudowana z ułożonych naprzemiennie deoksyryboz oraz grup fosforanowych, które połączone są ze sobą silnymi wiązaniami fosfodiestrowymi. Natomiast dwie komplementarne nici DNA utrzymywane są razem przez słabsze wiązania wodorowe. Podwójna nić DNA może być rozseparowana przez odpowiednie enzymy, może też służyć jako wzorzec

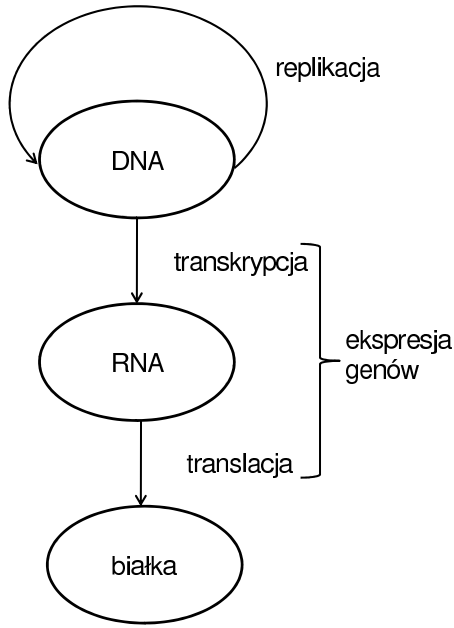


Rysunek 2.7. Schemat procesu budowania nici komplementarnej przez przesuwający się enzym polimerazy

w procesie replikacji DNA lub transkrypcji DNA na RNA. Oba procesy replikacji oraz transkrypcji są niesymetryczne, jeśli chodzi o dwie nici DNA. W procesie replikacji obie rozseparowywane nici DNA są kopiowane. Jednak, ponieważ rozseparowaniu (rozpleceniu) podlega zawsze tylko fragment łańcucha DNA, jak łatwo się zorientować, jedna z rozplatanych nici (nić $3'$) jest kopiowana w sposób ciągły, natomiast druga z nich (nić $5'$) jest kopiowana fragmentami nazywanymi fragmentami Okazakiego. Nić kopiowana w sposób ciągły nazywa się nicią wiodącą, a druga nić nicią opóźnioną. Z kolei w procesie transkrypcji kopiowana na mRNA jest tylko jedna z nici DNA nazywana nicią antysensowną. Druga nić, sensowna nie jest kopiowana w procesie transkrypcji. Sekwencje DNA, które zapisane są w genomowych bazach danych, są to zwykle sekwencje antysensowne kodujących fragmentów DNA. Zauważmy, że dla dwóch nici na rysunku 2.5 nie da się określić, która z nich jest sensowna, a która antysensowna. Jest to w DNA zdeterminowane przez kontekst, to znaczy, między innymi, przez informacje niezamieszczone na rysunku 2.5, określone przez położenie sekwencji kontrolujących proces transkrypcji sąsiadujących z kodującymi fragmentami DNA.

Dla funkcjonowania żywych organizmów informacja zapisana w sekwencjach DNA musi być efektywnie użyta dla zbudowania białek. Sposób przepływu informacji w tym procesie określa zasada, którą nazywa się Centralnym Dogmatem Biologii Molekularnej [13]. Nazwa Dogmat pochodzi stąd, że zasadę tę sformułowano w latach 50-60 ubiegłego wieku, gdy nie było jeszcze eksperymentalnych dowodów potwierdzających tę zasadę. Treść Centralnego Dogmatu Biologii Molekularnej ilustruje rysunek 2.8. Zgodnie z Dogmatem dla funkcjonowania żywych organizmów najistotniejsze są trzy procesy: replikacji, transkrypcji i translacji, częściowo już omówione w poprzednich fragmentach tego rozdziału.

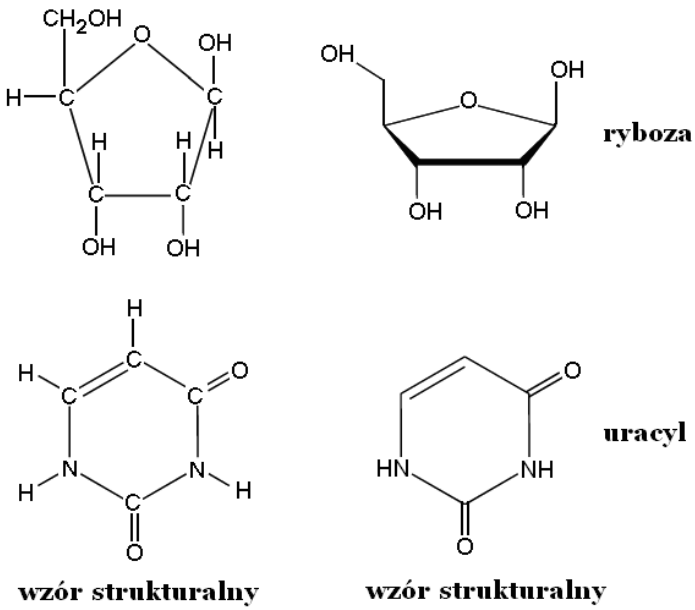
W pierwszym z procesów - replikacji - cały genom komórki jest duplikowany i dwie kopie są rozdzielane pomiędzy dwie komórki potomne. W drugim z procesów - transkrypcji - sekwencja nukleotydów fragment nici DNA jest kopiowana, w wyniku czego powstaje polimer RNA. RNA jest polimerem, który składa się z powtarzających się elementów nazywanych rybonukleotydami. Struktura RNA wykazuje dużo podobieństw do struktury DNA. Podobnie jak



Rysunek 2.8. Ilustracja Centralnego Dogmatu Biologii Molekularnej

DNA, kwas RNA zawiera rdzeń fosforowo-cukrowy, zbudowany z powtarzających się na przemian cząstek cukru - rybozy oraz reszt fosforanowych, a także związane z tym rdzeniem cząsteczki zasad azotowych. Podstawowe różnice strukturalne pomiędzy DNA i RNA to: (1) cukier deoksyroboza występujący w DNA, jest w RNA zastąpiony przez inny cukier - rybozę, (2) zamiast tyminy, która występuje w DNA, w RNA występuje inna zasada azotowa - uracyl. Chemiczne schematy strukturalne tych dwóch molekuł charakterystycznych dla RNA przedstawione są na rysunku 2.9. Polimer RNA jest przeważnie jednoniciowy. W odróżnieniu od DNA, molekuły RNA wykazują się znacznym zróżnicowaniem struktur przestrzennych, przez to pełnią różnorodne funkcje w żywych organizmach.

Trzeci proces - translacja - polega na budowaniu cząsteczek białek na podstawie informacji zapisanych w sekwencjach rybonukleotydów w RNA. Proces ten zachodzi w organellach komórkowych, rybosomach. Powstające w tym procesie białka są podstawowymi konstrukcyjnymi i funkcjonalnymi elementami komórek żywych organizmów. Oba procesy, transkrypcji i translacji nazywają się procesem ekspresji genów, albo procesem ekspresji informacji genetycznej. Mimo symetrycznej budowy dwuniciowej cząsteczki DNA, żaden z procesów replikacji ani transkrypcji nie jest symetryczny. W trakcie replikacji dwa procesy syntezy wzdłuż dwóch nici DNA zachodzą jednocześnie. Rozseparowanie dwóch, komplementarnych nici DNA wiąże się z przesuwaniem się wzdłuż łańcucha DNA tak zwanych widełek replikacyjnych. Ponieważ replikacja za-



Rysunek 2.9. Składniki polimeru RNA: cukier ryboza oraz zasada azotowa uracyl

chodzi zawsze wzdłuż kierunku od 5′ do 3′ (zobacz rysunek 2.7), replikacja wzdłuż nici 3′ zachodzi w sposób ciągły (syntezowany łańcuch komplementarny postępuje za oddalającymi się widełkami replikacyjnymi). Z kolei replikacja zachodzi wzdłuż nici 5′, w trakcie której widełki replikacyjne oddalają się od miejsca syntezy nowego łańcucha DNA, postępuje fragmentami o długości od 50 do 100 nukleotydów. Nitka DNA, wzdłuż której odbywa się replikacja o charakterze ciągłym, nazywa się nicią wiodącą, a nić wzdłuż której replikacja odbywa się fragmentami, nazywa się nicią opóźnioną. Z kolei w procesie transkrypcji tylko jedna z dwóch nici DNA jest używana do przepisania sekwencji na polimer RNA. Nić ta nazywa się nicią antysensowną. Druga z nici nazywa się nicią sensowną.

Żaden z procesów przedstawionych na rysunku 2.8 nie może oczywiście zachodzić spontanicznie. Aby komórka (organizm) mogła poprawnie funkcjonować, procesy replikacji, transkrypcji oraz translacji muszą być skoordynowane z wieloma procesami zachodzącymi w komórce. Replikacja musi być kontrolowana w szeregu punktach, musi być także zsynchronizowana z fazami i procesami cyklu komórkowego. Procesy transkrypcji i translacji muszą być ze sobą skoordynowane, ponieważ produkt pierwszego procesu jest substratem dla drugiego. Także żaden z tych procesów nie może przebiegać bez kontroli (ujemnego sprzężenia) od poziomu produktu, który jest wytwarzany. W przeciwnym przypadku prowadziłoby to do wytwarzania zbyt dużej ilości produktów tych procesów.

2.1.2 Struktura genomu

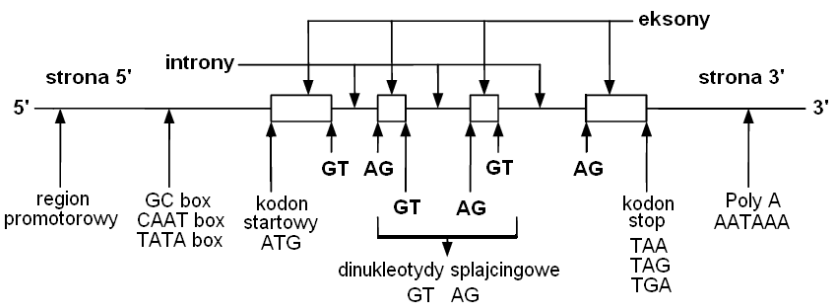
Genomy organizmów eukariotycznych składają się z chromosomów, nośnikiem informacji genetycznej są w nich geny. Sposób zapisu informacji genetycznej pozwalającej na budowanie białek jest określony przez kod genetyczny. Pojęcia te są omówione bardziej szczegółowo poniżej.

Chromosomy

Genomy komórek eukariotycznych są zorganizowane w postaci chromosomów, które są długimi polimerami DNA. Chromosomy znajdują się w jądrze komórki eukariotycznej. Są nawinięte na białka (histony), co pozwala na uzyskanie ciasnego upakowania nitek DNA. Nitki DNA nawinięte na pierścienie histonowe formują strukturę wewnątrz jądra komórkowego nazywaną chromatyną.

Geny

Geny są to odcinki DNA, które (których fragmenty) mogą być w procesie transkrypcji kopiowane do RNA, a następnie używane do konstrukcji białek. W procesach transkrypcji i translacji informacja zapisana w sekwencji nukleotydów genu jest używana do wytworzenia wielu polimerów RNA i wielu molekuł białek. Schemat struktury genu organizmu eukariotycznego jest przedstawiony na rysunku 2.10. Region w DNA położony w kierunku 5′ w stosunku do sekwencji genu nazywa się regionem powyżej genu, a region położony w kierunku 3′ regionem poniżej genu. Opisane terminy „powyżej" i „poniżej" odnoszą się także do innych charakterystycznych miejsc lub fragmentów genu, np. używa się określenia region powyżej miejsca inicjacji transkrypcji. W strukturze genu eukariotycznego można wyróżnić kilka regionów. Pierwszy to region powyżej sekwencji kodujących genu związany z inicjacją procesu transkrypcji. Do tego regionu należą sekwencje DNA nazywane sekwencjami wzmacniaczy oraz sekwencjami promotorowymi. Sekwencje te oddziałują z odpowiednimi



Rysunek 2.10. Ilustracja struktury genu eukariotycznego

białkami i w odpowiednich warunkach. Oddziaływanie tych białek i tych sekwencji umożliwia rozpoczęcie procesu transkrypcji. Następny region związany jest z procesem translacji. Rozpoczyna się od kodonu startowego (ATG) związanego z aminokwasem metioniną. Zawiera ułożone naprzemian sekwencje nazywane eksonami i intronami. Eksony zawierają właściwą informację dotyczącą liniowych (pierwszorzędowych) struktur białek kodowanych przez gen, a introny są w dalszych etapach procesu ekspresji informacji genetycznej wycinane. Granice pomiędzy eksonami i intronami są sygnalizowane w DNA przez dinukleotydy (pary nukleotydów) nazywane donorami i akceptorami. Koniec translacji jest sygnalizowany przez odpowiedni kodon (kodon stopu). Region położony poniżej kodonu stopu jest regionem terminacji (zakończenia) procesu transkrypcji.

Kod genetyczny

W trakcie transkrypcji przez kopiowanie informacji zapisanej w DNA wytwarzane są polimery RNA, które oznaczana są skrótem mRNA (angielska nazwa messenger RNA) i nazywane są matrycowym albo informacyjnym RNA. Te łańcuchy RNA przemieszczają się następnie poza jądro komórkowe i w organellach komórkowych nazywanych rybosomami służą do wytwarzania białek. Słownik, który pozwala na tłumaczenie sekwencji RNA na liniową strukturę aminokwasów, nazywa się kodem genetycznym. Podstawowa zasada konstrukcji kodu genetycznego została odkryta przez Francisa Cricka na podstawie następującego prostego rozumowania. Załóżmy, że jeden aminokwas musi być ko-

Tabela 2.1. Kod genetyczny

		Druga baza			
		U	**C**	**A**	**G**
Pierwsza baza	**U**	UUU (Phe/F) UUC (Phe/F) UUA (Leu/L) UUG (Leu/L), *Start*	UCU (Ser/S) UCC (Ser/S) UCA (Ser/S) UCG (Ser/S)	UAU (Tyr/Y) UAC (Tyr/Y) UAA, *Stop* UAG, *Stop*	UGU (Cys/C) UGC (Cys/C) UGA, *Stop* UGG (Trp/W)
	C	CUU (Leu/L) CUC (Leu/L) CUA (Leu/L) CUG (Leu/L), *Start*	CCU (Pro/P) CCC (Pro/P) CCA (Pro/P) CCG (Pro/P)	CAU (His/H) CAC (His/H) CAA (Gln/Q) CAG (Gln/Q)	CGU (Arg/R) CGC (Arg/R) CGA (Arg/R) CGG (Arg/R)
	A	AUU (Ile/I) AUC (Ile/I) AUA (Ile/I) AUG (Met/M), *Start*	ACU (Thr/T) ACC (Thr/T) ACA (Thr/T) ACG (Thr/T)	AAU (Asn/N) AAC (Asn/N) AAA (Lys/K) AAG (Lys/K)	AGU (Ser/S) AGC (Ser/S) AGA (Arg/R) AGG (Arg/R)
	G	GUU (Val/V) GUC (Val/V) GUA (Val/V) GUG (Val/V)	GCU (Ala/A) GCC (Ala/A) GCA (Ala/A) GCG (Ala/A)	GAU (Asp/D) GAC (Asp/D) GAA (Glu/E) GAG (Glu/E)	GGU (Gly/G) GGC (Gly/G) GGA (Gly/G) GGG (Gly/G)

dowany przez pewną sekwencję nukleotydów o pewnej stałej długości. Gdyby ta długość wynosiła 2, wówczas pojemność kodu genetycznego (liczba możliwych do zakodowania aminokwasów) wynosiłaby 16 (cztery do potęgi drugiej, ponieważ są cztery możliwe nukleotydy). Liczba wszystkich możliwych aminokwasów wynosi 20, czyli nie byłoby to wystarczające do kodowania wszystkich aminokwasów. Należy zatem przyjąć, że długość ta wynosi 3. Wtedy pojemność kodu wynosi 64 (cztery do potęgi trzeciej), co z nadmiarem zapewnia możliwość zakodowania wszystkich możliwych aminokwasów, i dodatkowo umożliwia kodowanie różnych sekwencji sygnalizacyjnych (kontrolnych), jak np. koniec translacji. Sekwencja trzech kolejnych nukleotydów, która koduje aminokwas albo jest sekwencją kontrolną, nazywa się kodonem. Znaczenie wszystkich możliwych kodonów, czyli inaczej budowa kodu genetycznego przedstawiona jest w tabeli 2.1. Znaczenia wszystkich kodonów przedstawione w tabeli 2.1 zostały odkryte przez Nirnberga i Matthaei w cyklu eksperymentów, w których wytwarzano łańcuchy mRNA i obserwowano powstające z nich aminokwasy. Wyniki tych eksperymentów opublikowano m.in. w pracy [46]. Badania te zostały uhonorowane nagrodą Nobla w 1968 roku.

2.2 Proteomika

Proteomika jest dziedziną nauk biologicznych, której przedmiotem badań są białka. Białka są podstawowymi elementami, które zarówno budują żywe organizmy, jak też biorą udział we wszystkich aspektach ich funkcjonowania. Białka biorą udział we wszystkich procesach w żywych organizmach.

Jak już wspomniano w rozdziale o genomice, procesem biologicznym, który prowadzi do powstawania białek, jest translacja. Translacja odbywa się w organellach komórkowych nazywanych rybosomami. Rybosomy są to kompleksy zbudowane z białek i molekuł RNA. W procesie translacji w rybosomach zaangażowane są dwa typy molekuł RNA: mRNA (matrycowy albo informacyjny RNA) oraz tRNA (transportowy RNA). W matrycowym RNA zapisany jest zasadniczy plan konstrukcji białka, to znaczy liniowa sekwencja aminokwasów tego białka. Transportowy RNA są to niewielkie molekuły RNA o specyficznej budowie, których zadaniem jest transport cząsteczek aminokwasów dla potrzeb budowania struktur białkowych.

Szacuje się, że liczba białek w organizmach eukariotycznych jest o rząd większa niż liczba genów w tych organizmach. Na przykład w ludzkim organiźmie liczba białek prawdopodobnie wynosi 200-300 tys. Jeden gen w sekwencji DNA może być użyty do konstrukcji wielu różnych białek. Proces molekularny, który na to pozwala, nazywa się alternatywnym splajcingiem. W tym procesie różne białka są produkowane z jednego genu przez układanie kopii eksonów w genie w różnej kolejności.

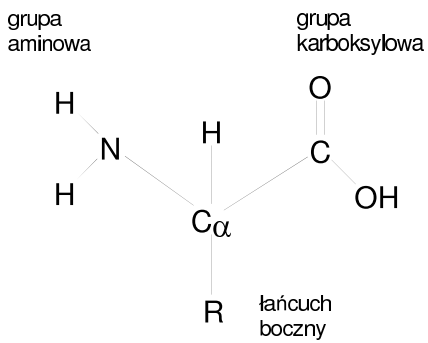
Białka są molekułami o złożonych przestrzennych kształtach. Aspekty funkcjonalne białek są związane w dużym stopniu właśnie z ich przestrzenną budową. Strukturę przestrzenną białek nazywa się też strukturą trzeciorzę-

dową białek. Obok struktury trzeciorzędowej, dla opisu białek stosuje się też własności ujmowane w postaci struktury pierwszo i drugorzędowej. Są one dokładniej opisane w dalszej części tego punktu.

Informacje o strukturach pierwszo, drugo i trzeciorzędowych białek, a także o aspektach funkcjonalnych białek zapisywane są w proteomicznych bazach danych, takich jak Swiss-Prot [67], ExPASy [64], PDB [65].

2.2.1 Struktura białek

Białka są zbudowane z aminokwasów. Strukturę molekularną aminokwasu przedstawiono na rysunku 2.11. Wszystkie aminokwasy posiadają jednakowy fragment struktury molekularnej, który pozwala na ułożenie dowolnych aminokwasów w łańcuch. Ten fragment struktury molekularnej składa się z centralnie położonego atomu węgla oznaczanego zwykle C_α, grupy aminowej NH_2 (o charakterze zasadowym), grupy karboksylowej $COOH$ (o charakterze kwasowym) oraz atomu wodoru H. Dwie molekuły aminokwasów mogą zostać połączone wiązaniem, w którym bierze udział grupa karboksylowa jednego oraz grupa aminowa drugiego aminokwasu, nazywanym wiązaniem peptydowym.



Rysunek 2.11. Struktua molekularna aminokwasu

Do centralnego atomu węgla oprócz atomu wodoru przyłączona jest także dodatkowa część aminokwasu nazywana łańcuchem bocznym (grupą boczną), oznaczona na rysunku 2.11 przez R. Aminokwasy różnią się między sobą łańcuchami bocznymi. Różne aminokwasy mają różne łańcuchy boczne.

Aminokwasy

W molekułach białek spotyka się 20 różnych rodzajów aminokwasów różniących się od siebie łańcuchami bocznymi. Nazwy, stosowane dla nich skróty i symbole, a także podstawowe własności aminokwasów zamieszczono w tabeli 2.2. Natomiast na rysunku 2.12 przedstawiono chemiczne schematy struk-

Tabela 2.2. Tabela aminokwasów, skróty, symbole oraz podstawowe własności

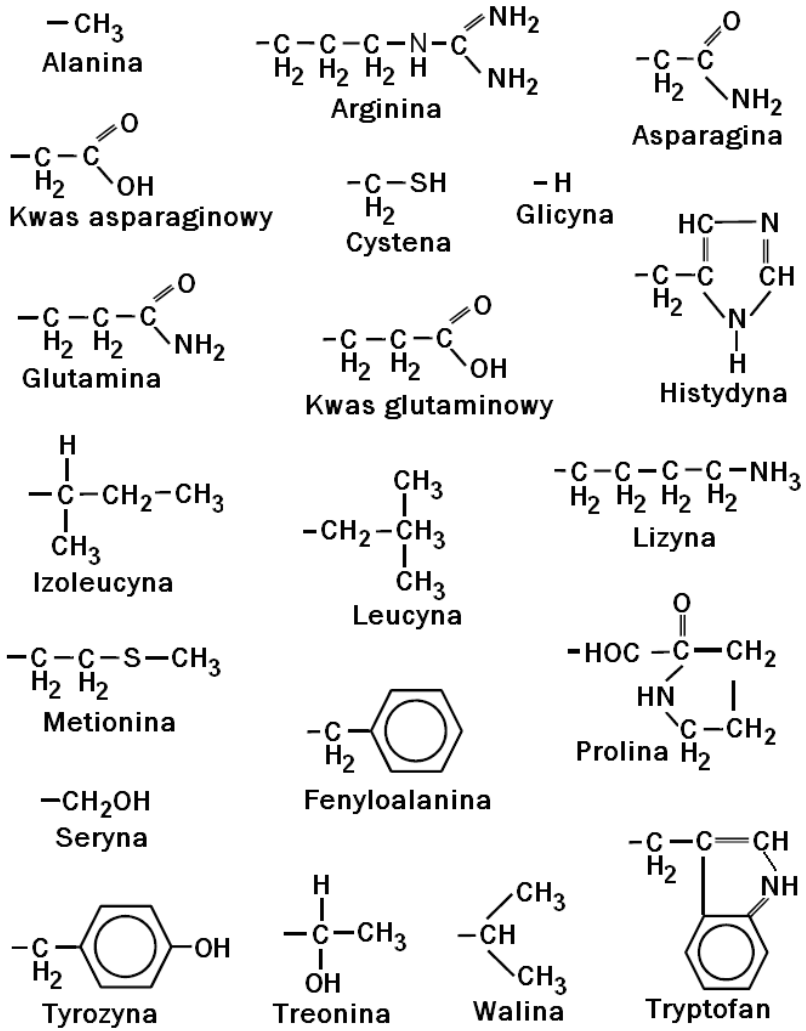
Aminokwas	Skrót	Symbol	Własności
Alanina	ALA	A	Niepolarny, hydrofobowy
Arginina	ARG	R	Polarny, hydrofilowy
Asparagina	ASN	N	Polarny, hydrofilowy
Kwas asparaginowy	ASP	D	Polarny, hydrofilowy
Cysteina	CYS	C	Polarny, hydrofilowy
Glutamina	GLN	Q	Polarny, hydrofilowy
Kwas glutaminowy	GLU	E	Polarny, hydrofilowy
Glicyna	GLY	G	Polarny, hydrofilowy
Histydyna	HIS	H	Polarny, hydrofilowy
Izoleucyna	ILE	I	Niepolarny, hydrofobowy
Leucyna	LEU	L	Niepolarny, hydrofobowy
Lizyna	LYS	K	Polarny, hydrofilowy
Metionina	MET	M	Niepolarny, hydrofobowy
Fenyloalanina	PHE	F	Niepolarny, hydrofobowy
Prolina	PRO	P	Niepolarny, hydrofobowy
Seryna	SER	S	Polarny, hydrofilowy
Treonina	THR	T	Polarny, hydrofilowy
Tryptofan	TRP	W	Niepolarny, hydrofobowy
Tyrozyna	TYR	Y	Polarny, hydrofilowy
Walina	VAL	V	Niepolarny, hydrofobowy

turalne łańcuchów bocznych wszystkich 20 aminokwasów. Własności aminokwasów są wyznaczane przez własności ich łańcuchów bocznych. Aminokwasy oraz ich łańcuchy boczne różnią się wielkością, strukturą, rozkładem potencjału elektrostatycznego na powierzchni oraz aktywnością chemiczną. Dwie ostatnie własności są zwykle opisywane przez dwa parametry: polarność i ładunek elektryczny.

Na podstawie wymienionych własności aminokwasy dzielone są na kilka grup wymienionych poniżej.

Aminokwasy niepolarne

Niepolarność jest związana z takim rozmieszczeniem ładunków elektrycznych na powierzchni łańcucha bocznego aminokwasu, że nie istnieje żadna asymetria pomiędzy dodatnim i ujemnym ładunkiem. Takie aminokwasy nie mają żadnego powinowactwa chemicznego do spolaryzowanych elektrostatycznie cząsteczek wody. Z tego powodu znajdują się one często w wewnętrznych częściach molekuł białek, nie mając kontaktu z otaczającymi białko cząsteczkami wody. Aminokwasy niepolarne nazywa się także hydrofobowymi albo wewnętrznymi. Grupa aminokwasów niepolarnych zawiera 8 aminokwasów: alaninę, leucynę, izoleucynę, metioninę, fenyloalaninę, prolinę, walinę oraz tryptofan.



Rysunek 2.12. Łańcuchy boczne aminokwasów

Aminokwasy polarne nienaładowane

Aminokwasy te mają łańcuchy boczne w których występuje polaryzacja ładunku elektrostatycznego. Jednak całkowity ładunek jest bliski zera. Dzięki spolaryzowaniu ładunku elektrostatycznego aminokwasy z tej grupy mają większe powinowactwo do cząsteczek wody od aminokwasów z poprzednio omawianej grupy aminokwasów niepolarnych i częściej są umieszczone na powierzchni białek. Aminokwasy, które wykazują powinowactwo do cząsteczek wody, nazywa się także aminokwasami hydrofilowymi. Do grupy aminokwasów po-

larnych nienaładowanych należy 6 aminokwasów: asparagina, cysteina, glutamina, tyrozyna, treonina i seryna.

Aminokwasy polarne naładowane

Aminokwasy te mają łańcuchy boczne, w których występuje zarówno polaryzacja ładunku elektrostatycznego jak też niezerowy całkowity ładunek. Aminokwasy te występują najczęściej na powierzchni białek, często też należą do regionów (miejsc) aktywnych w białkach. Do grupy aminokwasów polarnych naładowanych należą arginina, lizyna i histydyna naładowane dodatnio oraz kwas asparaginowy i glutaminowy naładowane ujemnie.

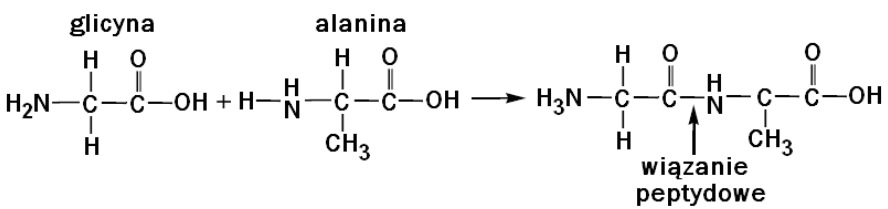
Aminokwasy aromatyczne

Do grupy tej należą aminokwasy, których boczne łańcuchy zawierają węglowe pierścienie aromatyczne. Jak widać z rysunku 2.12, są trzy takie aminokwasy: tyrozyna, fenyloalanina oraz tryptofan. Aminokwasy aromatyczne należą do największych i najcięższych aminokwasów. Fenyloalanina oraz tryptofan są niepolarne, hydrofobowe. Tyrozyna, jako jedyna spośród aminokwasów aromatycznych ma spolaryzowany łańcuch boczny. Jednak jest on spolaryzowany jedynie w niewielkim stopniu.

Jako niepolarne aminokwasy aromatyczne znajdują się zwykle wewnątrz struktur trójwymiarowych białek. Ich duża masa często też przynosi efekt stabilizacji własności mechanicznych struktur białkowych.

Wiązania peptydowe

Każda para aminokwasów może zostać połączona przez wiązanie peptydowe, które tworzy się pomiędzy resztą aminową jednego aminokwasu oraz resztą karboksylową drugiego. Przykład tworzenia się wiązania peptydowego pomiędzy alaniną a glicyną przedstawiony jest na rysunku 2.13. Poprzez szereg wiązań peptydowych aminokwasy mogą układać się w łańcuchy, które tworzą liniową, pierwszorzędową strukturę białek. Łańcuchy aminokwasów połączonych wiązaniami peptydowymi mają dobrze zdefiniowany kierunek (orientację), końcówka, w której znajduje się wolna grupa aminowa, nazywa się kierunkiem (końcem) N lub kierunkiem aminowym. Kierunek przeciwny definiowany przez końcówkę, w której znajduje się wolna grupa karboksylowa, nazywa się



Rysunek 2.13. Schemat wiązania peptydowego

kierunkiem (końcem) C lub kierunkiem karboksylowym. Powszechnie przyjmuje się konwencję, zgodnie z którą kierunek aminowy umieszczany jest po lewej stronie rysunku lub symbolu peptydu (białka). Taka sama konwencja przyjmowana jest dla reprezentacji symbolicznej aminokwasów. Zatem, sekwencją symboli dla reprezentacji dipeptydu z rysunku 2.13 jest GA. Łańcuch atomów węgla oraz azotu połączonych wiązaniami peptydowymi nazywa się szkieletem białka.

Struktura pierwszorzędowa białek

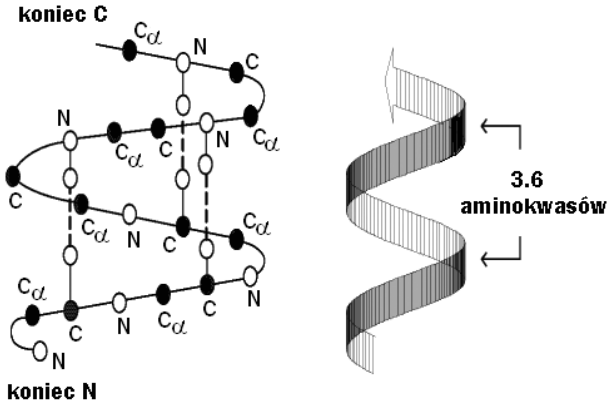
Białka składają się z jednego lub więcej łańcuchów polipeptydowych. Struktura pierwszorzędowa białka jest określona przez sekwencję aminokwasów pojawiających się wzdłuż linii szkieletu białka. Przy tym, jak już wspomniano, ich kolejność przyjmowana jest zgodnie z konwencją, według której początek wyznaczony jest przez koniec N łańcucha, a koniec przez koniec C. W bioinformatycznych (proteomicznych) bazach danych zapisane są wszystkie sekwencje aminokwasów tworzące struktury pierwszorzędowe znanych białek. W połączeniu z informacjami o aspektach funkcjonalnych oraz (ewentualnie) strukturalnych tych białek, bazy te tworzą dane odniesienia, dzięki którym można dalej rozwijać wiedzę o funkcjach i strukturach nowo odkrywanych białkach. Często białka, które wykazują podobieństwo w swoich strukturach pierwszorzędowych, posiadają także podobne struktury przestrzenne oraz podobne funkcje. Zatem podstawowym krokiem, który podejmuje się przy badaniu nowo-odkrytego białka, jest porównanie jego sekwencji aminokwasów do sekwencji aminokwasów w bazach danych białek. Ocenia się, że 40-procentowe podobieństwo sekwencji tworzących struktury pierwszorzędowe białek (badanego oraz zapisanego w bazie danych) jest już wystarczające do wiarygodnego przewidywania struktur przestrzennych białek.

Struktura drugorzędowa białek

Przez strukturę drugorzędową białka rozumie się sekwencje form przestrzennych tworzonych przez szkielet białka. Formy te wykazują duże podobieństwa pomiędzy różnymi białkami. Mechanizmami ich tworzenia jest pojawianie się wiązań wodorowych pomiędzy atomami tlenu i azotu w szkieletach białek. Poniżej są one dokładniej opisane.

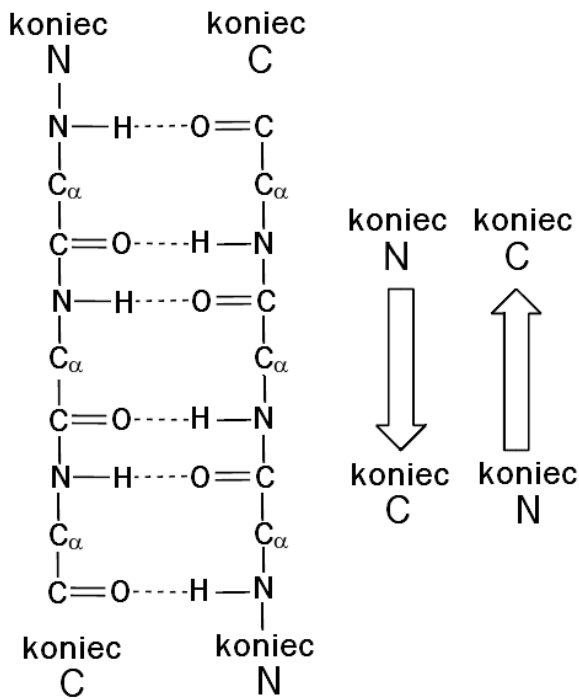
Helisy alfa

Helisa alfa jest to struktura przestrzenna stabilizowana przez wiązania wodorowe, w której łańcuch szkieletu białka tworzy w przestrzeni kształt helisy. Na rysunku 2.14 przedstawiono przykładowy kształt helisy alfa. Łańcuchy boczne aminokwasów tworzących helisę alfa są zawsze skierowane na zewnątrz tej struktury. Najczęściej spotykane helisy alfa są prawoskrętne, przy czym skok helisy alfa to znaczy liczba aminokwasów przypadająca na jeden obrót he-



Rysunek 2.14. Helisa alfa

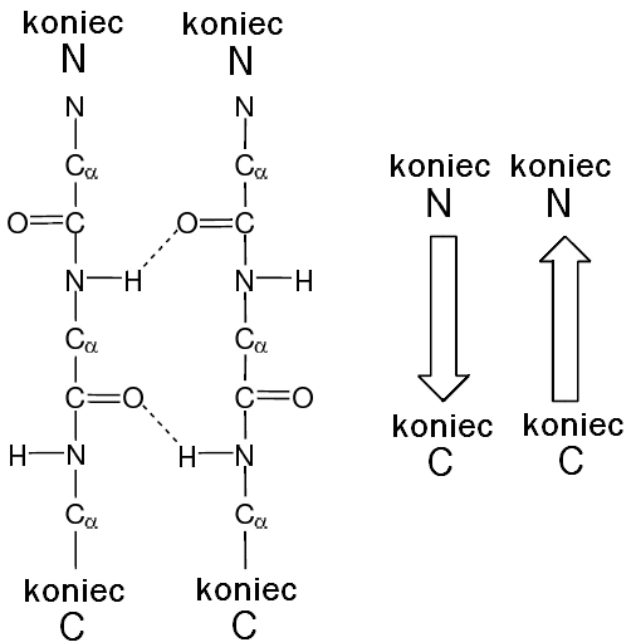
lisy wynosi ok 3.6. Taka helisa jest przedstawiona na rysunku 2.14. Rzadko spotykane są helisy lewoskrętne oraz takie, w których skok helisy przyjmuje wartość około trzech lub pięciu aminokwasów.



Rysunek 2.15. Beta kartka (płachta) typu antyrównoległego. Po lewej schemat strukturalny. Po prawej reprezentacja symboliczna

Beta kartki

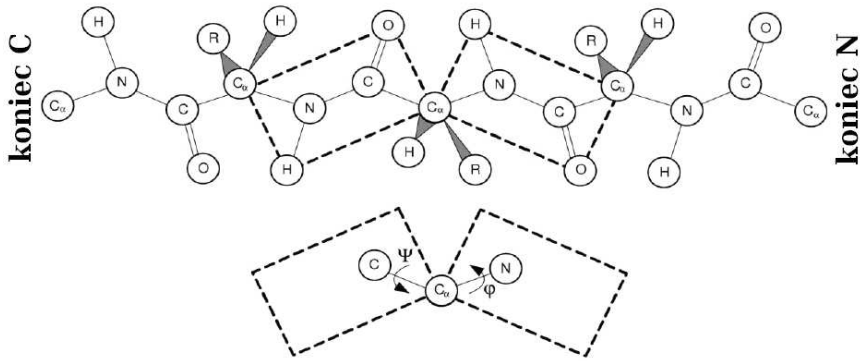
Drugą często spotykaną strukturą są tak zwane beta kartki, beta płachty lub beta harmonijki. W przeciwieństwie do alfa helis tworzonych przez jeden zwijający się łańcuch polipeptydowy beta kartki tworzone są przez dwa położone obok siebie łańcuchy polipeptydowe. Możliwe są dwie sytuacje: łańcuchy ze sobą sąsiadujące są albo równoległe, to znaczy kierunki $N-C$ są w nich zgodne, albo też antyrównoległe, gdy kierunki $N-C$ są przeciwne. Oba te rodzaje beta kartek są przedstawione odpowiednio na rysunkach 2.15 oraz 2.16. Podobnie jak w przypadku alfa helis, kształty tych struktur drugorzędowych są przestrzennie stabilizowane przez wiązania wodorowe pomiędzy atomami tlenu i azotu.



Rysunek 2.16. Beta kartka (płachta) typu równoległego. Po lewej schemat strukturalny. Po prawej reprezentacja symboliczna

Inne struktury

Poza dwoma powyżej opisanymi, w drugorzędowych strukturach białek występują także inne motywy. Połączenia pomiędzy alfa helisami i beta harmonijkami są realizowane przez odcinki łańcuchów polipeptydowych nazywane gamma pętlami. Istnieją także inne motywy drugorzędowe takie jak palce cynkowe, mostki siarczkowe itd. Istnieją bazy danych struktur i motywów drugorzędowych białek, np. [51].



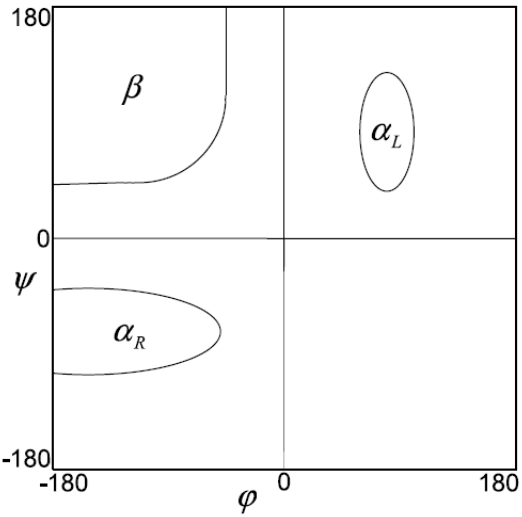
Rysunek 2.17. Rysunek ilustrujący definicję kątów Ramachandrana. Powyżej fragment szkieletu łańcucha aminokwasów. Przerywanymi liniami zaznaczono płaszczyznę tworzoną przez atomy C^{α}, C, N, O i H. Poniżej ilustracja kątów Ramachandrana

Struktura trzeciorzędowa białek

Struktura trzeciorzędowa białek jest to przestrzenne ułożenie łańcucha polipeptydowego. Jest ona zdefiniowana przez trójwymiarowe współrzędne środków wszystkich atomów tworzących dane białko. Dane dotyczące struktur trzeciorzędowych białek są znane tylko dla niewielkiej części wszystkich znanych białek. Są one zapisywane w bazie danych PDB (Protein Data Bank) [65]. Na strukturę trzeciorzędową białka bardzo duży wpływ ma kształt szkieletu białka. Przykład ułożenia fragmentu szkieletu białka przedstawiony jest na rysunku 2.17. Charakterystyczną cechą geometrii przestrzennej szkieletu białka jest to, że środki atomów leżących wzdłuż szkieletu, C^{α}, C i N, a także środki dwóch dalszych atomów O i H, związanych z atomami C oraz N, leżą w przybliżeniu w jednej płaszczyźnie. Ułożenie dwóch kolejnych płaszczyzn tworzonych przez środki atomów C^{α}, C, N, O i H, należących do dwóch kolejnych aminokwasów, względem siebie oraz względem łańcuchów bocznych opisane są przez dwa kąty φ i ψ, tak jak to przedstawiono na rysunku 2.17. Wykres kolejnych wartości tych kątów przedstawiony na płaszczyźnie φ - ψ nazywa się wykresem Ramachandrana. Płaszczyzna Ramachandrana φ - ψ przedstawiona jest na rysunku 2.18. Na płaszczyźnie tej występują charakterystyczne obszary odpowiadające strukturom drugorzędowym białek α_R - prawoskrętnym alfa - helisom, α_L - lewoskrętnym alfa - helisom, β - beta kartkom.

2.3 Transkryptomika

Transkryptomika jest to dziedzina biologii molekularnej, której obszarem badań są molekuły i sekwencje RNA powstałe w komórce w procesie transkrypcji. Zgodnie z Centralnym Dogmatem Biologii Molekularnej, przedstawionym na rysunku 2.8, molekuły RNA służą głównie do przekazywania informacji



Rysunek 2.18. Wykres Ramachandrana

genetycznej zapisanej w DNA do rybosomów, gdzie wykorzystuje się ją do produkowania białek. Jednakże w toku badań biologii molekularnej staje się coraz bardziej jasne, że Centralny Dogmat Biologii Molekularnej nie ujmuje w odpowiedni sposób ważności różnych rodzajów molekuł RNA w procesach zachodzących w żywych organizmach oraz w ewolucji.

2.3.1 Hipoteza świata RNA

Jak już wspominano, procesy replikacji, transkrypcji oraz translacji muszą podlegać bardzo wielu mechanizmom kontroli. Aby mogły one zachodzić konieczny jest udział wielu enzymów (białek), które katalizują te procesy. Z drugiej jednak strony, białka, aby powstały, muszą wykorzystywać informację zapisaną w DNA oraz przeniesioną przez molekuły RNA. Z punktu widzenia ewolucji molekularnej występuje tu paradoks, w jaki sposób mechanizmy replikacji, transkrypcji oraz translacji zostały wykształcone w toku ewolucji, które z molekuł, DNA, RNA czy białka powstały jako pierwsze? Wczesne teorie ewolucji dawały pierwszeństwo cząstkom białek lub peptydów. Jednak hipotezy wywodzące ewolucję od białek i peptydów napotykały coraz większe trudności związane głównie z brakiem wystarczająco silnych argumentów dla istnienia w toku ewolucji mechanizmów pozwalających na samoreplikację białek. Obecnie jednak przewagę zyskuje tak zwana hipoteza świata RNA, zgodnie z którą w procesie ewolucji molekuły RNA poprzedzały molekuły DNA i białek [23]. Udaje się potwierdzać możliwość samoreplikacji cząstek RNA w badaniach eksperymentalnych [58], co stanowi silny argument za tym, że wczesne etapy ewolucji mogły przebiegać za sprawą samoreplikacji cząstek RNA.

2.3.2 Funkcje molekuł RNA w komórce

Na bazie aspektów funkcjonalnych molekuły RNA mogą być podzielone na dwie grupy, kodujące oraz niekodujące. Kodujące funkcje pełni mRNA (messenger RNA), nazywane matrycowym lub informacyjnym. Sekwencje rybonukleotydów w mRNA są kopiami sekwencji genów. Przy tworzeniu molekuł mRNA zachodzą procesy transkrypcji, a także splicingu (alternatywnego splicingu) oraz poliadenylacji. Najwcześniej poznane niekodujące molekuły RNA to tRNA (transferowy RNA) oraz rRNA (rybosomowy RNA). Molekuły tRNA składają się z krótkich sekwencji rybonukleotydów (od 74 do 93) i służą do transportu aminokwasów do rybosomów. Molekuły rRNA są składnikiem rybosomów.

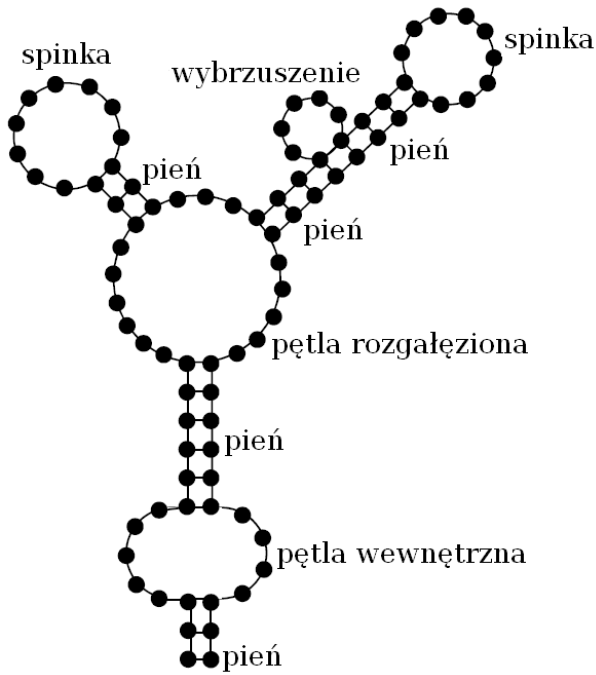
W ostatnich latach udowodniono, że molekuły RNA, które powstają w wyniku transkrypcji w niekodujących regionach chromosomów, pełnią różnorodne, bardzo ważne dla funkcjonowania komórki role. Do nowo odkrytych rodzajów molekuł RNA należą np. miRNA, siRNA (antysensowne RNA, interferencyjne RNA) pełniące funkcje regulacyjne w procesach ekspresji genów, snRNA (małe jądrowe RNA) wspomagające wycinanie intronów z transkryptów.

2.3.3 Struktura RNA

Molekuły RNA charakteryzują się dość dużą różnorodnością, jeśli chodzi o ich przestrzenny kształt. Podobnie jak w przypadku białek wyróżnia się pierwszo, drugo i trzeciorzędowe struktury molekuł RNA. Struktura pierwszorzędowa RNA jest zdefiniowana przez sekwencję rybonukleotydów. Analiza porównawcza sekwencji rybonukleotydów jest podstawowym krokiem w studiach dotyczących przewidywania funkcji lub przynależności do rodzin nowo odkrytych molekuł RNA.

Kształt przestrzenny cząstek RNA jest w dużym stopniu uwarunkowany przez sekwencje wiązań wodorowych, które tworzą się pomiędzy zasadami azotowymi wzdłuż polimeru RNA. Wiązania wodorowe tworzą się pomiędzy komplementarnymi parami zasad, tak jak w DNA. Dodatkowo, energetycznie korzystne jest wiązanie wodorowe G-U. Powstawanie tych wiązań w dalszej konsekwencji prowadzi do obecności charakterystycznych motywów drugorzędowych, występujących w większości cząstek RNA. Motywy występujące w drugorzędowych strukturach RNA to pętla typu spinka do włosów (hairpin loop), pień (stem), wybrzuszenie (bulge), pętla rozgałęziona (multibranched loop), pętla wewnętrzna (internal loop). Struktura drugorzędowa RNA może być zilustrowana przez schemat przedstawiony na płaszczyźnie. Ilustracja graficzna motywów drugorzędowych RNA pokazana jest na rysunku 2.19.

Struktura trzeciorzędowa RNA podobnie jak struktura trzeciorzędowa białek opisana jest przez współrzędne przestrzenne środków atomów należących do molekuł tworzących polimer RNA.



Rysunek 2.19. Struktura drugorzędowa RNA

3

Narzędzia bioinformatyczne

W rozdziale niniejszym przedstawiony zostanie przegląd metod algorytmicznych i obliczeniowych bioinformatyki. Omówione zostaną cztery podstawowe grupy zagadnień należące do zakresu bioinformatyki.: algorytmy przeszukiwania sekwencji molekularnych, metody odtwarzania topologii i metryki drzew filogenetycznych, algorytmy uliniowienia sekwencji molekularnych oraz algorytmy asemblcji (składania z fragmentów) sekwencji DNA.

3.1 Przeszukiwanie sekwencji molekularnych

Jednym z podstawowych zadań bioinformatyki jest przeszukiwanie sekwencji molekularnych, czyli sekwencji nukleotydów w polimerach kwasu DNA, sekwencji rybonukleotydów w polimerach kwasu RNA oraz sekwencji aminokwasów w pierwszorzędowych strukturach białek. Pod nazwą przeszukiwanie sekwencji molekularnych rozumie się różnorodne problemy, takie jak wyszukiwanie zadanych podciągów w długich sekwencjach molekularnych czy też w bazach danych takich sekwencji, przybliżone wyszukiwanie zadanych podciągów, wyszukiwanie pewnych charakterystycznych sekwencji, np. sekwencji tandemowych powtórzeń, jak również wyszukiwanie sekwencji unikalnych, tzn. takich, które nie występują w żadnym innym miejscu ani w żadnej innej sekwencji w bazie danych, poza analizowanym fragmentem zadanej sekwencji molekularnej.

Do rozwiązywania zadań przeszukiwania sekwencji molekularnych w bardzo dużej mierze wykorzystywane są standardowe algorytmy przeszukiwania i analizy tekstów. Jednak wiele rozwinięć ma już charakter wyspecjalizowany do analizowanego problemu biologii molekularnej. Dotyczy to zwłaszcza analiz, w których przeszukiwanie powiązane jest ze statystycznymi badaniami częstości występowania pewnych wzorców czy też stopnia podobieństwa pomiędzy sekwencjami.

W niniejszym punkcie omówione zostaną podstawowe algorytmy przeszukiwania sekwencji (tekstów). Omówienie to zostanie tak zbudowane, aby

przede wszystkim naświetlać zastosowania tych algorytmów do przeszukiwania sekwencji molekularnych. Jak wspomniano, jest kilka rodzajów sekwencji molekularnych. Jednak na pewno najważniejsze oraz najtrudniejsze jest przeszukiwanie genomowych sekwencji DNA, z uwagi na ich rozmiar.

Wyszukiwanie wzorców w sekwencjach molekularnych

Standardowym problemem w zakresie algorytmów komputerowych jest wyszukiwanie w sekwencjach symboli (w tekstach) pewnych zadanych podciągów (słów, wzorców). Symbole należące do tych sekwencji są elementami pewnego alfabetu, który oznaczany jest przez Σ. Na przykład dla sekwencji nukleotydów alfabet zawiera cztery symbole $\Sigma = \{a, c, g, t\}$. Długą sekwencję symboli, która jest obszarem przeszukiwania, oznaczać będziemy przez S, a krótki (krótszy) podciąg (wzorzec), który ma być wyszukany lub też ma zostać stwierdzony brak takiego wzorca w sekwencji, oznaczać będziemy przez P. Oznaczmy dodatkowo długość sekwencji S przez K^S, długość sekwencji P przez K^P, a także przez $P(i)$ oraz $S(j)$ odpowiednio i-ty symbol sekwencji P oraz j-ty symbol sekwencji S.

Prosty algorytm wyszukiwania wzorców

Oczywisty pomysł na zbudowanie algorytmu, który rozwiązuje przedstawiony powyżej problem, polega na przesuwaniu wzorca P wzdłuż sekwencji S oraz porównywaniu odpowiadających sobie symboli sekwencji S i przesuniętego wzorca P. Ten prosty algorytm wyszukiwania można przedstawić w postaci następującego pseudokodu:

Algorytm proste wyszukiwanie
for $j = 1$ to $K^S - K^P$
    $i = j$
    while symbol_cmp$[P(i - j + 1), S(i)] == 1$
        $i = i + 1$
        if $i == j + K^P$
            przerwij pętlę i jako wynik wydrukuj wyszukanie wzorca w pozycji $i = j$
        endif
    endwhile
endfor
jako wynik wydrukuj brak wzorca P w sekwencji S.

Powyższy algorytm wykorzystuje funkcję symbol_cmp$[s_1, s_2]$, która zwraca 0, jeśli symbole $[s_1, s_2]$ się różnią i 1, jeśli są jednakowe. Ponieważ powyższy algorytm zawiera dwie zagnieżdżone pętle, jego złożoność obliczeniowa, którą definiujemy jako liczę porównań (liczbę wywołań funkcji symbol_cmp), może być ograniczona od góry przez $K^S K^P$. Można skonstruować złośliwe przy-

kłady wyszukiwania wzorca, dla których konieczna liczba porównań istotnie jest bliska tego ograniczenia, np. zdefiniujmy $P = aaab$ oraz

$$S = aaac\_aaac\_aaac \ldots aaac\_aaab \tag{3.1}$$

Jednak w praktycznych problemach wyszukiwania wzorców takie złośliwe, specjalnie konstruowane zadania wyszukiwania raczej się nie zdarzają. Liczba porównań symboli w wewnętrznej pętli zależy od struktury sekwencji P i S. Zwykle też liczba porównań konieczna do wyszukania wzorca w tekście jest tylko rzędu K^S.

Szybki algorytm wyszukiwania wzorców, Boyera-Moore'a

Powyższy prosty algorytm wyszukiwania wzorca można ulepszyć. Bardziej zaawansowane algorytmy wyszukiwania wzorca w tekście [11], [38] są skonstruowane w taki sposób, że kolejność wykonywania operacji porównywania symboli, a także odległość, o jaką przesuwa się sekwencję wzorca zależą od wyników porównywania symboli w P i S. Poniżej zostaną przedstawione główne idee algorytmu Boyera-Moore'a [11]. Podobnie jak w prostym algorytmie wyszukiwania opisanym w poprzednim podpunkcie, przy zastosowaniu algorytmu Boyera-Moore'a wykonuje się porównania symboli z P i S, a także przesuwa się wzorzec P wzdłuż sekwencji S. Poprawę efektywności uzyskuje się przez następujące dwie zmiany w stosunku do prostego algorytmu. Pierwsza zmiana polega na tym, że porównywanie symboli sekwencji S i przesuniętej sekwencji wzorca P dokonywane jest od ostatniego symbolu sekwencji P, a nie jak poprzednio od pierwszego. Druga zmiana polega na tym, że przesunięcia wzorca P wzdłuż tekstu S wykonywane są nie zawsze o jeden symbol, lecz na ogół o większą liczbę symboli, zależną od wyników porównywania symboli sekwencji S i przesuniętej sekwencji wzorca P.

Algorytm Boyera-Moore'a ma bardziej złożoną konstrukcję od prostego algorytmu z poprzedniego podpunktu. Dlatego jego przedstawienie w postaci pseudokodu jest dość mało czytelne. Zamiast tego, konstrukcja algorytmu Boyera-Moore'a zostanie przedstawiona na przykładzie zadania wyszukania wzorca $P = ekst$ w sekwencji $S = korekta\_tekstu$. Algorytm wykorzystuje wskaźnik pozycji wzdłuż sekwencji S. Kolejne kroki algorytmu są następujące:

Krok 1. Sekwencje P i S znajdują się w początkowej pozycji. Porównywanie symboli w sekwencjach rozpoczyna się w miejscu pokazywanym przez wskaźnik $\uparrow$ i postępuje wstecz, to znaczy od strony prawej do lewej. Wskaźnik $\uparrow$ pokazuje odpowiadające sobie symbole sekwencji P i S: t oraz e. Ponieważ symbole są różne od siebie, porównywanie zostaje zakończone i następuje przesunięcie wzorca P wzdłuż sekwencji S. Liczba (miejsc) symboli, o którą przesuwa się sekwencję wzorca P, zależy od porównywanych symboli i od wyników porównań. Ponieważ symbol e, który występuje w sekwencji S na pozycji wskaźnika $\uparrow$, występuje także w sekwencji wzorca P (na pierwszej pozycji), dokonywane jest takie przesunięcie wskaźnika $\uparrow$, aby oba symbole e znalazły się naprzeciw siebie. Odpowiada to przesunięciu o trzy pozycje.

Krok 2. Na początku kroku 2 wskaźnik ↑ ustawiony jest zatem na symbol a sekwencji S i symbolowi temu odpowiada teraz symbol t sekwencji wzorca P. Ponieważ porównywane symbole są różne od siebie oraz, dodatkowo, symbol a nie występuje nigdzie w sekwencji wzorca P, wykonywane jest przesunięcie sekwencji wzorca P o pełną długość tej sekwencji, to znaczy o 4 pozycje.

Krok 3. Na początku kroku 3 wskaźnik ↑ ustawiony jest na ostatnim symbolu k sekwencji S. Symbolowi temu odpowiada symbol t sekwencji wzorca P. Porównywane symbole są różne od siebie, zatem dokonywane jest kolejne przesunięcie sekwencji wzorca P. Ponieważ symbol k, który występuje w sekwencji S na pozycji wskaźnika ↑, występuje także w sekwencji wzorca P (na drugiej pozycji), dokonywane jest takie przesunięcie wskaźnika ↑, aby oba symbole e znalazły się naprzeciw siebie. Odpowiada to przesunięciu o dwie pozycje.

Krok 4. Na początku kroku 4 wskaźnik ↑ ustawiony jest na ostatnim symbolu, t, sekwencji S. Symbolowi temu odpowiada symbol t sekwencji wzorca P. Przez dokonanie czterech kolejnych porównań symboli w P i S stwierdza się, że wzorzec P występuje w sekwencji S począwszy od 10 pozycji.

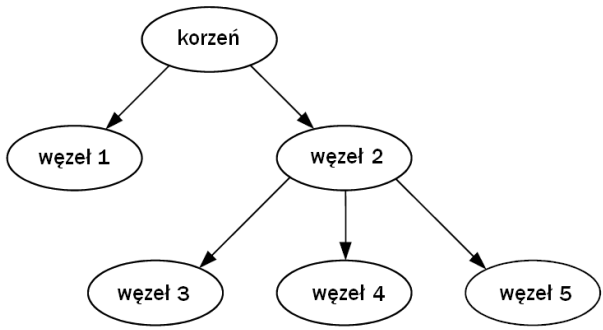
Jak widać z powyższego przykładu, stosując algorytm Boyera-Moore'a można wyszukać wzorzec P w sekwencji S w czterech krokach i z wykonaniem siedmiu porównań symboli. Gdyby do tego samego problemu zastosować prosty algorytm wyszukiwania opisany w poprzednim podpunkcie, to potrzebne byłoby 10 kroków algorytmu i 14 porównań symboli.

W typowych zadaniach wyszukiwania wzorców złożoność obliczeniowa algorytmu Boyera-Moore'a wynosi cK^S, gdzie c jest liczbą mniejszą od 1. Liczba porównań konieczna do wyszukania wzorca jest mniejsza w stosunku c od długości sekwencji S, dlatego że często przy przejściu z kolejnego do następnego kroku wykonuje się przesunięcie o więcej niż jedną pozycje. Oczywiście, podobnie jak w przypadku prostego algorytmu, czas obliczeń algorytmu Boyera-Moore'a zależy od struktury sekwencji P i S. Interesującą własnością algorytmu Boyera-Moore'a jest to, że im dłuższa jest sekwencja wzorca P, tym, na ogół, mniej porównań potrzeba do jej wyszukania. Algorytmy o złożoności obliczeniowej cK^S, gdzie $c < 1$, nazywane są czasem algorytmami subliniowymi.

Należy także zauważyć, że oprócz porównywania symboli należących do sekwencji P i S, algorytm Boyera-Moore'a wymaga także wyszukiwania symboli (wzorców) w sekwencji wzorca P. Operacje te wykonywane są wielokrotnie i muszą być odpowiednio efektywnie zaprogramowane aby cały algorytm działał efektywnie. Osiąga się to przez indeksowanie sekwencji wzorca P. Metody indeksowania tekstów (sekwencji) są omówione w następnych podpunktach.

Struktury indeksowe, drzewa, drzewa sufiksów, tablice sufiksów

Algorytmy wyszukiwania wzorca w sekwencji opisane w poprzednim podpunkcie zakładały sytuację, w której zarówno wzorzec P, jak i sekwencja S stanowią dane wejściowe. Inaczej mówiąc, algorytm jest budowany tak, jakby miał



Rysunek 3.1. Przykładowa struktura drzewa

być wykonany, dla konkretnych danych tylko raz. Jednak często spotykaną sytuacją jest powtarzanie zadań wyszukiwania różnych sekwencji wzorców P w tej samej sekwencji S. W takiej sytuacji jedną z metod poprawienia efektywności wyszukiwania jest zbudowanie pewnych struktur indeksowych, które przyspieszają działanie algorytmów. Metody rozwijania niektórych z takich struktur indeksowych przedstawione są w niniejszym podpunkcie [37], [54]. Przed przedstawieniem szeregu metod konstrukcji struktur indeksowych powinno się zwrócić uwagę na trzy aspekty oceny efektywności tych struktur. Pierwszy aspekt to złożoność obliczeniowa algorytmu budowania samej struktury indeksowej dla sekwencji S. Drugi aspekt to złożoność pamięciowa tej struktury indeksowej. Wreszcie trzeci aspekt to złożoność obliczeniowa algorytmu wyszukiwania wzorca P w sekwencji S, z wykorzystaniem zbudowanej struktury indeksowej.

Struktura drzewa

Bardzo często struktura indeksowa tworzona dla usprawnienia przeszukiwania sekwencji S ma postać drzewa. Drzewo jest to struktura budowana pamięci komputera, która z jednej strony pozwala bardzo efektywnie i ekonomicznie wykorzystywać zasoby pamięciowe komputera, a z drugiej strony istnieją efektywne algorytmy przeszukiwania informacji zapisanych w postaci drzewa. Drzewo jest to specjalny rodzaj grafu, w którym dla każdych dwóch węzłów istnieje dokładnie jedno połączenie pomiędzy nimi. Struktura (przykładowa) drzewa przedstawiona jest na rysunku 3.1. Drzewo przedstawione na rysunku 3.1 składa się z węzłów i krawędzi. Dla krawędzi używa się także terminu gałęzie. Wszystkie krawędzie (gałęzie) są skierowane. Jeśli dwa węzły mają wspólną gałąź, to istnieje między nimi relacja rodzic-dziecko. Węzeł, który nie posiada rodzica, nazywa się korzeniem drzewa, a węzły, które nie posiadają dzieci nazywa się liśćmi drzewa. W drzewie na rysunku 3.1 korzeniem jest węzeł 0, a liśćmi są węzły 3, 4 i 5. Liczba wszystkich węzłów wyznacza parametr, który nazywa się rozmiarem lub wielkością drzewa.

W typowych zastosowaniach w węzłach drzewa zapisuje się informacje istotne dla zadań wyszukiwania, a także informacje dotyczące adresów w pamięci komputera, w których zapisane są ich dzieci (także ew. rodzice). Na przykład w drzewie na rysunku 3.1 w węźle 2 powinny być zapisane adresy węzłów 3, 4 i 5. Dzięki takiej konwencji można łatwo konstruować algorytmy przeszukiwania struktury i zawartości drzew.

Drzewa słownikowe

Opis struktur indeksowych zaczynamy od najprostszego przypadku drzew słownikowych. Załóżmy, że dana jest pewna lista słów, na przykład taka jak podana poniżej.

$$\begin{aligned} &sea\$ \\ &seal\$ \\ &search\$ \\ &string\$ \\ &trie\$ \end{aligned} \tag{3.2}$$

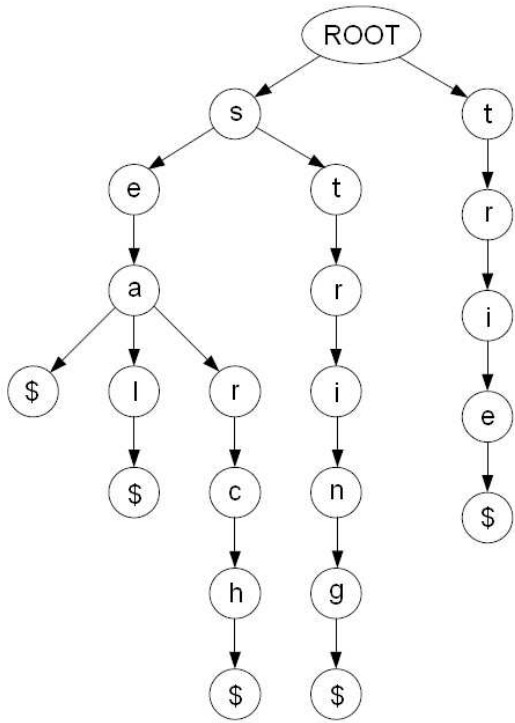
Jako służąca jedynie ilustracji lista jest bardzo krótka. Jednak w praktycznych zastosowaniach listy słów mogą składać się z setek tysięcy, a nawet milionów pozycji. Problemem, który należy rozwiązać, jest sprawdzenie, czy zadany wzorzec P jest jedną z pozycji na liście. Naiwnym, bardzo nieefektywnym podejściem byłoby porównywanie sekwencji wzorca P z kolejnymi słowami w liście. Bardziej efektywne podejście jest takie samo, jakie wszyscy stosują, wyszukując słowa w słownikach czy encyklopediach. Na przykład wyszukując słowo *drzewo*, przechodzi się w encyklopedii do litery *d*, następnie do *dr*, potem do *drz* itd.

Aby wykorzystać opisaną wyżej ideę, należy ułożyć słowa wypisane w (3.2) w drzewo. Drzewo słownikowe odpowiadające liście (3.2) przedstawione jest na rysunku 3.2. Dla oznaczenia korzenia drzewa użyto angielskiego terminu ROOT. Przed ułożeniem w drzewo, na końcu każdego ze słów dodaje się sztuczny symbol \$, pozwalający na zaznaczenie końca słowa. Aby sprawdzić, czy słowo *drzewo* występuje na liście, zaczyna się od korzenia i sprawdza się, czy wśród jego dzieci (bezpośrednich potomków) występuje litera *d*. Następnie sprawdza się, czy wśród bezpośrednich potomków węzła *d* występuje litera *r* itd.

Opracowanie programu komputerowego, który przeszukuje zawartość drzewa słownikowego jest dość łatwe i dzięki takiemu programowi można w efektywny sposób realizować zadania wyszukiwania.

Skompresowane drzewa słownikowe

Można zauważyć, że rezerwowanie oddzielnego węzła dla każdej litery słowa zapisywanego do drzewa jest raczej nieefektywne i prowadzi do niepotrzebnego



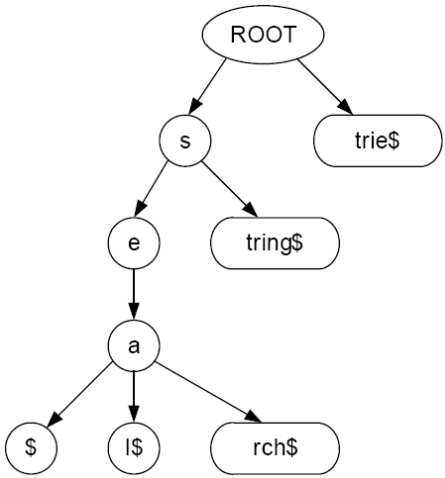
Rysunek 3.2. Drzewo słownikowe

zwiększania rozmiaru drzewa słownikowego. Bardziej efektywną konstrukcją jest skompresowane drzewo słownikowe [42], w którym nowy węzeł tworzony jest tylko wtedy, gdy konieczne jest to dla utworzenia rozgałęzienia pomiędzy słowami. Inaczej mówiąc, jeśli węzły w drzewie słownikowym, takim jak przedstawione na rysunku 3.2, tworzą sekwencję bez rozgałęzień, to łączy je się w jeden węzeł. Skompresowane drzewo słownikowe odpowiadające liście słów (3.2) przedstawione jest na rysunku 3.3.

Opracowanie programu komputerowego dla przeglądania skompresowanego drzewa słownikowego jest trochę trudniejsze niż dla zwykłego drzewa słownikowego, dlatego że trzeba przewidzieć możliwość występowania więcej niż jednego symbolu w każdym węźle drzewa. Jednak uzyskuje się oszczędność złożoności pamięciowej dzięki ograniczeniu liczby węzłów drzewa, co ma duże znaczenie szczególnie w przypadku bardzo długich list (słowników).

Drzewa sufiksowe i skompresowane drzewa sufiksowe

Dla zastosowań w bioinformatyce duże znaczenie mają jednak zadania przeszukiwania sekwencji (tekstów) inne od opisanych powyżej. Przykłady takich zadań to wyszukiwanie unikalnych sekwencji, wyszukiwanie powtórzeń, powtórzeń tandemowych, wyszukiwanie wzorców, które występują jednocześnie



Rysunek 3.3. Skompresowane drzewo słownikowe

w dwóch lub większej liczbie sekwencji. Tego typu zadania wyszukiwania mają wiele zastosowań w analizie sekwencji molekularnych. Dla ich rozwiązywania bardzo użyteczną strukturą indeksową jest struktura zdefiniowana przez drzewo sufiksowe. Idee konstrukcji drzewa sufiksowego składają się z kilku kroków, które są opisane poniżej.

Drzewo sufiksowe dla zadanej sekwencji S jest to drzewo słownikowe, w którym słownik składa się ze wszystkich sufiksów sekwencji S. Sufiks sekwencji S jest to podsekwencja, która zaczyna się w pewnym miejscu sekwencji S i kończy się na końcu sekwencji S. Na przykład dla sekwencji

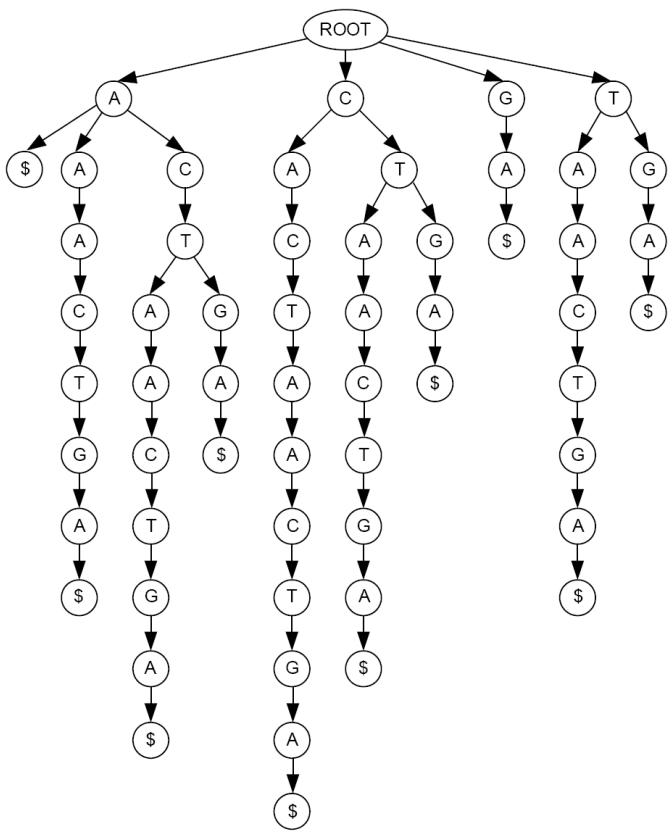
$$S = CACTAACTGA \tag{3.3}$$

która może reprezentować sekwencję nukleotydów fragmentu łańcucha DNA, lista wszystkich sufiksów jest następująca:

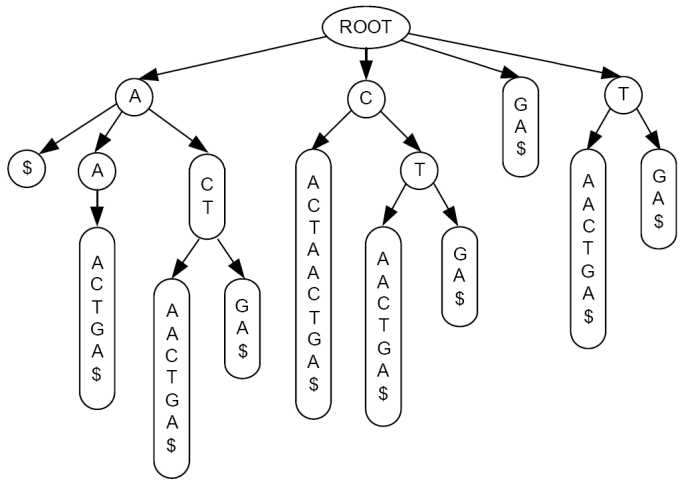
$$\begin{aligned}
&CACTAACTGA\$\\
&ACTAACTGA\$\\
&CTAACTGA\$\\
&TAACTGA\$\\
&AACTGA\$\\
&ACTGA\$\\
&CTGA\$\\
&TGA\$\\
&GA\$\\
&A\$.
\end{aligned} \tag{3.4}$$

Drzewo sufiksowe otrzymuje się jako drzewo słownikowe z powyższej listy. Dla utworzenia tego drzewa wygodnie jest ułożyć powyższe sufiksy w porządku alfabetycznyn, tzn.

$$
\begin{aligned}
&A\$\\
&AACTGA\$\\
&ACTAACTGA\$\\
&ACTGA\$\\
&CACTAACTGA\$\\
&CTAACTGA\$\\
&CTGA\$\\
&GA\$\\
&TAACTGA\$\\
&TGA\$
\end{aligned}
\tag{3.5}
$$

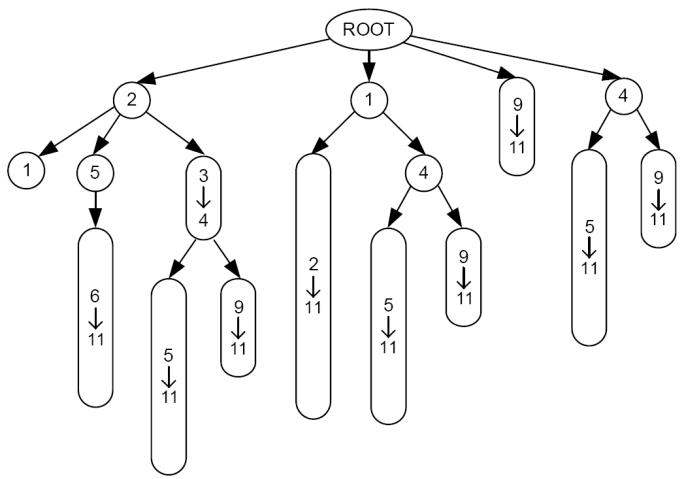


Rysunek 3.4. Zwykłe drzewo sufiksowe dla łańcucha S=CACTAACTGA



Rysunek 3.5. Skompresowane drzewo sufiksowe dla łańcucha S=CACTAACTGA

W obu przedstawionych listach na końcach słów został umieszczony dodatkowy, sztuczny symbol \$. Na podstawie listy (3.5) można utworzyć (zwykłe) drzewo sufiksowe, tak jak to przedstawiono na rysunku 3.4, a także przez łączenie węzłów, skompresowane drzewo sufiksowe, tak jak to przedstawiono na rysunku 3.5. Drzewo sufiksowe z rysunku 3.4 zawiera 58 węzłów, a skompresowane drzewo sufiksowe z rysunku 3.5 - tylko 18 węzłów. Można jednak mieć wątpliwości, czy idea kompresowania drzewa prowadzi do istotnych oszczędności pamięciowych, dlatego że co prawda węzłów jest mniej, ale każdy z połączonych węzłów musi teraz zajmować większą objętość pamięci. Dla zredu-



Rysunek 3.6. Skompresowane drzewo sufiksowe zapisane z użyciem indeksów do łańcucha S=CACTAACTGA

kowania objętości pamięci zajmowanej przez węzły skompresowanego drzewa sufiksowego stosuje się jeszcze jedną ideę. Mianowicie, zamiast przechowywać w pamięci sekwencje symboli, w węzłach skompresowanego drzewa sufiksowego zapisuje się jedynie zakresy, tzn. wskaźniki do początku i do końca podsekwencji zapisanych w węzłach. Inaczej mówiąc, skompresowane drzewo sufiksowe dla sekwencji (3.3) w praktyce wygląda tak, jak pokazano na rysunku 3.6, to znaczy w każdym węźle zapisane są tylko dwie liczby.

Tablice sufiksowe

Mimo że udaje się skompresować drzewo sufiksów, tak jak to opisano w poprzednich podpunktach, to jednak jest to struktura indeksowa, która zajmuje dość dużo pamięci (cN^2, gdzie N jest długością sekwencji), co zwłaszcza w zastosowaniach bioinformatycznych jest jej dużą wadą. Dlatego stosuje się jeszcze inne struktury o lepszej wydajności, jeśli chodzi o złożoność pamięciową.

Strukturą indeksową, która charakteryzuje się mniejszą złożonością pamięciową od drzew sufiksowych, jest struktura tablicy sufiksowej. Tablica sufiksów jest to uporządkowana leksykograficznie lista numerów wszystkich sufiksów analizowanej sekwencji. A zatem jeżeli przyporządkujemy numery od 1 do 10 wszystkim sufiksom (3.4), przy czym 1 odpowiada najkrótszemu sufiksowi a 10 najdłuższemu, to tablica sufiksowa dla sekwencji S zadanej formułą (3.3) będzie miała postać

$$10\ 5\ 2\ 6\ 1\ 3\ 7\ 9\ 4\ 8.$$

Jak widać powyżej, złożoność pamięciowa dla tablicy sufiksowej wynosi N.

Algorytmy przeszukiwania drzew sufiksowych

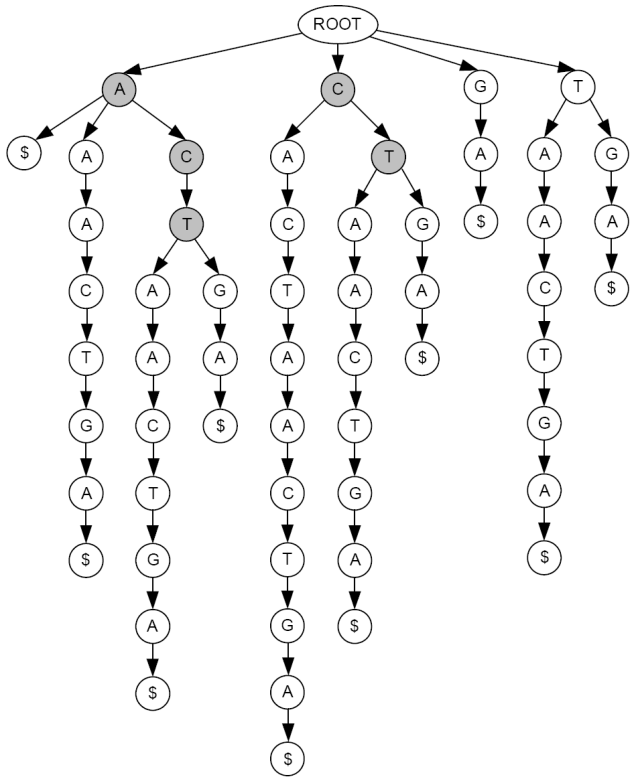
Powyżej przedstawiono kilka struktur indeksowych, które mogą znacznie przyspieszać wykonywania różnorodnych zadań przeszukiwania sekwencji. W niniejszym paragrafie zostaną przedstawione algorytmy dla zadań przeszukiwania sekwencji wykorzystujące te struktury indeksowe. Dla większej przejrzystości przedstawione zostaną algorytmy bazujące na nieskompresowanych drzewach sufiksowych. Algorytmy te pokazują zasadnicze idee i możliwości rozwiązywania zadań wyszukiwania. Należy także podkreślić, że wykorzystanie struktur indeksowych pozwala na realizację zadań wyszukiwania bardziej złożonych niż tylko wyszukiwanie wzorca w sekwencji. Zostanie to także omówione poniżej.

Wyszukiwanie wzorca w sekwencji Załóżmy, że dla sekwencji S przedstawionej w (3.3) zostało utworzone drzewo sufiksowe przedstawione na rysunku 3.4. Należy teraz rozwiązać dwa zadania wyszukiwania, wyszukania w sekwencji S wzorca

$$P_1 = ACT$$

oraz wyszukania w sekwencji S wzorca

$$P_2 = CT.$$



Rysunek 3.7. Wyszukiwanie wzorców z wykorzystaniem drzewa sufiksowego

Jak widać, oba wzorce P_1 i P_2 występują (każdy dwukrotnie) w sekwencji S. Na rysunku 3.7 przedstawiono graficznie wyniki wyszukiwania obu wzorców na bazie drzewa sufiksowego z rysunku 3.4. Położenie wzorców P_1 i P_2 zaznaczono przez zaciemnienie węzłów drzewa sufiksowego.

Poniżej przedstawiony został pseudokod programu do wyszukiwania wzorca P w sekwencji S. Podobnie jak poprzednio, k-ty symbol wzorca P oznaczany jest jako $P(k)$, K jest długością sekwencji wzorca.

Program wyszukiwanie wzorca na bazie drzewa sufiksowego

```
set node = ROOT
for k = 1 to K
    c_list ← get_children(node)
    if P(k) ∈ c_list
        node ← child_index[P(k)]
    elseif
        return(BRAK_DOPASOWANIA)
        break
```

```
    endif
endfor
return(ZNALEZIONO_DOPASOWANIE)
```

Program wykorzystuje funkcję get_children(*node*) zwracającą listę węzłów - potomków węzła *node* oraz funkcję child_index[$P(k)$], która w liście c_list wyszukuje węzeł zawierający symbol $P(k)$.

Złożoność obliczeniowa algorytmu wyszukiwania wzorca P w sekwencji symboli S jest proporcjonalna do długości sekwencji wyszukiwanego wzorca P. Zatem, jak widać, wykorzystanie struktury indeksowej drzewa sufiksów pozwala na znacznie efektywniejsze rozwiązywanie zadań wyszukiwania wzorca, niż było to możliwe z zastosowaniem poprzednio opisanych algorytmów, gdzie złożoność obliczeniowa była proporcjonalna do długości sekwencji przeszukiwanej S.

Liczba wystąpień wzorca, pozycje wystąpień wzorca Obie sekewencje wzorca, których użyto w przykładzie na rys. 3.7, występowały dwukrotnie w sekwencji S. Liczba wystąpień wzorca P w sekwencji S jest równa liczbie sufiksów sekwencji S, które zaczynają się od sekwencji wzorca P (dla których P jest prefiksem). A zatem liczba wystąpień wzorca w sekwencji jest równa liczbie liści drzewa sufiksowego zawierających wśród swych przodków sekwencję wzorca P. Pozycja (pozycje) wzorca P w sekwencji S jest wyznaczona przez długość sufiksu (sufiksów) S, dla których P jest prefiksem. Widać to na rysunku 3.7, gdzie dla obu sekwencji wzorców liczba liści drzewa sufiksów generowanych przez te wzorce wynosi 2.

Unikalne wzorce W wielu zagadnieniach biologii obliczeniowej i bioinformatyki bardzo ważnym zagadnieniem jest wyszukiwanie możliwie krótkich unikalnych sekwencji DNA, to znaczy takich sekwencji, które nie występują nigdzie indziej poza jednym miejscem w chromosomie (organiźmie). Takie problemy mają zastosowanie przy: projektowaniu starterów dla reakcji PCR, projektowaniu sond dla mikromacierzy DNA, identyfikacji organizmów na podstawie fragmentów DNA, czy też przy wyszukiwaniu charakterystycznych miejsc w DNA pomagających wyznaczać pozycje wzdłuż łańcuchów DNA. Zadanie wyszukania unikalnego wzorca sprowadza się do przeszukiwania lub wielokrotnego przeszukiwania sekwencji S. Ponieważ zastosowanie struktur indeksowych usprawnia zadania przeszukiwania, wszystkie algorytmy poszukiwania unikalnych wzorców wykorzystują struktury indeksowe.

Powtarzające się wzorce Zadaniem, które często rozwiązuje się w analizach sekwencji DNA, jest poszukiwanie powtarzających się wzorców. Powtórzenia mogą być dokładne lub przybliżone. Wyróżnia się także klasę tzw. powtórzeń tandemowych, to znaczy takich powtórzeń, dla których powtarzane motywy (wzorce) układają się bezpośrednio jeden za drugim. Formułuje się także zadania wyszukania np. najdłuższej powtarzającej się podsekwencji w sekwencji S. Dla tego typu zadań rozwijane efektywne algorytmy także głównie bazują na odpowiednio definiowanych i wykorzystywanych strukturach indeksowych.

Algorytm budowania drzewa sufiksowego

Zanim jednak wykorzystamy strukturę indeksową drzewa sufiksowego dla operacji przeszukiwania takich jak opisane powyżej, najpierw musi zostać utworzona w pamięci komputera. W tym podpunkcie opisany jest prosty algorytm konstrukcji drzewa sufiksowego [54]. Idea działania tego algorytmu przedstawiona jest na rysunku 3.8. Zakłada się, że sekwencją, dla której należy zbudować drzewo sufiksowe, jest $S = CATCA$. A zatem drzewo buduje się z następujących sufiksów:

$$CATCA\$$$
$$ATCA\$$$
$$ATCA\$$$
$$TCA\$$$
$$CA\$$$
$$A\$$$

Podobnie jak poprzednio dla zaznaczenia końca każdego sufiksu dodano unikalny symbol \$.

Dla zbudowania drzewa nastepujące operacje wykonywane są kolejno dla pobieranych z powyższej listy sufiksów:

(1) Znajdź węzeł drzewa sufiksowego, do którego należy dodać pobrany sufiks (którego potomkiem ma być pobrany sufiks).

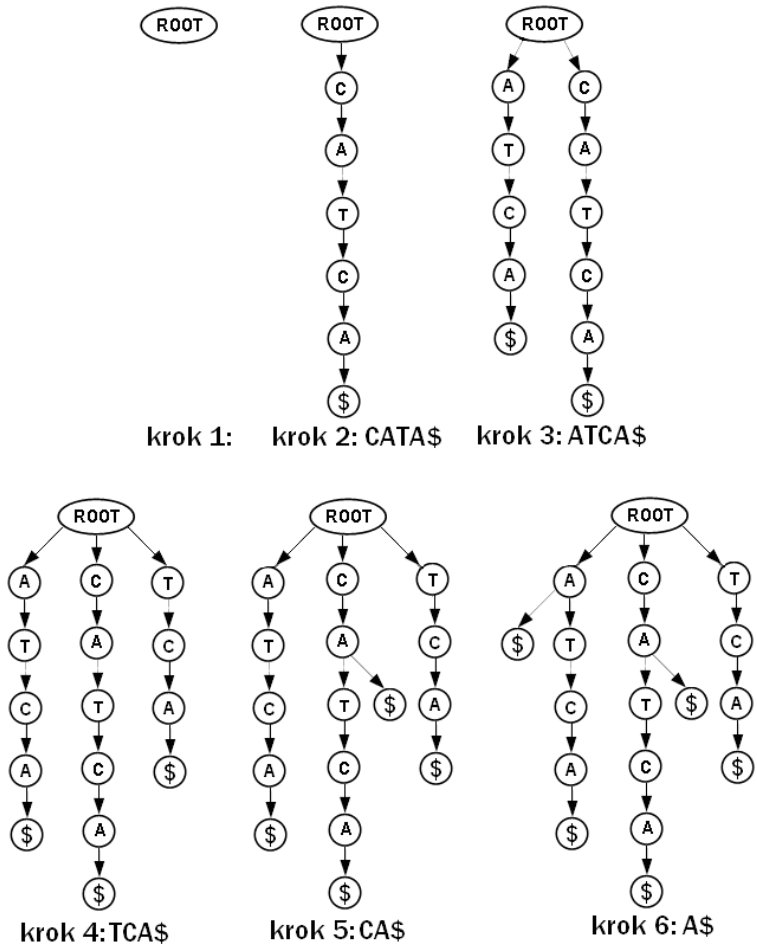
(2) Dodaj sufiks do drzewa przez dołożenie potomka do znalezionego węzła.

(3) Pobierz nowy sufiks z tabeli i wróć do (1).

Operacja (1) znajdowania węzła drzewa sufiksowego, dla którego należy utworzyć potomka, jest podobna do opisanego już algorytmu wyszukania wzorca na bazie drzewa sufiksowego. Jeśli prefiks analizowanego sufiksu nie występuje w utworzonym już drzewie sufiksowym, tak jak w krokach 2, 3 i 4 na rysunku 3.8, wtedy zapisywany jest jako potomek korzenia drzewa. Jeśli prefiks analizowanego sufiksu występuje w drzewie, tak jak w krokach, 5 i 6 na rysunku 3.8, wtedy nowy potomek tworzony jest w węźle, w którym kończy się możliwość dalszego dopasowywania znaków.

Transformacja Burrowsa-Wheelera

Jak już wspominano, w zastosowaniach w genomice, operacje przeszukiwania wykonywane są na bardzo długich sekwencjach. Dlatego przy konstrukcji algorytmów przeszukiwania bardzo istotne znaczenie ma zarówno złożoność obliczeniowa jak też złożoność pamięciowa. W ostatnich latach coraz większym zainteresowaniem cieszą się metody przeszukiwania sekwencji genomowych wykorzystujące transformację Burrowsa-Wheelera [12]. Transformacja Burrowsa-Wheelera z jednej strony zapewnia strukturę indeksową pozwalającą na wykonywanie zadań przeszukiwania, a z drugiej strony pozwala na jednoczesną kompresje przeszukiwanych sekwencji.



Rysunek 3.8. Algorytm konstrukcji drzewa sufiksowego

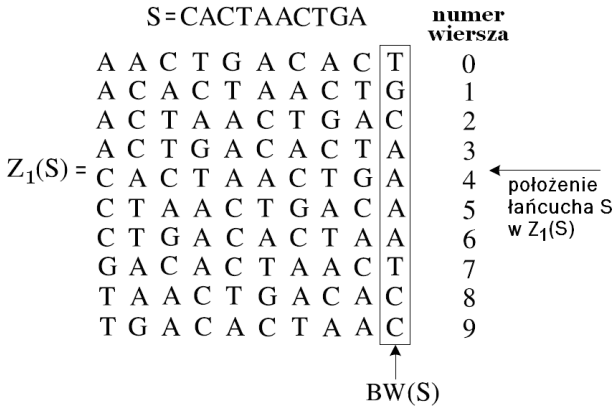
Dla zilustrowania konstrukcji transformacji Burrowsa - Wheelera załóżmy następującą sekwencję o długości $n = 10$ symboli:

$$S = CACTAACTGA. \tag{3.6}$$

Transformacja Burrowsa-Wheelera (BW) sekwencji S jest konstruowana następująco. Najpierw, na podstawie sekwencji S budowana jest tablica $n \times n$ wymiarowa $Z(S)$, taka że każdy wiersz tej tablicy (wiersze są numerowane od 0 do $n-1$) jest otrzymywane przez kolejne przesunięcia cykliczne sekwencji S. Tablica $Z(S)$ odpowiadająca sekwencji S przedstawiona jest na rysunku 3.9. W następnym kroku wiersze tablicy $Z(S)$ są sortowane alfabetycznie (leksykograficznie). W ten sposób otrzymuje się następną tablicę o wymiarach $n \times n$, którą oznacza się $Z_1(S)$, przedstawioną na rysunku 3.10. Ostatnia kolumna

		numer wiersza
	C A C T A A C T G A	0
	A C A C T A A C T G	1
	G A C A C T A A C T	2
	T G A C A C T A A C	3
Z(S) =	C T G A C A C T A A	4
	A C T G A C A C T A	5
	A A C T G A C A C T	6
	T A A C T G A C A C	7
	C T A A C T G A C A	8
	A C T A A C T G A C	9

Rysunek 3.9. Tablica $Z(S)$ powstająca przez cykliczne przesunięcia łańcucha S



Rysunek 3.10. Tablica $Z(S)_1$ powstająca przez uporządkowanie leksykograficzne wierszy tablicy $Z(S)$

tablicy $Z_1(S)$ jest transformatą Burrowsa-Wheelera sekwencji S, oznaczaną przez $BW(S)$. Zatem

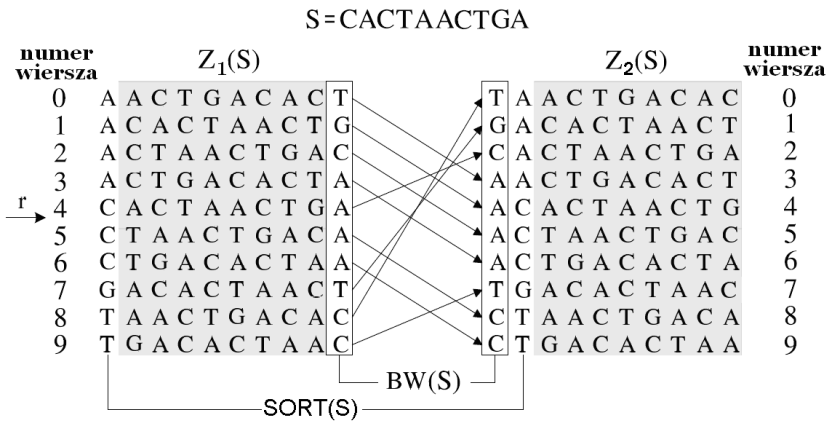
$$BW(S) = TGCAAAATCC. \tag{3.7}$$

Czy jest możliwe obliczenie odwrotnej transformacji Burrowsa-Wheelera, to znaczy odtworzenie sekwencji S na podstawie transformaty $BW(S)$? Transformacja BW sekwencji S, a także każdej innej sekwencji powstałej z S przez przesunięcie cykliczne jest taka sama i zadana jest przez sekwencję $BW(S)$ opisaną we wzorze (3.7). W tym sensie transformacja BW jest nieodwracalna. Jednak na podstawie transformacji BW udaje się odtworzyć oryginalną sekwencję, jeśli dodatkowo założy się pewne informacje pozwalające na uzyskanie danych o fazie (to znaczy o liczbie przesunięć cyklicznych w stosunku do oryginału). Na przykład dla uzyskania dokładnej transformacji odwrotnej może posłużyć dodatkowa informacja, w którym wierszu tablicy $Z_1(S)$ znajduje się

oryginalna sekwencja S. Na rysunku 3.10 sekwencja S znajduje się w 4 wierszu tablicy $Z_1(S)$. Ta informacja wystarcza do dokładnego odtworzenia sekwencji S na podstawie transformaty $BW(S)$. Bardziej praktyczną metodą na odwracanie transformacji BW jest wprowadzenie dodatkowego specjalnego (unikalnego) symbolu, np. \$ i umieszczenie go na końcu sekwencji S, a następnie dokonanie transformacji prostej BW sekwencji S\$. Dodatkowo zakłada się że symbol ten ma najniższy priorytet w porządku leksykograficznym. Stanowi on swojego rodzaju znacznik pozwalający na uzyskanie informacji o fazie (przesunięciu zadanego wiersza macierzy $Z(S)$ w stosunku do sekwencji S.

Odwrotna transformacja Burrowsa-Wheelera

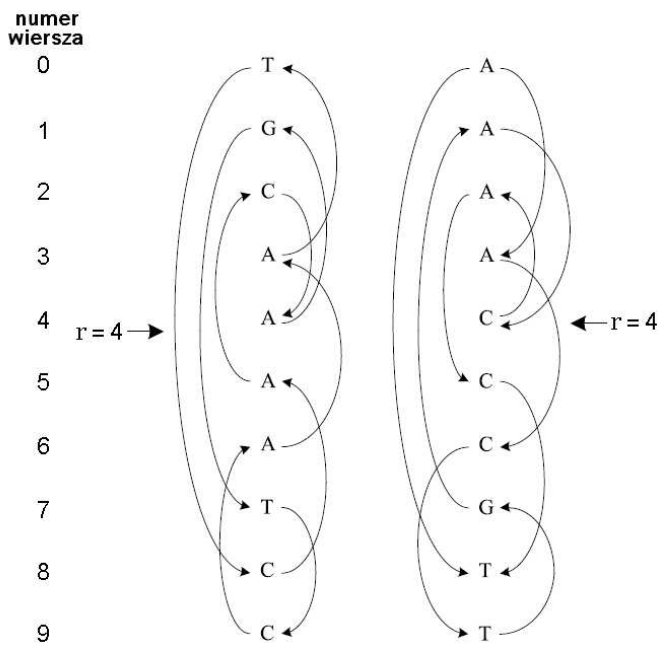
Aby opisać sposób odwracania transformacji BW, utwórzmy jeszcze jedną tablicę oznaczoną przez $Z_2(S)$, którą uzyskuje się przez przesunięcie ostatniej kolumny tablicy $Z_1(S)$ na początek. Obie tablice, $Z_1(S)$ i $Z_2(S)$, obliczone dla sekwencji S, są przedstawione na rysunku 3.11. Jak widać, pierwsza kolumna tablicy $Z_2(S)$ i jednocześnie ostatnia kolumna tablicy $Z_1(S)$ to transformata $BW(S)$ sekwencji S. Łatwo także zauważyć, że pierwsza kolumna tablicy $Z_1(S)$ i jednocześnie druga kolumna tablicy $Z_2(S)$ to kolumna, którą otrzymuje się przez posortowanie alfabetyczne symboli sekwencji S. Kolumna ta jest zatem na rysunku oznaczana przez $SORT(S)$. Na rysunku 3.11 zastosowano dodatkowo następującą konwencję. Kolumny obu tablic $Z_1(S)$ i $Z_2(S)$, których nie da się bezpośrednio uzyskać na podstawie sekwencji $BW(S)$ są zaciemnione. Na podstawie konstrukcji wiadomo, że obie tablice $Z_1(S)$ i $Z_2(S)$ zawierają wszystkie przesunięcia cykliczne sekwencji S. Jednak w tablicach $Z_1(S)$ i $Z_2(S)$ te przesunięcia cykliczne występują w różnym porządku. Można teraz zapytać: czy da się uzyskać przyporządkowanie pomiędzy odpowiadającymi sobie wierszami tablic $Z_1(S)$ i $Z_2(S)$? Wiersze w tablicy $Z_2(S)$ nie są uporządkowane alfabetycznie. Jednak jeśli spośród wierszy tablicy $Z_2(S)$



Rysunek 3.11. Odwrotna transforomacja Burrowsa-Wheelera

wybierzemy takie, które rozpoczynają się od tego samego symbolu, na przykład od litery C, to w ramach tego bloku uporządkowanie alfabetyczne będzie zachowane. To spostrzeżenie pozwala na uzyskanie odpowiedniości pomiędzy wierszami tablic $Z_1(S)$ i $Z_2(S)$. Odpowiedniość uzyskuje się w ten sposób, że w obu tablicach $Z_1(S)$ i $Z_2(S)$ wybieramy bloki zaczynające się od tego samego symbolu i definiujemy przyporządkowanie kolejno od góry do dołu. Tak zbudowane przyporządkowanie jest zaznaczone na rysunku 3.11 symbolami strzałek. Dla tego przyporządkowania (odwzorowania) będzie w dalszym ciągu stosowane oznaczenie $\Upsilon(i)$, np. $\Upsilon(0) = (3)$, $\Upsilon(1) = 4$ itd.

Odwzorowanie $\Upsilon(.)$ ma następującą własność, jeśli sekwencja występująca w i-tym wierszu tablicy $Z_1(S)$ zostanie przesunięta cyklicznie o jedną pozycję od lewej do prawej, to przesunięta sekwencja znajduje się w $j = \Upsilon(i)$ wierszu tablicy $Z_1(S)$. Z kolei odwzorowanie odwrotne $\Upsilon^{-1}(.)$ ma taką własność, że przesunięcie cykliczne o jedną pozycję od prawej do lewej zmienia położenie sekwencji z wiersza j do wiersza $i = \Upsilon^{-1}(j)$. Oba te odwzorowania mogą być użyte do odtworzenia oryginalnej sekwencji. Z rysunku 3.11 widać, że jeżeli będziemy sekwencyjnie stosowali odwzorowanie $\Upsilon(i)$ począwszy od $i = r$, to znaczy obliczali $\Upsilon(i)$, $\Upsilon(\Upsilon(i))$, itd. dla symboli sekwencji $SORT(S)$, to w efekcie uzyskamy oryginalną sekwencję S. Podobnie, jeśli będziemy sekwencyjnie stosowali odwzorowanie $\Upsilon^{-1}(i)$ począwszy od $i = r$, dla symboli sekwencji $BW(S)$, to w efekcie także uzyskamy oryginalną sekwencję S. W tym drugim



Rysunek 3.12. Odwracanie transformacji Burrowsa-Wheelera przez iterowanie odwzorowania $\Upsilon(.)$

przypadku symbole sekwencji S odtwarzane są w odwrotnej kolejności. Procedura odtwarzania oryginalnej sekwencji S przez iterowanie odwzorowania $\Upsilon(.)$ lub odwzorowania $\Upsilon^{-1}(.)$ jest przedstawiona na rysunku 3.12.

Transformacja BW jako narzędzie kompresji

Długość sekwencji - transformaty BW, $BW(S)$, jest oczywiście równa długości sekwencji S. Jednak jeśli sekwencja S jest tekstem napisanym w języku naturalnym, to okazuje się, że transformata $BW(S)$ jest sekwencją bardzo łatwą do skompresowania. Zwykle sekwencja $BW(S)$ ma postać długich ciągów powtarzających się symboli. Załóżmy na przykład, że transformacji podlega tekst angielski. W tekście tym wielokrotnie powtarza się słowo „the". Wśród przesuniętych cyklicznie sekwencji wiele będzie zaczynało się od „the", wiele będzie się także zaczynać od „he", a na ich końcach będzie występować symbol „t". Po uporządkowaniu alfabetycznym, symbole „t" na końcach tych sekwencji utworzą blok łatwy do kompresji.

Własności kompresyjne transformacji BW w odniesieniu do sekwencji DNA zostały oszacowane w pracy [27]. Tablica sufiksowa dla całego ludzkiego genomu zajmuje około 12 gigabajtów pamięci. Odpowiada to 3 miliardom (czterobajtowych) liczb całkowitych. Natomiast z zastosowaniem transformacji BW sekwencję całego ludzkiego genomu daje się skompresować do objętości około 1 gigabajta pamięci.

Intuicyjnie jest także jasne, że efektywność kompresji z zastosowaniem transformacji BW wzrasta wraz ze wzrostem długości kompresowanych sekwencji pochodzących z języków naturalnych lub z DNA.

Transformacja BW jako narzędzie przeszukiwania tekstów

W paragrafie tym zostanie opisana metoda zastosowania transformacji BW jako narzędzia przeszukiwania tekstu. Zakłada się, że analizowana jest sekwencja S zadana w (3.6). Dla zaznaczenia końca sekwencji dodawany jest znak \$, to znaczy analizowana jest sekwencja

$$S\$ = CACTAACTGA\$. \tag{3.8}$$

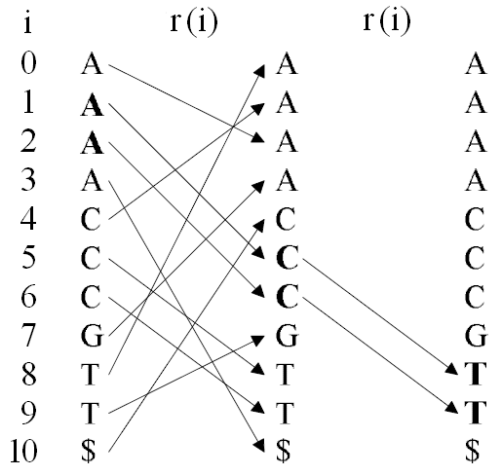
Transformacja BW sekwencji $S\$$ jest następująca:

$$BW(S\$) = TCAG\$AATCCA.$$

Na podstawie $BW(S\$)$ oraz $SORT(S\$)$ otrzymujemy odwzorowanie $\Upsilon(.)$ zadane następującą tabelą:

i	$=$	0	1	2	3	4	5	6	7	8	9	10
$\Upsilon(i)$	$=$	2	5	6	10	1	8	9	3	0	7	4

Rozważamy teraz problem wyszukania w sekwencji $S\$$ wzorca $P = ACT$. Schemat algorytmu wyszukania tego wzorca jest pokazany na rysunku 3.13.



Rysunek 3.13. Schemat algorytmu wyszukania wzorca z wykorzystaniem transformacji Burrowsa-Wheelera

Wyszukujemy w $SORT(S)$ wszystkie znaki A. Następnie dla pozycji wyszukanych znaków wyliczamy $\Upsilon(i)$ i sprawdzamy, jakie zostały otrzymane symbole. Wśród tych symboli dwukrotnie występuje C. Dla pozycji tych dwóch symboli C ponownie wyznaczamy $\Upsilon(i)$ i sprawdzamy, czy w otrzymanej liście symboli występuje z kolei T. W ten sposób stwierdzamy, że wzorzec ACT występuje w sekwencji (3.8) dwukrotnie.

Transformacja BW jako skompresowana pamięć asocjacyjna

Transformacja BW może być także uważana za rodzaj skompresowanej pamięci asocjacyjnej. Często występuje taka sytuacja, że z dużego dokumentu czy też zbioru należy zdekompresować tylko fragment. Na przykład ze skompresowanej książki należy odczytać tylko jeden rozdział, który zaczyna się zdaniem: „W niniejszym rozdziale opisano techniki kompresji". Stosując procedurę opisaną w poprzednim podpunkcie i przedstawioną na rysunku 3.13, można odnaleźć to zdanie i dalej wyczytywać treść rozdziału, wykorzystując odwzorowanie $\Upsilon(i)$.

Złożoność obliczeniowa transformacji BW

Na pierwszy rzut oka wydaje się, że złożoność pamięciowa transformacji BW jest proporcjonalna do kwadratu długości transformowanej sekwencji, N^2, dlatego że trzeba utworzyć tablice $Z_1(S)$ i $Z_2(S)$. Jednak łatwo zauważyć, że algorytm sortowania wierszy tablicy $Z_1(S)$ nie wymaga dostępu do całej tablicy $Z_1(S)$, w jednym kroku porównywane są tylko dwa wiersze tej tablicy. Dlatego można łatwo zrealizować transformację BW o złożoności pamięciowej proporcjonalnej do N oraz złożoności obliczeniowej $N\log(N)$.

Haszowanie

Haszowanie jest to technika wykorzystywana w różnorodnych zagadnieniach informatyki, polegająca na wykorzystaniu funkcji haszujących (mieszających) dla uzyskania szybkiego i randomizowanego dostępu do pamięci. Po raz pierwszy technika haszowania była zastosowana do adresowania pamięci na potrzeby funkcjonowania języków interpretacyjnych, np. języka Basic. Na przykład, jeśli wprowadzono następującą linię komendy:

$$LENGTH = 10,$$

to ma być utworzona nowa zmienna o nazwie $LENGTH$ i ma być jej przypisana wartość 10. Wygodną i bardzo skuteczną metodą zrealizowania tego wymagania jest zastosowanie funkcji haszującej, która przypisze nazwie zmiennej $LENGTH$ jakieś miejsce w pamięci. Przy następnym napotkaniu nazwy $LENGTH$ zostanie powtórzone to samo przypisanie. Na przykład jako funkcję haszującą można zastosować sumę wartości kodów ASCII odpowiadających literom występującym w nazwie zmiennej.

Jak widać, idea haszowania pozwala na bardzo szybki dostęp do pamięci. Problemem, który zawsze pojawia się przy haszowaniu, jest możliwość wystąpienia kolizji. Kolizje wynikają z faktu, że możliwe jest, że funkcja haszująca przypisze tę samą wartość różnym nazwom czy też sekwencjom. Aby w praktyce realizować ideę haszowania, konieczne jest opracowanie odpowiednich procedur rozwiązywania kolizji. Istnieje szereg algorytmów rozwiązywania kolizji, np.:

- Algorytmy liniowe. Jeśli adres wskazany przez funkcję haszującą jest już zajęty przez inną nazwę od przetwarzanej, to jako kolejną próbę bierze się następny adres w pamięci.
- Podwójne (wielokrotne haszowanie). Jeśli adres wskazany przez funkcję haszującą jest zajęty, to stosuje się inną funkcję haszującą.
- Haszowanie paczkami. Stosuje się taki system rezerwacji pamięci, że po jednym adresem można zapisać wiele zmiennych.

Idea haszowania umożliwia nie tylko rezerwację pamięci dla zmiennych, ale także rozwiązywanie wielu innych problemów w informatyce i bioinformatyce. Stosowanie funkcji haszujących pozwala na rejestrowanie, a potem na analizę występowania różnych charakterystycznych elementów w bardzo długich sekwencjach molekularnych. Pozwala to na rozwiązywanie na przykład następujących problemów:

- Porównywanie sekwencji molekularnej z sekwencjami zapisanymi w bazie danych. Dokonuje się haszowania np. wszystkich podsekwencji o długości 20 nukleotydów wszystkich sekwencji znajdujących się w bazie danych. Wyniki zapisuje się do tablicy haszującej. Każde pole tablicy zawiera informację o tym, z jakiej sekwencji molekularnej pochodzi konkretny 20 nukleotyd. Te same funkcje haszujące stosuje się dla analizowanej sekwencji molekularnej.

Badając zawartość pól tablicy haszującej, do której wskazują funkcje haszujące, można stwierdzić, z jakimi już znanymi sekwencjami molekularnymi ma wspólne podsekwencje analizowana sekwencja.

- Sekwencjonowanie (asemblacja) genomów. Algorytmy sekwencjonowania genomów, które będą jeszcze dalej opisywane w tym tekście, wymagają wyszukiwania przekrywających się par wśród bardzo wielu sekwenecji nukleotydów o długościach rzędu kilkuset nukleotydów. Rozwiązanie tego problemu można otrzymać, stosując haszowanie, podobnie jak przy poprzednim problemie.

- Wyszukiwanie powtarzających się fragmentów w sekwencjach molekularnych. Wyszukiwanie unikalnych fragmentów. Te zadania także można rozwiązać, stosując techniki haszowania.

3.2 Rekonstrukcja drzew filogenetycznych

Ważnym działem bioinformatyki jest rekonstrukcja drzew filogenetycznych. Rekonstrukcja genealogii jako problem jest rozważana od bardzo dawna w naukach biologicznych. Drzewa filogenetyczne rekonstruowano (oceniano), bazując na różnych wskaźnikach biometrycznych, które odtwarzały różnice pomiędzy gatunkami czy też organizmami. Wraz z rozwojem biologii molekularnej bardzo powszechnie dostępne stają się sekwencje molekularne, DNA, RNA oraz pierwszorzędowych struktur białek, odpowiadające badanym gatunkom czy organizmom. Obecnie, ocenę genealogii dla gatunków lub populacji najczęściej opiera się na danych dotyczących sekwencji molekularnych. Uważa się, że dane w postaci sekwencji molekularnych pozwalają na najbardziej wiarygodną ocenę historii ewolucji gatunków lub populacji. Dane takie pozwalają na wnioskowanie statystyczne dotyczące najbardziej podstawowych zdarzeń ewolucyjnych, mutacji w łańcuchach DNA lub podstawień w sekwencjach aminokwasów.

Dostępność sekwencji molekularnych pobudziła, i nadal pobudza, rozwój algorytmów obliczeniowych odtwarzania topologii i metryki drzew filogenetycznych. Algorytmy odtwarzania struktury drzew filogenetycznych dzieli się na trzy grupy: algorytmy bazujące na macierzy odległości, metody parsimonii oraz metody największej wiarygodności.

W rozdziale tym zostaną omówione oraz zilustrowane przykładami algorytmy obliczeniowe należące do tych trzech grup. Przedstawione zostaną zasady tworzenia algorytmów rekonstrukcji, a także przykłady ich użycia.

3.2.1 Podstawowe pojęcia

Drzewo jest to graf składający się z gałęzi (krawędzi) i węzłów, który ma tę własność, że dowolne dwa jego węzły są połączone dokładnie jedną ścieżką zbudowaną z gałęzi. Gałęzie w drzewach mogą być ukierunkowane lub nie. W drzewach, w których gałęzie są ukierunkowane istnieją relacje przodek - potomek

oraz rodzic - dziecko, pomiędzy węzłami. Drzewo binarne jest to drzewo, w którym gałęzie są ukierunkowane, takie że każdy węzeł ma nie więcej niż dwa węzły - dzieci. Węzły, które nie mają potomków, nazywa się liśćmi drzewa lub węzłami zewnętrznymi drzewa. Węzły, które nie są węzłami zewnętrznymi, nazywa się węzłami wewnętrznymi.

Dla drzew o ukierunkowanych gałęziach istnieje dokładnie tylko jeden węzeł, taki że jest on przodkiem wszystkich pozostałych. Nazywa się on korzeniem drzewa.

W drzewach filogenetycznych węzły i gałęzie mają interpretacje w postaci gatunków, osobników lub odpowiadających im sekwencji molekularnych oraz w postaci związków ewolucyjnych pomiędzy nimi.

Węzły. W drzewach filogenetycznych węzły nazywa się zwykle jednostkami taksonomicznymi (taxonomic units, TU).

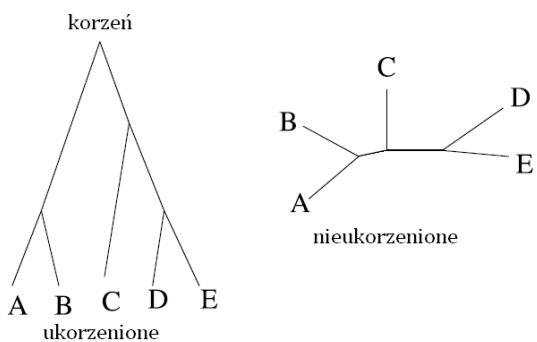
Gałęzie. Gałęzie obrazują relacje przodek - potomek, pomiędzy jednostkami taksonomicznymi.

Liście - nazywa się terminalnymi lub operacyjnymi jednostkami taksonomicznymi (operational taxonomic units, OTU), terminalnymi węzłami albo terminalnymi jednostkami taksonomicznymi.

Węzły wewnętrzne - wewnętrzne jednostki taksonomiczne.

Korzeń. Dla drzew filogenetycznych korzeń drzewa jest najwcześniejszym wspólnym przodkiem wszystkich terminalnych jednostek taksonomicznych.

Drzewa ukorzenione i nieukorzenione. Zgodnie z opisem powyżej, w drzewach filogenetycznych o ukierunkowanych gałęziach istnieje zawsze korzeń. Gdy strukturę drzewa przedstawia się graficznie, korzeń zwykle umieszcza się u góry. Wtedy wszystkie kierunki gałęzi wskazują kierunek od góry do dołu, który jest jednocześnie kierunkiem upływu czasu ewolucyjnego. W przypadku obrazowania graficznego drzew filogenetycznych nieukorzenionych drzewo rysuje się w taki sposób, aby podkreślić brak znajomości kierunku upływu czasu ewolucyjnego. Przykłady graficzne drzewa filogenetycznego ukorzenionego i nieukorzenionego przedstawiono na rysunku 3.14.



Rysunek 3.14. Przykład drzewa ukorzenionego (po lewej) i nieukorzenionego (po prawej)

Topologia drzewa. Topologia drzewa jest zdefiniowana przez relacje (przynależności) pomiędzy węzłami oraz gałęziami drzewa, analogicznie jak to ma miejsce przy definiowaniu topologii grafu. Podobnie jak w przypadku grafów, liczba możliwych różnych topologii drzew rośnie w sposób kombinatoryczny w funkcji ich liczby operacyjnych jednostek taksonomicznych. Wykorzystując zasadę indukcji matematycznej, można wyprowadzić następujące wzory na liczby możliwych topologii drzew ukorzenionych:

$$N(n) = \frac{(2n-3)!}{2^{n-2}(n-2)!}, \tag{3.9}$$

oraz nieukorzenionych

$$N(n) = \frac{(2n-5)!}{2^{n-3}(n-3)!}. \tag{3.10}$$

W powyższych wzorach N oznacza liczbę możliwych różnych topologii, a n - liczbę operacyjnych jednostek taksonomicznych.

Metryka drzewa. Metryka drzewa jest wyznaczona przez długości gałęzi. W drzewach filogenetycznych długości gałęzi mierzy się w jednostkach czasu ewolucyjnego. Jednak, oczywiście upływ czasu ewolucyjnego nie jest bezpośrednio obserwowalny. W trakcie ewolucji kumulują się mutacje, których efekty można bezpośrednio obserwować. Wielkość upływu czasu ewolucyjnego jako miarę długości gałęzi drzew filogenetycznych ocenia się na podstawie liczby mutacji (zdarzeń ewolucyjnych), które można przypisać do danej gałęzi.

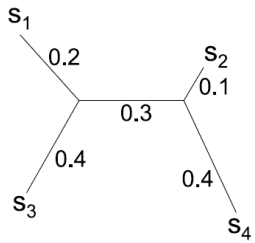
3.2.2 Metoda macierzy odległości

Metody oceny topologii i metryki drzew filogenetycznych wykorzystujące metody odległości (macierzy odległości) bazują na hipotezie, że odległość pomiędzy dwoma dowolnie wybranymi jednostkami taksonomiczymi, mierzona na bazie obserwowanych różnic sekwencji, może stanowić podstawę do oceny odległości ewolucyjnej pomiędzy tymi jednostkami taksonomicznymi, to znaczy wielkości czasu ewolucyjnego do najbliższego wspólnego przodka tych jednostek taksonomicznych [17], [21], [41], [50].

Jeśli zadane jest drzewo filogenetyczne, na przykład nieukorzenione drzewo filogenetyczne z czterema operacyjnymi jednostkami taksonomicznymi, przedstawione po lewej stronie na rysunku 3.15 oraz dla każdej gałęzi znana jest jego długość, to w trywialny sposób można policzyć macierz odległości pomiędzy wszystkimi operacyjnymi jednostkami taksonomicznymi, tak jak to jest przedstawione na rysunku 3.15 po prawej stronie. Dla macierzy odległości będzie też stosowane oznaczenie

$$D(s_1, s_2, ..., s_n) = [d(s_i, s_j)], \tag{3.11}$$

gdzie $d(s_i, s_j)$ jest odległością pomiędzy sekwencjami (operacyjnymi jednostkami taksonomicznymi) s_i oraz s_j.



	s1	s2	s3	s4
s1		0.6	0.6	0.9
s2			0.5	0.8
s3				1.1
s4				

Rysunek 3.15. Po lewej nieukorzenione drzewo o znanych długościach gałęzi. Po prawej odpowiadająca mu macierz odległości

W niniejszym podpunkcie rozważane jest zadanie odwrotne, to znaczy zadana jest macierz odległości, tak jak po prawej stronie rysunku 3.15, a problem polega na odtworzeniu topologii i metryki drzewa filogenetycznego na podstawie $D(s_1, s_2, ..., s_n)$. Przy tym konieczne jest rozważanie sytuacji, gdy prawdziwe wartości odległości $d(s_i, s_j)$ nie są bezpośrednio zadane, ale są oceniane na podstawie porównania sekwencji s_i i s_j. Wprowadza to pewne błędy, które będą występować w ocenach odległości. Poniżej przedstawione są dwa podstawowe podejścia do tego problemu i związane z nimi algorytmy.

Własność ultrametryczności, zegar molekularny

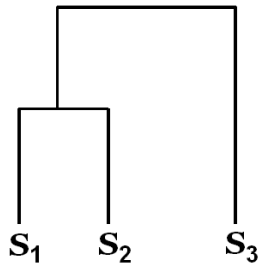
Mówimy, że macierz odległości (3.11) spełnia warunek ultrametryczności, jeśli dla dowolnych trzech operacyjnych jednostek taksonomicznych s_i, s_j i s_k zawsze dwie spośród trzech odległości $d(s_i, s_j)$, $d(s_i, s_k)$, $d(s_j, s_k)$ są sobie równe, a trzecia odległość jest mniejsza od wartości pary równych odległości. Własność ultrametryczności jest zilustrowana na rysunku 3.16. Jak widać, dla sekwencji s_1, s_2 i s_3 na tym rysunku mamy

$$d(s_1, s_3) = d(s_2, s_3) \tag{3.12}$$

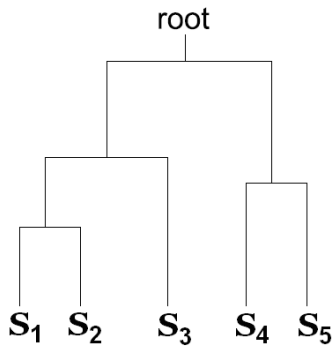
oraz

$$d(s_1, s_2) < d(s_1, s_3), d(s_1, s_2) < d(s_2, s_3). \tag{3.13}$$

Drzewa, których macierze odległości spełniają warunek ultrametryczności, nazywane są też drzewami zegara molekularnego. Określenie zegar molekularny oznacza tempo upływu czasu ewolucyjnego mierzone przez intensywność procesu mutacji, którego efekty z kolei kumulują się w obserwowanych sekwencjach molekularnych. Termin drzewo zegara molekularnego ma w sobie jeszcze następujące dodatkowe założenia: (1) wszystkie sekwencje molekularne odpowiadające operacyjnym jednostkom taksonomicznym zostały zaobserwowane obecnie, (2) intensywność procesu mutacji jest jednakowa dla wszystkich gałęzi drzewa. Drzewa zegara molekularnego (drzewa ultrametryczne) są drzewami ukorzenionymi, o ukierunkowanych gałęziach. Przykład drzewa zegara molekularnego jest przedstawiony na rysunku 3.17.



Rysunek 3.16. Ilustracja własności ultrametryczności. Zakłada się, że długości gałęzi drzewa mierzone są jedynie wzdłuż kierunku pionowego



Rysunek 3.17. Drzewo zegara molekularnego (drzewo ultrametryczne)

Drzewa ultrametryczne mają następującą własność ważną dla metodologii algorytmów odtwarzania topologii i metryki: dwie operacyjne jednostki taksonomiczne s_i, s_j, których odległość $d(s_i, s_j)$ jest minimalną odległością w macierzy odległości $D(s_1, s_2, ..., s_n)$, sąsiadują ze sobą w topologii drzewa.

Algorytm UPGMA

Algorytm UPGMA (Unweighted Pair Group Method with Arithmetic Mean) [41] służy do odtwarzania topologii i metryki drzew filogenetycznych na podstawie znanych macierzy odległości pomiędzy operacyjnymi jednostkami taksonomicznymi (sekwencjami). Algorytm w swojej konstrukcji wykorzystuje ideę łączenia najbliższych sąsiadów. Można udowodnić że w przypadku drzew ultrametrycznych oraz macierzy odległości wyznaczonej bez błędów, algorytm UPGMA pozwala na dokładne (bezbłędne) odtworzenie topologii i metryki drzewa.

Algorytm UPGMA działa iteracyjnie. W każdym kroku wykonywana jest jedna operacja łączenia. Łączenie jednostek taksonomicznych prowadzi do powstawania grup zbudowanych z operacyjnych jednostek taksonomicznych (sekwencji). Dlatego też dla odpowiedniego zaprojektowania algorytmu UPGMA

konieczne jest zdefiniowanie odległości pomiędzy grupami operacyjnych jednostek taksonomicznych (sekwencji). Dla dwóch grup sekwencji C_i i C_j, takich że $C_i \cap C_j = \emptyset$ odległość między nimi $d(C_i, C_j)$ zdefiniowana jest jako średnia arytmetyczna ze wszystkich odległości pomiędzy sekwencjami z grup C_i i C_j, to znaczy

$$d(C_i, C_j) = \frac{1}{\#C_i \#C_j} \sum_{x \in C_i} \sum_{y \in C_j} d(x, y). \tag{3.14}$$

W powyższym wzorze $\#C_i$ oraz $\#C_j$ oznaczają odpowiednio liczności grup C_i oraz C_j. Zwróćmy uwagę, że wykorzystywanie wzoru (3.14) jest dość niewygodne, zwłaszcza dla dużych drzew filogenetycznych, wymaga wykonywania operacji podwójnego sumowania po długich listach indeksów.

Istnieje iteracyjna metoda obliczania odległości $d(C_i, C_j)$. Ma ona następującą postać. Niech $C_m = C_i \cup C_j$ (tzn. C_m powstało przez połączenie C_i oraz C_j). Wtedy odległość pomiędzy C_m a dowolną inną grupą C_l można wyliczyć z następującego wzoru:

$$d(C_m, C_l) = \frac{d(C_i, C_l)\#C_i + d(C_j, C_l)\#C_j}{\#C_i + \#C_j}. \tag{3.15}$$

Na bazie tego wzoru można już opisać ostateczną postać algorytmu UPGMA, która wygląda następująco.

Inicjalizacja: Definiujemy n grup C_1, C_2,..., C_n, z których każda składa się z jednej z sekwencji s_1, s_2, ... s_n. Obliczamy macierz odległości $D(s_1, s_2, ..., s_n)$.

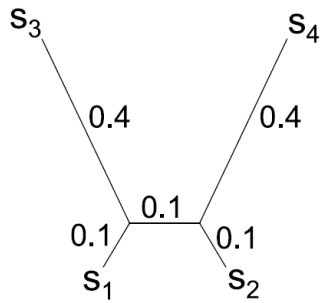
Krok iteracji: Łączymy grupy o najmniejszej odległości $d(C_i, C_j)$. Po połączeniu zmniejszamy liczbę grup o jeden, oraz przeliczamy na nowo macierz odległości pomiędzy grupami wykorzystując wzór (3.15).

Powyższa procedura iteracyjna jest powtarzana tak długo aż pozostanie jedna grupa zawierająca wszystkie sekwencje.

Algorytmy łączenia sąsiadów

Opisany w powyższym paragrafie algorytm UPGMA jest najprostszym algorytmem odtwarzania topologii oraz metryki drzew filogenetycznych. Jest bardzo często w praktyce stosowany. Przed zastosowaniem algorytmu UPGMA na ogół sprawdza się czy spełnione są warunki ultrametryczności (3.12)-(3.13) lub też, czy spełnione są one z pewnym zadanym poziomem dokładności (tolerancji).

Jeżeli drzewa filogenetyczne, które są badane, nie spełniają warunku ultrametryczności, to zastosowanie algorytmu UPGMA prowadzi na ogół do dużych błędów w ocenie ich topologii. Przyczyną jest to, że w ogólnym przypadku z faktu, że dla danej pary sekwencji s_i, s_j odległość między nimi $d(s_i, s_j)$ jest minimalną odległością w macierzy odległości $D(s_1, s_2, ..., s_n)$, nie wynika, że sekwencje te sąsiadują ze sobą w topologii drzewa. Przykład takiej sytuacji



Rysunek 3.18. Przykład drzewa, dla którego relacja sąsiedztwa nie wynika bezpośrednio z macierzy odległości

jest przedstawiony na rysunku 3.18. W topologii drzewa na tym rysunku węzły s_1 i s_3 sąsiadują ze sobą. Jeśli jednak policzymy odległości, wzdłuż gałęzi, pomiędzy wszystkimi parami operacyjnych jednostek taksonomicznych (sekwencji), to okazuje się, że najmniejsza odległość (0.3) występuje pomiędzy sekwencjami s_1 i s_2, które nie sąsiadują ze sobą w topologii drzewa.

Widać zatem, że nawet w przypadku dokładnej znajomości odległości pomiędzy terminalnymi węzłami zastosowanie zasady minimalizacji po elementach macierzy odległości nie prowadzi do wyznaczenie prawdziwego sąsiedztwa w topologii drzewa. Istnieje jednak prosta metoda modyfikacji zasady łączenia sąsiadów, która pozwala prawidłowo oceniać topologię drzewa. Metoda ta bazuje na zmodyfikowanej odległości $\delta(x, y)$ pomiędzy węzłami terminalymi x i y, zdefiniowanej następująco [50]:

$$\delta(x, y) = (N - 4)d(x, y) - \sum_{z \neq x,y} |d(x, z) + d(y, z)|. \tag{3.16}$$

Powyższy wzór obowiązuje dla drzew, które mają co najmniej 4 węzły terminalne. W przypadku drzew o trzech lub mniej liściach sąsiedztwo topologiczne zachodzi dla każdej pary węzłów. Zmodyfikowana odległość ma własność polegającą na tym, że jeśli w macierzy zmodyfikowanych odległości znajdziemy element minimalny, to para węzłów terminalnych związana z tym elementem pozostaje do siebie w relacji sąsiedztwa w topologii drzewa.

Zmodyfikowana odległość (3.16) może przyjmować wartości ujemne oraz, ogólnie nie spełnia aksjomatów odległości. Jednak mimo to może być zastosowana do oceny sąsiedztwa w drzewach filogenetycznych, bez konieczności przyjmowania założenia o ultrametryczności. Na przykład dla drzewa z rysunku 3.18 zmodyfikowane odległości pomiędzy węzłami są następujące:

$$\begin{aligned}
\delta(s_1, s_2) &= -2.2, \\
\delta(s_1, s_3) &= -2.4, \\
\delta(s_1, s_4) &= -2.2,
\end{aligned}$$

$$\delta(s_2, s_3) = -2.2,$$
$$\delta(s_2, s_4) = -2.4,$$
$$\delta(s_2, s_3) = -2.2.$$

Widać więc, że minimalna wartość, czyli -2.4 jest przyjmowana dla par węzłów s_1, s_3 oraz s_2, s_4, które ze sobą sąsiadują według topologii drzewa.

Algorytm łączenia sąsiadów, podobnie jak poprzednio opisywany algorytm UPGMA musi uaktualniać macierze odległości lub zmodyfikowanej odległości nie tylko pomiędzy węzłami terminalnymi, ale także pomiędzy dowolnymi innymi węzłami powstającymi w toku łączenia sąsiadów. W konstrukcji algorytmu zmodyfikowana macierz odległości jest wykorzystywana do wyszukiwania sąsiedztwa według topologii drzewa, a w każdym kroku uaktualniana jest macierz odległości. Uaktualnianie macierzy odległości bazuje na następującym wzorze

$$d(s, z) = \frac{1}{2}[d(x, s) + d(y, s) - d(x, y)]. \tag{3.17}$$

W powyższym wzorze $d(s, z)$ oznacza odległość pomiędzy węzłem z powstałym przez połączenie węzłów x i y, oraz dowolnym innym węzłem s.

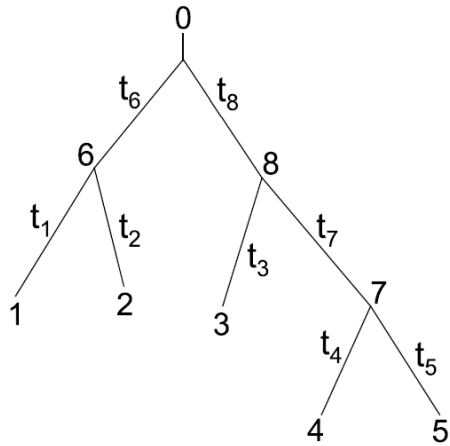
3.2.3 Drzewa największej wiarygodności

Do oceny topologii oraz metryki drzew filogenetycznych można także użyć metod maksymalizacji wiarygodności. Metoda największej wiarygodności oceny topologii i metryki drzew [18], [19], [21], [20] wykorzystuje modele Markowa substytucji nukleotydów lub aminokwasów. Na bazie odpowiednio wybranego modelu substytucji konstruuje się funkcje wiarygodności, a następnie tak dobiera się topologię i metrykę drzewa, żeby uzyskać maksymalną wartość tej funkcji.

Załóżmy na początku, że interesuje nas ewolucja tylko jednego nukleotydu ze zbioru sekwencji DNA, które tworzą operacyjne jednostki taksonomiczne hipotetycznego drzewa. Pozycja tego nukleotydu wzdłuż sekwencji oznaczana jest przez k. Niech i oraz j będą dwoma węzłami połączonymi gałęzią drzewa oraz niech kierunek tej gałęzi będzie od i do j. Przez $s_i(k)$ oraz $s_j(k)$ oznaczać będziemy stan na pozycji k-tej odpowiednio dla węzła i oraz j. Na przykład jeśli w modelu ewolucji w węźle i-tym na k-tej pozycji występuje adenina, to $s_i(k) = A$. Zakładamy jeszcze, że długość gałęzi łączącej węzły i oraz j wynosi t jednostek czasu ewolucyjnego. Wykorzystując model Markowa procesu substytucji, można policzyć prawdopodobieństwo warunkowe $P_{s_i(k), s_j(k)}(t)$ (prawdopodobieństwo, że stan na pozycji k-tej dla węzła j wynosi $s_j(k)$ pod warunkiem, że dla węzła i wynosił $s_i(k)$). Prawdopodobieństwo to jest elementem macierzy tranzycji $P(t)$

$$P(t) = \exp(Qt), \tag{3.18}$$

znajdującym się na przecięciu i-tego wiersza i j-tej kolumny. Postać macierzy intensywności Q we wzorze 3.18 zależy od przyjętego modelu substytucji.



Rysunek 3.19. Oznaczenia dla drzewa o pięciu OTU dla ilustracji metody największej wiarygodności

Powyżej opisany sposób pozwala na policzenie wiarygodności dla pojedynczej gałęzi. Łatwo go jednak rozszerzyć, tak aby można było liczyć wiarygodność całych drzew. Rozważmy np. drzewo posiadające pięć operacyjnych jednostek taksonomicznych, na rysunku 3.19. Procedura budowania funkcji wiarygodności dla drzewa z rysunku 3.19 przebiega następująco.

1. Zakładamy, że ewolucja dla każdej pozycji sekwencji molekularnej przebiega niezależnie. Wiarygodność (prawdopodobieństwo) dla całej sekwencji otrzymuje się, mnożąc prawdopodobieństwa dla poszczególnych pozycji. Zatem prawdopodobieństwo zaobserwowania w węźle j sekwencji s_j pod warunkiem, że w węźle i występowała sekwencja s_i, wylicza się jako

$$L_{s_k}^{(k)} = \left(\sum_{s_i} P_{s_k s_i}(t_i) L_{s_i}^{(i)} \right) \left(\sum_{s_j} P_{s_k s_j}(t_j) L_{s_j}^{(j)} \right). \tag{3.19}$$

2. Gdyby stany w każdym węźle drzewa były znane, to prawdopodobieństwo ich wystąpienia byłoby iloczynem odpowiednich prawdopodobieństw warunkowych (prawdopodobieństw przejść), zgodnie z poniższym wzorem

$$\begin{aligned} L = {} & \pi_{s_0} P_{s_0 s_6}(t_6) P_{s_6 s_1}(t_1) P_{s_6 s_2}(t_2) \\ & \times P_{s_0 s_8}(t_8) P_{s_8 s_3}(t_3) P_{s_8 s_7}(t_7) P_{s_7 s_4}(t_4) P_{s_7 s_5}(t_5). \end{aligned} \tag{3.20}$$

3. Jednak znane są stany tylko dla operacyjnych jednostek taksonomicznych (węzły 1, 2, 3, 4, 5). Zatem dla obliczenia wiarygodności konieczne jest posumowanie po wszystkich możliwych stanach w węzłach 0, 6, 7, 8, zgodnie z poniższym wzorem

$$L = \sum_{s_0}\sum_{s_6}\sum_{s_7}\sum_{s_8}[\pi_{s_0}P_{s_0s_6}(t_6)P_{s_6s_1}(t_1)P_{s_6s_2}(t_2) \times P_{s_0s_8}(t_8)P_{s_8s_3}(t_3)P_{s_8s_7}(t_7)P_{s_7s_4}(t_4)P_{s_7s_5}(t_5)]. \tag{3.21}$$

4. Powyższy wzór jest raczej niemożliwy do zastosowania w obliczeniach dla dużych drzew, z powodu konieczności wielokrotnego sumowania. Jednak operacje sumowania można przesunąć w taki sposób, że w sumach znajdują się tylko stany, po których występuje sumowanie. Zauważmy, że jeśli węzeł k jest bezpośrednim przodkiem węzłów i oraz j, wiarygodność drzewa którego korzeniem jest węzeł k, oznaczaną jako $L_{s_k}^{(k)}$, można wyrazić wzorem:

$$L_{s_k}^{(k)} = \left(\sum_{s_i} P_{s_ks_i}(t_i)L_{s_i}^{(i)}\right)\left(\sum_{s_j} P_{s_ks_j}(t_j)L_{s_j}^{(j)}\right). \tag{3.22}$$

W powyższym wzorze $L_{s_i}^{(i)}$ oraz $L_{s_j}^{(j)}$ oznaczają odpowiednio wiarygodności drzew, których korzeniami są węzeł i oraz węzeł j. Wzór ten pozwala na wyliczanie wiarygodności drzewa filogenetycznego w sposób iteracyjny. Dla drzewa z rysunku 3.19 najpierw liczymy $L_{s_6}^{(6)}$ i $L_{s_7}^{(7)}$, wykorzystując stany węzłów terminalnych 1 i 2 oraz 4 i 5, następnie liczymy $L_{s_8}^{(8)}$, wykorzystując stan węzła terminalnego i policzoną wcześniej wiarygodność (tablicę wiarygodności) $L_{s_7}^{(7)}$ oraz ostatecznie $L_{s_0}^{(0)} = L$, wykorzystując $L_{s_6}^{(6)}$ i $L_{s_8}^{(8)}$. Prowadzi to do następującego wzoru na wiarygodność:

$$L = \sum_{s_0}\pi_{s_0}\left(\sum_{s_6}P_{s_0s_6}(t_6)P_{s_6s_1}(t_1)P_{s_6s_2}(t_2)\right) \times\left(\sum_{s_8}P_{s_0s_8}(t_8)P_{s_8s_3}(t_3)\left(\sum_{s_7}P_{s_8s_7}(t_7)P_{s_7s_4}(t_4)P_{s_7s_5}(t_5)\right)\right). \tag{3.23}$$

Zasada dźwigni

Najczęściej jako model substytucji (przejść) pomiędzy stanami $s_i(k) \to s_j(k)$ wykorzystuje się model łańcucha Markowa, który posiada własność odwracalności. Na przykład modele Jukesa-Cantora, Felsensteina oraz HKY opisywane w rozdziale o modelach substytucji, są modelami odwracalnymi. Własność odwracalności modelu przejść pomiędzy stanami prowadzi do tzw. zasady dźwigni [19], która mówi, że możliwe jest przesuwanie położenia korzenia drzewa pomiędzy różnymi węzłami, bez zmiany wartości funkcji wiarygodności.

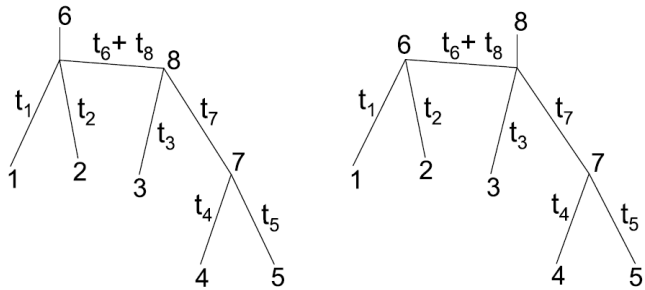
Zasadę dźwigni zilustrujemy na przykładzie drzewa z rysunku 3.19. Wykażemy, że korzeń drzewa może być przesunięty od węzła 0 do węzła 6. Z odwracalności procesu substytucji wynika istnienie następującego lokalnego bilansu:

$$\pi_{s_0}P_{s_0s_6}(t_6) = \pi_{s_6}P_{s_6s_0}(t_6). \tag{3.24}$$

Podstawiając (3.24) do wzoru (3.21) oraz zmieniając kolejność sumowania po s_6 i s_0, dostajemy następującą równoważną postać funkcji wiarygodności:

$$L = \sum_{s_6}\sum_{s_0}\sum_{s_7}\sum_{s_8}[\pi_{s_6} P_{s_6 s_0}(t_6) P_{s_6 s_1}(t_1) P_{s_6 s_2}(t_2) \times P_{s_0 s_8}(t_8) P_{s_8 s_3}(t_3) P_{s_8 s_7}(t_7) P_{s_7 s_4}(t_4) P_{s_7 s_5}(t_5)]. \tag{3.25}$$

Łatwo teraz zauważyć, że powyższa funkcja wiarygodności odpowiada drzewu przedstawionemu na rysunku 3.20 po lewej stronie. W drzewie tym korzeń umieszczony jest w węźle 6. Postępując analogicznie jak powyżej, możemy znowu przesunąć korzeń na przykład z węzła 6 do węzła 8; otrzymamy w ten sposób drzewo przedstawione na rysunku 3.20 po prawej stronie.



Rysunek 3.20. Zasada dźwigni

Przedstawione powyżej rozumowanie wykazuje, że opisane sformułowanie odpowiada maksymalizacji wiarygodności drzew nieukorzenionych. Aby z zastosowaniem metody największej wiarygodności odtwarzać topologię i metrykę drzew ukorzenionych, trzeba wprowadzić jakieś dodatkowe ograniczenia, na przykład wprowadzić wymaganie, aby metryka odtwarzanego drzewa spełniała warunek ultrametryczności.

Ocena topologii i metryki drzew największej wiarygodności

W przedstawionej w poprzednim paragrafie konstrukcji funkcji wiarygodności przyjęto założenie, że topologia drzewa oraz jego metryka (długości gałęzi) są znane. Aby zbudować algorytm odtwarzania topologii i metryki drzewa z zastosowaniem metody największej wiarygodności, konieczne jest dodatkowe rozbudowanie algorytmu opisanego w poprzednim paragrafie o procedury maksymalizacji po różnych topologiach oraz po różnych metrykach drzew. Dla zadanej topologii udaje się optymalizować funkcję wiarygodności drzewa podług metryki przez zastosowanie szybko zbieżnych procedur iteracyjnych. Natomiast optymalizacja podług różnych topologii wykonywana jest przez różnego rodzaju algorytmy heurystyczne lub przez kombinacje algorytmów heurystycznych oraz metod przeszukiwania stochastycznego typu symulowanego wyżarzania.

3.2.4 Drzewa maksymalnej parsimonii

Metoda maksymalnej parsimonii [21], [22], [39] polega na poszukiwaniu drzew oraz związanych z nimi scenariuszy ewolucyjnych, które wymagają jak najmniejszej liczby zdarzeń ewolucyjnych (mutacji) dla wytłumaczenia obserwowanych sekwencji operacyjnych jednostek taksonomicznych. Ponieważ samo zliczanie mutacji bez rozróżniania ich wag czy też prawdopodobieństw jest podejściem mniej precyzyjnym niż metody, które różnym zdarzeniom przypisują różne wagi lub różne prawdopodobieństwa, metodę maksymalnej parsimonii stosuje się głównie do oceny topologii drzew filogenetycznych dla sekwencji DNA. W tym przypadku założenie o tym, że dowolne substytucje są tak samo oceniane, może być zaakceptowane jako pewne przybliżenie rzeczywistości. W przypadku drzew filogenetycznych budowanych dla sekwencji aminokwasów uważa się, że takie przybliżenie już jest zbyt odległe od rzeczywistości.

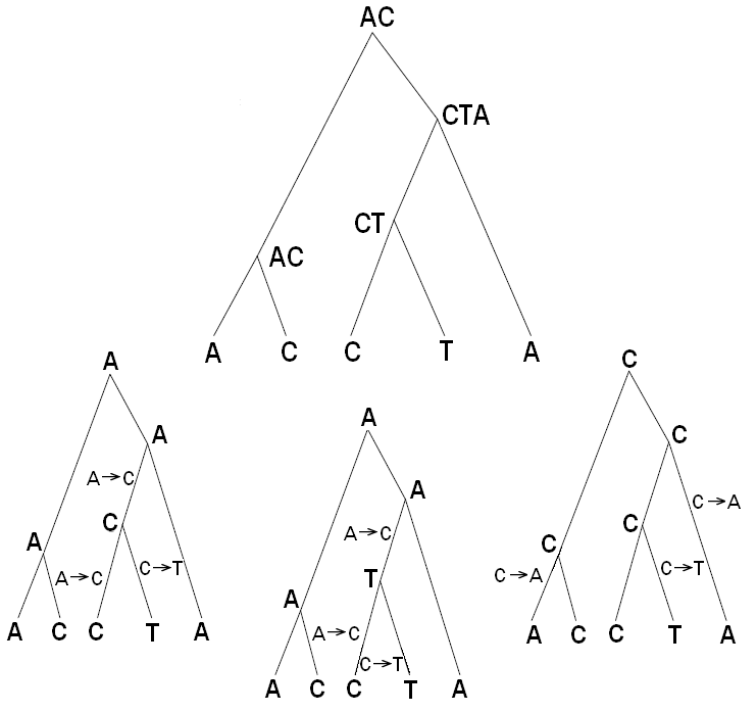
Podobnie jak w przypadku drzew największej wiarygodności wygodnie jest, przy analizie, najpierw skupić się na pojedynczej pozycji w sekwencjach DNA. Okazuje się, że dla pewnych konfiguracji występowania nukleotydów na pewnych pozycjach sekwencji DNA, niezależnie od przyjętej topologii drzewa rozwiązanie zadania minimalizacji liczby zdarzeń ewolucyjnych zawsze daje tę samą wartość. Takie konfiguracje nazywa się konfiguracjami nieinformatywnymi. Konfiguracje, dla których dla różnych topologii uzyskuje się różne wyniki zadania minimalizacji liczby zdarzeń ewolucyjnych, nazywa się konfiguracjami informatywnymi. Rozsądne jest poszukiwanie drzewa maksymalnej parsimonii ograniczyć tylko do pozycji sekwencji DNA, dla których występują konfiguracje informatywne. Warunkiem na to aby dana konfiguracja była informatywna jest aby występowały co najmniej dwa warianty nukleotydów oraz żeby dwa warianty występowały dla co najmniej dwóch operacyjnych jednostek taksonomicznych.

Algorytm dla wyznaczania minimalnej liczby zdarzeń ewolucyjnych

Konstrukcja algorytmu zostanie opisana za pomocą przykładu. Rozważamy drzewo przedstawione na rysunku 3.21, w górnej części. Drzewo to ma pięć operacyjnych jednostek taksonomicznych, rozważamy tylko jedną pozycję w sekwencjach operacyjnych jednostek taksonomicznych. Stany (nukleotydy) na tej pozycji są, jak widać, informatywne. Celem jest teraz zaproponowanie takich stanów w węzłach, które odpowiadają przodkom sekwencji operacyjnych jednostek taksonomicznych, aby liczba zdarzeń ewolucyjnych potrzebnych do zamodelowania stanów obserwowanych w operacyjnych jednostkach taksonomicznych była minimalna.

Reguła, która pozwala w sytuacjach takich jak przedstawiona na rysunku 3.21 znaleźć scenariusz ewolucyjny z minimalną liczbą mutacji, polega na zastosowaniu następującej alternatywy:

- jeśli zbiory stanów w dwóch węzłach drzewa nie mają żadnego wspólnego elementu (stanu), to jako hipotetyczny zbiór stanów ich bezpośredniego przodka przyjmuje się sumę logiczną zbiorów stanów tych dwóch węzłów,



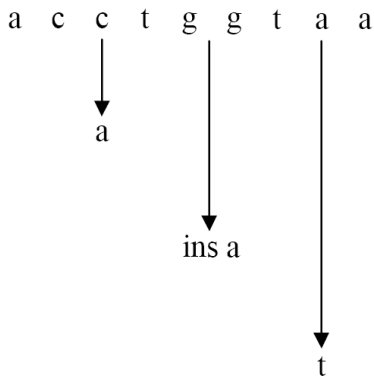
Rysunek 3.21. Powyżej - ilustracja zasady parsimonii. Poniżej - trzy scenariusze substytucji o jednakowej liczbie zdarzeń ewolucyjnych

- jeśli zbiory stanów w dwóch węzłach mają wspólne elementy, to jako hipotetyczny zbiór stanów dla ich najbliższego wspólnego przodka przyjmuje się część wspólną tych dwóch zbiorów stanów.

Powyższa reguła została zastosowana dla drzewa z górnej części rysunku 3.21. Wynikiem tego zastosowania są zbiory stanów odpowiadające wszystkim węzłom wewnętrznym drzewa, wypisane na rysunku. Na tej podstawie można wyznaczyć trzy różne scenariusze ewolucyjne, wszystkie zawierające trzy zdarzenia mutacyjne. Te scenariusze ewolucyjne przedstawione są na rysunku 3.21 w dolnej części.

Posiadając algorytm umożliwiający dla zadanej topologii drzewa obliczenie minimalnej liczby zdarzeń mutacyjnych potrzebnych do zamodelowania stanów operacyjnych jednostek taksonomicznych, można skonstruować heurystyczny algorytm poszukiwania topologii maksymalnej parsimonii. Ma on następującą postać:

(i) Załóż topologie drzewa.
(ii) Dla założonej topologii policz minimalną liczbę zdarzeń mutacyjnych koniecznych dla zamodelowania stanów operacyjnych jednostek taksonomicznych.



Rysunek 3.22. Schemat ilustrujący trzy zdarzenia ewolucyjne, które mogły wystąpić między sekwencjami s i s_1

(iii) Zmodyfikuj topologię drzewa i wróć do kroku (ii).

Istnieją heurystyczne metody modyfikacji topologii drzewa prowadzące do zmniejszania, z kroku na krok, liczby zdarzeń mutacyjnych w drzewie.

Wadą metody maksymalnej parsimonii jest to, że jej zastosowanie na ogół prowadzi do uzyskiwania dużej liczby różnych topologii, które wszystkie odpowiadają tej samej liczbie zdarzeń mutacyjnych. Wadę tę próbuje się eliminować przez konstrukcję tak zwanych drzew konsensusowych.

3.3 Uliniowienie sekwencji molekularnych

Problem uliniowienia sekwencji molekularnych polega na ocenie odpowiedniości pomiędzy nukleotydami (rybonukleotydami) lub kodonami w sekwencjach DNA albo RNA, lub pomiędzy aminokwasami w sekwencjach opisujących pierwszorzędowe struktury białek. Uliniowienie sekwencji molekularnych ma bardzo duże znaczenie w bioinformatyce, ma zastosowania w genomice, proteomice, transkryptomice. Przez rozwiązanie zadania uliniowienia można ocenić stopień podobieństwa pomiędzy dwoma sekwencjami, można także stwierdzić, że jedna sekwencja jest częścią drugiej lub że mają wspólną część. Zadanie uliniowienia może dotyczyć pary (dwóch) sekwencji molekularnych lub też ich większej liczby. W tym ostatnim przypadku mówi się o uliniowieniu blokowym.

Problem uliniowienia jest jednym z podstawowych w bioinformatyce i poświęcona jest mu bardzo szeroka literatura, np. [16], [31], [32], [40], [61]. W zależności od tego, jaki wariant problemu uliniowienia jest rozwiązywany, stosowane metody matematyczne i obliczeniowe mogą się różnić. Dla zadania uliniowienia dwóch sekwencji podstawową metodą, którą się stosuje, jest metoda programowania dynamicznego [17], [45], [53], [59].

Gdy rozwiązuje się problem uliniowienia dwóch lub więcej sekwencji molekularnych, należy mieć na uwadze fakt, że obserwowane podobieństwa i róż-

nice w sekwencjach wynikają z procesów ewolucji molekularnej, w której przez replikację tworzone są nowe, identyczne kopie sekwencji DNA, jednak w procesie replikacji mogą zachodzić błędy (mutacje) prowadzące do różnic pomiędzy sekwencjami. Najprostsze zdarzenia mutacyjne, które stosuje się w modelach uliniowienia sekwencji, to substytucje, insercje oraz delecje. Na przykład, jeśli w trakcie ewolucji sekwencji nukleotydów

$$s = acctggtaaa \tag{3.26}$$

nastąpi, na trzeciej pozycji, substytucja $c \to a$, na przedostatniej pozycji substytucja $a \to t$ oraz insercja nukleotydu c pomiędzy parę tt, tak jak to przedstawiono na rysunku 3.22, prowadzi do następującej sekwencji:

$$s_1 = acatgcgtata. \tag{3.27}$$

Sekwencje s i s_1 wykazują podobieństwo, ponieważ druga powstała z pierwszej w trakcie ewolucji, w której występowały głównie replikacje. Istniejące różnice wynikają z błędów w procesie replikacji, czyli zdarzeń mutacyjnych. Uliniowienie sekwencji zwykle przedstawia się z zastosowaniem następującego zapisu:

$$\begin{array}{l} s = \ a\ c\ c\ t\ g - g\ t\ a\ a\ a \\ \qquad\ :\ :\ \ :\ :\ \ \ :\ :\ :\ \ :\ \\ s_1 = a\ c\ a\ t\ g\ c\ \ g\ t\ a\ t\ \ a \end{array} \tag{3.28}$$

W powyższej reprezentacji znak dwukropka przedstawia dokładne dopasowanie dwóch znaków, znak myślnika insercję. Jeżeli w dwóch sekwencjach odpowiadające sobie symbole są różne, to znaczy, że zakłada się, że nastąpiło zdarzenie substytucji.

Oczywiście znana historia ewolucyjna przedstawiona na rysunku 3.22 jednoznacznie wyznacza uliniowienie pomiędzy sekwencjami s i s_1 przedstawione we wzorze (3.28). Jednak w zagadnieniach analizy danych w biologii molekularnej zawsze występuje odwrotny problem. To znaczy: zadane są dwie sekwencje, np. s i s_1 i trzeba dokonać ich uliniowienia i przedstawić wynik w postaci takiej jak we wzorze (3.28). W dalszym ciągu tego punktu omówione zostaną metody rozwiązywania tego problemu.

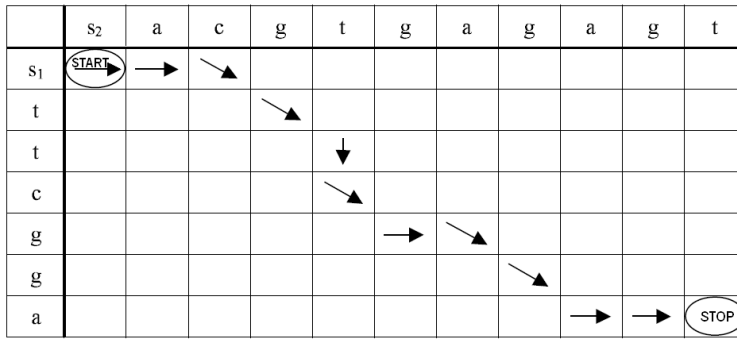
Interpretacja problemu uliniowienia jako wyboru drogi w tablicy

Opis metod rozwiązywania problemu uliniowienia zaczniemy od reprezentacji uliniowienia dla dwóch sekwencji jako drogi w dwuwymiarowej tablicy. Rozważmy problem uliniowienia dwóch sekwencji

$$s_1 = ttcgga$$

oraz

$$s_2 = acgtgagagt.$$



Rysunek 3.23. Reprezentacja uliniowienia jako wyboru drogi w tablicy

Utwórzmy dwuwymiarową tablicę, której każdy wiersz odpowiada kolejnemu symbolowi w sekwencji s_1, a każda kolumna kolejnemu symbolowi sekwencji s_2. Każde uliniowienie sekwencji s_1 i s_2 można przedstawić w postaci ścieżki (drogi) przez tę tablicę, trzy dozwolone posunięcia to ruch w prawo, ruch w dół i ruch prawo - dół. Ruch w prawo odpowiada wstawieniu insercji w sekwencji s_1, ruch w dół odpowiada wstawieniu insercji w sekwencji s_2, a ruch w prawo - dół odpowiada ustaleniu odpowiedniości pomiędzy symbolami sekwencji s_1 i s_2, przy czym, jeśli odpowiadające sobie symbole są jednakowe, to jest to dopasowanie, a jeśli są różne, to jest to równoważne założeniu wystąpienia zdarzenia mutacyjnego - substytucji. Taka reprezentacja uliniowienia jest, dla sekwencji s_1 i s_2, zilustrowana na rysunku 3.23. Ścieżka jest przedstawiona za pomocą strzałek, a poniżej tablicy jest wydrukowane odpowiadające tej ścieżce uliniowienie sekwencji s_1 i s_2. Uliniowienie to przedstawione jest z wykorzystaniem zapisu (3.28).

Tablice kropkowe

Tablice kropkowe są bardzo prostym i jednocześnie bardzo użytecznym pomysłem na praktyczne rozwiązywanie zadań uliniowienia sekwencji molekularnych. Załóżmy, że zadane są dwie sekwencje DNA

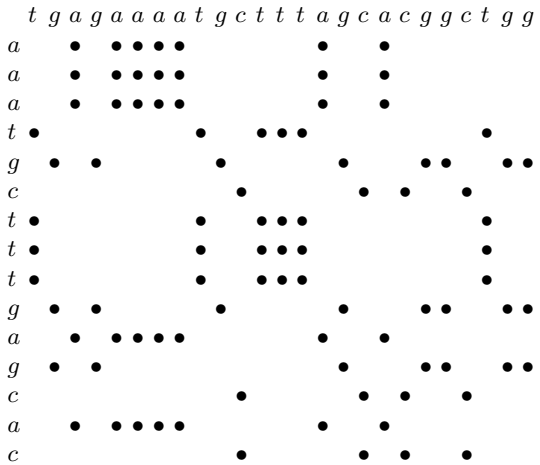
$$s_1 = tgagaaaatgctttagcacggctgg$$

oraz

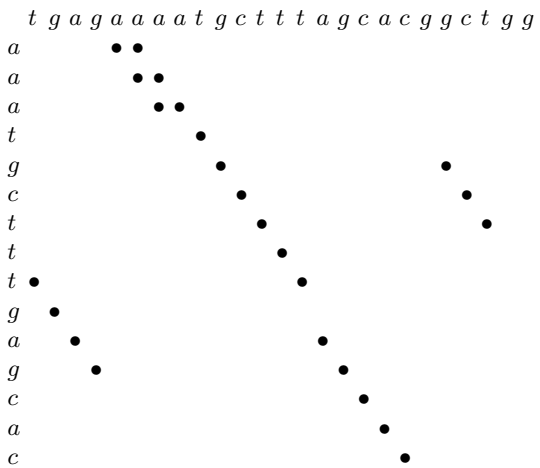
$$s_2 = aaatgctttgagcac.$$

Należy dokonać uliniowienia tych sekwencji.

Podobnie jak w poprzednim podpunkcie, układamy prostokątną tablicę, której kolumny odpowiadają symbolom w pierwszej sekwencji s_1, zaś kolumny symbolom w drugiej z sekwencji s_2. Następnie na przecięciach wierszy i ko-



Rysunek 3.24. Macierz kropkowa dla sekwencji $s_1 = tgagaaaatgctttagcacggctgg$ i $s_2 = aaatgctttgagcac$



Rysunek 3.25. Macierz kropkowa dla sekwencji $s_1 = tgagaaaatgctttagcacggctgg$ i $s_2 = aaatgctttgagcac$, z której usunięto część kropek zgodnie z zasadą filtrowania

lumn odpowiadających tym samym symbolom wstawiamy kropki. Tablica kropkowa odpowiadająca sekwencjom s_1 i s_2 przedstawiona jest na rys. 3.24. Jak widać z tego rysunku, interpretacja powstałego wzoru kropek jest jeszcze dość trudna. Dużo kropek odpowiada przypadkowym miejscom zgodności pomiędzy sekwencjami. Dla uzyskania wygodniejszego zobrazowania odpwied-

niości pomiędzy sekwencjami s_1 i s_2 można użyć dodatkowego kryterium, które pozwoli na eliminację części z przypadkowych kropek z rysunku 3.24. Kryterium to polega na pozostawianiu tylko tych kropek, które występują w co najmniej k sąsiadujących ze sobą ciągach, wzdłuż kierunku przekątnej prawo - dół. Zastosowanie takiego kryterium, przy $k = 3$ dla tablicy kropkowej z rysunku 3.24 prowadzi do przefiltrowanej tablicy kropkowej przedstawionej na rysunku 3.25. Jak widać przefiltrowana w ten sposób tablica kropkowa jest znacznie łatwiejsza do intepretacji i wykorzystania. Kierując się intuicyjnym kryterium uzyskiwania jak najdłuższych ciągów kropek wzdłuż kierunku przekątnej prawo - dół, można zaproponować następujące rozsądne uliniowienie sekwencji s_1 i s_2 :

$$\begin{array}{lc} s_1 = & a\ a\ a\ t\ g\ c\ t\ t\ t\ g\ \ a\ g\ c\ a\ c \\ & : : : : : : : : : : \quad : : : : : \\ s_2 = t\ g\ a\ g\ a\ a\ a\ a\ t\ g\ c\ t\ t\ t - a\ g\ c\ a\ c\ g\ g\ c\ t\ g\ g \end{array}$$

Funkcja jakości uliniowienia

Opisana powyżej metoda tablic kropkowych pozwala na skonstruowanie prostych i wygodnych metod uliniowienia sekwencji. Jednak w wielu sytuacjach potrzebne są algorytmy, które z jednej strony nie potrzebują nadzoru człowieka dla wyboru optymalnego uliniowienia, a z drugiej strony pozwalają na ilościową ocenę stopnia (jakości) dopasowania. Takie sytuacje pojawiają się gdy dokonuje się uliniowienia bardzo długich sekwencji DNA, gdy dokonuje się uliniowienia sekwencji kodonów lub sekwencji aminokwasów, gdy dokonuje się blokowego uliniowienia sekwencji, czy też gdy porównuje się wyniki różnych metod uliniowienia.

Bardziej zaawansowane od poprzednio opisanych i jednocześnie powszechnie w praktyce stosowane algorytmy uliniowienia sekwencji bazują na odpowiednich definicjach funkcji jakości uliniowienia. Często stosowana prosta funkcja jakości uliniowienia ma następującą postać:

$$S = n_m w_m + n_s w_s + n_g w_g \tag{3.29}$$

gdzie w_m, w_s i w_g są odpowiednio dobieranymi współczynnikami wagowymi, a n_m, n_s i n_g oznaczają odpowiednio liczbę dokładnie dopasowanych symboli, liczbę substytucji (przyporządkowanych sobie różnych symboli) oraz liczbę insercji w uliniowieniu. W prostych systemach uliniowienia system jakości zwykle konstruuje się w taki sposób, że dokładne dopasowania są premiowane, to znaczy $w_m > 0$, a substytucje i insercje są penalizowane, to znaczy $w_s < 0$ i $w_g < 0$.

Wydaje się, że powyżej skonstruowana funkcja jakości uliniowienia jest raczej intuicyjna, jednak istnieje dla niej uzasadnienie wynikające z modelowania procesu ewolucji sekwencji molekularnych. Załóżmy, że w procesie ewolucji możliwe są następujące zdarzenia: (1) uzyskanie dokładnej kopii elementu sekwencji, prawdopodobieństwo tego zdarzenia wynosi p_m, (2) substytucja,

z prawdopodobieństwem p_s oraz (3) insercja (delecja) z prawdopodobieństwem p_g. Wtedy, przy dodatkowym założeniu, że replikacje wzdłuż sekwencji molekularnej są od siebie niezależne, funkcja wiarygodności ma, z dokładnością do czynnika skalującego, następującą postać:

$$l = (p_m)^{n_m}(p_s)^{n_s}(p_g)^{n_g}.$$

Logarytmując, otrzymujemy logarytmiczną funkcje wiarygodności:

$$L = n_m \ln p_m + n_s \ln p_s + n_g \ln p_g, \tag{3.30}$$

która, jak widać, ma taką samą postać jak (3.29), przy czym należy podstawić $w_m = \ln p_m$, $w_s = \ln p_s$, i $n_g = \ln p_g$.

Często praktycznie stosowane funkcje jakości uliniowienia są bardziej skomplikowane od powyżej opisanej. Zwykle zakłada się, że zadany jest alfabet możliwych symboli Ξ, które mogą występować w uliniowianych sekwencjach. Przyjmuje się, że przyporządkowanie sobie symboli $\xi, \eta \in \Xi$ daje następujący składnik do funkcji jakości uliniowienia:

$$d(\xi, \eta).$$

Powyższa funkcja jest symetryczna $d(\xi, \eta) = d(\eta, \xi)$. Dodatkowo zadaje się składniki funkcji jakości przy przyporządkowywaniu symboli do przerw (insercji), w postaci $d(\xi, -) = d(-, \xi)$, gdzie symbol $-$ oznacza przerwę. Przy modelowaniu kosztów przerw (insercji), w bardziej zaawansowanych funkcjach kosztów, w przypadku występowania sekwencji przerw, pierwsza przerwa jest penalizowana z większym współczynnikiem niż pozostałe. Taka zasada budowania funkcji jakości wynika z argumentów ewolucyjnych, brania pod uwagę możliwości wklejenia się do sekwencji DNA nie tylko jednego, ale także kilku nukleotydów. Składniki funkcji jakości wynikające z występowania przerw często także uzależnia się od tego, czy przerwy występują na początku sekwencji (w tym przypadku nie penalizuje się ich) czy w środku.

Stworzenie algorytmu do (automatycznego) uliniowiania sekwencji molekularnych wiąże się z dwoma zagadnieniami: (1) należy wybrać lub oszacować, odpowiednią funkcję jakości uliniowienia, (2) należy rozwiązać problem optymalizacji (maksymalizacji) funkcji jakości uliniowienia podług wszystkich możliwych uliniowień. Rozwiązanie problemu (2) sprowadza się do zastosowania algorytmu programowania dynamicznego polegającego na znalezieniu optymalnej drogi przez tablicę dwuwymiarową. Rozwiązanie problemu (1) można uzyskiwać z zastosowaniem różnych podejść. Często dla zaproponowania rozwiązania problemu (1) opracowuje się i dopasowuje do danych modele procesu substytucji nukleotydów lub aminokwasów. Często też stosuje się podejście iteracyjne, rozwiązuje się zadania uliniowienia sekwencji z różnymi funkcjami jakości, a potem dobiera się taką funkcję jakości, przy zastosowaniu której wyniki w najlepszym stopniu odpowiadają oczekiwaniom. Oba zagadnienia (1) i (2) zostaną poniżej opisane.

Uliniowienie sekwencji przez zastosowanie metody programowania dynamicznego

Jako pierwsze zagadnienie zostanie opisane uliniowienie sekwencji metodą programowania dynamicznego. W tym podpunkcie zakłada się zatem, że funkcja jakości uliniowienia została już opracowana i decyduje o parametrach algorytmu maksymalizacji poniżej opisanego.

Programowanie dynamiczne [5], [15] jest to metoda rozwiązywania pewnego typu zadań optymalizacji, w której charakterystycznym elementem jest uszeregowanie decyzji optymalizacyjnych w sekwencję i organizacja wyliczania optymalnych decyzji w sposób rekurencyjny. Metoda programowania dynamicznego jest bardzo szeroko stosowana, a także bardzo użyteczna w matematyce stosowanej. Zaproponowanie dla jakiegoś problemu rozwiązania w postaci algorytmu programowania dynamicznego często nie jest oczywiste, wymaga przeprowadzenia badań, podjęcia różnych prób rozwiązania. Dla zbudowania algorytmu programowania dynamicznego pomocna jest wiedza o strukturze takich algorytmów i o pewnych ogólnych regułach ich konstruowania. (1) Problem optymalizacji musi udać się zdekomponować na ciąg oddzielnych decyzji, które można podejmować etapowo. (2) Powinno się zdefiniować stan systemu optymalizacji. Stan jest to wielkość (liczba) lub wektor, który na każdym etapie podejmowania decyzji podsumowuje efekt już podjętych decyzji optymalizacyjnych. (3) Powinno się sformułować wskaźnik (funkcję) jakości w taki sposób, żeby można go było obliczać (uaktualniać) rekurencyjnie, z jednego etapu decyzji na następny.

Możliwość rekurencyjnego uaktualniania wartości wskaźnika jakości jest podstawową cechą problemów optymalizacji dynamicznej. Uzyskanie odpowiedniej rekurencji jest podstawowym podejściem do rozwiązania. Dla uzyskania rekurencji stosuje się następującą zasadę nazywaną Zasadą Optymalności Bellmana [5]:

„Polityka optymalna ma tę własność, że niezależnie od początkowego stanu początkowych decyzji pozostałe decyzje muszą stanowić politykę optymalną ze względu na stan wynikający z początkowych decyzji".

Przez politykę optymalną rozumie się sekwencję optymalnych decyzji. Rekurencję wynikającą z powyższej Zasady Optymalności Bellmana nazywa się równaniem Bellmana . Poniżej metoda programowania dynamicznego zostanie zilustrowana przykładem, a następnie zostanie przedstawione zastosowanie tej metody do zadania uliniowienia pary sekwencji.

Ilustracja metody programowania dynamicznego

Jako ilustrację metody programowania dynamicznego rozważmy zadanie podejmowania optymalnych decyzji dla układu dynamicznego zapisanego w ogólnej postaci funkcyjnej

$$x_{k+1} = f_k(x_k, u_k), \tag{3.31}$$

gdzie $k = 0, 1, \ldots, K$ są indeksami oznaczającymi kolejne momenty czasu dyskretnego oraz jednocześnie kolejne etapy podejmowania decyzji, u_k oznaczają

zmienne decyzyjne, przez x_k oznacza się zmienne stanu, funkcja $f_k(x_k, u_k)$ jest modelem dynamiki układu. Funkcja $f_k(x_k, u_k)$ nazywana jest także funkcją albo regułą tranzycji stanu. Zmienne decyzyjne oraz stanu u_k, x_k mogą być skalarami bądź też wektorami w większej liczbie wymiarów. Zakłada się, że decyzje muszą się ograniczać do pewnych zbiorów

$$u_k \in U_k(x_k), \tag{3.32}$$

które mogą zależeć zarówno od stanów x_k, jak i od chwil czasu dyskretnego k.

Znajomość stanu w chwili $k-$tej oraz znajomość decyzji u_k, u_{k+1}, ... pozwala na wyliczenie przyszłych stanów x_{k+1}, x_{k+2}, ... bez względu na to, jakie były przeszłe stany x_{k-1}, x_{k-2}, ... oraz przeszłe decyzje u_{k-1}, u_{k-2}.

W układzie (3.31) należy podjąć decyzje w taki sposób, aby zminimalizować następujący wskaźnik (funkcję strat):

$$I(x_0) = \sum_{k=1}^{K} s_k(x_k, u_k) \tag{3.33}$$

przy dodatkowym założeniu, że w chwili początkowej $k = 0$ układ znajduje się w stanie x_0.

Do rozwiązania powyżej postawionego problemu (3.31)-(3.33) stosujemy zasadę optymalności Bellmana. Zdefiniujmy optymalny cząstkowy wskaźnik jakości jako minimum podług wszystkich decyzji u_k, u_{k+1}, ... sumy (3.33) liczonej od k-tego składnika, to znaczy następująco:

$$I_k^{opt}(x_k) = \min_{u_k, u_{k+1}, \ldots, u_K} \sum_{i=k}^{K} s_i(x_i, u_i). \tag{3.34}$$

Dla skrócenia zapisu w powyższym wzorze opuszczono warunek przynależności decyzji do zbiorów ograniczeń (3.32), jednak oczywiście zakłada się, że ograniczenia (3.32) obowiązują. Zastosowanie Zasady Optymalności pozwala na sformułowanie procesu podejmowania optymalnych decyzji w sposób rekurencyjny. Załóżmy, że układ znajduje się w ostatniej chwili czasowej $k = K$ oraz że stan układu jest zadany przez x_K. Na podstawie Zasady Optymalności można wyliczyć wartość optymalnego cząstkowego wskaźnika jakości następująco:

$$I_K^{opt}(x_K) = \min_{u_K \in U_K(x_K)} s_K(x_K, u_K), \tag{3.35}$$

a optymalna decyzja $u_K^{opt}(x_K)$ wynika z rozwiązania powyższego zadania optymalizacji statycznej, to znaczy

$$u_K^{opt}(x_K) = \arg \min_{u_K \in U_K(x_K)} s_K(x_K, u_K). \tag{3.36}$$

Zauważmy, że zadanie optymalizacji (3.35)-(3.36) obejmuje tylko jedną decyzję, u_K, jest to zatem zadanie optymalizacji statycznej, istotnie łatwiejsze do

rozwiązania niż zadania optymalizacji (dynamicznej) obejmujące wiele zmiennych decyzyjnych, jak (3.34).

Rozwiązania problemu (3.35)-(3.36), to znaczy optymalnej wartości cząstkowego wskaźnika jakości $I_K^{opt}(x_K)$ można użyć do rozwiązania kolejnego problemu, to znaczy problemu (3.34) przy $k = K-1$. Kolejne kroki tego podejścia prowadzą do iteracyjnej procedury, w której optymalną wartość cząstkowego wskaźnika dla czasu dyskretnego k uzyskuje się rozwiązując zadanie optymalizacji w której występuje rozwiązanie problemu optymalizacji (3.34) dla $k+1$ o postaci

$$\begin{aligned} I_k^{opt}(x_k) &= \min_{u_k \in U_k(x_k)} s_k(x_k, u_k) + I_{k+1}^{opt}(x_{k+1}) \\ &= \min_{u_k \in U_k(x_k)} s_k(x_k, u_k) + I_{k+1}^{opt}[f_k(x_k, u_k)], \end{aligned} \tag{3.37}$$

w którym optymalna decyzja $u_k^{opt}(x_k)$ ma postać

$$u_k^{opt}(x_k) = \arg \min_{u_k \in U_k(x_k)} s_k(x_k, u_k) + I_{k+1}^{opt}[f_k(x_k, u_k)]. \tag{3.38}$$

Równanie iteracyjne, w którym występuje operacja optymalizacji (minimalizacji) statycznej, nazywa się równaniem Bellmana. Iterując powyższe równanie Bellmana, otrzymujemy rozwiązanie sformułowanego zadania optymalizacji dynamicznej, to znaczy $I_0^{opt}(x_0)$.

Należy też zwrócić uwagę na pewną dodatkową trudność, która występuje w iteracjach (3.35)-(3.38). Każdy z sekwencji problemów optymalizacji, które trzeba rozwiązać, jest problem optymalizacji statycznej, argumentem optymalizacji jest jedna decyzja (ew. jeden wektor decyzyjny) u_k. Jednak wymienione problemy optymalizacji zawierają w sobie dodatkowe zmienne, zmienne stanu x_k. Trzeba rozwiązywać te problemy dla wielu różnych wartości x_k, są to zatem tzw. problemy optymalizacji parametrycznej. W praktyce zwykle trzeba rozwiązać zadania (3.35)-(3.38) dla całej przestrzeni zmiennych stanu, w której może znajdować się optymalna trajektoria układu dyskretnego (3.31).

Algorytm Needlemana-Wunscha

Algorytm uliniowienia dwóch sekwencji molekularnych Needlemana i Wunscha [45] wykorzystuje powyżej opisaną ideę programowania dynamicznego. Tak jak poprzednio, oznaczmy dwie sekwencje, które mają być uliniowione przez s_1 i s_2. Zakładamy, że długości tych sekwencji wynoszą odpowiednio n i m. Również oznaczamy i -ty symbol sekwencji s_1 przez $s_1(i)$, $i = 1, 2, ..., n$, a j-ty symbol sekwencji s_2 przez $s_2(j)$, $j = 1, 2, ..., m$. Te symbole (litery) należą do alfabetu Ξ.

W opisie algorytmu będzie używana ogólna notacja możliwa do zastosowania zarówno do sekwencji nukleotydów, jak i do sekwencji aminokwasów czy kodonów. Podobnie jak poprzednio dodajemy dodatkowy symbol „−” dla

oznaczenia przerwy związanej z insercją lub delecją. Zakładamy, że została stworzona funkcja jakości pozwalająca na ocenę uliniowienia, zadana przez sumę składników $d(\xi,\eta)$ oraz $d(\xi,-)$, $\xi,\eta\in\Xi$. Tak, jak to już opisano w podpunkcie 3.3, każde uliniowienie dwóch sekwencji s_1 i s_2 może być reprezentowane jako droga w tablicy dwuwymiarowej o rozmiarze $n\times m$, której wiersze są przyporządkowane symbolom sekwencji s_1, a kolumny - symbolom sekwencji s_2. Zaplanowanie drogi przez tablicę jest równoważne podjęciu sekwencji decyzji optymalizacyjnych $u_0,u_1,\ldots,u_L$, z których każda może być następującej postaci:

$$u_k=\begin{cases}\searrow & \text{prawo-dół},\\ \rightarrow & \text{prawo},\\ \downarrow & \text{dół}.\end{cases}$$

Z sekwencją decyzji $u_0,u_1,\ldots,u_L$ związana jest sekwencja stanów dla problemu programowania dynamicznego, $x_1,x_2...x_L$. Stan jest wektorem dwuwymiarowym zbudowanym z indeksów (współrzędnych) wiersza i kolumny, tzn.

$$x_k=[x_k^r\ x_k^c].$$

W powyższym wzorze x_k^r oznacza współrzędną stanu zadaną przez indeks wiersza, a x_k^c współrzędną stanu zadaną przez indeks kolumny. Reguła tranzycji stanu, którą oznaczać będziemy przez $\Phi(x_k,u_k)$, ma zatem następującą postać

$$x_{k+1}=\Phi(x_k,u_k)=\begin{cases}[x_k^r+1\ x_k^c+1] \text{ jeśli } u_k=\searrow\\ [x_k^r\ x_k^c+1] \text{ jeśli } u_k=\rightarrow\\ [x_k^r+1\ x_k^c] \text{ jeśli } u_k=\downarrow\end{cases}\tag{3.39}$$

Funkcja ta wiąże stan x_{k+1} ze stanem x_k oraz z decyzją u_k. Reguła (3.39) powinna być jeszcze dodatkowo uprecyzyjniona, mianowicie nie można inkrementować współrzędnych stanu dla granicznych pozycji, to znaczy

$$x_k^r=n\Rightarrow u_k=\rightarrow$$

oraz

$$x_k^c=m\Rightarrow u_k=\downarrow .$$

Dla zadanych $u_0,u_1,\ldots,u_L$ oraz $x_1,x_2...x_L$ funkcja jakości uliniowienia ma następującą postać:

$$I=\sum_{k=1}^{L}f(x_{k+1},u_k).\tag{3.40}$$

Każdy składnik tej funkcji, to znaczy $f(x_{k+1},u_k)$ oblicza się następująco:

$$f(x_{k+1},u_k)=\begin{cases}d(s_1(x_{k+1}^r),s_2(x_{k+1}^c)) \text{ jeśli } u_k=\searrow\\ d(-,s_2(x_{k+1}^c)) \text{ jeśli } u_k=\rightarrow\\ d(s_1(x_{k+1}^r),-) \text{ jeśli } u_k=\downarrow\end{cases}.$$

Wprowadzone powyżej definicje pozwalają na sformułowanie zadania uliniowienia sekwencji s_1 i s_2 jako zadania programowania dynamicznego w po-

staci przedstawionej w poprzednim podpunkcie. Wprowadzamy jeszcze definicję l-tej optymalnej wartości skumulowanego cząstkowego wskaźnika jakości

$$I_l^{opt}(x_l) = \max_{u_l, u_{l+1}, \ldots, u_L} \sum_{k=l}^{L} f(x_{k+1}, u_k)$$

i możemy sformułować rozwiązanie w postaci iteracji równania Bellmana

$$I_l^{opt}(x_l) = \max_{u_l \in \{\searrow, \rightarrow, \downarrow\}} f(\Phi(x_l, u_l), u_l) + I_{l+1}^{opt}(\Phi(x_l, u_l)). \tag{3.41}$$

Dodatkowo dla współrzędnych stanu zadane są wartości graniczne $x_0 = [0, 0]$ oraz $x_L = [n, m]$.

Przyjmując $I_L^{opt}(x_L) = 0$, możemy teraz wykorzystywać równanie Bellmana rekurencyjnie i w ten sposób, zaczynając od $x_L = [n, m]$, wyliczać $I_l^{opt}(x_l)$ dla wszystkich pozycji w tablicy stanów.

Jako przykład zastosowania algorytmu Needlemana i Wunscha dokonano uliniowienia dwóch sekwencji nukleotydów

$$s_1 = ttcgga \tag{3.42}$$

i

$$s_2 = acgtgagagt, \tag{3.43}$$

które występowały już na rysunku 3.23. Przyjęto następujący system oceny jakości uliniowienia: dla dokładnych dopasowań $d(\xi, \xi) = 3$, $\xi = a, c, t, g$ oraz dla substytucji i przerw $d(\xi, \eta) = -1, \xi \neq \eta$, $d(\xi, -) = -1$. Stosując ten system oceniania, zadanie uliniowienia sekwencji s_1 i s_2 zadanych przez (3.42)-(3.43) można sformułować jako przejście (od górnego lewego do dolnego prawego rogu) przez tablicę wyników przedstawioną na rysunku 3.26, w każdym polu której znajduje się odpowiedni składnik oceny jakości uliniowienia. Dalej, wykonując iteracje równania Bellmana (3.41), wyliczamy wartości optymalnych cząstkowych wskaźników jakości i zapisujemy je do tablicy $n \times m$ wymiarowej. Tablica ta jest przedstawiona na rysunku 3.27. Z tablicy tej wynika, że optymalna wartość wskaźnika jakości wynosi $I_l^{opt}(x_1) = 5$. Przy rozwiązywaniu zadania optymalizacji dynamicznej (3.39)-(3.40) zapełnia się jeszcze jedną tablicę, a mianowicie tablicę optymalnych decyzji. Tablica ta dla problemu uliniowienia sekwencji (3.42)-(3.43) ma postać przedstawioną na rysunku 3.28. Optymalne decyzje są reprezentowane w postaci strzałek. Jak widać z rysunku, w niektórych polach tablicy wpisanych jest więcej niż jedna optymalna decyzja. Jest to związane z sytuacją, gdzie kilka różnych decyzji prowadzi do rozwiązania o tej samej wartości wskaźnika jakości. Na podstawie tablicy optymalnych decyzji z rysunku 3.28 można znaleźć wszystkie rozwiązania zadania uliniowienia. Okazuje się, że dla zadanego problemu istnieje 7 różnych rozwiązań (uliniowień) prowadzących do jednakowej maksymalnej wartości wskaźnika jakości. Rozwiązania te są przedstawione na rysunku 3.29.

	S_2	a	c	g	t	g	a	g	a	g	t
S_1											
t		-1	-1	-1	3	-1	-1	-1	-1	-1	3
t		-1	-1	-1	3	-1	-1	-1	-1	-1	3
c		-1	3	-1	-1	-1	-1	-1	-1	-1	-1
g		-1	-1	3	-1	3	-1	3	-1	3	-1
g		-1	-1	3	-1	3	-1	3	-1	3	-1
a		3	-1	-1	-1	-1	3	-1	3	-1	-1

Rysunek 3.26. Tablica wyników

	S_2	a	c	g	t	g	a	g	a	g	t
S_1	5	6	7	7	3	0	0	-2	-2	-2	-2
t	6	6	7	8	4	1	1	-1	-1	-1	-5
t	6	7	4	5	5	2	2	0	0	-4	-4
c	2	3	4	5	6	3	3	1	1	-3	-3
g	-2	-1	0	1	2	3	4	1	2	-2	-2
g	-6	-5	-4	-3	-2	-1	0	1	-2	-1	-1
a	-11	-10	-9	-8	-7	-6	-5	-4	-2	-1	0

Rysunek 3.27. Tablica optymalnych wskaźników cząstkowych

	S_2	a	c	g	t	g	a	g	a	g	t
S_1	↓↘	↓↘	↘	↓	↓	↓↘	↓	↓↘	↓↘	↓	↓
t	↘	↓→	→	↘	↓	↓↘	↓	↓↘	↓	↘	↓
t	→	↘	↓→	↘	↓	↓↘	↓	↓↘	↓	↓↘	↓
c	→	→	↓→	→	↘	↘	↓	↘	↓	↓↘	↓
g	→	→	↓→	→	↘	→	↘	→	↘	↓↘	↓
g	→	→	→	→	→	→	→	↘	↓↘	↘	↓
a	→	→	→	→	→	→	→	→	→	→	STOP

Rysunek 3.28. Tablica optymalnych decyzji

s_1 : – – t t c g – g a – –
s_2 : a c g t – g a g a g t

s_1 : – t – t c g – g a – –
s_2 : a c g t – g a g a g t

s_1 : t – – t c g – g a – –
s_2 : a c g t – g a g a g t

s_1 : t t c – – g – g a – –
s_2 : a – c g t g a g a g t

s_1 : t t c g – g – – a – –
s_2 : a – c g t g a g a g t

s_1 : t t c – – g – g a – –
s_2 : – a c g t g a g a g t

s_1 : t t c g – g – – a – –
s_2 : – a c g t g a g a g t

Rysunek 3.29. Rozwiązania problemu uliniowienia metodą Needlemana-Wunscha

Algorytm Smitha-Watermana

Przedstawiony w poprzednim podpunkcie przykład pokazuje z jednej strony prostotę i łatwość zastosowania metody programowania dynamicznego do zadania uliniowienia sekwencji, a z drugiej strony jednak pewne wady zastosowanej funkcji jakości oceny uliniowienia. Problem z sekwencjami (3.42)-(3.43) został dobrany celowo, aby to uwypuklić. W wyniku rozwiązania tego problemu uzyskuje się dużo różnych uliniowień, silnie się od siebie różniących, z których trudno wybrać jedno. Można zauważyć, że fakt ten wynika przede wszystkim z tego, że uliniowiane sekwencje są różnej długości i jednocześnie algorytm uliniowienia zawiera karę za przerwy występujące na początku i końcu dłuższej sekwencji.

Sposobem ominięcia powyżej opisanych trudności jest odpowiednie zmodyfikowanie funkcji jakości. Odpowiednią konstrukcję funkcji jakości zaproponowali Smith i Waterman [53]. Ich algorytm zakłada następujące modyfikacje w stosunku do algorytmu Needlemana i Wunscha:

1. Jeśli obliczona z iteracji równania Bellmana wartość optymalna cząstkowego wskaźnika jakości staje się ujemna, to jest ona ustawiana na wartość zero.
2. Punkt początkowy uliniowienia jest przyjmowany w pozycji tablicy, dla której występuje maksymalna wartość optymalnego cząstkowego wskaźnika jakości.

Jeśli zastosuje się powyższe reguły do problemu z poprzedniego podpunktu, to otrzymuje się tablicę optymalnych wartości cząstkowych wskaźnika jakości przedstawioną na rysunku 3.30. Na tym samym rysunku przedstawiona jest strzałkami jednoznacznie wyznaczona optymalna droga przez tablicę. Tej drodze odpowiada następujące optymalne uliniowienie:

$$\begin{array}{c} s_1 = t\ \ t\ \mathbf{c}\ \mathbf{g} - \mathbf{g}\ \mathbf{a} - - - - \\ \qquad\quad : :\quad\ : : \\ s_2 = -\ a\ \mathbf{c}\ \mathbf{g}\ t\ \ \mathbf{g}\ \mathbf{a}\ g\ \ a\ \ g\ \ t \end{array}$$

Wartość wskaźnika jakości dla optymalnego uliniowienia wynosi 11.

	$\mathbf{S}_2$	**a**	**c**	**g**	**t**	**g**	**a**	**g**	**a**	**g**	**t**
$\mathbf{S}_1$		9	5	5	2	2	2	1	2	3	0
t	4	10	6	6	3	3	3	1	2	3	0
t	10	11	7	7	4	4	4	2	2	0	0
c	6	7	8	7	8	5	5	2	3	0	0
g	2	3	4	5	6	5	6	2	3	0	0
g	3	0	0	1	2	3	2	3	0	0	0
a	0	0	0	0	0	0	0	0	0	0	0

Rysunek 3.30. Macierz optymalnych wskaźników cząstkowych oraz optymalne rozwiązanie dla metody Smitha-Watermana

Strategia doboru funkcji jakości uliniowienia

Aby można było obliczyć wiarygodność (funkcję jakości) uliniowienia, należy ocenić wartości prawdopodobieństw substytucji nukleotydów lub aminokwasów oraz insercji/delecji występujące we wzorach (3.29)-(3.30). Prawdopodobieństwa te ocenia się na podstawie danych obserwacyjnych, z wykorzystaniem odpowiednich modeli matematycznych. Modele matematyczne dla procesów substytucji mają postać łańcuchów lub procesów Markowa. Ich konstrukcje, a także metody dopasowania ich do danych obserwacyjnych zostaną opisane w niniejszym podpunkcie.

Ocena prawdopodobieństw substytucji nukleotydów

Zakładamy, że proces ewolucji pojedynczego nukleotydu w polimerze DNA może być opisany przez model łańcucha Markowa, w którym możliwe stany, zaznaczane jako A, C, G, T odpowiadają nukleotydom. Zatem równanie ewolucji rozkładu prawdopodobieństwa dla pojedynczego nukleotydu ma następującą postać:

$$\pi(k+1) = \pi(k) \begin{bmatrix} p_{AA} & p_{AC} & p_{AG} & p_{AT} \\ p_{CA} & p_{CC} & p_{CG} & p_{CT} \\ p_{GA} & p_{GC} & p_{GG} & p_{GT} \\ p_{TA} & p_{TC} & p_{TG} & p_{TT} \end{bmatrix}, \tag{3.44}$$

przy czym elementy macierz tranzycji $p_{AA}, ..., p_{TT}$ opisują (warunkowe) prawdopodobieństwa substytucji, np. $p_{AC} = P[A \to C]$. Wektor $\pi(k)$ opisuje rozkład prawdopodobieństwa stanów A, C, G, T , wektor ten ma następujące składowe:

$$\pi(k) = \left[\pi_A(k) \, , \pi_C(k), \, \pi_G(k), \, \pi_T(k) \right].$$

Zdarzenia substytucji są dalej klasyfikowane jako tranzycje, gdy dotyczą zmian puryna → puryna lub pirymidyna → pirymidyna, oraz transwersje dotyczące zmian puryna → pirymidyna lub pirymidyna → puryna, tzn.

$$\left.\begin{array}{l} A \to G, \; G \to A \\ C \to T, \; T \to C \end{array}\right\} \text{ tranzycje}$$

$$\left.\begin{array}{l} A \to C, \; T \to A \\ C \to G, \; G \to C \end{array}\right\} \text{ transwersje.}$$

Problem, który nas interesuje, polega na ocenie prawdopodobieństw p_{AA}, $\ldots, p_{TT}$ występujących w macierzy tranzycji stanu we wzorze (3.44). Gdyby było możliwe rejestrowanie wszystkich kolejnych stanów w ewolucji opisanej przez model (3.44), to możliwa byłaby ocena wartości prawdopodobieństw $p_{AA}, ..., p_{TT}$ metodą największej wiarygodności, w następujący sposób

$$\hat{p}_{AA}^{ml} = \frac{\#\text{ wszystkie przejścia z } A \text{ do } A}{\#\text{ wszystkie wystąpienia } A} \tag{3.45}$$

lub

$$\hat{p}_{CT}^{ml} = \frac{\#\text{ wszystkie przejścia z } C \text{ do } T}{\#\text{ wszystkie wystąpienia } C}, \tag{3.46}$$

przy czym symbol # oznacza liczbę elementów zbioru.

Jednak zastosowanie wzorów (3.45)-(3.46) jest w praktyce niemożliwe, dlatego że dostępne dane wynikają jedynie z porównywania (uliniowienia) homologicznych fragmentów DNA. Konsekwencją tego jest (1) brak wiedzy o kierunku substytucji oraz (2) konieczności uwzględnienia rekurencyjnych substytucji, to znaczy na przykład, jeśli obserwuje się odpowiedniość $A - C$, to może

ona wynikać zarówno z substytucji (transwersji) $A \to C$, jak też z $A \to G \to C$ (lub jeszcze innej sekwencji substytucji). Metody oceny prawdopodobieństw substytucji opisane dalej muszą brać te fakty pod uwagę.

Parametryczne modele procesu substytucji nukleotydów

Z powodu specyfiki posiadanych danych obserwacyjnych , dla modelowania procesu substytucji nukleotydów duże znaczenie mają modele parametryczne. Modele takie z jednej strony redukują wymiar przestrzeni parametrów koniecznych do ocenienia, z drugiej zaś przez swoją konstrukcję zapewniają pożądane własności modelowanego procesu substytucji, ergodyczność i odwracalność. Obszerne omówienie różnych modeli parametrycznych dla procesu substytucji nukleotydów zamieszczone jest w monografii [17]. W tym punkcie zostaną przedstawione trzy różne modele parametryczne procesu substytucji: najprostszy model Jukesa - Cantora [30], bardziej elastyczny model Felsensteina [19], oraz najbardziej złożony model zwykle opisywany skrótem HKY od nazwisk autorów (Hasegawa, Kishino, Yano) [25].

Wszystkie opisywane modele posiadają zarówno wersje dyskretne dla modeli łańcuchów Markowa z czasem dyskretnym, jak również ciągłe, dla modeli w postaci procesów Markowa z dyskretną przestrzenią stanów oraz z czasem ciągłym. W poniższych modelach zawsze przez P oznacza się macierz tranzycji modelu łańcucha Markowa, a przez Q macierz intensywności przejść.

Model Jukesa-Cantora

Wersja dyskretna tego modelu jest opisana następującą macierzą prawdopodobieństw przejść:

$$P = \begin{bmatrix} 1-3\alpha & \alpha & \alpha & \alpha \\ \alpha & 1-3\alpha & \alpha & \alpha \\ \alpha & \alpha & 1-3\alpha & \alpha \\ \alpha & \alpha & \alpha & 1-3\alpha \end{bmatrix}. \tag{3.47}$$

Ponieważ w powyższym modelu jednokrokowym wartość parametru α jest zwykle bardzo mała, często więc stosuje się model Jukesa-Cantora procesu substytucji nukleotydów z czasem ciągłym, w którym macierz intensywności ma postać

$$Q = \begin{bmatrix} -3\alpha & \alpha & \alpha & \alpha \\ \alpha & -3\alpha & \alpha & \alpha \\ \alpha & \alpha & -3\alpha & \alpha \\ \alpha & \alpha & \alpha & -3\alpha \end{bmatrix}. \tag{3.48}$$

Model Jukesa-Cantora (3.47) lub (3.48) jest ergodyczny i odwracalny. Jak widać z (3.47), (3.48) wszystkie substytucje są w nim jednakowo prawdopodobne, rozkład stacjonarny jest jednostajny. Model ten zależy tylko od jednego parametru α i dla wielu zastosowań jest jednak zbyt uproszczony.

Model Felsensteina

Model Felsensteina jest opisany następującą macierzą prawdopodobieństw przejść:

$$P = \begin{bmatrix} 1-u\varphi_{CTG} & u\varphi_C & u\varphi_G & u\varphi_T \\ u\varphi_A & 1-u\varphi_{AGT} & u\varphi_G & u\varphi_T \\ u\varphi_A & u\varphi_C & 1-u\varphi_{ACT} & u\varphi_T \\ u\varphi_A & u\varphi_C & u\varphi_G & 1-u\varphi_{ACG} \end{bmatrix}. \tag{3.49}$$

Niezależne parametry tego modelu to u oraz prawdopodobieństwa φ_A, φ_C, φ_G, φ_T, przy czym

$$\varphi_A + \varphi_C + \varphi_G + \varphi_T = 1.$$

Dodatkowo we wzorze (3.49) wprowadzono oznaczenia

$$\begin{aligned} \varphi_{CTG} &= \varphi_C + \varphi_G + \varphi_T, \\ \varphi_{AGT} &= \varphi_A + \varphi_G + \varphi_T, \\ \varphi_{ACT} &= \varphi_A + \varphi_C + \varphi_T, \\ \varphi_{ACG} &= \varphi_A + \varphi_C + \varphi_G. \end{aligned} \tag{3.50}$$

Rozkład stacjonarny w modelu Felsensteina (3.49) zadany jest wektorem $[\varphi_A, \varphi_C, \varphi_G, \varphi_T]$. Wynika z tego, że model Felsensteina można dopasować do dowolnego rozkładu stacjonarnego prawdopodobieństw wystąpień nukleotydów. Model Felsensteina jest, jak łatwo sprawdzić, ergodyczny i odwracalny. Parametr u opisuje intensywność procesu substytucji, która jest jednakowa dla wszystkich przejść. Wersja ciągła modelu Felsensteina jest dana następującą macierzą intensywności:

$$Q = \begin{bmatrix} -u\varphi_{CTG} & u\varphi_C & u\varphi_G & u\varphi_T \\ u\varphi_A & -u\varphi_{AGT} & u\varphi_G & u\varphi_T \\ u\varphi_A & u\varphi_C & -u\varphi_{ACT} & u\varphi_T \\ u\varphi_A & u\varphi_C & u\varphi_G & -u\varphi_{ACG} \end{bmatrix}. \tag{3.51}$$

Model HKY

Najbardziej elastyczny z rozważanych tu modeli to model HKY. Powstaje on jako uogólnienie modelu Felsensteina w ten sposób, że zamiast jednakowej intensywności wszystkich przejść teraz przyjmuje się dwie różne intensywności u - dla tranzycji i v - dla transwersji. Macierz prawdopodobieństw przejść dla tego modelu ma następującą postać:

$$P = \begin{bmatrix} 1-\psi_{CTG} & v\varphi_C & u\varphi_G & v\varphi_T \\ v\varphi_A & 1-\psi_{AGT} & v\varphi_G & u\varphi_T \\ u\varphi_A & v\varphi_C & 1-\psi_{ACT} & v\varphi_T \\ v\varphi_A & u\varphi_C & v\varphi_G & 1-\psi_{ACG} \end{bmatrix}. \tag{3.52}$$

Model zawiera następujące niezależne parametry, intensywności u i v oraz podobnie jak poprzednio prawdopodobieństwa φ_A, φ_C, φ_G, φ_T, przy czym

$$\varphi_A + \varphi_C + \varphi_G + \varphi_T = 1.$$

Dodatkowo, oznaczenia wykorzystywane we wzorze (3.52) są następujące:

$$\begin{aligned}
\psi_{CTG} &= u\varphi_G + v(\varphi_C + \varphi_T),\\
\psi_{AGT} &= u\varphi_T + v(\varphi_A + \varphi_G),\\
\psi_{ACT} &= u\varphi_A + v(\varphi_C + \varphi_T),\\
\psi_{ACG} &= u\varphi_C + v(\varphi_A + \varphi_A).
\end{aligned}$$

Rozkład stacjonarny, tak jak poprzednio może być dowolny, jest zadany wektorem $[\varphi_A, \varphi_C, \varphi_G, \varphi_T]$.

Model HKY jest ergodyczny i odwracalny. Wersja ciągła tego modelu jest opisana macierzą intensywności

$$Q = \begin{bmatrix}
-\psi_{CTG} & v\varphi_C & u\varphi_G & v\varphi_T\\
v\varphi_A & -\psi_{AGT} & v\varphi_G & u\varphi_T\\
u\varphi_A & v\varphi_C & -\psi_{ACT} & v\varphi_T\\
v\varphi_A & u\varphi_C & v\varphi_G & -\psi_{ACG}
\end{bmatrix}. \tag{3.53}$$

Równania ewolucji rozkładów prawdopodobieństwa dla modeli substytucji nukleotydów

Dla powyżej opisanych modeli interesować nas będzie problem rozwiązania równania ewolucji rozkładu prawdopodobieństwa dla przypadku czasu ciągłego. Równanie to ma następującą postać:

$$P(t) = \exp(Qt), \tag{3.54}$$

w której t oznacza czas ewolucyjny, Q jest występującą w modelach (3.48), (3.51), (3.53) macierzą intensywności, $P(t)$ jest macierzą prawdopodobieństw przejść, zależną od czasu t. W powyższym wzorze exp() oznacza macierzową funkcję wykładniczą.

Równanie (3.54) jest macierzowym równaniem liniowym. Interesującą i bardzo ważną dla zastosowań własnością wszystkich opisanych modeli jest możliwość jawnego wyliczenia rozwiązania równania (3.54), co zostanie opisane poniżej. Rozwiązanie równania (3.54) bazuje na rozkładzie kanonicznym Jordana macierzy substytucji Q [4] o postaci

$$Q = U_Q D_Q V_Q^T. \tag{3.55}$$

W powyższym rozkładzie U_Q oraz V_Q^T są nieosobliwymi macierzami transformacji, zbudowanymi z kolumnowych i wierszowych wektorów własnych ma-

cierzy Q, które związane są zależnością

$$U_Q = (V_Q^T)^{-1},$$

przy czym dla macierzy 4×4 wymiarowych U_Q oraz V_Q^T stosuje się dodatkowo notację

$$U_Q = \left[u_{Q1}\ u_{Q2}\ u_{Q3}\ u_{Q4} \right]$$

oraz

$$V_Q^T = \begin{bmatrix} v_{Q1}^T \\ v_{Q2}^T \\ v_{Q3}^T \\ v_{Q4}^T \end{bmatrix}.$$

Symbol T oznacza transpozycję macierzy. W powyższych wzorach $u_{Q1}...u_{Q4}$ oznaczają wektory własne kolumnowe, a $v_{Q1}^T...v_{Q4}^T$ wektory własne wierszowe macierzy intensywności Q. Macierz D_Q występująca w dekompozycji (3.55) jest macierzą diagonalną, na jej przekątnej znajdują się wartości własne macierzy intensywności Q

$$D_Q = diag(q_1, q_2, q_3, q_4).$$

Wykorzystując dekompozycję (3.55), wzór (3.54) można przekształcić do postaci

$$P(t) = U_Q \exp(D_Q t) V_Q^T, \tag{3.56}$$

gdzie wartość macierzowej funkcji wykładniczej wylicza się w postaci macierzy diagonalnej

$$\exp(D_Q t) = diag\left[\exp(q_1 t), \exp(q_2 t), \exp(q_3 t), \exp(q_4 t)\right].$$

Wykorzystując wzór (3.56), można wyliczyć zależność od czasu dla prawdopodobieństwa substytucji dla dowolnej pary nukleotydów. To znaczy dla dowolnych $i, j \in \{A, C, G, T\}$ mamy

$$p_{ij}(t) = \mathbf{1}_i^T P(t) \mathbf{1}_j = u_Q^{Ti} P(t) v_Q^j. \tag{3.57}$$

W powyższym wzorze użyto oznaczenia $\mathbf{1}_i$ dla wektora kolumnowego, który na i - tej pozycji posiada jedynkę, a na pozostałych same zera.

Zaletą wszystkich opisanych modeli substytucji jest możliwość analitycznego wyznaczenia postaci dekompozycji (3.55). Wzory opisujące dekompozycje są podane poniżej. Dla modelu Jukesa-Cantora wartości własne macierzy intensywności prowadzą do następującej macierzy diagonalnej D_Q :

$$D_Q = diag(0, -4\alpha, -4\alpha, -4\alpha), \tag{3.58}$$

macierz U_Q kolumnowych wektorów własnych ma postać

$$U_Q = \begin{bmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \tag{3.59}$$

a macierz V_Q^T wierszowych wektorów własnych ma postać

$$V_Q^T = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ -0.25 & 0.75 & -0.25 & -0.25 \\ -0.25 & -0.25 & 0.75 & -0.25 \\ -0.25 & -0.25 & -0.25 & 0.75 \end{bmatrix}. \tag{3.60}$$

Dla modelu Felsensteina (3.51) macierz D_Q ma postać

$$D_Q = diag(0, -u, -u, -u), \tag{3.61}$$

wektory własne kolumnowe tworzą następującą macierz U_Q :

$$U_Q = \begin{bmatrix} 1 & -\varphi_C/\varphi_A & -\varphi_G/\varphi_A & -\varphi_T/\varphi_A \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \tag{3.62}$$

a wektory własne wierszowe tworzą następującą macierz V_Q^T :

$$V_Q^T = \begin{bmatrix} \varphi_A & \varphi_C & \varphi_G & \varphi_T \\ -\varphi_A & \varphi_{AGT} & -\varphi_G & -\varphi_T \\ -\varphi_A & -\varphi_C & \varphi_{ACT} & -\varphi_T \\ -\varphi_A & -\varphi_C & -\varphi_G & \varphi_{ACG} \end{bmatrix}. \tag{3.63}$$

W powyższych wzorach użyto notacji opisanej w (3.50).

Dla modelu HKY (3.53) macierz D_Q jest postaci

$$D_Q = diag[0, -v, -v(\varphi_A+\varphi_G)-u(\varphi_C+\varphi_T), -v(\varphi_C+\varphi_T)-u(\varphi_A+\varphi_G)], \tag{3.64}$$

wektory własne kolumnowe tworzą następującą macierz U_Q :

$$U_Q = \begin{bmatrix} 1 & -\dfrac{\varphi_C+\varphi_T}{\varphi_A+\varphi_G} & 0 & -\dfrac{\varphi_G}{\varphi_A} \\ 1 & 1 & -\dfrac{\varphi_T}{\varphi_C} & 0 \\ 1 & -\dfrac{\varphi_C+\varphi_T}{\varphi_A+\varphi_G} & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \tag{3.65}$$

a wektory własne wierszowe tworzą następującą macierz V_Q^T :

$$V_Q^T = \begin{bmatrix} \varphi_A & \varphi_C & \varphi_G & \varphi_T \\ -\varphi_A & \dfrac{\varphi_C(\varphi_A+\varphi_G)}{\varphi_C+\varphi_T} & -\varphi_G & \dfrac{\varphi_T(\varphi_A+\varphi_G)}{\varphi_C+\varphi_T} \\ 0 & -\dfrac{\varphi_C}{\varphi_C+\varphi_T} & 0 & \dfrac{\varphi_C}{\varphi_C+\varphi_T} \\ -\dfrac{\varphi_A}{\varphi_A+\varphi_G} & 0 & \dfrac{\varphi_A}{\varphi_A+\varphi_G} & 0 \end{bmatrix}. \tag{3.66}$$

Podstawiając powyższe wzory (3.58)-(3.66) do (3.57), można analitycznie wyznaczyć prawdopodobieństwa przejścia dla dowolnych par nukleotydów. Poniżej przedstawiono kilka przykładów takich obliczeń.

Dla modelu Jukesa-Cantora (3.48) prawdopodobieństwa przejść dane są wzorami

$$p_{ij}(t) = 0.25 - 0.25\exp(-\alpha t),\ i \neq j \tag{3.67}$$

oraz

$$p_{ii}(t) = 0.25 + 0.75\exp(-\alpha t), \tag{3.68}$$

gdzie $i, j \in \{A, C, G, T\}$.

Dla modelu Felsensteina (3.51) mamy

$$p_{ij}(t) = \varphi_j[1 - \exp(-ut)],\ i \neq j \tag{3.69}$$

oraz

$$p_{ii}(t) = \varphi_i + (1-\varphi_i)\exp(-ut), \tag{3.70}$$

gdzie, podobnie jak poprzednio $i, j \in \{A, C, G, T\}$.

Dla modelu HKY dla transwersji $A \to C$ obowiązuje wzór

$$p_{AC}(t) = \varphi_C[1 - \exp(-vt)], \tag{3.71}$$

dla tranzycji $G \to A$ mamy

$$\begin{aligned} p_{GA}(t) = \varphi_A &+ \frac{\varphi_A(\varphi_C+\varphi_T)}{\varphi_A+\varphi_G}\exp(-vt) \\ &- \frac{\varphi_A}{\varphi_A+\varphi_G}\exp[-v(\varphi_C+\varphi_T)t - u(\varphi_A+\varphi_G)t], \end{aligned} \tag{3.72}$$

a dla zachowania tej samej bazy $A \to A$ mamy

$$\begin{aligned} p_{AA}(t) = \varphi_A &+ \frac{\varphi_A(\varphi_C+\varphi_T)}{\varphi_A+\varphi_G}\exp(-vt) \\ &+ \frac{\varphi_G}{\varphi_A+\varphi_G}\exp[-v(\varphi_C+\varphi_T)t - u(\varphi_A+\varphi_G)t]. \end{aligned} \tag{3.73}$$

Dopasowanie modeli substytucji do danych

Uliniowienie pary sekwencji s_1 i s_2 prowadzi do listy (nieukierunkowanych) odpowiedniości pomiędzy nukleotydami lub aminokwasami, a także do listy odpowiedniości pomiędzy symbolami jednej sekwencji oraz przerwami w drugiej. Na podstawie uliniowienia konstruuje się modele substytucji powyżej opisywane. Na podstawie odpowiedniości pomiędzy nukleotydami ocenia się prawdopodobieństwa przejść i dalej parametry modeli (3.48), (3.51) lub (3.53).

Rozważmy problem dopasowania modeli substytucji nukleotydów. Bierzemy pod uwagę tylko odpowiedniości pomiędzy nukleotydami (na tym etapie nie uwzględnia się odpowiedniości nukleotyd - przerwa). Zakładamy, że dla każdego z nukleotydów wzdłuż polimeru DNA proces substytucji przebiega niezależnie od innych. Jeśli przez k_{ij} oznaczymy liczby zaobserwowanych odpowiedniości pomiędzy nukleotydami, $i, j \in \{A, C, G, T\}$, $i \leqslant j$ (relacja mniejszości oznacza tu porządek alfabetyczny) oraz przez q_{ij} oznaczymy prawdopodobieństwa wystąpienia takich odpowiedniości, to dla danych wynikających z uliniowienia można zapisać następującą logarytmiczną funkcję wiarygodności (z pominięciem elementu skalującego):

$$L = \sum_{i<j} k_{ij} \ln q_{ij} + \sum_{i} k_{ii} \ln q_{ii}. \tag{3.74}$$

Powyższa funkcja wiarygodności odpowiada rozkładowi wielomianowemu, w którym liczba wszystkich obserwacji wynosi

$$K = \sum_{i} k_{ii} + \sum_{i} \sum_{j<i} k_{ij}. \tag{3.75}$$

Na podstawie (3.74) oceny największej wiarygodności prawdopodobieństw q_{ij} są następujące:

$$\hat{q}_{ii} = \frac{k_{ii}}{K} = \frac{k_{ii}}{\sum_i k_{ii} + \sum_i \sum_{j<i} k_{ij}} \tag{3.76}$$

oraz

$$\hat{q}_{ij} = \frac{k_{ij}}{K} = \frac{k_{ij}}{\sum_i k_{ii} + \sum_i \sum_{j<i} k_{ij}}. \tag{3.77}$$

Zwróćmy uwagę, że na podstawie powyższych ocen $\hat{q}_{ii}$, $\hat{q}_{ij}$, $i, j \in \{A, C, G, T\}$, $i \leqslant j$ nie uda się jeszcze policzyć (ocenić) prawdopodobieństw p_{AA}, p_{AC}, p_{AG},...,p_{TT} , dlatego że, jak już wspomniano, prawdopodobieństwa p_{AA}, p_{AC}, p_{AG},...,p_{TT} odpowiadają (ukierunkowanym) substytucjom, a prawdopodobieństwa q_{ij} odpowiadają nieukierunkowanym odpowiedniościom. Dlatego wyprowadzenie wzorów dla ocen wymaga wykonania jeszcze kilku kroków rozumowania przedstawionych poniżej.

Ocena prawdopodobieństw rozkładu stacjonarnego

Aby wykorzystać dane dotyczące odpowiedniości pomiędzy nukleotydami, rozważmy najpierw problem oceny prawdopodobieństw rozkładu stacjonarnego

w modelach Markowa, tzn. składowych wektora π^{ST}

$$\pi^{ST} = \left[\pi_A \ \pi_C \ \pi_G \ \pi_T \right]. \tag{3.78}$$

Oceny prawdopodobieństw $\hat{\pi}_i$, $i \in \{A, C, G, T\}$ wyliczane są z wzoru:

$$\begin{aligned} \hat{\pi}_i &= \frac{\#\text{ wystąpienia nukleotydu } i \text{ w } s_1 \text{ oraz w } s_2}{2K} = \\ &= \frac{2k_{ii} + \sum_{j \neq i} k_{ij}}{2(\sum_i k_{ii} + \sum_i \sum_{j<i} k_{ij})}. \end{aligned} \tag{3.79}$$

Przeliczanie prawdopodobieństwa odpowiedniości na prawdopodobieństwo substytucji

Dla zilustrowania pierwszej idei potrzebnej dla skonstruowania estymatorów załóżmy, że zajmujemy się jednokrokowym łańcuchem Markowa z czasem dyskretnym. Problem, który jest analizowany, polega na wyliczeniu prawdopodobieństw substytucji p_{AA}, p_{AC}, p_{AG},...,p_{TT} przy założeniu, że znane są prawdopodobieństwa zaobserwowania odpowiedniości pomiędzy nukleotydami, q_{ij}, i, $j \in \{A, C, G, T\}$, $i \leqslant j$. Prawdopodobieństwo zaobserwowania odpowiedniości pomiędzy bazami $i - i$ wyraża się zatem wzorem:

$$q_{ii} = \pi_i p_{ii}, \tag{3.80}$$

a prawdopodobieństwo zaobserwowania odpowiedniości $i - j$, $i < j$ wzorem

$$q_{ij} = \pi_i p_{ij} + \pi_j p_{ji}. \tag{3.81}$$

Strategia oceny prawdopodobieństw p_{ij} mogłaby polegać na zastąpieniu w równaniach (3.80)-(3.81) prawdopodobieństw q_{ij} ich ocenami $\hat{q}_{ij}$ i rozwiązaniu układu równań ze względu na p_{ij}. Jednak jest to niemożliwe, dlatego, że z dwóch równań (3.80)- (3.81) nie da się wyliczyć trzech niewiadomych p_{ii}, p_{ij} oraz p_{ji}. Rozsądne założenie, które pozwala usunąć ten problem, to założenie o odwracalności procesu substytucji. Założenie o odwracalności sprowadza się do dodatkowego równania (równania lokalnego bilansu)

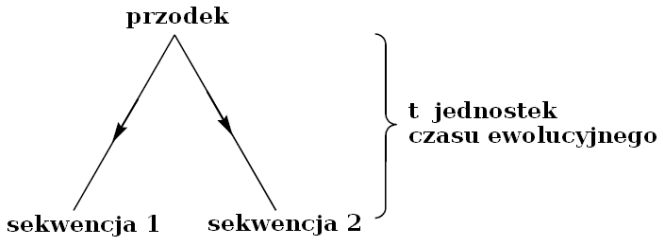
$$\pi_i p_{ij} = \pi_j p_{ji}. \tag{3.82}$$

Wykorzystując powyższą zależność jako trzecie, brakujące równanie, możemy obliczyć oceny w następujący sposób:

$$\hat{p}_{ii} = \frac{\hat{q}_{ii}}{\hat{\pi}_i} = \frac{2k_{ii}}{2k_{ii} + \sum_{j \neq i} k_{ij}} \tag{3.83}$$

oraz

$$\hat{p}_{ij} = \frac{\hat{q}_{ij}}{2\hat{\pi}_i} = \frac{k_{ij}}{2k_{ii} + \sum_{j \neq i} k_{ij}}, \ j \neq i. \tag{3.84}$$



Rysunek 3.31. Genealogia sekwencji

Założenie, że porównywane sekwencje (symbole w sekwencjach) są związane przez model jednokrokowego łańcucha Markowa, jest oczywiście nierealistyczne. Rozsądnym modelem związku pomiędzy symbolami dwóch sekwencji nukleotydów jest model przedstawiony na rysunku 3.31. Ponieważ liczba generacji, które wyznaczają związek pomiędzy obserwowanymi sekwencjami DNA, jest bardzo duża, posługiwać się będziemy procesem Markowa z czasem ciągłym, w którym macierz prawdopodobieństw przejść

$$P(t) = \exp(Qt) \tag{3.85}$$

jest zadana przez macierzową funkcję wykładniczą, gdzie wykładnikiem macierzowym jest macierz intensywności pomnożona przez czas ewolucyjny. Celem jest wyprowadzenie dla tego modelu wyrażeń analogicznych do (3.80)-(3.81). Prawdopodobieństwa q_{ij} zaobserwowania odpowiedniości pomiędzy symbolami i oraz j w sekwencjach nukleotydów s_1 oraz s_2 dla modelu przedstawionego na rysunku 3.31 wylicza się przez sumowanie po wszystkich możliwych stanach przodka obu sekwencji. Prowadzi to do następujących wzorów:

$$q_{ii} = \sum_{n\in\{A,C,G,T\}} \pi_n p_{ni}^2(t) \tag{3.86}$$

oraz

$$q_{ij} = \sum_{n\in\{A,C,G,T\}} 2\pi_n p_{ni}(t) p_{nj}(t),\ i < j. \tag{3.87}$$

Jeśli ponownie założyć odwracalność procesu substytucji, tzn. $\pi_n p_{ni}(t) = \pi_i p_{in}(t)$ oraz $\pi_n p_{nj}(t) = \pi_j p_{jn}(t)$, wówczas wykorzystując dodatkowo równanie Chapmana-Kołmogorowa, przekształcamy wzory (3.86)-(3.87) do postaci

$$q_{ii} = \pi_i p_{ii}(2t) \tag{3.88}$$

oraz

$$\frac{1}{2} q_{ij} = \pi_i p_{ij}(2t) = \pi_j p_{ji}(2t). \tag{3.89}$$

Podobnie jak poprzednio, na podstawie powyższych wzorów można wyliczyć oceny prawdopodobieństwa substytucji lub zachowania bazy

$$\hat{p}_{ii}(2t) = \frac{\hat{q}_{ii}}{\hat{\pi}_i} \tag{3.90}$$

oraz

$$\hat{p}_{ij}(2t) = \frac{\hat{q}_{ij}}{2\hat{\pi}_i}. \tag{3.91}$$

Powyżej wyprowadzone wzory można wykorzystać dla oceny parametrów parametrycznych modeli substytucji (3.48), (3.51) oraz (3.53).

Dla modelu Jukesa-Cantora, biorąc pod uwagę, że dla wszystkich i π_i, sumując obie strony równania (3.91) po wszystkich, i, j, $i < j$, i wreszcie wykorzystując (3.67), dostaje się następującą ocenę:

$$\widehat{\alpha t} = -\frac{1}{8} \ln \left(1 - \frac{4}{3}\hat{d}\right), \tag{3.92}$$

przy czym

$$\hat{d} = \frac{\sum_{j<i} k_{ij}}{K}. \tag{3.93}$$

Dla modelu Felsensteina, stosując podobną technikę, dostaniemy następujące oceny największej wiarygodności:

$$\hat{\varphi}_i = \hat{\pi}_i = \frac{2\sum_i k_{ii} + \sum_{j\neq i} k_{ij}}{2K}, \tag{3.94}$$

$$\widehat{ut} = -\frac{1}{2} \ln(1 - \hat{d}), \tag{3.95}$$

przy czym

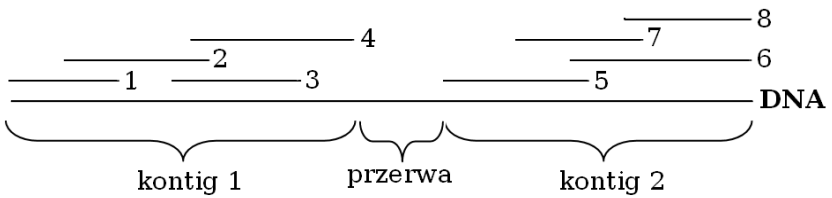
$$\hat{d} = \frac{\sum_{j<i} k_{ij}}{2\sum_{j<i} \hat{\varphi}_i \hat{\varphi}_j}. \tag{3.96}$$

Jak widać z powyższych wzorów, intensywność procesów substytucji można ocenić jedynie jako iloczyn parametrów α oraz u i czasu ewolucyjnego.

3.4 Algorytmy składania sekwencji genomowych

Odtworzenie sekwencji całych genomów bardzo wielu organizmów przyczyniło się do bardzo znacznego postępu w biologii molekularnej. Stało się ono możliwe dzięki wieloletniemu rozwojowi technik eksperymentalnych obejmujących między innymi następujące ważne techniki: 1. Zastosowanie enzymów restrykcyjnych do fragmentacji nici DNA. 2. Techniki elektroforezy pozwalające na separację fragmentów DNA o różnych długościach. 3. Reakcja łańcuchowa polimerazy, dzięki której można otrzymywać wiele kopii jednego fragmentu DNA. 4. Techniki klonowania DNA. 5. Sekwencjonowanie DNA metodą terminacji łańcucha DNA.

Powyżej opisane techniki eksperymentalne prowadzą do możliwości odczytania sekwencji z łańcucha DNA o długości maksymalnie do ok. 1000 nukleotydów. Natomiast długości odtwarzanych genomów organizmów są oczywiście dużo większe. Dlatego do odczytywania zawartości całych genomów stosuje się technikę nazywaną sekwencjonowaniem śrutowym (angielski termin shotgun sequencing). Polega ona na tym, że dokonuje się odczytów bardzo dużej liczby sekwencji. Każda z sekwencji nazywanych fragmentami albo odczytami położona jest w losowym miejscu w DNA. Następnie na podstawie zbioru odczytanych fragmentów odtwarza się sekwencję nukleotydów całego łańcucha DNA. Wykorzystuje się przy tym informację dotyczącą przekrywania się odczytywanych fragmentów. Tę strategię ilustruje rysunek 3.32, na którym przedstawiono przykładowy fragment łańcucha DNA oraz 8 odczytanych z tego łańcucha fragmentów. Na rysunku tym przedstawiono konfiguracje przekrywania się fragmentów występujące w rzeczywistych danych. Odcinki DNA, które są w całości pokryte przez odczytane fragmenty, nazywane są kontigami. Na rysunku 3.32 kontig 1 jest tworzony przez przekrywające się fragmenty 1, 2, 3 i 4, a kontig 2 przez fragmenty 5, 6, 7 i 8. Przy stosowaniu metody śrutowego sekwencjonowania zwykle nie uzyskuje się pokrycia całości długiego łańcucha DNA przez fragmenty. Zamiast tego otrzymuje się pewną liczbę kontigów, pomiędzy którymi występują przerwy. Na rysunku 3.32 są dwa kontigi i jedna przerwa pomiędzy nimi.



Rysunek 3.32. Sekwencjonowanie DNA metodą śrutową

Problem odtwarzania sekwencji DNA na podstawie odczytanych fragmentów nazywać będziemy problemem składania (asemblacji) DNA.

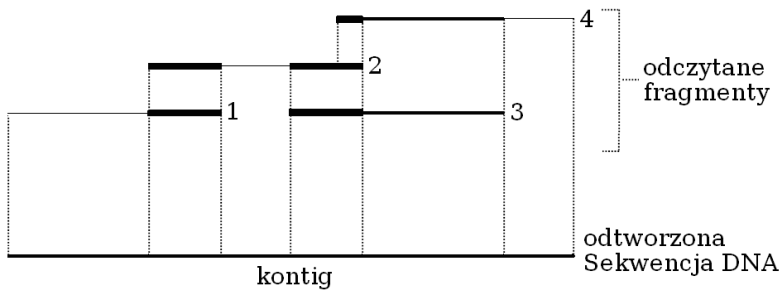
3.4.1 Algorytmy asemblacji sekwencji DNA

Istnieje wiele funkcjonujących aplikacji przeznaczonych do asemblacji (składania) całości sekwencji DNA na podstawie odczytów. Do narzędzi takich należą następujące programy i odpowiadające im serwery i usługi internetowe: Atlas Genome Assembly [26], Arachne [3], Celera Assembler [44], Jazz [14], Phusion [43], PCAP [28], Euler [47]. Algorytmy, na których bazują wymienione aplikacje, są złożone, mają w sobie szereg elementów heurystycznych, a także duże możliwości interakcji pomiędzy użytkownikiem (ekspertem) a funkcjonowaniem aplikacji.

Wszystkie algorytmy asemblacji sekwencji DNA bazują na detekcji przekrywania się pomiędzy fragmentami odczytanymi metodą śrutowego sekwencjonowania. Ponieważ liczba odczytywanych fragmentów jest bardzo duża, dla detekcji przekrywania się pomiędzy odczytami konieczne jest stosowanie odpowiednio efektywnych algorytmów. Często stosowaną metodą jest haszowanie odczytywanych fragmentów z wykorzystaniem krótkich (20-25 par baz) sekwencji nukleotydowych.

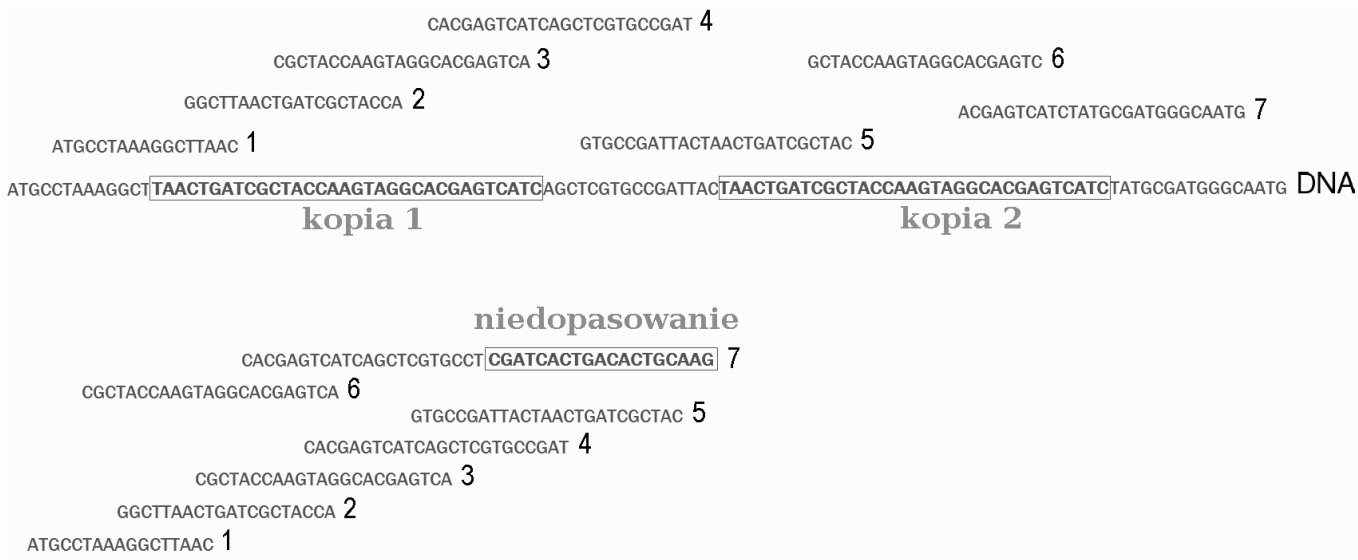
Asemblacja sekwencji na podstawie przekrywania się fragmentów

Na pierwszy rzut oka wydaje się, że zadanie asemblacji może być skutecznie rozwiązywane przez proste, heurystyczne algorytmy łączenia ze sobą przekrywających się odczytanych fragmentów DNA. Jeśli uda się uzyskać wielokrotne pokrycie sekwencji DNA przez fragmenty, to dzięki temu możliwe jest uzyskanie bardziej odpornych wersji algorytmów, z filtracją błędnych odczytów. Schemat tego typu algorytmu przedstawiony jest na rysunku 3.33, gdzie zobrazowano asemblację fragmentu DNA przez łączenie ze sobą przekrywających się odczytów DNA. Przekrycia pomiędzy odczytanymi fragmentami zaznaczone są na tym rysunku pogrubionymi liniami.



Rysunek 3.33. Asemblacja DNA metodą rozrostu kontigu

Jednak powyższe intuicyjne podejście do zadania asemblacji nie uwzględnia bardzo często występującej cechy sekwencji DNA, która może bardzo utrudniać jego asemblację, mianowicie istnienia powtarzających się fragmentów. Jak wiadomo, rzeczywiste sekwencje DNA zawierają bardzo dużą liczbę powtarzających się fragmentów [35], [48]. Jak łatwo zauważyć, istnienie powtarzających się fragmentów w DNA skutkuje występowaniem fałszywych przekryć pomiędzy odczytami. Jest to zilustrowane na rys. 3.34. W górnej części rysunku przedstawiono fragment DNA, w którym występują dwie kopie takiej samej podsekwencji. W górnej części rysunku przedstawiono też 7 fragmentów sekwencji odczytanych z łańcucha DNA. Jeśli do tych siedmiu fragmentów zastosuje się algorytm rekonstrukcji (asemblacji) na podstawie przekryć, przedstawiony na rys. 3.33, to jak widać z dolnej części rys. 3.34, otrzyma się błędnie zrekonstruowany łańcuch DNA, w którym obie powtarzające się kopie zostają



Rysunek 3.34. Błędy asemblacji DNA metodą rozrostu kontigu wynikające z powtórzeń w DNA

sprowadzone tylko do jednej kopii. Co więcej w trakcie wykonywania algorytmu asemblacji występuje brak możliwości dopasowania sekwencji numer 7 do już uzyskanego łańcucha otrzymanego przez asemblację sekwencji 1-6.

Przykład powyższy pokazuje, że do asemblacji łańcuchów DNA konieczne jest stosowanie algorytmów o bardziej złożonej postaci niż algorytm sklejania przekrywających się sekwencji przedstawiony na rysunku 3.33.

Zadanie asemblecji jako problem najkrótszego superciągu

Jedną z możliwości algorytmicznego sformułowania zadania asemblacji łańcucha DNA na podstawie zbioru uzyskanych fragmentów jest problem poszukiwania najkrótszego superciągu. Jest to jeden z dobrze znanych problemów optymalizacji kombinatorycznej, polegający na znalezieniu najkrótszej sekwencji symboli alfabetu s, która zawiera w sobie każdą z zadanych sekwencji s_1, $s_2, \ldots, s_n$. Problem poszukiwania najkrótszego superciągu [24], [33], [34], [52] jest probemem NP zupełnym. Jednak znanych jest szereg algorytmów przybliżonego rozwiązywania tego problemu [55], [57], które pozwalają na uzyskanie rozwiązania w czasie wielomianowym.

Jeśli zapiszemy problem najkrótszego superciągu dla zbioru odczytów z rysunku 3.34, tzn. następujących sekwencji:

$$\begin{aligned}
s_1 &: TATGCCTAAAGGCTTAAC \\
s_2 &: GGCTTAACTGATCGCTACCA \\
s_3 &: CGCTACCAAGTAGGCACGAGTCA \\
s_4 &: CACGAGTCATCAGCTCGTGCCGAT \\
s_5 &: GTGCCGATTACTAACTGATCGCTAC \\
s_6 &: GCTACCAAGTAGGCACGAGTC \\
s_7 &: ACGAGTCATCTATGCGATGGGCAATG
\end{aligned} \tag{3.97}$$

to rozwiązaniem problemu najkrótszego superciągu dla tych sekwencji będzie następująca sekwencja

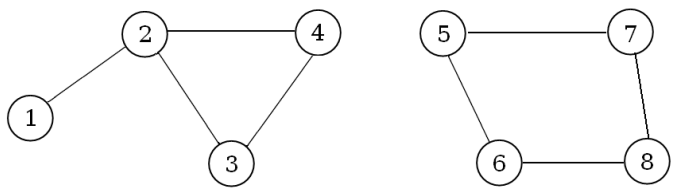
$$\begin{aligned}
s = {} & TATGCCTAAAGGCTTAACTGATCGCTACCAAGTAGGCA \\
& CGAGTCATCAGCTCGTGCCGATTACTAACTGATCGCTA \\
& CCAAGTAGGCACGAGTCATCTATGCGATGGGCAATG,
\end{aligned}$$

która, jak widać, stanowi prawidłowo odtworzoną sekwencję DNA z rys. 3.34.

Zadanie asemblacji jako problem poszukiwania ścieżki Hamiltona w grafie

Wygodnym i obrazowym sposobem przedstawienia struktury przekrywania się odczytanych fragmentów DNA jest wykorzystane grafu przekryć. W grafie tym węzły odpowiadają odczytanym fragmentom DNA, a gałęzie strukturze

przekryć pomiędzy węzłami. Inaczej mówiąc, pomiędzy dwoma węzłami występuje gałąź, jeśli odpowiadające im odczytane fragmenty się+ przekrywają. Przykład grafu przekryć przedstawiony jest na rysunku 3.35. Przedstawiony na tym rysunku graf przekryć odpowiada konfiguracji przekrywania się fragmentów z rysunku 3.32. Na rysunku 3.32 występują dwa rozłączne kontigi. Odbiciem tego faktu w grafie przedstawionym na rysunku 3.32 jest występowanie dwóch rozłącznych podgrafów grafu przekryć. Z kolei na rysunku 3.36 przedstawiony jest graf przekryć odpowiadający pokryciu sekwencji DNA fragmentami, przedstawionemu na górnej części rysunku 3.34. Graf ten jest grafem ważonym, to znaczy każdej gałęzi odpowiada waga zdefiniowana przez długość przekrycia pomiędzy fragmentami połączonymi przez tę gałąź. Ponadto, graf ten jest grafem skierowanym. Kierunek strzałki gałęzi łączącej dwa węzły odpowiadające fragmentom może być ustalony przez odpowiednią analizę sekwencji dwóch fragmentów i ich przekrycia.

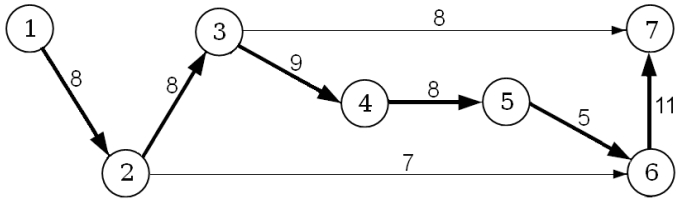


Rysunek 3.35. Graf przekryć

Problem poszukiwania ścieżki Hamiltona w grafie polega na przejściu przez wszystkie węzły grafu, przez każdy dokładnie jeden raz. Problem poszukiwania ścieżki Hamiltona jest znów problemem NP zupełnym [47], [49], [52], [62], trudnym do rozwiązania w dokładny sposób. Jednak podobnie jak poprzednio, istnieją dla tego problemu algorytmy przybliżone pozwalające na znalezienie suboptymalnego rozwiązania w czasie wielomianowym [1], [56]. Dla grafu przedstawionego na rysunku 3.36 istnieje dokładnie jedna ścieżka Hamiltona $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5 \rightarrow s_6 \rightarrow s_7$. Jest ona zaznaczona na rysunku przez pogrubione strzałki. Odtworzenie struktury przekrywania się fragmentów w kolejności wyznaczonej przez tę ścieżkę Hamiltona prowadzi do poprawnej asemblacji sekwencji łańcucha DNA. Dla poprawnego odtworzenia łańcucha DNA nie jest w tym przypadku potrzebna informacja o wagach w grafie przekrycia. Jednak w skomplikowanych przypadkach, bardziej zbliżonych do rzeczywistych problemów informacja o wagach może być przydatna do konstrukcji algorytmów bardziej odpornych na błędy w odczytach.

Asemblacja (sekwencjonowanie) przez hybrydyzację

Hybrydyzacja jest to nazwa całej rodziny metod stosowanych w laboratoriach biologii molekularnej, polegających na powiązaniu technik znaczenia molekuł (np. przez radioaktywne izotopy lub barwniki fluorescencyjne) z technikami

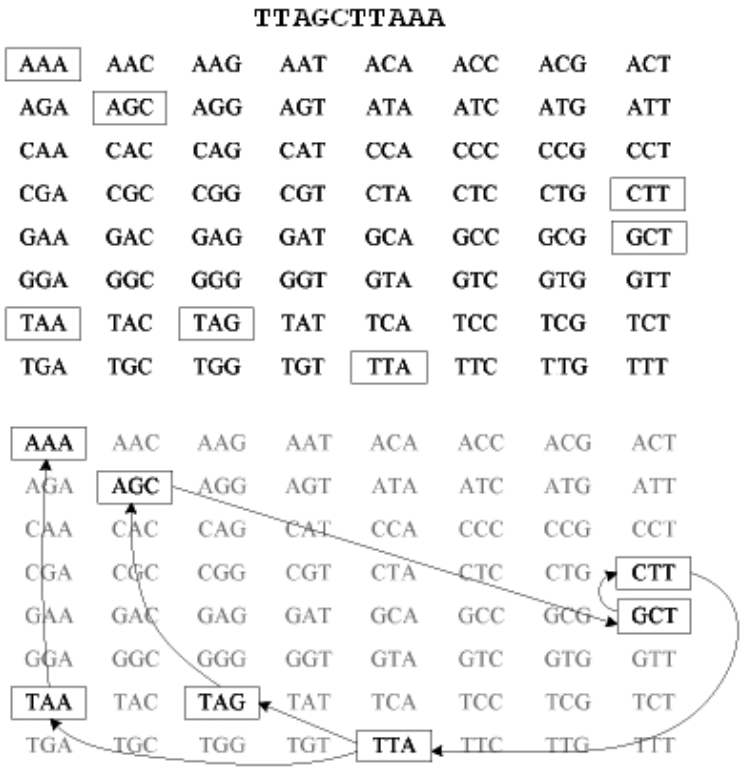


Rysunek 3.36. Asemblacja DNA jako problem poszukiwania ścieżki Hamiltona

eksperymentalnymi, w których wykorzystuje się różnego rodzaju powinowactwo pomiędzy molekułami, na przykład powinowactwo pomiędzy komplementarnymi sekwencjami DNA. Idea asemblacji poprzez hybrydyzację, [2], [29] polega na skonstruowaniu zbioru sond w postaci sekwencji fragmentów DNA, a następnie zbadaniu ich powinowactwa do analizowanego polimeru DNA. Taki zbiór sond nazywa się biblioteką hybrydyzacyjną. Obecnie biblioteki hybrydyzacyjne wykonuje się, wykorzystując technologie mikromacierzowe. Idea asemblacji przez hybrydyzację pokazana jest na rysunku 3.37. Na rysunku tym w górnej części przedstawiono matrycę sond zbudowaną ze wszystkich sekwencji trójnukleotydowych. Badamy powinowactwo sond tej tablicy z fragmentem DNA komplementarnym do $TTAGCTTAAA$.

Sondy, które wykazują powinowactwo do fragmentu DNA (ich sekwencje należą do sekwencji tego fragmentu), zaznaczone są na rysunku 3.37 przez obramowanie. Struktura przekrywania się sond jest przedstawiona w dolnej części rysunku 3.37. Struktura ta zadana jest w postaci grafu przekryć. Odtworzenie oryginalnej sekwencji molekuły DNA uzyskuje się przez rozwiązania zadania poszukiwania ścieżki Eulera, to znaczy ścieżki przechodzącej przez wszystkie gałęzie grafu, przez każdą dokładnie jeden raz. Dla grafu przedstawionego na rysunku 3.37 istnieje dokładnie jedna ścieżka Eulera, która prowadzi do wyznaczenia oryginalnej sekwencji badanej molekuły DNA.

Podstawowym problemem w technice asemblacji przez hybrydyzację jest wykładniczy wzrost rozmiarów bibliotek hybrydyzacyjnych, gdy rośnie długość ich sond. Problem ten powoduje, że w chwili obecnej technik tych nie stosuje się bezpośrednio do asemblacji DNA. Jednak istnieje szereg modyfikacji metod sekwencjonowania przez hybrydyzację, których rozwój pozwala na zastosowanie tej techniki do zagadnień związanych z badaniem sekwencji DNA. Jedną z tych modyfikacji jest izotermiczne sekwencjonowanie przez hybrydyzację. W metodzie tej nie buduje się całych bibliotek hybrydyzacyjnych, lecz tylko ich mniejsze wersje zbudowane z sond o odpowiednich własnościach dotyczących temperatury topnienia. Temperatura topnienia jest parametrem, który zależy od proporcji pomiędzy nukleotydami CG i AT. Wiele problemów związanych z konstrukcją algorytmów sekwencjonowania przez hybrydyzację oraz izotermicznego sekwencjonowania przez hybrydyzację zostało przedyskutowanych w pracach [6], [7], [8], [9], [10].



Rysunek 3.37. Metoda asemblacji (sekwencjonowania) przez hybrydyzację

3.4.2 Modele statystyczne procesu odczytywania fragmentów

Będziemy stosować następujące przybliżenie rzeczywistego procesu odczytywania krótkich fragmentów z długiego łańcucha DNA w procesie śrutowego sekwencjonowania. Zakładamy, że długość całego łańcucha DNA wynosi G baz. W procesie śrutowego sekwencjonowania odczytywanych jest N fragmentów, wszystkie o jednakowej długości L. Iloraz

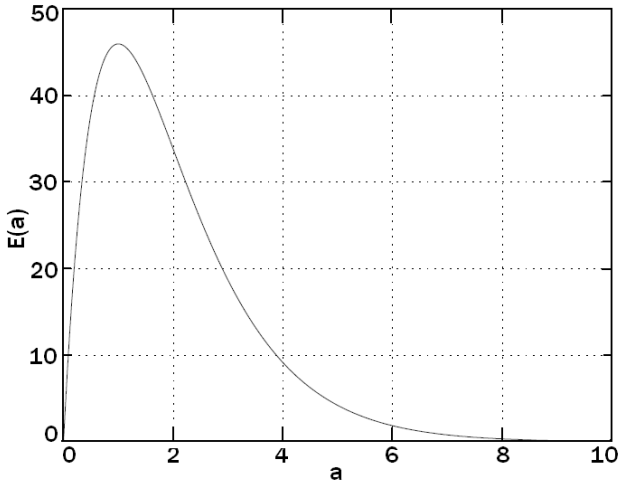
$$a = \frac{NL}{G} \tag{3.98}$$

nazywa się pokryciem w procesie śrutowego sekwencjonowania.

Prosty model, który pozwala powiązać liczbę kontigów w pokryciu z parametrem a, ma następującą postać. Rozważmy przedział w sekwencji DNA zdefiniowany przez fragment:

$$F = (x, x - L),$$

gdzie x jest losowo dobranym końcem fragmentu DNA. Zauważmy, że warunek na to, aby fragment F był prawym końcem kontigu jest aby żaden z prawych



Rysunek 3.38. Wykres zależności oczekiwanej liczbuy kontigów od paramteru pokrycia a

końców pozostałych fragmentów pokrycia nie należał do F. Prawdopodobieństwo "trafienia" w F przez prawy koniec losowo położonego fragmentu wynosi L/G. Zatem rozkład liczby trafień k jest rozkładem dwumianowym. Ponieważ długość L fragmentu F jest mała w porównaniu z długością G całego łańcucha DNA, zatem ten rozkład dwumianowy można bardzo dokładnie przybliżyć rozkładem Poissona z parametrem

$$\lambda = a = \frac{NL}{G}.$$

Prawdopodobieństwo, że konkretny fragment jest prawym końcem kontigu, wynosi

$$P[k = 0] = \exp(-\lambda) = \exp(-\frac{NL}{G}).$$

Wykorzystując powyższy wzór można wyliczyć oczekiwaną liczbę kontigów w pokryciu o parametrze a. Wynosi ona

$$\begin{aligned} E[\text{Liczba kontigów}] &= N \times \Pr[\text{zadany fragment jest końcem kontigu}] \\ &= N \times \exp(-NL/G) = (aG/L) \times \exp(-a). \end{aligned} \quad (3.99)$$

Wykres zależności oczekiwanej liczby kontigów od wartości parametru a przedstawiony jest na rysunku 3.38. Na postać wykresu oprócz wartości a wpływa także wartość ilorazu GŁ. Na rysunku 3.38 przyjęto $G = 100000$, $L = 800$. Przy takich założeniach wartością a, dla której uzyskuje się oczekiwaną liczbę kontigów równą jeden, jest $a = 8$. Jest to wartość dość często przyjmowana jako zalecane pokrycie w procesie śrutowego sekwencjonowania.

Interesującym i ważnym parametrem jest średnia liczba fragmentów w kontigu, a także średnia długość kontigu. Jeśli wszystkie lewe końce odczytów uporządkujemy według porządku narastania, to tworzenie się kontigu możemy uważać za ciąg prób Bernouliego kontynuowany tak długo aż lewy koniec kolejnego odczytu znajdzie się w odległości większej niż L od lewego końca poprzedniego odczytu. Zgodnie z tym spostrzeżeniem rozkład liczby fragmentów w kontigu jest rozkładem geometrycznym o prawdopodobieństwie $p = \exp(-a)$. Z kolei biorąc pod uwagę, że odległość pomiędzy lewymi końcami kolejnych odczytów jest zmienną losową o rozkładzie wykładniczym, możemy wyprowadzić następujący wzór na wartość oczekiwaną długości kontigu, $E(S)$:

$$E(S) = \frac{1 - \exp(-a)}{\exp(-a)} \int_0^L x\lambda \frac{\exp(-\lambda x)}{1 - \exp(-\lambda L)} dx + L = L\frac{\exp(a) - 1}{a}.$$

Literatura

1. Angluin D., Valiant L. G. (1979), Fast probabilistic algorithm for Hamiltonian circuits and matchings, J. Comput. System Sci., vol. 18, pp. 155–193.
2. Bains W., Smith G. C. (1988), A novel method for DNA sequence determination, J. Theor. Biol., vol. 135, pp. 303–307.
3. Batzoglou S., Jaffe D. B., Stanley K., Butler J., Gnerre S., Mauceli E., Berger B., Mesirov J. P., Lander E. S. (2002), ARACHNE: A whole-genome shotgun assembler. Genome Res., vol. 12, pp. 177–189.
4. Bellman R. E., (1970), *Introduction to Matrix Analysis*, McGraw-Hill.
5. Bellman R. E., Dreyfus S. E. (1962), *Applied Dynamic Programming*, Princeton University Press.
6. Blazewicz J., Kasprzak M., (2003), Complexity of DNA sequencing by hybridization, Theoretical Computer Science, Vol. 290, pp. 1459-1473.
7. Blazewicz J., Formanowicz P., Kasprzak M., Markiewicz W., Weglarz J., (2000), Tabu search for DNA sequencing with false negatives and false positives, European Journal of Operational Research, Vol. 125, pp. 257-265.
8. Blazewicz J., Formanowicza P., Kasprzak M., Markiewicz W., Swiercz A.,(2004), Tabu search algorithm for DNA sequencing by hybridization with isothermic libraries, Computational Biology and Chemistry, Vol. 28, pp. 11-19.
9. Blazewicz J., Formanowicza P., Kasprzak M., Markiewicz W., (2004), Sequencing by hybridization with isothermic oligonucleotide libraries, Discrete Applied Mathematics, Vol. 145, pp. 40-51.
10. Blazewicz J., Glover F., Kasprzak M., Markiewicz W., OĞuz C., Rebholz-Schuhmanne D., Swiercz A., (2006), Dealing with repetitions in sequencing by hybridization, Computational Biology and Chemistry, Vol. 30, pp. 313-320.
11. Boyer R. S., Moore J. S. (1977), A fast string searching algorithm, Commun. ACM, vol. 20, pp. 762–772.
12. Burrows M., Wheeler D. J. (1994), A block sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, Palo Alto, CA.
13. Crick F. (1970), Central dogma of molecular biology, Nature, vol. 227, pp. 561–563.
14. Dehal P., Satou Y., Campbell R. K., Chapman J., Degnan B., De Tomaso A., Davidson B., Di Gregorio A., Gelpke M., Goodstein D.M., et al. (2002), The

draft genome of *Ciona intestinalis*: Insights into chordate and vertebrate origins. Science, vol. 298, pp. 2157–2167.
15. Dreyfus S. E., Law A. M. (1977), *The Art and Theory of Dynamic Programming*, Academic Press.
16. Dumas J. P., Ninio J. (1981), Efficient algorithm for folding and comparing nucleic acid sequences, Nucleic Acids Res., vol. 10, no. 1., pp. 197–206.
17. Ewens W. J., Grant G. R., (2001), *Statistical Methods in Bioinformatics*, Springer.
18. Felsenstein J. (1992), Estimating effective population size from samples of sequences, inefficiency of pairwise and segregating sites as compared to phylogenetic estimates. Genet. Res., vol. 59, pp. 139–147.
19. Felsenstein J. (1981), Evolutionary trees from DNA sequences: a maximum likelihood approach, J. Mol. Evol., vol. 17, pp. 368–376.
20. Felsenstein J. (1985), Confidence limits on phylogenies: An approach using the bootstrap. Evolution vol. 39, pp. 783–791.
21. Felsenstein J., (2004), *Inferring Phylogenies*, Sinauer Associates.
22. Fitch W. (1971), Toward defining the course of evolution: minimum change for a specified tree topology. Syst. Zool., vol. 20. pp. 406–416.
23. Gesteland R. F., Cech T. R., Atkins J. F. (eds.), (1993), *The RNA World : the Nature of Modern RNA Suggests a Prebiotic RNA*, Cold Spring Harbor Laboratory Press.
24. Gusfield D. (1997), *Algorithms on strings, trees and sequences*, Cambridge University Press.
25. Hasegava M., Kishino M., Yano T. (1985), Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. J. Mol. Evol., vol. 22, pp. 160-174.
26. Havlak P., Chen R., Durbin K. J., Egan A., Ren Y., Song X-Z. Weinstock M., Gibbs R. A. (2004), Tha Atlas genome assembly system, Genome Res., vol. 14, pp. 721–732.
27. Healy J., Thomas E. E., Schwartz J. T., Wigler M. (2003), Annotating large genomes with exact word matches. Genome Res., vol. 13, pp. 2306–2315.
28. Huang X., Wang J., Aluru S., Yang S. P. Hillier L. (2003), PCAP: A whole-genome assembly program. Genome Res., vol. 13, pp. 2164–2170.
29. Idury R. M., Waterman M. S. (1995), A new algorithm for DNA Sequence Assembly, J. Comput. Biol., vol. 2, pp. 291–306.
30. Jukes T. H., Cantor C. R. (1969), Evolution of protein molecules, in H. N. Munro (ed.) *Mammalian Protein Metabolism*, Academic Press, New York, pp. 21–132.
31. Karlin S., Altschul S. F. (1990), Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. USA, vol. 87, pp. 2264–2268.
32. Karlin S., Altschul S. F. (1993), Applications and statistics for multiple high-scoring segments in molecular sequences. Proc. Natl. Acad. Sci. USA, vol. 90, pp. 5873–5877.
33. Kececioglu J.D., Myers E.W. (1995), Exact and approximate algorithms for the sequence reconstruction problem. Algorithmica, vol. 13, p. 7.
34. Kececioglu J.D., Myers E.W. (1995), Combinatorial algorithms for DNA sequence assembly, Algorithmica, vol. 13 p. 51.
35. Kimmel M., Axelrod D. E. (2002), *Branching processes in biology*, Springer, New York.

36. Kimura M. (1980), A simple method for estimating evolutionary rate in a finite population due to mutational production of neutral and nearly neutral base substitution through comparative studies of nucleotide sequences. J. Molec. Biol., vol. 16, pp. 1501–1531.
37. Knuth D. E. (1973), *The Art of Computer Programming*, Addison-Weseley.
38. Knuth D. E., Morris J. H., Pratt V. R. (1977), Fast pattern matching in strings. SIAM J. Comput., vol. 6, no. 2, pp. 323–350.
39. Li W. H. (1997), *Molecular Evolution*, Sinauer Associates.
40. Lipman D. J., Pearson W. R. (1985), Rapid and sensitive protein similarity searches. Science, vol. 227, pp. 1435–1441.
41. Michener C. D., Sokal, R. R. (1957), A quantitative approach to a problem in classification. Evolution, vol. 11, pp. 130–162.
42. Morrison D. R. (1965), PATRICIA–practical algorithm to retrieve information coded in alphanumeric, J. ACM, vol. 15, pp. 514–534.
43. Mullikin J. C., Ning Z. (2003), The Phusion assembler, Genome Res., vol. 13, pp. 81–90.
44. Myers E. W., Sutton G. G., Delcher A. L., Dew I. M., Fasulo D. P., Flanigan M. J., Kravitz S. A., Mobarry, C.M., Reinert, K.H., Remington K. A., et al. (2000), A whole-genome assembly of Drosophila. Science vol. 287, pp. 2196–2204.
45. Needleman S. B., Wunsch C. D. (1970), A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol., vol. 48, pp. 443–453.
46. Nirenberg M. W., Matthaei J. H. (1961), The dependence of cell-free protein synthesis in *E. coli* upon naturally occurring or synthetic polyribosomes. Proc. Nat. Acad. Sci. USA, vol. 47, pp. 1588–1602.
47. Pevzner P., Tang H., Waterman M. (2001), An Eulerian path approach to DNA fragment assembly. Proc. Nat. Acad. Sci. USA, vol. 98, pp. 9748–9753.
48. Primrose S.B. (1998), *Principles of Genome Analysis*, Blackwell Science.
49. Rosen K. (2002), *Discrete Mathematics and its Applications*, McGraw-Hill.
50. Saitou N, Nei M. (1987), The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol., vol. 4, pp. 406–225.
51. Schulz G. E., Schrimer R. H. (1990), *Principles of Protein Structure*, Springer.
52. Skiena S. S. (1998), *The Algorithm Design Manual*, Springer.
53. Smith T. F., Waterman M. S. (1981), Identification of common molecular subsequences. J. Mol. Biol., vol. 197, pp. 147–195.
54. Stephen G. A., (1994), *String Searching Algorithms*, World Scientific.
55. Teng S.H., Yao F. F. (1997), Approximating shortest superstrings. SIAM J. Comput., vol. 26, pp. 410–417.
56. Thomason A. (1989), A simple linear expected time for finding a Hamiltonian Path, Discrete Math., vol. 75, pp. 373-379.
57. Turner J. (1989), Approximation algorithms for the shortest common superstring problem. Inf. Comput., vol. 83, p. 20.
58. Vlassov A. V., Kazakov S. A., Johnston B. H., Landweber L. F. (2005), The RNA world on ice: A new scenario for the emergence of RNA information. J. Mol. Evol., vol. 61, pp. 264–273.
59. Waterman M.S., (1995), Introduction to computational biology. Chapman and Hall/CRC, Boca Raton, FA.
60. Watson J., Crick F. (1953), A structure for deoxyribose nucleic acid, Nature, vol. 171, pp. 737–738.

61. Wilbur W. J., Lipman D. J. (1983), Rapid similarity searches of nucleic acid and protein data banks. Proc. Natl. Acad. Sci. USA, vol. 80, pp. 726–730.
62. Wilf H. S. (2002), *Algorithms and Complexity*, A. K. Peters, http://www.cis-.upenn.edu/~wilf/
63. 3DNA. A software package for the analysis, rebuilding, and visualization of three-dimensional nucleic acid structures, http://rutchem.rutgers-.edu/~xiangjun/3DNA/
64. ExPASy, Expert protein analysis system, proteomics server, http://us-.expasy.org/
65. PDB, protein data bank, http://www.rcsb.org/pdb/
66. Ras Mol, molecular visualization freeware, http://www.umass.edu/microbio/-rasmol/
67. Swiss-Prot, protein knowledgebase, http://us.expasy.org/sprot/śprot-top.html

Indeks

ポーランド日本情報工科大学

POLSKO-JAPOŃSKA WYŻSZA SZKOŁA TECHNIK KOMPUTEROWYCH

Jedna z najlepszych uczelni w Polsce – wyróżniana przez pracodawców, studentów i media.

Akredytacja Państwowej Komisji Akredytacyjnej

Informatyka

Studia inżynierskie, magisterskie uzupełniające, podyplomowe, studia doktoranckie, uprawnienia habilitacyjne, studia przez internet.

Specjalizacje:

animacja 3D, bazy danych, eksploracja www, inteligentne systemy przetwarzania danych, inżynieria oprogramowania i baz danych, multimedia, programowanie aplikacji biznesowych, programowanie gier, programowanie systemowe i sieciowe, robotyka, sieci urządzeń mobilnych, systemy rozproszone i równoległe.

Architektura Wnętrz

Studia licencjackie

Kultura Japonii

Studia licencjackie

Sztuka Nowych Mediów (Grafika)

Studia licencjackie, magisterskie uzupełniające

Zarządzanie Informacją

Studia inżynierskie

Kursy:

Akademia Sieciowa CISCO; LPI Linux; Microsoft

Akademickie Liceum Ogólnokształcące przy PJWSTK

02-008 Warszawa, ul. Koszykowa 86
tel.: 22 58 44 500, fax: 22 58 44 501
e-mail: pjwstk@pjwstk.edu.pl
www.pjwstk.edu.pl

PJWSTK w Bytomiu
Informatyka
Sztuka Nowych Mediów, Grafika
Studia inżynierskie, licencjackie
41-902 Bytom, Aleja Legionów 2
tel.: 32 387 16 60
e-mail: bytom@pjwstk.edu.pl
www.bytom.pjwstk.edu.pl

PJWSTK w Gdańsku
Informatyka
Studia inżynierskie
80-045 Gdańsk, ul. Brzegi 55
tel. 58 683 59 75
e-mail: gdansk@pjwstk.edu.pl
www.gdansk.pjwstk.edu.pl

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt **"Nowoczesna kadra dla e-gospodarki"** – program rozwoju Wydziału Zamiejscowego Informatyki w Bytomiu Polsko-Japońskiej Wyższej Szkoły Technik Komputerowych, współfinansowany przez Unię Europejską ze środków Europejskiego Funduszu Społecznego w ramach Podziałania 4.1.1. "Wzmocnienie potencjału dydaktycznego uczelni" Programu Operacyjnego Kapitał Ludzki

ISBN 978-83-89244-91-8

Egzemplarz bezpłatny