VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

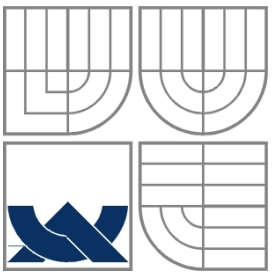
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TECHNIKY PRO ZÍSKÁVÁNÍ DAT V GENOMICE

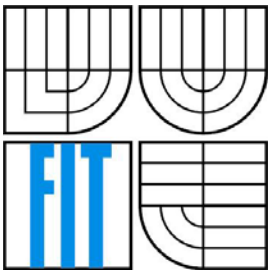
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE BC. Petr Jaša
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

TECHNIKY PRO ZÍSKÁVÁNÍ DAT V GENOMICE

TECHNICS FOR DATAMINING IN GENOMICS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Petr Jaša
AUTHOR

VEDOUCÍ PRÁCE — Ing. Ivana Rudolfová
SUPERVISOR

BRNO 2007

Zadání diplomové práce/3993/2006/xjasap02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2006/2007

Zadání diplomové práce

Řešitel: **Jaša Petr, Bc.**
Obor: Informační systémy
Téma: **Techniky pro získávání dat v genomice**
Kategorie: Databáze

Pokyny:

1. Seznamte se s oblastmi genomiky, ve kterých se využívají algoritmy pro získávání dat.
2. Seznamte se s technikami, které se používají pro získávání dat v genomice.
3. Některou z metod implementujte ve vhodném programovacím jazyce a ověřte její funkčnost na vhodném vzorku dat. Pod vedením vedoucího DP navrhněte vlastní metodu založenou na již existujících technikách.
4. Zhodnoťte dosažené výsledky, porovnejte obě metody a diskutujte další možné pokračování práce.

Literatura:

- J. T. L. Wang, M. J. Zaki, H. T. T. Toivonen, D. Shasha: Data Mining in Bioinformatics, Springer-Verlag, 2005

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním paměťovém médiu (disketa, CD-ROM), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rudolfová Ivana, Ing.**, UIFS FIT VUT
Datum zadání: 28. února 2006
Datum odevzdání: 22. května 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Jaroslav Zendulka, CSc.
vedoucí ústavu

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami

1. **Pan**

Jméno a příjmení: **Bc. Petr Jaša**
Id studenta: 47540
Bytem: Kamenice 404, 588 23 Kamenice u Jihlavy
Narozen: 15. 05. 1983, Jihlava
(dále jen "autor")

a

2. **Vysoké učení technické v Brně**

Fakulta informačních technologií
se sídlem Božetěchova 2/1, 612 66 Brno, IČO 00216305
jejímž jménem jedná na základě písemného pověření děkanem fakulty:

.....
(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
diplomová práce

Název VŠKP: Techniky pro získávání dat v genomice
Vedoucí/školitel VŠKP: Rudolfová Ivana, Ing.
Ústav: Ústav informačních systémů
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:
tištěné formě počet exemplářů: 1
elektronické formě počet exemplářů: 2 (1 ve skladu dokumentů, 1 na CD)

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizování výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti:
   - ☐ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor

Abstrakt

Tato práce si v první řadě klade za cíl představit některé běžné techniky pro získávání dat v genomice a v další řadě naimplementovat vlastní algoritmus vycházející z originální verze algoritmu BLAST jako zástupce zmíněných technik. Práce se konkrétně zaměřuje na DNA sekvence, které v sobě uchovávají genetickou informaci, jež je předlohou živých organismů. Pro rozluštění této informace se používá řada technik. Tento text popisuje algoritmus Fasta a algoritmy rodiny BLAST. Pomocí těchto algoritmů je možné získat o sekvencích DNA, u kterých je známá pouze jejich primární struktura, mnoho důležitých informací. Princip algoritmů spočívá v zarovnání jedné vzorové sekvence DNA, ze které chceme získat informace, s mnoha sekvencemi uloženými v databázi. Ze zarovnání je pak možné u vzorové sekvence, na základě míry její podobnosti se sekvencemi databáze, stanovit s určitou pravděpodobností mnoho různých vlastností.

Klíčová slova

BLAST, Fasta, DNA, RNA, sekvence DNA, nukleotid, báze, adenosin, cytosin, guanin, tymin, dynamické programování, porovnání, zarovnání, vyhledávání

Abstract

First of all, this thesis sets itself a goal to introduce some common technics for datamining in genomics and as a next step to implement own algorithm like algorithm BLAST. In the concrete, this work is pointed to sequences of DNA. The DNA sequence contains in itself genetic information, which is template for living organism. For explanation this information can be used number of technics. This paper describes algorithm Fasta and algorithms from BLAST family. With these algorithms, it is possible to gain a lot of important information even about such DNA sequences, where only primary structure is known. Principle of these algorithms is based on alignments of one query sequence, which we want to obtain some information from, with many sequences stored in database. According to result alignment, it is possible to determine many features of the query sequence.

Keywords

BLAST, Fasta, DNA, RNA, sequence of DNA, nucleotid, base, adenosine, cytosine, guanine, thymine, dynamic programming, comparison, alignment, searching

Citace

Jaša Petr: Techniky pro získávání dat v genomice. Brno, 2007, diplomová práce, FIT VUT v Brně.

Techniky pro získávání dat v genomice

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Ivany Rudolfové. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Petr Jaša
22. 5. 2007

Poděkování

Tato semestrální práce není výsledkem pouze mé práce a mého snažení. Chtěl bych poděkovat Ing. Ivaně Rudolfové za její podporu a trpělivost.

Obsah

1 Úvod

Současný rozvoj a přístupnost počítačových systémů umožnil, aby se tyto systémy začlenily do řady odvětví a oborů lidské činnosti. Jedním z takových oborů je i obor biologie. Zde počítačové technologie s rozvojem vědy a techniky nachází stále širší uplatnění. Tato práce se zaměřuje právě na spojení informatiky a biologie, které se stalo tak silné, že dalo vznik novému vědnímu oboru bioinformatice. Bioinformatika je disciplína, která se svým zaměřením a záběrem pohybuje na rozhraní počítačových věd, informačních technologií, matematiky a biologie. Jednou z oblastí biologie, kterou se bioinformatika zabývá, je obor genomika. Zde se počítačové technologie využívají zejména pro získávání a analýzu genomických dat (sekvence nukleových kyselin, sekvence proteinů apod.), pro jejich ukládání a praktické uchovávání nebo pro jejich grafické zobrazení.

Tento text se konkrétně soustřeďuje na oblast analýzy sekvencí nukleových kyselin, zejména na získávání informací o sekvencích, u kterých známe pouze jejich primární strukturu, tedy pouze pořadí molekul, ze kterých se sekvence skládá. V tomto směru se využívá principů komparativní genomiky, kdy se informace odhadují na základě porovnání „neznámé“ sekvence s již popsanými sekvencemi. Vzájemné porovnání dvou sekvencí mezi sebou přináší biologům cenné informace nejen o strukturální podobnosti sekvencí, ale i o podobnosti jejich vlastností nebo jejich vzájemné příbuznosti z hlediska evoluce apod. Postupů, jak mezi sebou sekvence porovnat, a tedy získat nové informace o neznámé sekvenci, je celá řada.

Cílem práce je prezentovat především informatické automatizované postupy porovnávání sekvencí, které biologům výrazně usnadňují práci při získávání informací o sekvencích DNA. V této práci se postupně rozebere několik různých přístupů k problému porovnání sekvencí. Hlavní důraz je však kladen na algoritmy z rodiny Blast, které jsou díky svým metodám využívajících různých heuristik, schopné porovnání sekvencí výrazně urychlit. Součástí práce je také vlastní implementace algoritmu, která vychází z původní verze algoritmu Blast1. Na této implementaci se ukážou některé rafinovanosti těchto algoritmů. Než se ale představí konkrétní principy algoritmů pro porovnávání sekvencí, popíší se nejprve základní biologické a informatické pojmy

V první kapitole nazvané „Základní pojmy“ se čtenář může seznámit převážně s biologickými pojmy, jako jsou genomika, dna, gen, bioinformatika apod. Dále se může dozvědět něco o historii biologie, o významných osobách a jejich objevech, které provázely vývoj biologie. Celá kapitola vyúsťuje částí, která objasňuje, jak spolu souvisí obory biologie a informatiky a jaké výhody přináší jejich spolupráce. V této části je možné si udělat lepší představu o tom, co se skrývá pod pojmem bioinformatika a jaký má význam pro dnešní lidskou společnost.

Druhá kapitola se konkrétně zaměřuje na makromolekulu DNA, která je nositelkou genetické informace všech organismů s výjimkou těch nebuněčných organismů, u nichž hraje tuto úlohu RNA.

Pro pochopení principů porovnávacích algoritmů je nezbytné vědět některé základní informace o struktuře a vlastnostech této makromolekuly.

V další kapitole nazvané „Struktura databází a formáty dat DNA“ je možné získat informace o současných databázích pro ukládání sekvencí DNA a organizacích, které tyto databáze spravují. Jsou zde také popsány formáty, v jakých se sekvence ukládají a distribuují.

Čtvrtá kapitola nazvaná „Porovnávání sekvencí“ popisuje, základní porovnávací postupy a problémy, které se k tomu vážou. Jsou zde představeny některé algoritmy dynamického programování a nástin dalších zajímavých sofistikovaných řešení.

Pátá kapitola detailněji popíše algoritmy z rodiny Blast. Jsou to chytré algoritmy, založené na heuristice, které jsou schopné provést porovnání sekvencí s mnohem menší spotřebou systémových prostředků.

Předposlední kapitola popíše implementaci vlastního algoritmu pro porovnávání sekvencí vycházející z algoritmů rodiny Blast. Představí strukturu programu a poukáže na některé problémy, které se při implementaci vyskytly. Zároveň také popíše i jejich řešení. V závěru kapitoly se provede analýza, jejímž úkolem je odhalit slabá místa vlastní implementace.

V závěru práce je možné nalézt porovnání a stručné zhodnocení výsledků porovnání jednotlivých algoritmů. Tato kapitola také shrnuje slabá místa vlastní implementace a následně se snaží navrhnout možné úpravy a vylepšení.

2 Základní pojmy

Jak již bylo zmíněno v úvodu, tato práce se zabývá technikami pro získávání informací z dat v genomice. Než se pustíme do vlastní práce s daty, musíme nejdříve přesně pochopit, jaká data se pod pojmem „data v genomice“ skrývají. Popíšeme si proto zásadní vědní obory, které se těmito daty zabývají, tedy obory, které zkoumají jejich strukturu, postup získávání, zpracování a uchovávání. Jelikož se jedná o data biologická, jsou to obory v zásadě biologické, které pro zpracování zkoumaných dat využívají informační technologie.

2.1 Biologie

Biologie je vědní obor, který vznikl jako reakce na lidskou touhu pochopit jak fungují živé organismy. V nejširším slova smyslu můžeme říci, že se biologie zabývá vším, co souvisí s organismy, od chemických dějů v organismech probíhajících na úrovni atomů a molekul, až po celé ekosystémy – tedy společenstva mnoha populací různých organismů a jejich vzájemné vztahy i vztahy k jejich životnímu prostředí.

2.1.1 Vybrané kapitoly z historie biologie

Za rozhodující krok v posunu biologie na exaktní moderní základ považují někteří historici práci Williama Harveye (1578-1657), který jako první experimentálně zkoumal funkci srdce a pohyb krve v těle. K obrovskému rozmachu biologie jakožto exaktní vědy došlo v devatenáctém století. Připomeňme si některé zásadní postavy tohoto období.

- Jan Evangelista Purkyně (1787-1869) zakladatel embryologie
- Theodor Schwann (1810-1882) představil buněčnou teorii
- Louis Pasteur (1822-1895) zkoumal nemoci, bakterie, očkování, pasterizaci
- Robert Koch (1843-1910) založil obor bakteriologie
- Elie Mečnikov (1845-1916) začal studovat imunitu
- Paul Ehrlich (1854-1915) položil na vědecký základ chemoterapii.
- Charles Darwin (1809-1882) vysvětlil základní mechanismus evoluce
- Gregor Mendel (1822-1884) objasnil základní principy dědění vlastností

Ve dvacátém století došlo k rychlému rozvoji biologie, zvláště v jeho druhé polovině. První polovině dominovaly převážně fyzika a chemie. Zásadním a snad nejvýznamnějším posunem v poválečné biologii byl vznik oboru molekulární biologie. Jako první impuls pro vznik oboru molekulární biologie mnozí považují práci Jamese D. Watsona a Francise Cricka, kteří jako první objasnili strukturu deoxyribonukleové kyseliny (DNA). Až do jejich výzkumné práce, prováděné v Cambridgi

na počátku padesátých let, nebyla úloha DNA v uchovávání a přenosu dědičné informace všeobecně uznávána. V té době se za základ života považovaly proteiny (bílkoviny) [12].

Právě Cambridge je považována za kolébku molekulární biologie, protože kromě objevu Watsona, Cricka a Wilkinse tam Max Perutz a John Kendrew rentgenostrukturní analýzou stanovili prostorové uspořádání prvních molekul proteinů. A bylo to také v Cambridgi, kde Frederick Sanger stanovil poprvé pořadí aminokyselin v proteinu (inzulínu) a kde vypracoval i metodu analýzy nukleových kyselin („čtení“ genetické informace).

S přibývajícími objevy se postupem času molekulární biologie začala vyvíjet dvěma doplňujícími se směry. Současná molekulární biologie je syntézou strukturní biologie a molekulární genetiky. Důležité je také zmínit i další syntézu, a sice molekulární biologii a studium buněk a celých organismů. Daří se tak porozumět funkcím buněk a organismů na úrovni molekul. Je tak možné sledovat, do jaké míry molekulární podstata organismů ovlivňuje jejich vlastnosti.

2.1.2 Genomika

Než si vysvětlíme, jak obor genomiky vznikl, a co je jeho hlavní náplní výzkumu, vysvětlíme si stručně některé pojmy, na kterých celá genomika stojí.

DNA (deoxyribonukleová kyselina)

DNA je makromolekula, skládající se ze dvou kolem sebe obtočených řetězců, často v tomto stavu nazývaných jako dvoušroubovice. Každý tento řetězec se skládá ze čtyř typů poměrně jednoduchých organických molekul, pospojovaných kovalentními vazbami. Těmito molekulami jsou deoxyadenosinfosfát (označován písmenem A), deoxycytidinfosfát (C), deoxyguanosinfosfát (G) a deoxythymidinfosfát (T). Souborně se tyto molekuly nazývají nukleotidy. Pořadí nukleotidů, kterých je v každém vlákně DNA mnoho tisíc až miliónů, tvoří dědičnou informaci. Sled a kombinaci těchto „písmen“ umí organismus interpretovat jako návod ke svému životu [12].

Gen

Informace zapsaná v pořadí nukleotidů je rozdělena do úseků, podobně jako je psaný text rozdělen do vět. Gen je úsek DNA, kódující vznik jednoho typu proteinu. A proteiny jsou těmi molekulami, které řídí (katalyzují) chemické reakce v organismu. Například DNA jednoduché bakterie obsahuje několik tisíc genů; k životu bakterie tedy postačuje několik tisíc různých proteinů.

Chromozóm

Genová výbava člověka obsahuje přibližně 3,2 x 10^9 vazebných párů nukleotidů (bází – také označováno zkratkou bp). DNA je v jádru vázána s bílkovinami a tvoří útvary zvané chromozómy. Chromozóm se sestává z histonových bílkovin, které tvoří jakousi kostru, na níž se molekula DNA

namotává. Zároveň se podílí na různých dalších úkolech (replikace DNA, ochrana DNA, regulace replikace atd.). Člověk má 23 souhlasných párů chromozomů.

Genom

Genom je veškerá genetická informace uložená v DNA (u některých virů v RNA) konkrétního organismu. Zahrnuje všechny geny a nekódující sekvence. Termín genom je možné chápat úžeji jako kompletní výčet veškeré jaderné DNA, avšak může být do něj také navíc zahrnuta i veškerá DNA těch organel, které dědí svou vlastní DNA. Tedy soubor veškeré buněčné DNA obsahuje vedle vlastní jaderné DNA také mitochondriální a u rostlin plastidovou DNA.

Genomika

Nyní se již zaměříme konkrétně na genomiku. V posledních desetiletích se v biologii věnuje výrazná pozornost výzkumu nukleových kyselin a proteinů. Právě tyto obrovské molekuly a složité sítě jejich vzájemných interakcí tvoří podstatu živých organizmů. Vlastnost molekuly DNA udržet v sobě určitou genetickou informaci, která je pro každý organismus jedinečná, umožnila organismům se neustále vyvíjet a přizpůsobovat životním podmínkám.

Bez evoluce by jen stěží mohly vzniknout tak složité objekty, jako je třeba pampeliška nebo člověk. Instrukce pro vytvoření známých autonomních forem života jsou zakódovány v primárních strukturách jejich molekul DNA. Máme-li tyto instrukce pochopit, musíme dokázat přečíst primární struktury molekuly DNA a dekódovat jejich biologický smysl. Tímto se právě zabývá obor genomika. Definici genomiky tedy můžeme formulovat podle [12] takto: „Genomika je prudce se rozvíjející vědní obor, jehož snahou je určit úplnou primární strukturu (pořadí bází) DNA studovaného druhu a vytěžit z této znalosti maximum biologicky relevantních informací, zejména identifikovat jednotlivé geny a určit jejich funkční vztahy.“ Někdy se genomika rozděluje na tři základní směry:

- **strukturní genomika** - snaha o stanovení sledu nukleotidů genomu organismu.
- **funkční genomika** - experimentem, například vyřazením nějakého genu z činnosti, se snaží přiřadit funkci neznámým genům, případně funkci genů studovat.
- **bioinformatika** - interpretace přečtené dědičné informace počítačovými metodami ve spolupráci s databázemi.

Jedním z největších pokroků biologie je vypracování metod pro stanovení sledu (sekvence) nukleotidů v molekulách DNA. Toto tzv. „sekvenování“ DNA nám doslova umožňuje „přečíst si“ dědičnou informaci organismů písmenko po písmenku.

Dnes již známe úplnou dědičnou informaci více než 20 organismů. Většina těchto organismů jsou bakterie (Tabulka 1). Bakteriální genomy jsou poměrně malé, složené „pouze“ z miliónů nukleotidů, je tedy snazší je rozluštit. Nejen proto se tyto genomy získávají. Pro člověka mají zásadní význam, neboť bakterie jsou často příčinou chorob, rychle se množí a mutují a je tedy možné

zkoumat v poměrně krátkém časovém horizontu jejich vývoj a pochopit tak různé vlastnosti nukleových kyselin ve vztahu k vývoji a životu organismů.

26. června 2000 dva nezávisle na sobě pracující vědecké týmy ze Spojených států a z Velké Británie (*the National Human Genome Research Institute*, *Celera Genomics*) oznámily dokončení mapování lidského genomu. Tento vědecký mezník je přirovnáván k objevu štěpení atomu ve fyzice nebo k chůzi člověka po povrchu Měsíce.

ČESKÝ HOVOROVÝ NÁZEV	LATINSKÝ NÁZEV	POČET BÁZÍ
kvasinka	Saccharomyces cerevisiae	12 Mbp
hlístice	Caenorhabditis elegans	97 Mbp
muška octomilka	Drosophila melanogasther	137 Mbp
člověk	Homo sapiens	3,2 Gbp

Tabulka 1: Přehled počtů nukleotidů

Lidský genom je sice přečten, avšak nejobtížnějším krokem je popis tohoto genetického kódu, kdy vědci musí identifikovat každý gen a jeho funkce. Očekává se, že tento poslední krok bude dokončen během několika následujících let. Lékaři budou schopni vypracovat genetický profil svých pacientů, z něhož budou schopni určit náchylnost vůči různým chorobám a zvolit vhodnou strategii léčby. V tomto směru již biologům velmi pomáhá bioinformatika.

Slibné výsledky dnes poskytuje srovnávací genomika. Genomy – a zejména geny člověka, myši a např. mušky octomilky – jsou si mnohem podobnější, než by napovídaly obrovské rozdíly v jejich vlastnostech a stavbě těla. Myš je blízko člověku, a přitom je to laboratorní objekt s velmi krátkou generační dobou, takže je možné dělat i populační experimenty apod. Na základě takových experimentů a studií je možné vytvářet rozumné extrapolace a experimentálně testované hypotézy o analogických vlastnostech člověka.

2.2 Bioinformatika

Exponenciální růst poznatků na poli molekulární biologie s sebou jednoznačně přináší integraci výpočetních technologií do tohoto oboru. Nejen, že jsou počítače zapojené při samotném bádání a provádění experimentů ve výzkumných laboratořích, kde hlavní úlohu hraje automatizovaný hardware, dnes se začínají využívat i pro simulace, modelování, zpracování a ukládání obrovského množství dat, jejichž množství roste stále rychleji právě i díky genomickému výzkumu, kde čtení genomu organismů pokračuje ve stále rychlejším tempu, a stále rychleji tak přibývají nové přečtené genomy (myš, krysa, šimpanz …).

Ohromné množství nukleotidů v jednotlivých sekvencích DNA, jak ukazuje tabulka 1, již nedovolí jejich manuální prohledávání za účelem najít v nich například nějaké podobnosti, které by

mohli prozradit něco o evoluci organismů na Zemi. Nastává tedy problém. Biolog potřebuje aplikaci, která by řešila jeho problémy a ulehčila mu tak významně jeho práci, ale nemá potřebné programátorské znalosti. Na druhé straně informatik tyto znalosti má, ale nedokáže se orientovat v problémech biologa, nerozumí problémům biologie, a tak nedokáže naprogramovat aplikaci, která by dané problémy řešila. Tomuto se právě věnuje obor bioinformatika.

Bioinformatika je mnohostranná disciplína kombinující mnoho vědních oborů: výpočetní biologii, statistiku, matematiku, molekulární biologii a genetiku. Umožňuje biologům využít existující výpočetní technologie, aby uceleně ukládali, dolovali, získávali a analyzovali data získané genomickými a proteomickými technikami. Toho je dosaženo vytvořením jednotných datových modelů, standardizovaným datovým rozhraním, vývojem strukturovaných slovníků, vytvářením nových datových vizualizačních metod a zaznamenáváním podrobných metadat, která popisují různé aspekty experimentálních návrhů a rozborů metod.

Mezi typické úkoly bioinformatiky patří sestavování vysoce kvalitní a odpovídající geonomové sekvence z fragmentů DNA s využitím sekvenování, studium genové regulace s využitím dat z mikročipů (microarrays) nebo hmotnostní spektrometrie a další. [1] V následující části se na některé vybrané oblasti zkoumání bioinformatiky podíváme podrobněji.

2.2.1 Vybrané oblasti zkoumání

Analýza sekvence

Od roku 1977, kdy byla sekvenována jedna z prvních DNA sekvencí, byly dekódovány a poté uloženy do databáze stovky DNA sekvencí organismů. Tato data jsou analyzována za účelem určení genů, které kódují proteiny. Porovnání genů v rámci jednoho druhu organismu nebo mezi různými druhy může ukázat podobnosti mezi funkcemi proteinů nebo vztahy mezi druhy. Se stoupajícím množstvím dat je velmi nepraktické je zpracovávat ručně. V současnosti jsou pro hledání a objevování genomů organismů obsahující biliony nukleotidů používány počítačové programy. Tyto programy mohou vyrovnat mutaci (změnu, odstranění nebo vložení báze) v DNA sekvenci za účelem identifikace sekvencí, které jsou spřízněné ale ne identické.

Obdoba tohoto porovnávání sekvencí je využita v samotném sekvenovacím procesu. Konce sekvenovaných fragmentů molekuly DNA se překrývají a vhodným seřazením a zarovnáním je možné sestavit kompletní genom. Z důvodu obrovského množství fragmentů je tato úloha celkem náročná. Algoritmy pro sestavování kompletních genomů jsou dnes kritickou oblastí bioinformatického výzkumu.

Dalším aspektem bioinformatiky v oblasti analýzy sekvencí je automatické vyhledání a rozpoznání genů a řídících úseků sekvencí v rámci určitého genomu. Ne všechny nukleotidy genomu jsou geny. U vývojově vyšších organismů větší část DNA neslouží žádnému konkrétnímu účelu. Vědci se domnívají, že mnoho těchto částí do sekvence DNA vnesly viry.Někdy jsou tyto úseky

nazývány jako odpadní DNA. Přesto tyto části ale mohou obsahovat námi nerozeznané funkční elementy. Bioinformatika pomáhá přemostit mezery mezi projekty týkajících se genomů a projekty zabývající se proteiny. Například využití DNA sekvence pro identifikaci proteinu.

Analýza exprese genu

Genová exprese je proces, při kterém se informace uložená v daném genu převádí na konkrétní buněčnou strukturu nebo vlastnost. Děj probíhá ve dvou krocích, kdy se DNA nejprve přepíše do tzv. mRNA a ta je následně přeložena na protein. Analýzou exprese genu je možné zjistit, v jakých situacích se gen stává aktivní a jak svojí expresí může ovlivnit expresi jiných genů. Velmi používanou technikou analýzy exprese genu jsou tzv. mikročipy, kam se nanese velké množství naklonovaných genů a poté se naráz zkoumá jejich exprese. Pomocí takového biočipu je možné sledovat, jak se jednotlivé geny zapínají nebo vypínají v chorobných stavech, jak se liší chorobná tkáň od tkáně zdravé, které geny přispívají k různým dědičným chorobám apod.

Predikce struktury proteinu (Prediction of protein structure)

Predikce struktury proteinu je další důležitá oblast bioinformatiky. Sekvence aminokyselin, primární struktura proteinu, může být jednoduše určena ze sekvence nukleotidů genu, který ji kóduje. Ve většině případů tato primární struktura jednoznačně určuje strukturu ve svém přirozeném prostředí. Znalost této struktury je nezbytný k pochopení funkce proteinu.

Jedna z klíčových myšlenek v bioinformatice je pojem homologie. V genomické větvi bioinformatiky se homologie využívá k predikci funkce genu. Jestliže sekvence genu A, jehož funkce je známa, je homologická se sekvencí genu B, jehož funkce je neznámá, je možné usuzovat, že B může sdílet funkce A. Ve strukturní větvi bioinformatiky se homologie využívá pro určení, které části proteinu jsou důležité v strukturním uspořádání a interakci s ostatními proteiny. Tato informace se pak využívá k predikci struktury proteinu, jakmile je známa jedna struktura proteinu homologického.

Srovnávací genomika (Comparative genomice)

Jádro srovnávací genomické analýzy je založení shody a korespondence mezi geny nebo jinými genomickými rysy v různých organizmech. Jsou to intergenomické mapy, které umožňují sledovat evoluční procesy zodpovědné za rozdílnost dvou genů. Velké množství evolučních událostí vyskytujících se na různých strukturních úrovních, utvářejí geonomovou evoluci. Na nejnižší úrovni může mutace ovlivnit jeden konkrétní nukleotid. Na vyšší úrovni pak velká chromozómová část podstoupí duplikaci, inverzi, přesun, odstranění a vložení. Nakonec se celý genom účastní procesu hybridizace a dalších, což může vézt k rapidní specializaci – vzniku nového druhu.

2.2.2 Stručný souhrn

Aplikace bioinformatiky

- Získávání dat
- Nástroje pro přístup k databázím
- Mapování a srovnávání genomů
- Seřazení a porovnávání sekvencí
- Identifikace genů
- Funkční identifikace proteinů
- Molekulární evoluce
- Molekulární modelování
- Predikce struktur (bílkovin)
- Analýza exprese genů
- Srovnávání struktur
- Stanovení makromolekulárních struktur
- Vývoj léčiv na základě struktur

Jaká bioinformatická data se sbírají

- Sekvence DNA a RNA
- Sekvence proteinu
- Struktura proteinu
- Údaje o aktivitě genu - DNA mikročip
- Údaje o expresi proteinu - mikročip, hmotnostní spektrometrie
- Mapy interakcí mezi proteiny a DNA
- Mapy interakcí mezi proteiny navzájem

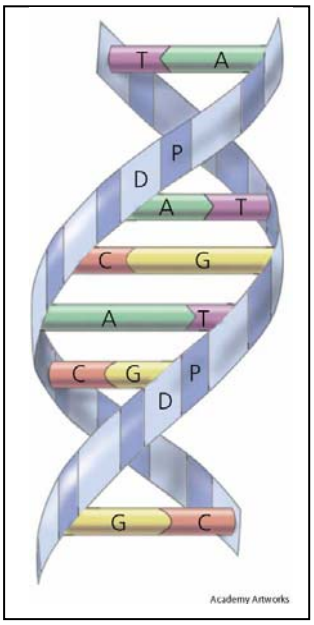
3 DNA (deoxyribonukleová kyselina)

Tato kapitola je celá věnovaná makromolekule DNA. Vysvětlíme si, co to DNA je, a kde ji můžeme nalézt. Postupně si popíšeme její strukturu od nejvyšší abstrakce, kdy na ni budeme nahlížet jako na dlouhé vlákno až po nejnižší úroveň abstrakce, kdy si popíšeme jednotlivé molekuly a atomy, ze kterých se skládá. Dále probereme její význam a nejdůležitější vlastnosti.

3.1 Obecný popis DNA

DNA je nositelkou genetické informace všech organismů s výjimkou těch nebuněčných organismů, u nichž hraje tuto úlohu RNA. DNA je pro život nezbytnou látkou, která ve své struktuře kóduje program života. Jsou zde kódovány plány pro syntézu nukleových kyselin a bílkovin, které pak ovlivňují vývoj a vlastnosti každé buňky žijícího organismů a tím zároveň kompletně vlastnosti celého organismu

DNA sekvence každé živé entity je unikátní a mezi jedinci jednoho živočišného či rostlinného druhu v ní nalezneme rozdíly. Na druhou stranu, prakticky všechny buňky, bez ohledu na to, z jaké části organismu pocházejí, nesou úplnou dědičnou informaci pro vývoj celého organismu: všechny mají stejnou DNA. Nicméně jen určitá část této informace, je v konkrétní buňce realizována. Tím se od sebe buňky navzájem v průběhu růstu diferencují. Buňky lidského těla tak tvoří různé tkáně, například játra, kůži, svaly apod. Pro každou konkrétní buňku je DNA určitou kuchařkou, podle níž realizuje svůj specifický program.



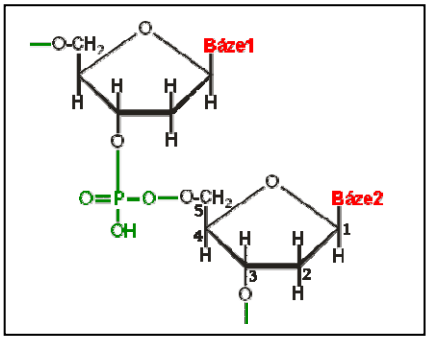
Obrázek 1: Molekula DNA; zdroj: [12]

DNA řídí a udržuje při životě celý organismus vydáváním pokynů buňce pro vytváření základních molekul bílkovin. Bílkoviny mají velmi různorodé vlastnosti a funkce. Zmiňme např. bílkoviny stavební (kosti, svaly) nebo regulační (enzymy). Úsek sekvence DNA pro kódování jedné bílkoviny se nazývá gen. Každý gen může mít jednu nebo několik forem - alel. Většina genů potřebných pro život se v eukaryotických buňkách nachází převážně v jádře na chromozómech, částečně pak v mitochondriích a u rostlin v chloroplastech. U prokaryotických organizmů (jednodušší organizmy jako jsou viry a bakterie) se genetická informace nachází na tzv. prokaryotním chromozómu a v plazmidech.

3.2 Struktura DNA

DNA je biologická makromolekula skládající se ze dvou dlouhých řetězců zkroucených do dvojité pravotočivé šroubovice tvořící kroucený žebřík. V každém řetězci se střídá pětiuhlíkový cukr

deoxyribóza se zbytkem kyseliny fosforečné (tzv. fosfátem). Na deoxyribózu je z vnitřní strany dvoušroubovice připojena heterocyklická sloučenina, tzv. báze. Na místě báze se mohou v DNA vyskytnout čtyři různé heterocyklické sloučeniny. Dvě z nich jsou odvozeny od struktury purinu a dvě od struktury pyrimidinu. Mezi purinové báze patří Adenin (A), Guanin (G), a mezi pyrimidinové patří Tymin (T), Cytosin (C). Báze do sebe specificky zapadají a tvoří tak vazebné páry, které při sobě drží pomocí vodíkových můstků a spojují tak oba řetězce molekuly DNA do pravotočivé dvoušrobovice.



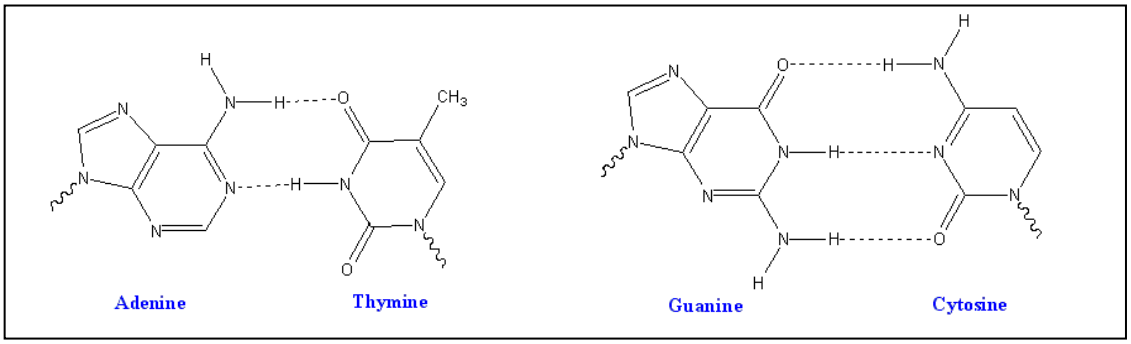
Obrázek 2: Část řetězce DNA; zdroj [12]

Dvouřetězcová struktura DNA je výhodná pro uchování genetické informace, protože je stabilnější a lépe odolává vnějšímu poškození. Genetická informace je v DNA zapsána v pořadí bází samozřejmě jen v jednom z jejích řetězců, druhý řetězec je jeho doplňkem (je tzv. komplementární). Je to dáno komplementaritou samotných bází, která vychází z jejich chemického složení. Tyto báze se mohou pojit pouze následujícím způsobem: A se pojí s T a C se pojí s G.

Trojice deoxyribóza + zbytek kyseliny fosforečné + báze, tzv. nukleotid, tvoří základní stavební jednotku (monomer) molekuly DNA. Protože na místě báze se mohou vyskytovat čtyři různé molekuly, odvozujeme pro nukleotid čtyři různé názvy podle obsažené báze: deoxyadenosinfosfát, deoxytymidinfosfát, deoxyguanosinfosfát a deoxycytidinfosfát. Pokud z trojice látek, ze kterých je složen nukleotid, odebereme zbytek kyseliny fosforečné, dostaneme tzv. nukleosid.

Pro zopakování ještě uveďme, že hřbet dvoušroubovice je tedy tvořen cukrem a fosfátem a vnitřek nukleovou kyselinou (nukleovými bázemi).

- Báze adenin (A), tymin (T), guanin (G), cytosin (C)
- Nukleosid báze + cukr (deoxyribosa)
- Nukleotid báze + cukr + zbytek kyseliny fosforečné

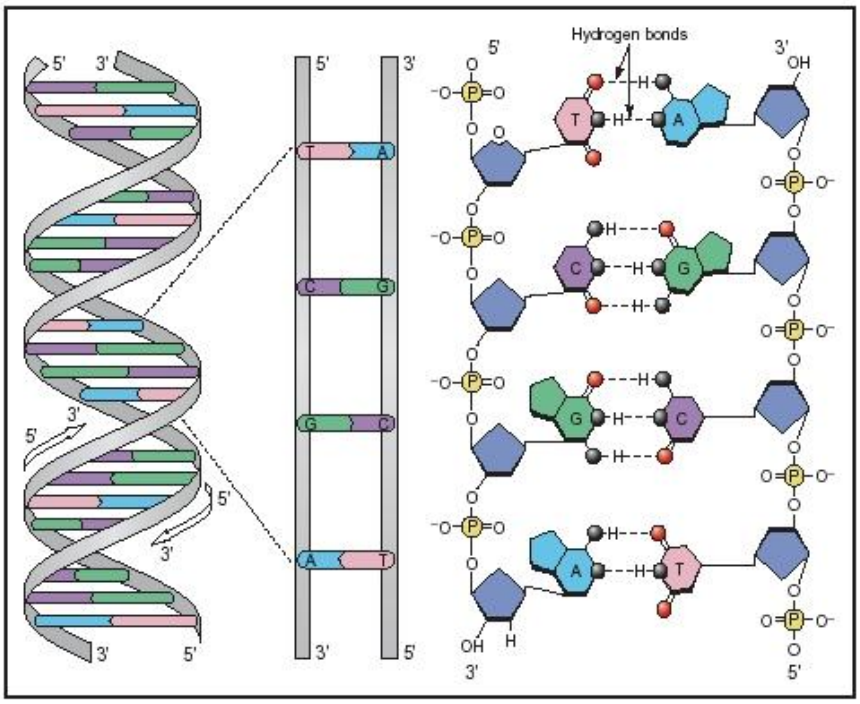


Obrázek 3: Ukázka vazeb mezi bázemi; zdroj: [12]

V nukleotidu má každý atom své číslo, což umožňuje těm, kteří systém číslování pochopili, orientaci ve složité molekule. Uhlík (chmická značka C) v cukerné složce, na němž je navázán zbytek kyseliny fosforečné, má číslo 5´, uhlík, na němž je navázána skupina OH, má číslo 3´ Každé vlákno, řetězec,

má podle konvence 3´-hydroxylový a 5´-fosfátový konec. Na obrázku 2 je zbytek kyseliny fosforečné zobrazen zelenou, cukerná složka černou a místo pro navázání jedné z báze červenou barvou V dvoušroubovici mají vlákna opačný směr, tj. jedno vlákno je orientováno z konce 3´ ke konci 5´ a druhé vlákno z konce 5´ ke konci 3´. Obě vlákna ve dvoušroubovici DNA jsou tedy vůči sobě nejenom komplementární, ale také opačně orientovaná (obrázek 4).

Podíváme-li se pozorně na schéma dvouřetězcové molekuly DNA (obrázek 4), můžeme si všimnout, že deoxyribózy jsou v protějších řetězcích orientovány opačnými směry. Směřuje-li v horním řetězci kyslík ve vrcholu pětiuhlíkového cyklu deoxyribózy vždy doleva, pak v dolním řetězci směřuje vždy doprava. Každý z řetězců tedy má svůj směr. Toho právě při čtení genetické informace



Obrázek 4: Přehled struktury molekuly DNA; zdroj: [12]

využívá enzym RNA-polymeráza. Kvůli této orientaci vláken se RNA polymeráza může po konkrétním řetězci vydat pouze jedním směrem, který je dán orientací 5'- a 3'-konce.

Z tohoto faktu vychází i základní konvence, kterou se řídí zápisy jakékoli sekvence bází v nukleových kyselinách v učebních a vědeckých textech - uváděné pořadí vždy představuje sled nukleotidů od 5'-konce vlákna ke 3'-konci vlákna. Pokud je zapisováno dvouvlákno, pak se toto pravidlo vztahuje vždy na horní vlákno.

Nyní se ještě podívejme na možné nahlížení na strukturu DNA. U nukleových kyselin rozlišujeme primární, sekundární a terciální strukturu.

Primární struktura

Primární struktura je dána pořadím nukleotidů v sekvenci. Nese v sobě genetickou informaci.

Sekundární struktura

Forma stočení dvoušroubovice. Vlákna DNA se přirozeně stáčí do dvoušroubovice, avšak forma stočení není vždy za všech podmínek stejná. Může se vyskytovat v několika formách, které mají vliv na reaktivitu molekuly:

- ds forma A - Pravotočivá; 10 párů bazí na otáčku; průměr vlákna je 2,3 nm
- ds forma B - Pravotočivá; 11 párů bazí na otáčku; průměr vlákna je 1,9 nm
- ds forma Z - Levotočivá; 12 párů bazí na otáčku; průměr vlákna je 1,8 nm

Terciární struktura (supercoiling)

Velmi dlouhá molekula DNA není jen neuspořádaným klubkem náhodně zamotaného vlákna. Celá molekula se velmi pečlivě několikanásobně navíjí a skládá. Podobně jako u bílkovin i dvojitá šroubovice nukleových kyselin může být prostorově stočena do nadšroubovicového vinutí - superhelixu.

Pro lepší představu o velikosti makromolekuly DNA si uveďme příklad. Pokud by se nám podařilo rozmotat a natáhnout celou molekulu DNA z jedné lidské buňky, měřila by téměř dva metry, a kdybychom ji zvětšili do tloušťky niti, byla by dlouhá asi 300 kilometrů.

3.3 Vlastnosti a význam DNA

Základní vlastní úlohou DNA je uchovávat a v případě potřeby dále předávat genetickou informaci (o pořadí aminokyselin v bílkovinách), která je zapsaná v její primární struktuře. Tato informace se přenáší (dědí) z generace na generaci. Člověk obdrží od každého rodiče 23 chromozomů. Prvním a zásadním krokem při uchovávání dědičné informace a při jejím přenosu z rodičů na potomky je její zdvojení.

3.3.1 Replikace DNA

Základní mechanizmus replikace, tedy zdvojení, deoxyribonukleové kyseliny je znám již od objevu struktury její molekuly začátkem padesátých let. Vlákna DNA jsou složitým pochodem za pomoci řady proteinů rozplétána a k uvolněným vláknům jsou prostřednictvím enzymů dosyntetizovány komplementární protějšky. Tak z jedné dvouvláknové molekuly DNA vznikají dvě identické dvouvláknové molekuly. Při dělení buňky pak každá z nich přechází do jedné buňky dceřiné. Tento proces musí proběhnout při každém dělení buňky.

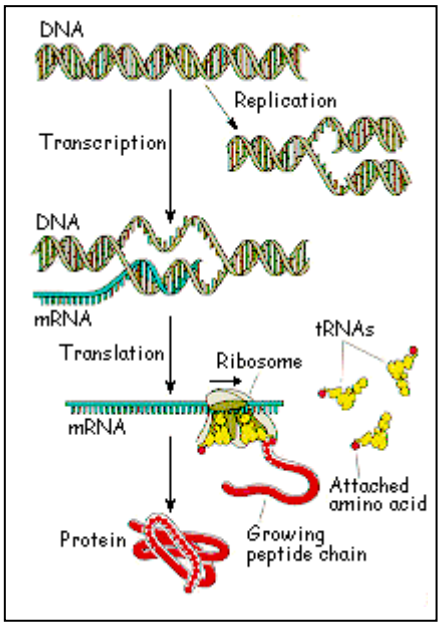
Vlastní syntézu vlákna DNA řídí enzym DNA-polymeráza, která katalyzuje přenos nukleotidů z deoxyribonukleosid-trifosfátů na rostoucí řetězec. Protože je bytostně důležité, aby kopie byly dokonale stejné, má DNA-polymerasa schopnost odstranit připojený nukleotid, pokud není připojen komplementárně. Tento proces se pak označuje jako reparace DNA.

3.3.2 DNA jako předloha bílkoviny

DNA se nachází v jádře, ale zařízení umožňující vznik bílkovin je umístěno v cytoplasmě na vnější straně jaderné membrány. DNA se dorozumívá s tímto zařízením prostřednictvím poslů, jimiž jsou molekuly RNA. Cesta od molekuly DNA k vytvoření proteinu vede přes několik fází. Jsou jimi transkripce, translace a proteosyntéza. Na obrázku 5 jsou docela přehledně znázorněny.

Transkripce (přepis genetické informace z DNA do RNA)

Transkripce je enzymatický proces, jehož průběh je závislý na katalytickém působení RNA-polymerázy. RNA-polymeráza je enzym, který přečte informaci z jednoho řetězce DNA a zkopíruje ji do pořadí nukleotidů v nově syntetizované molekule RNA. Tato molekula RNA se pak uvolní a putuje do cytoplasmy, kde je syntetizována bílkovina s určitým pořadím aminokyselin právě podle pořadí nukleotidů v molekule RNA.



Obr. 5 Postup syntézy bílkovin; zdroj [12]

Při transkripci se tvoří tzv. mRNA. Nejprve dochází k rozvolnění jednoho úseku DNA, a to v oblasti molekuly, která se nazývá promotor. Na základě komplementarity dusíkatých bází probíhá na jednom vlákně DNA přiřazování komplementárních nukleotidů ve směru 5' > 3'. Jejich vzájemným spojením diesterickými vazbami vzniká souvislý polynukleotidový řetězec (*RNA obsahuje nukleotid uracil místo tyminu*). V průběhu transkripce tak vzniká řetězec mRNA, který se uvolní, projde jadernou membránou do cytoplazmy a napojí se na ribozom. Transkripce končí v oblasti DNA zvané terminátor. Molekula mRNA tedy v sobě nese přepis genetické informace mezi promotorem a terminátorem. Rozsah této informace může být různý.

Translace

Translace je proces syntézy proteinů podle informace obsažené v molekulách mRNA. Uskutečňuje se na ribozomech a její průběh je katalyzován řadou enzymů. Na ribozom se mRNA navazuje zásadně 5´ koncem své molekuly. Vazba se děje na menší pod-jednotce ribozomu. Ribozom se po navázané mRNA posunuje ke konci 3´, přičemž volný konec 5´se zatím může napojovat na další a další ribozomy. Tento komplex postupujících ribozomů na molekule mRNA se nazývá polyribozom. Při posouvání se mRNA dostává na každém ribozomu do kontaktu s jeho dvěma vazebními místy. Tato místa odpovídají velikostí dvěma kodonům (*trojicím ribonukleotidů v mRNA*). V těchto místech

dochází k připojování aminokyselin a k jejich spojování do polypeptidového řetězce. Označují se jako aminoacylové místo (A site) a peptidylové místo (P site).

Translace se považuje za zahájenou teprve ve chvíli, kdy se pohybující se ribozom dostane svým P místem na iniciační kodon mRNA (*většinou AUG*). To je signálem připojení iniciační transferové RNA (tRNA) s navázanou iniciační aminokyselinou (*formylmetioninem nebo metioninem – záleží na „prokaryocitě" nebo „eukaryocytě" organismu*). Samozřejmě zde hrají významnou úlohu nejméně 3 různé iniciační enzymy. tRNA přináší pomocí komplexů k ribozomům aminokyseliny.

Syntéza polypeptidů na ribozomech

Po iniciačním kodonu se pohybem po molekule mRNA dostane ribozom do A-místa. To je signálem pro připojení dalšího komplexu tRNA s odpovídajícím antikodonem. Na ribozomu jsou vedle sebe tedy dva komplexy aminoacyl-tRNA v takovém vzájemném uspořádání, které umožňuje vznik peptidové vazby mezi karboxylovou skupinou iniciační aminokyseliny a aminoskupinou následující aminokyseliny. Současně se vznikem peptidové vazby se však ruší vazba iniciační aminokyseliny na iniciační tRNA a tato tRNA se z ribozomu uvolňuje. Vazebné P-místo je tedy volné a k A-místu je připojen dipeptidový komplex spojených s jednou molekulou tRNA. Dalším pohybem ribozomu po mRNA se opakuje stejný proces dál. Všechny aminoacyl-tRNA musí nejdříve projít A-místem, aby se dostaly do P-místa. Přemísťování komplexů je katalyticky podmíněno činností enzymů zvaných elongační faktory. Translace se ukončuje ve chvíli, kdy se do A-místa dostane tripletová oblast mRNA, která svým nukleotidovým složením odpovídá jednomu z terminačních kodonů (*UGA, UAA, UAG*). Pro tyto kodony neexistuje žádný odpovídající typ tRNA s komplementárním antikodonem. Nemožnost napojení další aminokyseliny na již začleněnou je tedy signálem pro uvolnění polypeptidového řetězce z ribozomu. Opět je katalyzováno terminačními faktory (*enzymy*).

Průměrná bílkovina je složena z řetězců z několika set aminokyselin. Počet kombinací jednotlivých bází a jejich kódu pro bílkoviny je nekonečný, DNA proto představuje stálý informační soubor buňky a nikdy neopouští jádro. Jeho hlavním a nejdůležitějším úkolem je udržovat a pečovat o genetickou informaci před poškozením či špatným přepsáním v nezměněné podobě z buňky do buňky z generace na generaci.

3.4 Získávání DNA z organismů

Nyní si ještě stručně vysvětlíme, jak se DNA získává z organizmů, jaké jednotlivé kroky je nutné udělat, aby se genetická informace uložená v podobě sekvence nukleotidů převedla do podoby textové sekvence zapsané kombinací čtyř písmen.

Izolace nukleových kyselin

Podstatou izolace nukleových kyselin je oddělení těchto kyselin od ostatního biologického materiálu, kterým jsou nukleové kyseliny obklopeny. Po izolaci nukleových kyselin je nutné ještě oddělit od sebe jednotlivé nukleové kyseliny. Pokud je například cílem izolovat molekulu DNA, musíme ještě z nukleových kyselin, které si jsou svoji chemickou stavbou velmi podobné, vybrat pouze a právě molekulu DNA. [1]

Štípání molekuly DNA

Pokud je DNA izolována z buněk šetrně, pak je představována celými chromosomy, tedy mimořádně dlouhými molekulami. Pro mnoho molekulárně-genetických technik je proto nejdříve nutné takové molekuly rozštípat na menší úlomky, se kterými lze snáze pracovat.

Mechanicky lze molekulu DNA „polámat“ velmi účinně ultrazvukem, který rozkmitá dlouhé molekuly. Ty se pak v místech největšího mechanického namáhání v průběhu kmitání lámou.. Místa zlomu jsou ovšem zcela nezávislá na konkrétní sekvenci DNA, jsou náhodná, takže v každém experimentu i v každé molekule dochází k lámání v jiných místech.

Nukleové kyseliny lze biochemicky štípat pomocí enzymů nukleáz. Revoluci v práci s nukleovými kyselinami přinesl objev tzv. restrikčních endonukleáz II. typu, které rozpoznávají v molekule DNA krátké specifické sekvence a v místě výskytu této sekvence molekulu štěpí. Revolučnost štěpení restrikčními endonukleázami není jen v tom, že jsme schopni štěpit reprodukovatelně, tj. ve stejném místě (pokud použijeme stejnou endonukleázu), ale také v tom, že rozštěpené molekuly můžeme znovu spojit.

Nikdy samozřejmě neštěpíme jedinou molekulu genomové DNA, ale velké množství těchto molekul, takže ve výsledné směsi je každý specifický fragment zastoupen v mnoha kopiích, odpovídajících počtu vstupních molekul DNA. Takovou směs nemá smysl znovu spojovat, neboť bychom dostali mnoho náhodných spojení mezi různými fragmenty v různě přeházeném pořadí. Aby mělo štípání molekul DNA smysl, musíme mít možnost ve směsi rozpoznat a izolovat určitý fragment, obsahující např. gen nebo určitou sekvenci, která nás zajímá.

Klonování DNA

Klonováním DNA se myslí množení určitého úseku DNA. V molekulární genetice se ke klonování DNA s výhodou využívají enzymatické systémy živých buněk, které jsou schopny konkrétní DNA kopírovat a klonovat prakticky bezchybně a s nízkými náklady. Prakticky technika klonování znamená, že je třeba přimět určité buňky, aby kopírovaly úsek cizí DNA, který jsme předtím připravili nějakou jinou molekulárně-genetickou metodou, a který po namnožení (klonování) v živých buňkách zase získáme zpět.

Sekvenování DNA

Termín sekvenování DNA znamená určení sekvence nukleotidů (resp. bází) v jednom z řetězců DNA. Důvodů pro zjišťování sekvence je mnoho. Pokud má například nějaká choroba genetický základ, pak je její konkrétní příčinou změna zdravé alely genu v chorobnou alelu. Podkladem této změny je právě konkrétní změna sekvence DNA. Podaří-li se takovou změnu po srovnání sekvencí zdravých a nemocných jedinců identifikovat, může to sloužit k vývoji diagnostického postupu, který bude tuto změnu namísto sekvenování identifikovat některou jednodušší a méně nákladnou molekulárně-genetickou metodou. Znalost změny sekvence, která je příčinou choroby, také otevírá cestu k budoucí cílené léčbě genovou terapií (náhradou „chorobné“ alely dodáním „zdravé“ alely).

Pro sekvenování byly vyvinuty dvě metody – chemická Maxam-Gilbertova a biochemická Sangerova, která dnes jednoznačně převládá. Tato metoda vyžaduje jednovláknovou DNA, která slouží jako vzor (templát) pro syntézu druhého (komplementárního) řetězce. Syntézu zprostředkuje enzymem polymeráza a reakce je upravena takovým způsobem, že v určitých místech dochází v závislosti na sekvenci templátového vlákna k ukončení (terminaci) syntézy. Reakce funguje tak, že k ukončení (terminaci) reakce dojde v okamžiku, kdy se v templátovém vlákně přečte určitý nukleotid. Délka nedokončeného úseku, zjištěná elektroforézou v gelu, pak odpovídá vzdálenosti od začátku vlákna, ve které se vyskytl určitý nukleotid.

Na základě tohoto ukončení se tedy identifikuje jedna ze čtyř bází. Pokud běží těchto reakcí v jeden okamžik paralelně více na několika dalších (naklonovaných) vláknech, je možné určit mnoho dalších bází, které se pak podle délky komplementárního řetězce, který polymeráza stačila nasyntetizovat než se zastavila, přesně zařadí pomocí počítačového programu do rekonstruované sekvence DNA. [1]

4 Struktura databází DNA

Nyní když máme lepší představu o tom, co se skrývá za sekvencí nukleotidů makromolekuly DNA, podíváme se na nejpoužívanější biologické databáze a formáty dat.

4.1 Instituce zabývající se shromažďováním dat

V současné době je prostřednictvím Internetu dostupných přibližně 550 databází zabývajících se shromažďováním bioinformací. Jejich přehled a popis je každoročně publikován ve specializovaném, volně dostupném čísle časopisu Nucleic Acids Research. K největším institucím zabývajících se správou dat a vývojem nástrojů pro jejich analýzu patří:

- ***Evropský institut pro bioinformatiku*** (EBI) se sídlem v Hinxtonu v UK, dostupný na webové adrese http://www.ebi.ac.uk/
- ***Národní centrum pro biotechnologické informace*** (NCBI) založené původně v rámci Národní lékařské knihovny (NLM) v USA (http://www.ncbi.nlm.nih.gov/)
- ***Centrum pro informační biologii*** (CIB) založené jako oddělení Národního genetického institutu (NIG) v Mishimě, Japonsko (http://www.cib.nig.ac.jp/).

4.2 Přístup k databázím

Existují tři prostředky na získávání informací, které umožňují textové vyhledávání v molekulárně biologických databázích. Tyto prostředky jsou vstupním bodem do mnoha integrovaných databází a každý z nich byl vyvinut v jednom ze tří hlavních center pro bioinformatiku. Navzájem se liší v databázích, které mohou prohledávat, ve vazbách, které vytvářejí mezi jednotlivými databázemi a ve vazbách vztahujících se k dalším informacím. [3]

- ***Entrez*** - vyvinutý v NCBI, adresa http://www.ncbi.nlm.nih.gov/Entrez/
- ***SRS*** (Sequence Retrieval System) - vyvinutý v EBI, adresa http://srs.ebi.ac.uk
- ***DBGET/Link DB*** – vyvinutý v Japonsku, adresa http://www.genome.ad.jp/dbget

4.3 Standardní kódy podle IUB/IUPAC

Organizace IUPAC (Mezinárodní unie pro čistou a užitou chemii) vydala dokument IUPAC-IUB, o zkratkách a symbolech užívaných v sekvencích DNA nebo proteinů. Tento dokument jednoznačně přiřazuje znaky chemickým sloučeninám, které se v těchto sekvencích mohou vyskytovat. Přehled je zobrazen v tabulkách 2 a 3.

Kódy pro sekvence aminokyselin	
A	Alanin
B	kys. asparagová nebo asparagin
C	Cystein
D	kys. Aspartová
E	kys. Glutamová
F	Fenylalanin
G	Glycin
H	Histidin
I	Izoleucin
K	Lysin
L	Leucin
M	Metionin
N	Asparagin
P	Prolin
Q	Glutamin
R	Arginin
S	Serin
T	Treonin
U	Selenocystein
V	Valin
W	Tryptofan
Y	Tyroxin
Z	kys. glutamová nebo glutamin
X	jakákoli aminokyselina
*	translační stop (terminační kodon)
-	mezera (gap) neurčené délky

Tabulka 2. Kódy pro sekvence aminokyselin

Kódy pro sekvence nukleotidů	
A	Adenosin
C	Cytidin
G	Guanidin
T	Tymidin
U	Uridin
R	G/A (purin)
Y	T/C (pirimidin)
K	G/T (nukleosid s keto sk.)
M	A/C (nukleosid s amino sk.)
S	G/C (silná vazba)
W	A/T (slabá vazba)
B	G/T/C
D	G/A/T
H	A/C/T
V	G/C/A
N	A/G/C/T (jakýkoli)
-	mezera (gap) neurčené délky

Tabulka 3. Kódy pro sekvence nukleotidů

4.4 EMBL databáze

Databáze EMBL je organizována Evropskou molekulárně biologickou laboratoří (EMBL). Je to veřejná evropská nukleotidová databáze se sídlem v Anglii na adrese http://www.ebi.ac.uk/embl. Databáze je v součinnosti vytvářena s ostatními nukleotidovými databázemi GENBANK (USA) a DDBJ (Japonsko) a je velmi dobře přístupná spolu s mnoha odvozenými a dalšími databázemi přes SRS. Databáze obsahuje všechna data zaslaná vědeckou komunitou, a to bez kontroly. Z tohoto důvodu může obsahovat určité procento chyb. Na této databázi si ukážeme, jak může vypadat formát uložených dat.

4.4.1 Vkládání dat do databáze EMBL-Bank

Většina vědeckých časopisů v dnešní době vyžaduje vložení informací o sekvencích nukleotidů do jedné z databází EMBL, GenBank nebo DDBJ před publikací článků, které se jimi zabývají. Díky tomu je zaručeno, že informace o sekvencích budou dostupné všem vědcům, už v době, kdy bude článek publikován. EMBL-Bank přidělí každému novému záznamu o sekvenci přístupové číslo, které jednoznačně a trvale identifikuje záznamy dané sekvence. Toto číslo také autoři článků zveřejňují ve svých příspěvcích a pomocí tohoto čísla mohou vkládat nové informace do záznamu o dané sekvenci. Preferovaný způsob vkládání informací do EMBL databáze je pomocí www systému *Webin*, který je dostupný na adrese http://www.ebi.ac.uk/submission/webin.html. Tento systém umožňuje vkládání informací o sekvencích interaktivní formou, vyplňováním několika www formulářů.

Informace o sekvencích je také možné zasílat pomocí emailu, ale data musí být poskytována ve speciálním formátu. Při vkládání většího množství podobných sekvencí, EBI doporučuje kontaktovat správce databáze, kteří pomohou se vkládáním těchto dat do databáze. [3]

4.4.2 Struktura uložených dat

Databáze je tvořena záznamy o sekvencích nukleotidů. Každý záznam odpovídá jedné sekvenci nukleotidů, která byla vložena do databáze nebo publikována v literatuře. Takový záznam tvoří samotná sekvence a odpovídající informace o ní (anotace sekvence). V některých případech záznam shrnuje informace z více publikovaných článků zabývajících se stejnou sekvencí nukleotidů. Naopak jeden článek může být zdrojem více záznamů v databázi, zejména při porovnávání homologních sekvencí různých organismů.

Třídy dat

Třída každého záznamu je indikována na prvním řádku (ID) záznamu o sekvenci. Všechny distribuované a veřejně přístupné záznamy jsou třídy *standard*. Vnitřně databáze používá i další třídy dat.

Divize databáze

Záznamy, které tvoří databázi, jsou shlukovány do divizí. Shlukování je založeno převážně na taxonomii. Výjimku tvoří divize EST (expressed sequence tags), kam patří sekvence, ke kterým obvykle neexistují téměř žádné doplňující informace a byly získány pomocí jednodušších metod. Tyto sekvence jsou také obvykle považovány za sekvence nižší kvality. Další výjimky tvoří divize STS (sequence tagged site), HTG (HighThroughput Geonome Sequence), HTC (High-Throughput cDNAs), GSS (genome survey sequences) a CON (construct). Do těchto divizí patří záznamy o sekvencích, které byly získány speciálními metodami [3]. Popis jednotlivých divizí je možné nastudovat na http://www.ebi.ac.uk/embl/Documentation/index.html.

Struktura záznamu

Záznamy v databázi jsou strukturovány tak, aby byly snadno zpracovatelné počítačovými programy a zároveň byly srozumitelné pro člověka. Všechny popisy dat, klasifikace i poznámky jsou v běžné angličtině. Jestliže je to možné, jsou použity symboly a označení, které se běžně používají v molekulární biologii.

Každý záznam v databázi obsahuje řádky různého typu. Každý typ řádku má definovaný formát a obsahuje určité informace o sekvenci pomocí specifikovaných typů dat. Na začátku každého řádku záznamu je dvouznakový kód, který určuje typ řádku, a tedy i informace, které tento řádek obsahuje [3]. Několik příkladů kódů řádků je možné vidět v tabulce 4.

ID	identification	begins each entry; 1 per entry
AC	accession number	>=1 per entry
SV	sequence vision	1 per entry
DT	Date	2 per entry
XX	spacer line	many per entry
//	termination line	ends each entry; 1 per entry

Tabulka 4.: Příklad dvouznakových kódů na začátcích řádků

V jednom záznamu se nemusí vyskytovat všechny typy řádků a naopak v jednom záznamu se jeden typ řádku může opakovat několikrát. Každý záznam začíná řádkem ID (identification line) a je ukončen řádkem //. Řádky se vyskytují v záznamu v přesně stanoveném pořadí, s výjimkou řádku XX, který slouží jako oddělovač a může být použit kdekoliv mezi řádky ID a SQ.

Příklad databázového záznamu

```
ID   sekvence187 standard; DNA; UNC; 262 BP.
DE   sekvence187
SQ   Sequence 262 BP;
     CCTGAGATAT AAAAGCGTAT AGAATATTCG AAGTAGTATA GTGTAGCCTG CAATAGCATC        60
     AGTGTAGTTC TCAGCATAGA GAGGCTAAGA TAAATGAAGA TTGTGTCAGC AGCTACAATG       120
     AATGATGGCG GAAAGACCTT TG                                               142
//
```

Formát řádků databázového záznamu

Každý záznam v databázi Každý řádek záznamu obsahuje na začátku dvouznakový kód, který určuje typ řádků. Tento kód je následován třemi mezerami, takže aktuální informace začínají na každém řádku na pozici znaku číslo 6.

Kódem řádků se samotnou sekvencí jsou 2 mezery, proto se zdá, že tyto řádky jsou bez kódu. Sekvence je zapsána po 60 bázi na jeden řádek. Vždy po deseti bázích je vložena jedna mezera. První báze řádku je na šesté pozici. Báze jsou vždy zapsány od konce 5’ ke konci 3’ a pokud je to možné, používá se nekódující řetězec. Na pozicích 73 – 80 každého řádku se nachází číslo báze pro snadnější

orientaci a snadnější vyhledávání oblastí v sekvenci. Čísla jsou zarovnána vpravo a obsahují číslo poslední báze na daném řádku.

4.5 GENBANK

Distribuční formát nukleotidové databáze GENBANK, který je podobný formátu EMBL je lépe čitelný. Místo dvoupísmenného identifikátoru používá celé slovo. Databáze GENBANK je nukleotidová databáze, kterou organizuje Národní institut zdraví (NIH) v USA. Díky výměnné spolupráci s ostatními nukleotidovými databázemi obsahuje v podstatě stejná data jako EMBL.
GENBANK je výborně propojena s mnoha dalšími databázemi. Bohužel, je třeba mít při práci na paměti, že (stejně jako EMBL) neobsahuje všechny dostupné sekvence, hlavně z velkých genomových projektů. Podrobnější informace o databázi je možné nalézt například na internetové adrese http://www.ncbi.nlm.nih.gov/Genbank/GenbankOverview.html

Příklad databázového záznamu

```
LOCUS       sekvence187    262 bp    DNA             UNA       30-Jan-2003
DEFINITION  sekvence187
ORIGIN
        1 CCTGAGATAT AAAAGCGTAT AGAATATTCG AAGTAGTATA GTGTAGCCTG CAATAGCATC
       61 AGTGTAGTTC TCAGCATAGA GAGGCTAAGA TAAATGAAGA TTGTGTCAGC AGCTACAATG
//
```

4.6 Formát dat

Jak je vidět formát ukládání sekvencí a informací o nich se v obou popsaných databázích mírně liší. Aby byly počítačové programy schopné data z databází jednoznačně přečíst, je nutné jim sdělit, v jakém formátu jsou data uložena. Rozlišuje se pak několik formátů.

Fasta

Velmi jednoduchý formát pro ukládání sekvencí. Obsahuje pouze identifikaci sekvence a samotnou sekvenci. Využívá se v situacích, kdy podrobné informace o sekvenci nejsou pro práci s touto sekvencí příliš důležité. Tento formát je vhodný např. pro porovnávání sekvencí apod.

EMBL

Formát určený databází EMBL.

GENBANK

Formát určený databází GENBANK.

5 Porovnávání sekvencí

Jak již jsme si řekli v předchozí kapitole, všechna data jako jsou sekvence nukleotidů a aminokyselin jsou uloženy ve stanoveném formátu v různých celosvětových databázích (EMBL, GENBANK atd.). Abychom pochopili, proč jsou pro nás tato data důležitá, vysvětlíme si pojem homologie. Homologie označuje strukturální podobnost dvou genů nebo celých sekvencí nukleotidů či aminokyselin. Míru strukturální podobnosti je možné určit vzájemným strukturálním porovnáním dvou sekvencí. Podle míry podobnosti se pak stanoví, jestli mezi sekvencemi nějaká homologie je nebo není.

Homologie naznačuje, že se dva porovnávané geny patrně vyvinuly ze společného předchůdce a že zřejmě kódují proteiny s obdobnou funkcí. Proto je pro nás důležitá např. i sekvenovaná DNA kopinatce nebo ježíka tetraodon. Kopinatec patří mezi bezlebečné a je to velmi zajímavý organismus. Má velmi jednoduchou stavbu těla a tato jednoduchá stavba těla odpovídá jednoduché struktuře jeho genů. Přitom jsou tyto geny velmi podobné genům obratlovců, například i genům člověka.

Zkoumání kratšího genomu s jednodušší strukturou genů může vést k rychlejšímu pochopení funkcí genů složitějších organismů. Člověk má přibližně 35 000 genů a DNA dlouhou 3,2 miliardy nukleotidů, avšak vědci zjistili, že více než 95 procent nukleotidů lidské DNA nekóduje vůbec nic. Největší část z této nekódující DNA tvoří opakující se krátké kousky, které do naší DNA vnesly viry již před milióny let. Tím se lidská DNA pro vědce stala poměrně nepřehlednou četbou, a je tedy někdy obtížné identifikovat geny, které jsou pro člověka zásadní. Vraťme se nyní k ježíkovi. Ježík má na rozdíl od člověka jen velmi malé množství DNA, které nic nekóduje. Takže když osekvenujeme ježíka a osekvenujeme člověka, a pak porovnáme, které části jsou stejné, máme docela dobrou nápovědu, že by to mohla být u člověka také kódující sekvence, že by to mohl být gen.

Porovnání dvou sekvencí je jednoduše porovnání shody mezi jednotlivými znaky každé sekvence. Aby toto porovnání mělo reálnou vypovídající hodnotu, musí být sekvence vůči sobě optimálně zarovnány. Jednou z možností, jak posoudit podobnost dvou sekvencí je tzv. metoda vykreslování teček, kdy je možné vizuálně sledovat oblasti stejných nebo podobných úseků porovnávaných sekvencí. Tato metoda je stručně popsána v podkapitole 5.1. Jedná se o jednoduchou metodu, která tvoří výchozí bod pro ostatní složitější a rychlejší metody porovnávání sekvencí. Tyto sofistikované metody budou představeny dále v této kapitole. Tomu ale předchází podkapitola 5.2, věnovaná problematice zarovnání sekvencí, kterou je nezbytné pochopit pro pochopení porovnávacích metod dynamického programování nebo metod rodiny BLAST.

5.1 Vykreslování teček (Dot Plots)

Jedna z nejjednodušších metod pro vyhodnocování podobnosti mezi dvěma sekvencemi je vizualizace regionů podobnosti tzv. vykreslením bodů (dot plots). Při konstrukci této metody se první sekvenci

určené k porovnání přiřadí horizontální osa prostoru pro vykreslení a druhé sekvenci se přiřadí osa vertikální. Tečky se potom vykreslují na místě, kde se jednotlivé úseky obou sekvencí sobě rovnají. Pokud se po sobě shoduje několik úseků sekvence, vznikají v oblasti pro vykreslování viditelné diagonální čáry, složené z vykreslených teček [4].

Pokud se porovnávají velké a velmi podobné sekvence, ve kterých se objevují často se opakující podobné motivy, stane se vykreslování složité, nepřehledné a zašuměné. Efektivní způsob, jak toto vyřešit, je tzv. sliding window, které v jednom okamžiku zohledňuje více než jednu pozici (písmeno) sekvence. Toto šoupající se okno může mít například velikost 10 znaků. Přidejme ještě mez podobnosti, kterou stanovíme na 8.

Potom průběh porovnání proběhne následovně. Prvních deset nukleotidů sekvence *A* se porovná s prvními deseti nukleotidy sekvence *B*, a pokud se alespoň 8 nukleotidů rovná, vykreslí se tečka na pozici (1,1), jinak nikoliv. Poté se okénko na sekvenci *A* posune o jeden nukleotid vpřed, tedy nyní se budou porovnávat nukleotidy 2-11 sekvence *A* s nukleotidy 1-10 sekvence *B*. Toto se opakuje, dokud se všechny desetiznakové podsekvence sekvence *A* neporovnají s podsekvencí 1-10 sekvence *B*. Poté se posune také okénko na sekvenci *B* o jeden znak dopředu, a takto se vše opakuje, dokud se neporovnají desetiznakové sekvence sekvencí *A* a *B* každá s každou. Velikost okna a hodnota meze podobnosti mohou být libovolně měněny v závislosti na podobnosti porovnávaných sekvencí.

5.2 Jednoduché zarovnávání

Zarovnání sekvencí spočívá v nalezení takových úseků sekvencí, které si jsou nejvíce podobné a tyto úseky zarovnat vůči sobě. Uvědomme si, že v průběhu evoluce může v sekvenci nukleotidů každého druhu docházet k různým změnám. Tyto změny mohou být tři:

- Mutace - záměna písmene (nukleotidu) za jiný
- Inserce - vložení (přidání) jednoho nebo více nových nukleotidů
- Delece - odebrání jednoho nebo více nukleotidů

Bylo zjištěno, že inserce a delece se v přírodě vyskytují mnohem méně často než mutace. Protože k inserci a deleci neexistuje žádná homologie, zavádí se pojem mezera, která při zarovnávání sekvencí reflektuje tyto změny.

Pokud zarovnáváme sekvence, kde se neuvažuje inserce a delece, nevyužívá se ani mezer. Zarovnání dvou sekvencí je potom pouze záležitostí výběru startovního bodu pro kratší z porovnávaných sekvencí. Uvažujme následující příklad [1]:

- sekvence A: AATCTATA
- sekvence B: AAGATA

Tyto sekvence mohou být zarovnány pouze třemi způsoby (příklad 1), pokud nebereme v úvahu mezery. Abychom mohli určit, které z těchto 3 zarovnání je optimální, musíme rozhodnout, jak ohodnotit každé zarovnání.

```
AATCTATA    AATCTATA    AATCTATA
AAGATA        AAGATA      AAGATA
```

Příklad 1: tři způsoby zarovnání sekvencí; zdroj [1]

Zatímco metoda vykreslování teček je užitečná pro vizuální prohlédnutí regionů podobnosti dvou sekvencí, číselný skorovací systém pro ohodnocení podobnosti sekvencí má výhody v objektivním určení optimálního zarovnání. V případě, kdy neuvažujeme mezery, je skorovací systém realizován funkcí [1], která určuje součet jednotlivých shod a neshod porovnávaných řetězců.

$$S = \sum_{i=1}^{n} \begin{cases} \text{Skóre shody; } seq1_i = seq2_i \\ \text{Skóre neshody; } seq1_i \neq seq2_i \end{cases} \quad (1)$$

kde *n* je délka delší sekvence. Pro náš případ jsou tedy hodnoty výsledku 4, 1 a 3.
Pokud vezmeme v úvahu případ, že v sekvenci nastaly inserce nebo delece, porovnání se výrazně zkomplikuje. Počet možných zarovnání sekvence značně vzroste. Například dvě sekvence z předchozího příkladu, které bylo možné zarovnat třemi způsoby, je s uvažováním mezer možné zarovnat 28 různými způsoby (příklad 2).

```
AATCTATA    AATCTATA    AATCTATA
AAG-AT-A    AA-G-ATA    AA--GATA
```

Příklad 2: tři možné způsoby zarovnání sekvencí; zdroj [1]

Při ohodnocování porovnání sekvencí, které mohou obsahovat mezery, se zavádí termín gap penalty, který určuje hodnotu penalizace v případě, kdy je znak sekvence zarovnán s mezerou jiné sekvence. Tato hodnota se musí začlenit do skórovacího systému. Funkce, kde *n* je délka delší sekvence, pak vypadá následovně [1]:

$$S = \sum_{i=1}^{n} \begin{cases} \text{Penalizace za mezeru; jestliže } seq1_i = \text{‘–‘ nebo } seq2_i = \text{‘–‘} \\ \text{Skóre shody; } seq1_i = seq2_i \\ \text{Skóre neshody; } seq1_i \neq seq2_i \end{cases} \quad (2)$$

Není vzácné, že při využívání gap penalty se při hledání najde více možností optimálního zarovnání dvou sekvencí. Metoda pro další rozlišení mezi těmito porovnáními je založena na rozlišování mezi takovými sekvencemi, které obsahují mnoho samostatných mezer a takovými, které obsahují delší ucelené úseky mezer. [4] Penalizaci za mezeru (gap penalty) pak rozdělujeme na dvě části:

- origination penalty - penalizace za zahájení nové série mezer vytvořením samostatné mezery mezi plnohodnotnými znaky sekvence.
- length penalty - penalizace, která závisí na počtu (délce) postupně souvisle chybějících znaků. Obecně je menší než origination penalty.

5.3 Substituční tabulka skóre

Podobně jako penalizace za mezeru může být i penalizace za neshodu rozdělena tak, aby hodnoty penalizace více odpovídali pravděpodobnosti změn objevující se v evoluci. Některé substituce znaků se vyskytují v přírodě častěji než jiné.

Uveďme si příklad na bílkovinách. Uvažujme dvě sekvence bílkovin. Jedna z nich má na určitém místě aminokyselinu alanin. Substituce na aminokyselinu jako je valin, bude mít v důsledku její struktury pravděpodobně menší účinek na změnu funkčnosti výsledného proteinu, než substituce na aminokyselinu jako lyzin. Proto při hodnocení zarovnání můžeme chtít ohodnotit pozici, na které je alanin zarovnán s valinem lépe, než pozici, kde je zarovnán alanin s lyzinem. Podobně tomu může být i u nukleotidů v DNA, kdy se zvýhodňuje vzájemná substituce pyrimidinových nebo purinových bází.

Jakmile je určeno skóre porovnání každé možné dvojice nukleotidů, výsledná substituční matice skóre se použije k ohodnocení každé pozice v zarovnání, kde se nevyskytuje mezera. Pro nukleotidové sekvence je tato matice celkem jednoduchá. Například v programu BLAST je využita matice, která shodu oceňuje hodnotou 5 a vše ostatní hodnotou -4. Jiný příklad může být matice, která nepatrně zvýhodňuje substituce nukleotidů, kdy purin (C nebo T) se mění na jiný purin nebo pyrimidin (A nebo G) se mění na jiný pirimidin, viz tabulka 5. Pro aminokyseliny jsou již tyto tabulky složitější.

	A	**C**	**T**	**G**
A	1	0	0	0
C	0	1	0	0
T	0	0	1	0
G	0	0	0	1

	A	**C**	**T**	**G**
A	5	-4	-4	-4
C	-4	5	-4	-4
T	-4	-4	5	-4
G	-4	-4	-4	5

	A	**C**	**T**	**G**
A	1	-5	-5	-1
C	-5	1	-1	-5
T	-5	-1	1	-5
G	-1	-5	-5	1

Tabulky 5: zleva tabulka identity, Blast tabulka, tabulka přechodů [1]

Nejběžnější a zřejmě nejobjektivnější metoda, jak odvodit substituční tabulku skóre, je sledovat poměr zastoupení různých substitucí přímo v přírodě. Jestliže je substituce mezi dvěma konkrétními aminokyselinami pozorována často, potom jsou pozice, ve kterých jsou tyto aminokyseliny takto zarovnány, ohodnoceny příznivěji a naopak. Nejčastěji používané matice pro sekvence aminokyselin jsou PAM (point accepted mutation) nebo BLOSUM [1]

Jakmile je metoda pro ohodnocení podobnosti jednotlivých dvojic nukleotidů dvou sekvencí vybrána, může být vytvořen algoritmus pro hledání nejlepšího zarovnání mezi těmito sekvencemi. Metoda, která se nejvíce nabízí, kompletní hledání všech možných zarovnání, není obecně proveditelná. Představme si dvě sekvence o ne příliš ohromující délce 100 a 95 nukleotidů. Pokud bychom chtěli navrhnout algoritmus, který by měl spočítat a ohodnotit všechna možná zarovnání, musel by testovat přibližně 55 milionů možností.

Následující podkapitola představí metody založené na technikách dynamického programování, které jsou schopné najít optimální zarovnání mezi sekvencemi v přijatelném čase. Porovnání sekvencí a stanovení podobnosti (homologie) u těchto metod záleží na zhodnocení výsledného zarovnání.

5.4 Dynamické programování

Jak roste délka sekvencí, roste i počet možností, které není možné již velmi brzy testovat v rozumném čase. Tento problém je možné řešit tzv. dynamickým programováním, což je metoda, která řešený problém rozloží na podproblémy o přijatelné složitosti a výsledky jednotlivých podproblémů použije k výpočtu konečného výsledku. S.Needleman a C.Wunsch byli první, kdo použili přístup dynamického programování na řešení problému zarovnání sekvencí. Jejich algoritmus, který je podobný tomu, který bude představen níže, je jedním ze základních kamenů bioinformatiky.

Klíč k pochopení přístupu dynamického programování k zarovnání sekvencí spočívá v tom, jak je problém rozložen na podproblémy. Uvažujme dvě sekvence CACGA a CGA. Pro jednoduchost budeme uvažovat jednotné penalizace za mezeru a za neshodu. Na první pozici máme při zarovnávání tři možnosti:

- můžeme vložit mezeru na první místo prvního řetězce (v našem případě nesmyslné)
- umístit mezeru na první místo v druhé sekvenci
- neumisťovat mezeru nikam

V prvních dvou případech obdržíme penalizaci za vložení mezery, v posledním případě dostaneme jako výsledek shodu. Nyní hodnota skóre závisí na tom, jak zarovnáme zbytek sekvencí. Pokud bychom předem znaly skóre nejlepšího zarovnání, kterého je možné se zbytky sekvencí dosáhnout, mohli bychom okamžitě spočítat celkové skóre. Vezměme v úvahu třetí možnost, tedy nevložení žádné mezery. Měli bychom spočítat skóre pro zarovnání sekvencí ACGA a GA.

Metoda dynamického programování nabízí tabulku pro ukládání částečných skóre zarovnání, která nám pomohou při výpočtu. Výpočet optimálního zarovnání pak spočívá ve vyplňování tabulky částečného skóre, dokud skóre pro celkové zarovnání není spočítáno. V tabulce 6 jsou vertikální a horizontální osy označeny písmeny porovnávaných sekvencí. Zarovnání dvou sekvencí je ekvivalentní s cestou z levého horního rohu do pravého dolního rohu. Horizontální pohyb představuje mezeru v sekvenci podél levé vertikální osy a vertikální pohyb reprezentuje mezeru v sekvenci podél horní horizontální osy. Diagonální pohyb označuje zarovnání nukleotidů z každé sekvence. První řádek a první sloupec jsou inicializovány hodnotami určující pokutu za vložení mezery. Samotné vyplňování tabulky tedy začíná na pozici (2,2). Vzpomeňme si nyní na tři možnosti zarovnání na první pozici. Od nich se odvíjí možnosti, které se nabízí při vyplňování této tabulky. Pro pozici (2,2) máme možnosti:

- můžeme vzít hodnotu vlevo na pozici (2,1) a k ní přičíst pokutu za vložení mezery do sekvence podél levé osy tabulky

- můžeme vzít hodnotu nahoře na pozici (1,2) a k ní přičíst pokutu za vložení mezery do sekvence podél horní osy tabulky
- můžeme vzít hodnotu po diagonále o jednu pozici blíže k levému hornímu rohu (1,1) a přidat k ní hodnotu porovnání nukleotidů (pro neshodu 0, pro shodu 1)

		A	**C**	**T**	**C**	**G**
	0	-1	-2	-3	-4	-5
A	-1	1				
C	-2					
A	-3					
G	-4					
T	-5					
A	-6					
G	-7					

Tabulka 6:Tabulka částečného skóre; zdroj [1]

Z těchto možností vybereme takovou, kde hodnota skóre je největší. Konkrétně pro pozici (2,2) máme na výběr: z hodnot -2, -2 a 1. Proto se na tuto pozici zapíše hodnota 1. Takovýto postup provedeme pro všechny zbývající prázdné pozice tabulky. Z výsledné tabulky (tabulka 7) pak lehce zjistíme optimální zarovnání zpětným sledováním cesty z pravého dolního rohu, kde je hodnota 2. Sledování probíhá tak, že ze současné pozice přejdeme na takovou pozici, ze které bylo možné spočítat ohodnocení na stávající pozici. Cesta je v tabulce 4 vyznačena červeně.

		A	**C**	**T**	**C**	**G**
	0	-1	-2	-3	-4	-5
A	-1	**1**	0	-1	-2	-3
C	-2	0	**2**	1	0	-1
A	-3	-1	**1**	2	1	0
G	-4	-2	**0**	1	2	2
T	-5	-3	-1	**1**	1	2
A	-6	-4	-2	0	**1**	1
G	-7	-5	-3	-1	0	**2**

Tabulka 7: Výsledná tabulka částečného skóre; zdroj [1]

Globální versus lokální porovnání

Základní zarovnávací algoritmus, jak jsme si ho ukázali dříve, prováděl právě globální zarovnávání. To znamená, že zarovnával dvě sekvence jako celky. Penalizace za mezeru je udělena i navzdory tomu, že nevíme, kde byla mezera umístěna. Zda byla někde uvnitř sekvence nebo na konci jedné nebo obou sekvencí. To není vždy nejžádanější způsob, jak porovnávat dvě sekvence. Může nastat

situace, kdy hledáme krátký úsek v celém genomu. Představme si, že chceme v sekvenci AAACACGTGTCT nalézt sekvenci ACGT. Bude nás tedy zajímat toto zarovnání:

```
AAACACGTGTCT
----ACGT----
```

Toto zarovnání jednoznačně ukazuje, že kratší sekvence je podsekvencí delší sekvence. Okrajové mezery jsou většinou důsledkem nekompletních dat, což nemá žádný biologický význam. Je proto dobré s těmito mezerami nakládat jinak než s mezerami uvnitř sekvence. Takový přístup se někdy nazývá pologlobální zarovnání. Takové zarovnání můžeme provést dříve popsaným algoritmem dynamického programování s využitím několika změn. Připomeňme si, že vertikální pohyb v tabulce je ekvivalentní s mezerou v sekvenci umístěné na horní horizontální ose. Takže by mělo být jasné, že vertikálním pohybem až na spodek tabulky a poté horizontálním pohybem až na pravý konec tabulky získáme následující zarovnání pro sekvence z tabulky 4.

```
-------ACTCG
ACAGTAG-----
```

Každý pohyb z levého horního rohu směrem dolů přidává na začátek jedné sekvence mezery. Protože každá mezera penalizuje hodnotu celkového skóre zarovnání, inicializuje se první řádek tabulky postupnou penalizací. My ale chceme, aby úvodní mezery nebyly penalizovány. Proto změníme inicializaci prvního sloupce na nuly. Ze stejného důvodu (ale pro druhou sekvenci) provedeme tuto inicializaci i v prvním řádku. Abychom umožnili vkládat mezery na konec sekvence, aniž by se penalizovalo skóre zarovnání, musíme mírně pozměnit výklad tabulky. Tím, že inicializujeme první řádek a sloupec této tabulky částečného skóre na nuly a dovolíme volný horizontální a vertikální pohyb v posledním řádku a sloupci v tomto pořadí, dostaneme modifikovaný Needleman a Wunsch algoritmus, který vyhledává pologlobálních zarovnávání.

Někdy ale ani semiglobální zarovnání nevyhovuje požadované flexibilitě. Představme si, že máme dlouhou sekvenci DNA a rádi bychom našli nějaké její podsekvence, které jsou podobné nějaké části genomu kvasinky. Pro tento druh porovnávání semiglobální porovnání nestačí, protože každé zarovnání bude penalizováno pro každou nesouhlasnou pozici. Dokonce, i když existuje zajímavá podsekvence, která se shoduje s částí genomu kvasinky, všechny neshodující se oblasti zajistí, že výsledné skóre nebude stát za nic. Vhodným nástrojem pro tento druh úlohy je lokální zarovnání, které najde nejlépe se shodující podsekvenci dvou porovnávaných sekvencí.

Uveďme si opět příklad. Mějme dvě sekvence AACCTATAGCT s GCGATATA. Podívejme se nejprve, jak by si s tímto příkladem poradil algoritmus pro semiglobal zarovnání.

```
AAC-CTATAGCT
-GCGATATA---
```

Už při první pohledu je patrné, že toto zarovnání je poměrně chabé. Čtyři z prvních pěti pozic jsou neshody nebo mezery, stejně jako poslední tři. Přesto ale tento algoritmus odhalil, že uprostřed se nachází podřetězec TATA, společný oběma sekvencím. S drobnou úpravou může být tato metoda

použita k identifikaci podsekvencí, tím že se budou ignorovat neshody a mezery před a za oblastí, která se shoduje. Výsledný algoritmus poprvé představili F. Smith a M. Waterman v roce 1981 a dnes je základní technikou v bioinformatice. Abychom mohli provést lokální zarovnání, provedeme drobnou změnu při vyplňování tabulky částečného skóre. Konkrétně, nulou nahradíme takové hodnoty, kde všechny ostatní metody spočítaly hodnotu skóre menší než nula. [4]

Dynamické programovací metody pro globální zarovnání dovolují dosáhnout optimálního zarovnání v čase, který je úměrný délkám dvou sekvencí, které jsou zarovnávány. Když jsou tedy aplikované na celou databázi, výpočetní čas roste lineárně s velikostí databáze. Se současnými rozsáhlými databázemi je porovnávání pomocí dynamického programování samostatně s každou sekvencí databáze příliš pomalé. To vedlo k vývoji BLAST a Fasta programů, které umožňují zarovnat jednu sekvenci proti všem sekvencím databáze v rozumném čase. Zároveň dovedou podle určitých statistických výpočtů zhodnotit významnost podobnosti podle vytvořeného zarovnání, tedy provést jakési porovnání a stanovit zda by mezi porovnávaným sekvencemi mohla existovat homologie.

Programy z rodiny BLAST a Fasta jsou založené na heuristice, kdy se výpočetní čas redukuje tím, že se obětuje jistá citlivost algoritmu. Ten totiž již během zarovnávání vyřazuje sekvence, které se na první pohled zdají být nevhodné, a vybírá pouze ty sekvence, které s hledanou sekvencí sdílí významnou podobnost, stanovenou nalezením podobných oblastí uvnitř sekvence. Tyto výběrové kroky umožňují omezit časově náročnou metodu porovnávání sekvencí pouze na podmnožinu sekvencí databáze a omezit hledání pouze na podregiony sekvencí. Za účelem rychlosti odhadují podobnost mezi sekvencemi přibližným způsobem, a tak riskují ztrátu a eliminaci sekvencí, jejichž podobnost lze s hledanou sekvencí obtížněji detekovat, nebo ukončení algoritmu s výsledkem, který není optimální. Následující kapitola se těmto algoritmům věnuje podrobněji.

6 Algoritmy využívající heuristik

V této kapitole se seznámíme s algoritmy, které pro porovnání sekvencí využívají různé heuristiky za účelem toto porovnání co nejvíce urychlit. Minulá kapitola již naznačila, že mezi tyto algoritmy patří například algoritmy z rodiny BLAST a algoritmus Fasta. Než se podrobněji pustíme do rozboru jejich algoritmů, stanovíme si základní termíny. Nechť A = {A, C, G, T, U} je abeceda znaků sekvencí. Nechť D je sekvence databáze nad abecedou A a Q je sekvence dotazu nad abecedou A. Toto označení bude použito ve všech následujících popisech algoritmů.

6.1 Senzitivita, selektivita a významnost

Hlavním důvodem pro použití heuristik je co nejvíce urychlit vzájemné porovnání dvou sekvencí. Cena za toto urychlení je jistá ztráta citlivosti (senzitivity). Míra citlivosti velmi závisí i na nastavení startovních parametrů algoritmu. Senzitivitu metody můžeme definovat jako její schopnost detekovat všechny významné podobnosti, a tak generovat výsledek s co nejvíce odpovídajícími sekvencemi. Analogicky je možné definovat selektivitu jako schopnost vybrat pouze sekvence považované za významně podobné, a tak generovat výsledek s co nejmenším počtem odpovídajících sekvencí.

Senzitivita a selektivita se tedy výrazně vztahují k pojmu významnosti (důležitosti) výsledku. Biologická významnost podobnosti mezi dvěma sekvencemi je představa (dojem), která závisí na biologickém kontextu, ve kterém je hledání prováděno, a tak je velice těžké a pravděpodobně nemožné odhadnout ji přímo ze skóre získaným porovnáním.

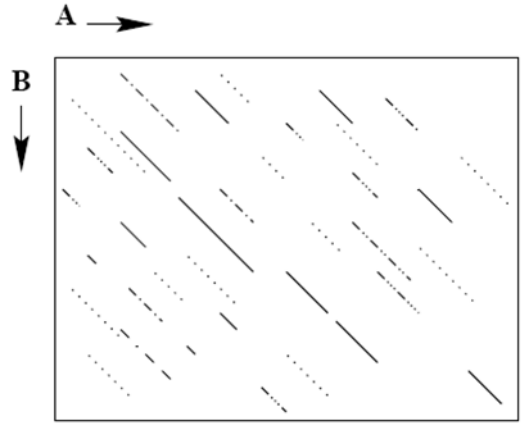
Programy nám stanoví míru statistické významnosti výsledku s ohledem na náhodný model. Výsledek bude považován za statisticky významný, jestliže pravděpodobnost jeho získání pomocí náhody je velmi malá. Skórovací systém a algoritmy, které se využívají, jsou vyvíjené s tím, že vyhledávaná podobnost je biologicky smysluplná a užitečná podobnost, a tak tyto výsledky, které jsou statisticky nejvíce významné, jsou zároveň také ty, které jsou nejvíce významné z hlediska biologického. Je ale důležité mít na paměti, že zarovnáním můžeme dostat skóre, které je statisticky významné, ale nemusí být významné z hlediska biologie. Naopak málo statisticky významné výsledky mohou korespondovat se zajímavými biologickými rysy. Detekovat a interpretovat výsledky nízké statistické významnosti je samozřejmě obtížná věc.

6.2 Fasta algoritmus

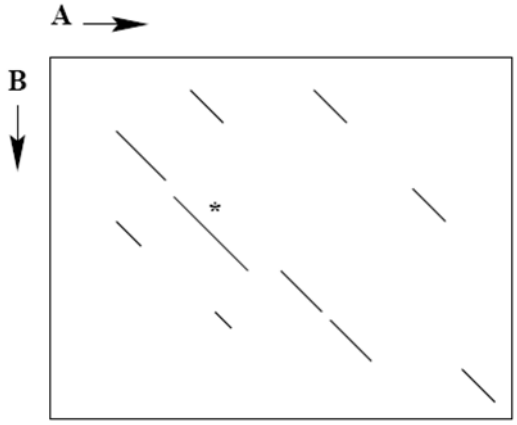
Algoritmus fasta sestává ze 4 kroků, které jsou znázorněny na obrázku 7. Je založen na diagonální metodě vyvinuté v roce 1983.

Identifikace podobnosti regionů

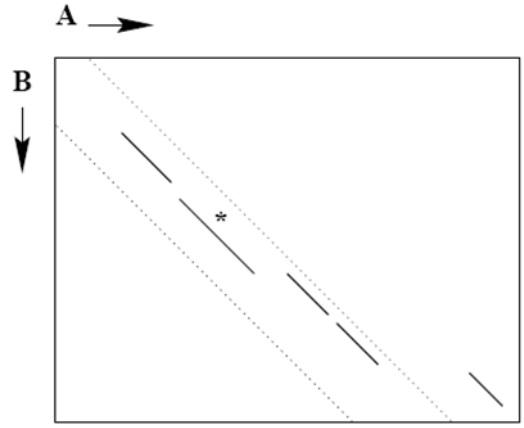
První krok spočívá v lokalizaci deseti nejlepších regionů podobnosti mezi sekvencí dotazu *Q* a každou sekvencí databáze. Proces popsaný níže je při porovnávání sekvencí databáze se sekvencí dotazu *Q* aplikován na každou sekvenci databáze *D*. V algoritmu se využívá termín „k-tice“, což je slovo nebo úsek sekvence o délce *k*. Všechny k-tice sekvence *Q* (získané postupným posouváním okna o délce *k* podél sekvence) jsou porovnány se všemi sekvencemi databáze a rovnosti detekované mezi k-ticemi jsou uloženy. Metoda může být reprezentována v dvoudimenzionální matici, kde každá shoda mezi dvěmi k-ticemi bude znázorněna jako tečka. Každý diagonální úsek odpovídá porovnání regionů dvou sekvencí bez mezer, jak se postupně k-tice shodují nebo neshodují.



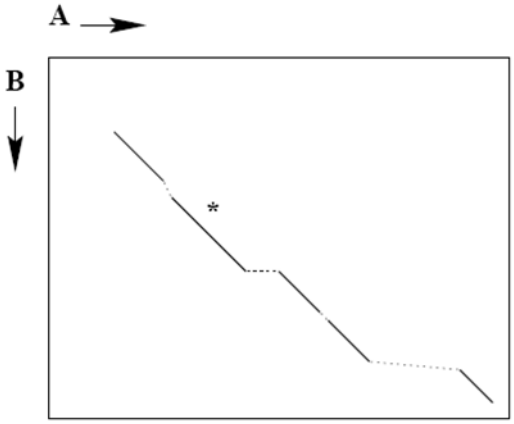
A: Určení všech shod k-tic



B: Přehodnocení 10 nejlepších regionů tabulky skór. Stanoveni init1 skóre



C: Provedení spojovací procedury. Stanovení initn skóre



D: Omezení DP. Stanovení opt skóre

Obrázek 7: Postup algoritmu Fasta. zdroj:[4]

Program nesestavuje matici, ale využívá více efektivní výpočetní struktury známé jako lookup table za účelem uložit všechny dvojice shodných k-tic. Na diagonálách jsou regiony postupných shod a

neshod k-tic. Tyto regiony jsou ohodnoceny, kde hodnota je výsledek jednoduché funkce určující příslušný počet shod a neshod v daném regionu. Deset regionů, obsahující nejvyšší hustotu shod k-tic, je vybráno pro další krok.

Tento krok je velice důležitý v souvislosti s citlivostí této metody. Pouze regiony, které jsou vybrány v tomto kroku, budou uvažovány v kroku dalším. Také hodnota zvolená pro parametr *k* je důležitá. Menší *k* je lepší pro větší citlivost, například pokud k = 1, pak se do výpočtu zahrnou všechny shody mezi jednotlivými nukleotidy (popř. aminokyselinami). Větší *k* znamená,, že celý program bude rychlejší, protože v dalším kroku bude uvažováno méně regionů. Volba *k* se tedy projeví na citlivosti a rychlosti.

Přehodnocení skóre deseti nejlepších regionů

Ve druhém kroku jsou regiony, vybrané´v předešlém kroku, přehodnoceny použitím tzv. matice podobnosti a je stanoven podregion, jehož skóre podobnosti je maximální (nemůže být zvýšeno zkrácením nebo rozšířením regionů). Skóre nejlepšího z deseti regionů je nazýváno init1 skóre.
Vzhledem k tomu, že výpočet těchto skóre podobností je proveden pouze na deseti regionech vybraných v prvním kroku (použitím hustoty shod k-tic), může se stát, že těchto deset regionů nebude deset nejpodobnějších regionů mezi sekvencemi.

Výběr „více podobných“ sekvencí

Starý Fastp program používal toto init1 skóre již přímo k ohodnocení sekvencí databáze. Fasta se ještě pokouší (pokud je to možné) spojit některé z vybraných regionů. Pro spojovací proces jsou uvažovány pouze regiony mající skóre nad stanovený práh. Kombinace kompatibilních regionů je ohodnocena součtem skóre regionů, které jsou ve spojení obsaženy a odečtením penále za úseky „spojení“, tj úsek mezer. Toto penále je analogií k penalizaci za mezery. Fasta využívá pro výpočet nejlepší kombinace regionů metodu dynamického programování. Ohodnocení nejlepší kombinace se nazývá initn skóre. Všechny sekvence databáze jsou zpracovány těmito třemi kroky a každá tak získá ohodnocení initn skóre. Potom jsou pro další zpracování vybrány pouze sekvence s vyšším initn skóre, ostatní jsou eliminovány (se současnou implementací nevstoupí do čtvrtého kroku pouze 10 procent sekvencí databáze).

Zarovnání vybraných sekvencí

Každá z vybraných sekvencí je porovnána s referenční sekvencí, použitím algoritmu, který je variací Smith-Watermanova algoritmu; lokální porovnání je propočteno metodou dynamického programování. Je ale omezeno na oblast kolem diagonálního regionu, který má nejlepší init1 skóre (tento algoritmus je nazýván „banded SW“). Skóre vypočítaného zarovnání je nazýváno opt skóre a je to právě to, které je používáno k ohodnocení zarovnání. Protože proces dynamického programování je

aplikován pouze na vybrané pásmo (jejichž šířka je parametr programu) uvnitř celé matice porovnání, může to vést k zarovnání, které je pouze suboptimální.

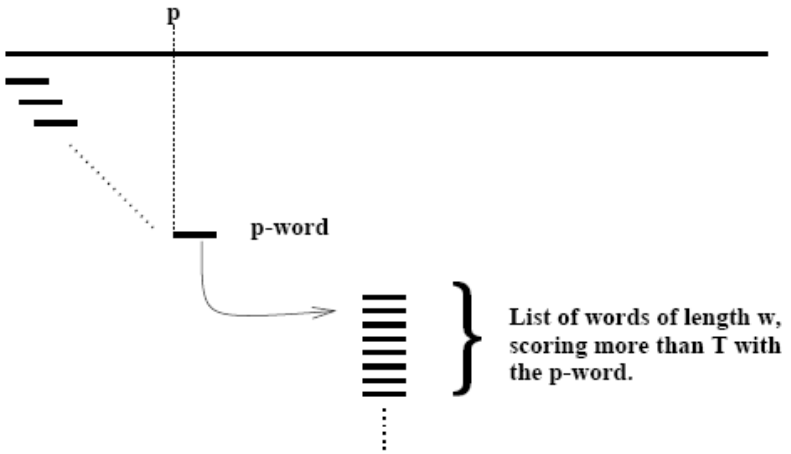
6.3 BLAST programy

Rodina programů BLAST se zrodila počátkem devadesátých let minulého století s vývojem programu Blast1, který je velice rychlý a zaměřený na hledání regionů lokálních podobností bez mezer. Je založen na principu nalezení tzv. semínek (seeds), výskytu krátkých řetězců a hledání podobnosti algoritmem dynamického programování jenom v jejich blízkosti.

V letech 1996 - 1997 se objevily dvě nové verze programu Blast, které se navzájem liší jen velmi málo, ale na rozdíl od Blast1 berou obě v úvahu vkládání mezer. Obě byly nazvané Blast2 a můžeme je rozlišit podle použitých iniciál institucí, do kterých patřili jejich autoři: NCBI-Blast2 a WU-Blast2 (National center fot biotechnology information a Washington University obě umístěny v USA). Nyní se detailněji podívejme na algoritmus Blast1, který je dle [4] složen ze čtyř kroků. První tři kroky jsou postupně zobrazeny na obrázku 8, 9 a 10.

Předzpracování dotazu

Jako u Fasta je cílem prvního kroku co nejrychleji lokalizovat podobné regiony bez mezer mezi referenční sekvencí a sekvencemi databáze. Všechna slova o délce *w* tzv. w-tice (nazvaná podle autora Blastu) sekvence dotazu *Q* jsou porovnány se slovy sekvencí z databáze. Nicméně, oproti Fasta, jsou v prvním kroku brány v potaz i podobnosti na úrovni jednotlivých aminokyselin.



Obrázek 8: Pro každou pozici p referenční sekvence nalezneme seznam slov délky w, která mají skóre větší než hodnota T, když jsou zarovnány s podřetězcem začínajícím na pozici p; zdroj [4]

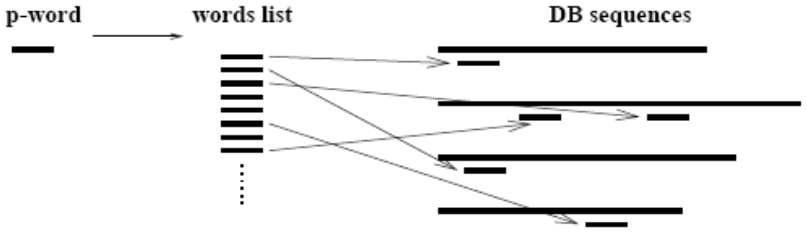
Určitý způsob jak toho docílit, je porovnat každé slovo sekvence *Q* s každým slovem sekvencí z databáze a spočítat skóre podobnosti pro každý pár slov (součtem skóre spárovaných nukleotidů,

které jsou částí spárovaného slova). Potom se musí na základě prahu provést rozhodnutí, že všechny páry slov se skóre větším nebo rovný tomuto prahu jsou považovány za páry podobných (stejných) slov. Je tak možné získat všechna slova v databázi, která jsou podobná s každým slovem sekvence dotazu *Q*.

Blast však používá efektivnější způsoby, jak toho dosáhnout, aby výsledek byl úplně stejný. Nejdříve jsou vygenerovaná všechna slova nad abecedou *A* o délce *w* (pokud $w = 4$, pak existuje 4^4 tedy 256 možných slov). Potom se postupně každé takto vygenerované slovo o délce *w* porovnává s každým podřetězcem (také o délce *w*) sekvence *Q*. Zároveň je nastaven práh T pro podobnost mezi slovy. Každá pozice sekvence dotazu *Q* je asociována se seznamem slov, které získaly porovnáním se slovem sekvence *Q* skóre větší, než práh T. Podobná slova jsou také nazývána „sousedé".

Generování zásahů (hits)

Po prvním kroku je nyní *Q* reprezentovaná seznamy slov (tzv. seznamy sousedů). Jeden seznam na každou pozici sekvence *Q*. V tomto kroku je každé slovo ze seznamu sousedů každé pozice sekvence *Q* porovnáno s každým podřetězcem sekvence *D*, a jestliže jedno ze sousedních slov na dané pozici sekvence *Q* je identické slovu z *D*, je zaznamenána shoda tzv. zásah (obrázek 8B). Tedy pro každé slovo ze seznamu sousedů stanovíme všechny konkrétní zásahy do sekvence *D*. Někdy se těmto zásahům říká semínka (seeds). Zásahy jsou tvořeny jedním nebo několika následnými (překrývajícími se) páry podobných slov a jsou charakterizovány jejich pozicí v každé ze dvou sekvencí. Všechny existující shody mezi sekvencí *Q* a sekvencí *D* jsou počítány tímto způsobem.



Obrázek 9: Pro každé slovo listu stanovíme všechny konkrétní shody se sekvencemi databáze; zdroj:[4]

Prodloužení zásahů

Každá shoda, která byla generována v minulém kroku, je v tomto kroku bez mezer rozšířena za účelem stanovit, zdali tato shoda může být částí většího úseku podobnosti. Každá shoda je rozšiřována v obou směrech. Rozšiřování je okamžitě zastaveno, jakmile skóre rozšířené shody klesne o více než hodnotu *X* (*X* je parametr programu) v porovnání s nejlepší dosaženou hodnotou skóre během tohoto procesu rozšiřování. Výsledkem jednoho rozšíření je jeden rozšířený zásah s nejlepší hodnotou skóre, které při procesu rozšiřování dosáhl. Všimněme si, že tento způsob hledání nejlépe ohodnoceného párového segmentu obsahující shodu je pouze přibližný a nezaručuje, že

výsledný pár podobnosti je nejlepší. Je to způsobeno požadavkem zahodit vše, co se liší více než o *X* (včetně úseku, který způsobil pokles skóre o hodnotu *X*).
Každá rozšířená segmentová dvojice, která má skóre stejné nebo lepší než S (parametr programu) je uchována a nazvaná jako HSP (High scoring Segment Pair). HSP lze chápat jako úsek stejné délky dvou sekvencí se skóre, které už nelze zlepšit prodloužením. Nejlépe ohodnocená HSP se nazývá MSP (Maximal Segment Pair) [4].

Obrázek 10: Pro každou shodu slov (zásah) rozšíříme zarovnání do obou směrů bez mezer. Zastavíme se, když S klesne více než o hodnotu X vůči nejvyšší dosažené hodnotě S; zdroj [4]

Spojení rozšířených zásahů

V tomto kroku je cílem vhodně spojit HSP segmenty do jedné zarovnané dvojice sekvencí. Není to lehký úkol, neboť se mezi HSP segmenty mohou stále vyskytovat takové, které do výsledného zarovnání nepatří. Je proto nutné je správně určit, respektive správně vybrat pouze ty, které jsou součástí výsledného nejlépe ohodnoceného zarovnání.

6.4 Varianty algoritmu BLAST

Algoritmus BLAST se neustále vyvíjí a zdokonaluje. Zkoumají se možné způsoby, jak BLAST ještě více urychlit a přitom se klade důraz na to, aby co nejvíce vyhovoval potřebám biologů. Na druhou stranu, je zřejmé, že není možné postihnout všechny potřeby biologů jedním algoritmem. Z původního algoritmu BLAST vznikla celá řada variant. Následující přehled, čerpající z [11], stručně shrnuje pomocí různého členění nejznámější varianty algoritmu BLAST

6.4.1 Členění algoritmu podle toho, jaké porovnává sekvence

Nukleotid - nukleotid BLAST

Porovnává nukleotidovou sekvenci s databází nukleotidových sekvencí. Do této třídy se řadí algoritmy blastn, megablast, discountiguous megablast.

Protein – protein BLAST

Porovnává sekvenci aminokyselin s databází sekvencí aminokyselin. Do této třídy se řadí algoritmy blastp, psi-blast, phi-blast.

BLASTX

Porovnává přeloženou nukleotidovou sekvenci na sekvenci aminokyselin proti aminokyselinovým sekvencím databáze.

TBLASTN

Porovnává sekvenci aminokyselin s přeloženými sekvencemi nukleotidů dané databáze na sekvence aminokyselin.

TBLASTX

Porovnává přeloženou nukleotidovou sekvenci na sekvenci aminokyselin s přeloženými sekvencemi nukleotidů dané databáze na sekvence aminokyselin. Je to jeden z nejpomalejších algoritmů rodiny BLAST. Využívá se pro odhalení velmi vzdálených vztahů mezi sekvencemi nukleotidů.

6.4.2 Varianty algoritmu BLAST zaměřené na speciální oblast

Existuje také celá řada ještě specializovanějších Variant, které se soustředí opravdu pouze na jednu oblast hledání.

IgBLAST

BLAST specializovaný na porovnávání imunoglobulinu

CDS (conserved domains in sequence)

Každý řetězec aminokyselin je možné pomyslně rozdělit na funkční oblasti, domény. Právě s těmito doménami pracuje tato varianta algoritmu BLAST.

6.4.3 Algoritmy vycházející z původního algoritmu BLAST

PSI-BLAST (Position-Specific Iterative BLAST)

Jeden ze současných algoritmů rodiny BLAST. Využívá se pro hledání vzdálených vztahů mezi proteiny. Nejprve je vygenerován seznam blízce spřízněných proteinů. Poté jsou tyto proteiny zkombinovány do „profilu“, což je jakási průměrná sekvence. Tento profil se potom použije jako sekvence dotazu při porovnávání s databází proteinů. Při tom se najde velké množství proteinů, ze kterých se vytvoří nový profil a celý proces se pak opakuje s tímto novým profilem [11].

MegaBLAST (Large numbers of query sequences)

MegaBLAST využívá poznatku, že porovnávání velkého množství vstupních sekvencí je mnohem rychlejší než spouštět program BLAST několikrát po sobě. Několik vstupních sekvencí se před prohledáním databáze konkatenuje do jednoho velkého řetězce a ten se porovnává s databází. Potom se provede post-analýza výsledku, aby se shromáždila všechna samostatná zarovnání a jejich statistické hodnoty [11].

BLAT (BLAST-like alignment tool)

Program BLAT napsal Jim Kent. Stejně jako většina jeho programů, je i BLAT volně dostupný pro akademické, osobní a neziskové účely na jeho osobních stránkách http://www.soe.ucsc.edu/~kent. Je navržen tak, aby co nejrychleji našel DNA sekvence s nejméně 95% podobností délky 40 a více bází. U proteinů najde sekvence s nejméně 80% podobností o délce minimálně 20 aminokyselin. V praxi pracuje DNA BLAT dobře s DNA sekvencemi primátů a protein BLAT se sekvencemi suchozemských obratlovců.

DNA BLAT pracuje na principu uchovávání indexů celého genomu v paměti počítače. Index sestává ze všech nepřekrývajících se 11-tic. Vlastní genom v paměti není. Index zabere přibližně gigabyte paměti RAM. Index je použit pro hledání oblastí pravděpodobných homologií, které jsou poté vloženy do paměti pro provedení detailnějšího zarovnání. Protein BLAT pracuje stejným způsobem, pouze místo 11-tic používá 4-tice. Index proteinů zabere v paměti něco přes 2 GB [5].

Některé prameny [12] tvrdí, že BLAT je v porovnání se současnými existujícími nástroji pro porovnávání asi 500 krát rychlejší pří práci s DNA sekvencemi a asi 50 krát rychlejší při práci se sekvencemi proteinů s nastavením sensitivity, které se typicky používá při práci se sekvencemi obratlovců.

ORACLE BLAST

Oracle v rámci Oracle Database 10g implementuje varianty algoritmů BLASTN, BLASTP, BLASTX, TBLASTN a TBLASTX Oracle rozšířil RDBMS o podporu biologické domény a integroval tam ODM BLAST, který umožňuje analytikům využít jazyk SQL ke spuštění BLAST funkcí, které jsou součástí databáze Oracle [9].

NCBI-BLAST

Program NCBI-Blast2 byl vyvinut na NCBI (National Center for Biotechnology Information) [11] a jeho implementace byla poprvé publikována v roce 1997. První dva kroky, vedoucí ke generování primárních shod, jsou stejné jako u původního algoritmu BLAST popsaný v podkapitole 6.3. První důležitý rozdíl se týká výběru shod, které se mají rozšiřovat (prodlužovat). Záměr autorů je generovat lokální *mezerové* porovnání ze zjištěných shod, což vyžaduje přídavný krok (a čas) pro výpočet

zarovnání s mezerami. Na druhou stranu, shody, které nemohou být spojené skrze bezmezerové rozšíření, mohou být částí stejného porovnání s mezerami, a to nemusí být nezbytné rozšiřovat všechny z nich.

Za účelem ušetřit čas, vybrali méně shod pro další bezmezerové rozšiřování využívající podmínky pro rozšíření shody. Požadují, aby na stejné diagonále byla další shoda ve vzdálenosti menší než A (parametr programu, může být změněn uživatelem) Protože tento nový požadavek dělá program méně citlivým (ne všechny shody uspokojí tuto novou podmínku), doporučují použití menší hodnoty pro parametr T (práh pro podobnost mezi slovy používanými v prvním kroku, když byl generován seznam sousedů), v zájmu generovat více shod ve druhém kroku. Samozřejmě tyto dva parametry (A a T) neovlivňují citlivost na stejné úrovni.

Všechny tyto shody, které plně splňují tyto požadavky, jsou vybrány pro bezmezerové rozšíření (jak se dělá ve třetím kroku programu BLAST). Mezi generovanými HSP, ty, které mají vyšší skóre než práh, spustí rozšiřování s mezerami. Jsou použité jako startovní bod pro provádění lokálního porovnání dynamickým programováním. Algoritmus, který je použit pro výpočet tohoto lokálního porovnání s mezerami je modifikace Smith-Waterman algoritmu. Matice je prohledávána oběma směry začínající ze středního bodu nejvýše ohodnocené 11-tice v rámci HSP. Hledání optimální cesty je omezeno na takové buňky matice, u kterých skóre zarovnání neklesne více než o X_g vůči maximálnímu dosaženému skóre od začátku rozšiřování (stejná myšlenka jako v případě bezmezerového rozšiřování)

WU-BLAST (Washington University BLAST)

WU BLAST [10] je rozšířením originální verze algoritmu BLAST. Byl vyvinut pod vedením Warren Gish z Washingtonské Univerzity. V současnosti existuje WU-BLAST ve verzi 2.2. Jedná se o balíček programů BLAST pro identifikaci proteinů a genů, využívající podobnostního prohledávání databází sekvencí nukleotidů nebo proteinů. Balíček obsahuje programy blastp, blastn, blastx, tblastn a tblastx. WU-BLAST byl vypuštěn v roce 1996, jeden rok před NCBI-BLAST. Jeho algoritmus nebyl nikdy zveřejněn a následující popis čerpá pouze z dostupné dokumentace.

WU-BLAST program nabízí několik parametrů umožňující emulaci různých variant BLAST. Jeho standardní chování je takové, že 3 první kroky jsou stejné jako u původního BLAST. Ve čtvrtém kroku spouští u HSP, mající skóre větší než práh, prodlužování slov s mezerami. Algoritmus použitý pro rozšíření s mezerami je směsice „banded (to co dělá fasta)“ a „score-limited“ (jako u NCBI-BLAST) algoritmů dynamického programování. Jestliže se vybere volba „nogap“ tento poslední krok se neprovede a WU-BLAST se stane identický s původním BLAST. WU-BLAST také nabízí volbu zvanou „hitdist“, která, když je aktivována, dovolí zahrnout do výpočtu stejné „two – hits“ požadavky jako u NCBI-BLAST ve třetím kroku.

6.5 Statistika výsledků algoritmu BLAST

Aby bylo možné podle výsledku zarovnání snáze usuzovat, zda se mezi porovnávanými sekvencemi nachází homologie, zavádí se proměnné, které mohou o výsledku leccos napovědět ze statistického hlediska. V následující části si některé proměnné, jak je uvádí [16], stručně popíšeme.

S-hodnota

Tato hodnota určuje hrubé skóre zarovnání. Je to prakticky pouhý součet všech hodnot jednotlivých shod a neshod. Čím větší *S*-hodnota, tím lepší zarovnání. Protože ale hodnoty shody a neshody jsou parametrem programu v rámci substituční matice, může se tato S-hodnota v závislosti právě na těchto hodnotách výrazně měnit.

Bit skóre

Hrubé skóre má velmi slabý význam pro toho, kdo nemá detailněji znalosti o použitém skórovacím systému. Proto se hrubé skóre normalizuje a tuto hodnotu právě vyjadřuje tzv. Bit skóre. Při normalizaci *S*-hodnoty se zohledňuje skórovací systém, tudíž by mělo být Bit skóre pro dvě sekvence s využitím různých substitučních matic vždy stejné. To umožňuje porovnávat vzájemně Bit skóre různých zarovnání. Pro výpočet Bit skóre se uvádí následující vzorec:

$$S' = \frac{\lambda S - lnK}{ln2} \tag{3}$$

Kde *S* je hodnota hrubého skóre a lambda a *K* jsou Karlin-Altschul parametry [16], charakterizující skórovací systém.

E-hodnota

Tato hodnota se interpretuje jako horní hranice očekávané frekvence náhodného výskytu úseků HSP se skóre větším, než je hodnota *S*, v kontextu prohledávání celé databáze sekvencí. Tato hodnota naznačuje statistickou významnost daného zarovnání a reflektuje použitý skórovací systém a velikost databáze. Čím nižší hodnota, tím vyšší významnost. Zarovnání sekvencí s *E*-hodnotu rovné 0.05, znamená, že tato podobnost může náhodně nastat v 5 případech ze 100. Tato hodnota pomáhá určit významnost zarovnání, ale nelze ji zaměňovat s významností biologickou. Vzorec pro výpočet vypadá následovně:

$$E = K|D||Q|e^{-\lambda S} \tag{4}$$

nebo

$$E = |D||Q|2^{-S'} \tag{5}$$

P-hodnota

Počet náhodných HSP segmentů se skóre větším nebo rovným než hodnota *S* je popsáno Poissonovým rozložením. To znamená, že pravděpodobnost nalezení přesně daného HSP segmentu se skóre větším nebo rovným hodnotě *S* je dána vztahem:

$$P = e^{-E}\frac{E^a}{a!} \qquad (6)$$

kde *E* je *E*-hodnota skóre *S* daná rovnicí (4). Speciálně, šance nalezení 0 HSP segmentů se skóre >= *S* je dán vztahem e^{-E}, a pravděpodobnost nalezení nejméně jednoho takového HSP je dána vztahem:

$$P = 1 - e^{-E} \qquad (7)$$

Například, jestliže se očekává nalezení třech HSP se skóre >= *S*, je pravděpodobnost nalezení alespoň jednoho rovna 0,95 [16]. Programy BLAST raději ve výsledku zobrazují *E*-hodnotu, protože je jednoduší si uvědomit rozdíly, které vyjadřuje. Lépe si lze představit vztah mezi *E*-hodnotami 5 a 10 než mezi *P*-hodnotami 0.993 a 0.99995. Avšak v případě, kdy $E < 0.01$, *P*-hodnota a *E*-hodnota si jsou téměř rovny.

7 Vlastní implementace

Pro implementaci algoritmu BLAST, kde velmi záleží na rychlosti, se většinou používají jazyky jako C, C++, které se kompilují přímo do instrukcí daného procesoru. V poslední době, rozmachu vyšších, plně objektově orientovaných programovacích jazyků, se pro programování stále častěji začíná používat například i C# nebo Java.

Každý ze zmiňovaných jazyků má své výhody a nevýhody. Pro implementaci byl nakonec vybrán jazyk Java, který je dnes velice rozšířený a vysoce kompatibilní. Z výběru tohoto jazyka vyplývají jisté nevýhody. Hlavní nevýhodou se může zdát rychlost běhu výsledného programu, která bude zřejmě pomalejší než u programů, které jsou kompilované přímo do konkrétního binárního formátu, určeného pouze pro konkrétní systém. Tato nevýhoda je však dnes již zmírňována tím, že současné implementace kompilátorů Javy provádí poměrně slušnou optimalizaci, která zaručí dostatečnou rychlost.

Tato nevýhoda je pak dále kompenzována jinými výhodami jazyka Java. Velkou výhodou Javy je její hardwarová nezávislost, neboť je překládaná do speciálního mezikódu (to jí ovšem činí pomalejší). Tento mezikód je možné spustit na libovolné platformě, pro kterou existuje běhové prostředí Javy (JRE, Java Runtime Environment). Jazyk Java je možné využívat zdarma, neboť je součástí projektu open source. Nejen proto je tento jazyk tak rozšířen. Jeho další velkou výhodou je dostupnost knihoven, podporující nástroje pro zpracování biologických dat. Jedním z projektů, který se zabývá vývojem takových nástrojů je open source projekt BioJava. Proto byl tedy pro implementaci zvolen jazyk Java.

Podobně jako BioJava vznikají bio-knihovny i pro jiné jazyky. Relativně známé jsou projekty BioPython [15] a BioPerl [14]. V poslední době vzniká i projekt BioPHP a dokonce i programátoři v C# mají možnost využít vlastní biologické knihovny.

BioJava

Projekt BioJava je open source projekt distribuován s podmínkami GNU Lesser GPL [13]. Jedná se o knihovnu, která poskytuje Java nástroje pro práci s biologickými daty. Tato knihovna obsahuje mnoho užitečných tříd, které zapouzdřují různé biologické i bioinformatické pojmy jako jsou například sekvence, abecedy symbolů, samotné symboly, souborové parsery. Obsahuje celý balíček týkající se dynamického programování, balíček týkající se zarovnávání sekvencí, apod. Dále je pro programátory užitečná i tím, že podporuje CORBA, DAS klienta a server, přístup do BioSQL a Ensembl databází.

7.1 Implementace

Cílem této práce bylo vytvořit algoritmus podobný algoritmu BLASTN, následně vyzkoušet jeho funkčnost a porovnat zjištěné výsledky zarovnávání sekvencí s již existujícími algoritmy. Aplikace, která vznikla, je naprogramována v jazyce Java. Ovládání aplikace se provádí přes grafické uživatelské rozhraní.

7.1.1 Návrh

Celý program je rozdělen do třech balíčků: gui, alignment a settings. Jednotlivé balíčky budou postupně popsány podrobněji.

Balíček gui

Třídy, které tento balíček obsahuje, zajišťují obsluhu grafického rozhraní, tedy vykreslení okna a odchytávání událostí.

- ApplicationSeqAlignment - třída programu
- WindowSeqAlignment - třída hlavního okna
- WindowAbout - třída okna, zobrazující informace o programu
- WindowHelp - třída okna, zobrazující nápovědu k programu

Balíček settings

Třídy balíčku settings v sobě uchovávají nastavení programu.

- Constants - třída uchovává rozměry okna, titulek programu, výchozí adresář, apod.
- ExtensionFilter - třída zajišťující výběr souborů pouze s danou příponou

Balíček alignment

Tento balíček zabaluje třídy, které jsou využívány při provádění zarovnávání sekvencí. Následuje stručný přehled tříd a následně detailnější popis některých zajímavějších tříd.

- BlastAligner - třída, která provádí zarovnání sekvencí nukleotidů
- FastaReader - třída, která čte soubory ve formátu Fasta
- SegmentHSP - třída, která uchovává informace o segmentu HSP
- ListHSP - třída, která implementuje list objektů třídy SegmentHSP
- Score - třída pro skóre segmentu HSP, implementuje rozhraní Comparable
- ReverseScoreComparator - třída implementuje obrácený komparátor, pro zajištění řadění skóre sestupně

BlastAligner

Objekt této třídy provede zarovnání sekvence dotazu se všemi sekvencemi databáze a vybere nejlépe ohodnocené porovnání.

SegmentHSP

Objekty této třídy představují během provádění algoritmu nejprve zásahy (HIT) a poté úseky HSP. Jeden takový objekt uchovává informace o dosaženém skóre, případně o dalších statistických hodnotách (p-hodnota, e-hodnota) – ty se ale momentálně v dané verzi algoritmu nevyužívají. Dále se tu zaznamenává, kde zásah začíná a končí v sekvenci dotazu a v sekvenci databáze. Aby bylo možné stanovit, ke které sekvenci databáze zásah patří, obsahuje objekt také identifikační řetězec sekvence databáze. Do objektu je také možné zaznamenat konkrétní nukleotidové úseky daného zásahu sekvence dotazu i sekvence databáze.

ListHSP

Objekty třídy SegmentHSP se ukládají do seznamů segmentů HSP *ListHSP*. Pro každou diagonálu v matici, kde jednu osu tvoří sekvence dotazu a druhou sekvence databáze, existuje jeden *ListHSP*. Každý list navíc uchovává informaci o posledním nalezeném zásahu, o celkovém součtu všech skóre jednotlivých HSP a o nejlépe ohodnoceném segmentu HSP.

7.1.2 GUI

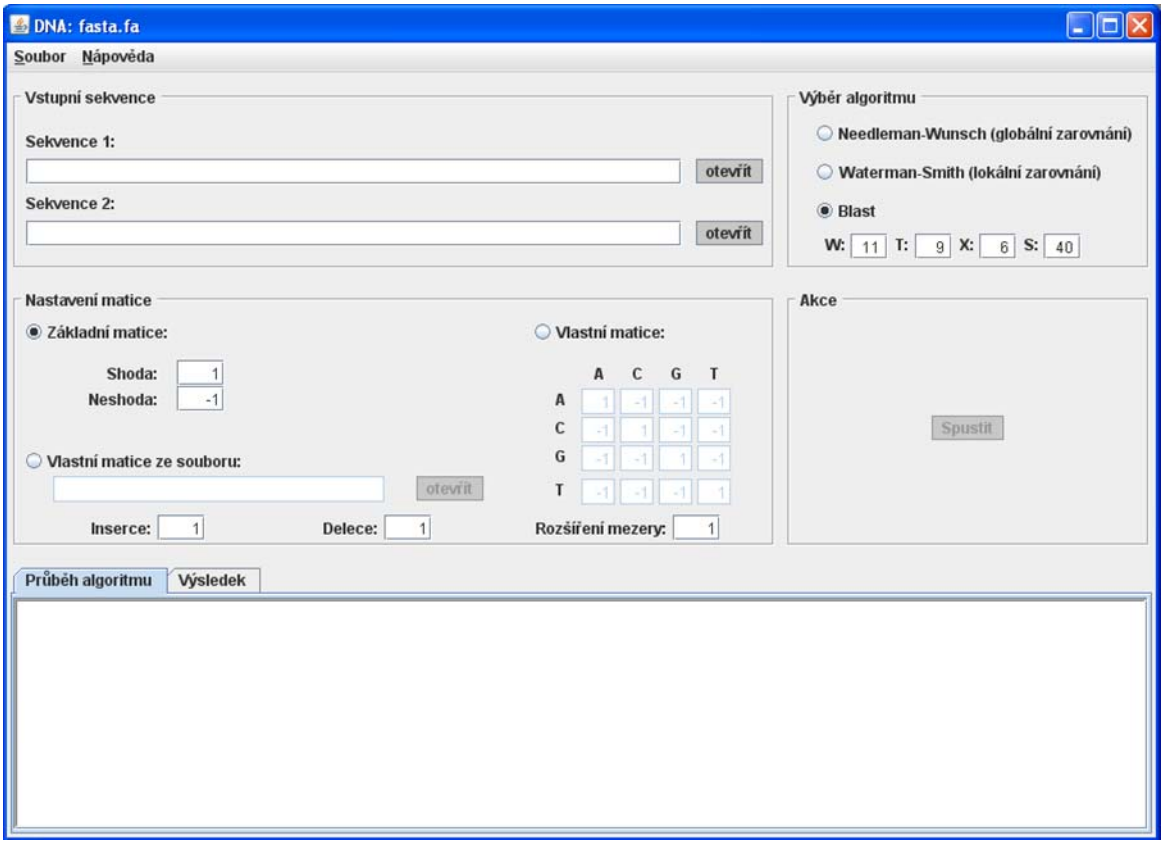
Grafické rozhraní je v zásadě tvořeno pouze jedním hlavním oknem programu. Na obrázku 11 je možné vidět, z jakých prvků se okno skládá.

V části *Vstupní sekvence* se zadávají soubory se vstupními sekvencemi. Program očekává, že *Sekvence 1* bude soubor obsahující jednu sekvenci DNA ve formátu Fasta. To bude sekvence dotazu. Na místě *Sekvence 2* může být soubor obsahující několik sekvencí DNA ve formátu Fasta. Tyto sekvence poslouží jako databáze sekvencí.

V další části *Výběr algoritmu* je možné zvolit algoritmus, kterým se provede zarovnání sekvencí. V případě výběru položky Blast je možné nastavit 4 vstupní parametry. Tyto parametry odpovídají parametrům, jak byly popsány v kapitole 6.3 o algoritmu BLAST. Pro připomenutí:

- **Parametr W** - délka slova, které se bude porovnávat
- **Parametr T** - práh shody, měl by být vždy menší nebo rovný než délka slova
- **Parametr X** - hodnota, o kterou může skóre klesnout při rozšiřování zásahu v porovnání s nejvyšší dosaženou hodnotou
- **Parametr S** - Skóre, kterého musí dosáhnout rozšířený zásah, aby byl označen jako HSP a postoupil do dalšího kroku algoritmu

V části *Nastavení matice* je možné upřesnit substituční matici, která se použije pro výpočet skóre zarovnání. Výchozím nastavením je matice, kde diagonála (shoda) je nastavena na samé jedničky a ostatní pole matice (neshoda) jsou nastaveny na hodnotu -1. Tyto hodnoty je možné kdykoli měnit. V případě potřeby zvýhodnění nějaké mutace, je možné nastavit celou matici podle vlastní potřeby. Stačí zaškrtnout radiobutton *Vlastní matice* a vyplnit pole, popřípadě je možné načíst matici ze souboru.



Obrázek 11: Grafické uživatelské rozhraní aplikace

Část *Akce* je velmi jednoduchá. Obsahuje pouze tlačítko S*pustit*. Toto tlačítko nelze stisknout, dokud nejsou nastaveny potřebné parametry programu. Ve spodní části okna jsou dvě záložky. První záložka *Průběh algoritmu* informuje uživatele o akcích, které právě algoritmus provádí, a o případných chybách, které v průběhu provádění vznikly. Druhá záložka v případě úspěšného dokončení algoritmu zobrazuje výsledek zarovnání.

7.1.3 Vlastní algoritmus

Vlastní algoritmus vychází z popisu původního algoritmu BLAST, a proto byl také implementován v tomto duchu a rozdělen na čtyři hlavní fáze: předzpracování, nalezení zásahu, prodlužování a spojování. Nyní se postupně po jednotlivých fázích podíváme na vlastní řešení algoritmu. Při popisu použijeme definice z kapitoly 6. Stanovíme ještě, že $|Q|$ je délka sekvence Q a $|D|$ je délka sekvence D. Dále si také pro lepší představu o časové náročnosti algoritmů zavedeme pojem *jednoznakové*

porovnání. Tento pojem vyjadřuje vzájemné porovnání právě dvou znaků. Takové porovnání trvá vždy stejnou konstantní dobu. Pro konkrétní výpočty budou použity hodnoty $w = 11$, $|Q| = 15$, $|D| = 15$.

Předzpracování

V této fázi originální BLAST generuje všechna slova o délce *w* nad DNA abecedou *A* a poté vytvoří seznamy sousedů pro každou pozici sekvence dotazu *Q*. Takové generování a vytváření seznamů ovšem zabere obrovské množství času. V případě, že délka *w* bude nastavena na 11, což je standardní hodnota pro nukleotidové sekvence, je potřeba vygenerovat 4^{11} (tj. 4 194 304) slov. To je celkem velké množství. Největší problém ale vzniká při vytváření seznamů sousedů. Pokud chceme pro každou pozici sekvence *Q* určit takový seznam, je nutné projít všechna vygenerovaná slova a podle prahu *T* odfiltrovat taková slova, která do seznamu nepatří.

Počet jednoznakových porovnání v takovém případě roste velmi rychle. Můžeme ho vyjádřit vztahem: $(|Q| - w + 1) \cdot 4^w \cdot w$. V našem konkrétním případě, kdy má sekvence *Q* pouhopouhých 15 znaků a parametr *w* standardní hodnotu 11, se počet jednoznakových porovnání rovná hodnotě $55 \cdot 4^{11}$. A to jsme ještě neporovnávali žádný znak se sekvencemi databáze Je zřejmé, že takto to v samotném algoritmu není možné aplikovat. Na provedení kompletního zarovnání dvou sekvencí o délkách $|Q|$ a $|D|$ pomocí dynamického programování stačí takových jednoznakových porovnání právě $|Q| \cdot |D|$, což je v našem ukázkovém příkladě rovno hodnotě 225. Rozdíl je jistě patrný. Z tohoto důvodu se žádné předzpracování neprovádí.

Nalezení zásahů

Protože se neprovedlo žádné předzpracování, není možné porovnávat podřetězce sekvencí databáze o délce *w* se slovy ze seznamů sousedů. Přesto se ale sekvence porovnávají postupně po slovech o délce *w,* a pokud je skóre porovnání slov větší než práh T, je zaznamenán zásah. Počet jednoznakových porovnání je v případě, kdy se okno o délce *w* posouvá po sekvencích po jednom znaku, možné vyjádřit vztahem $(|Q| - w + 1) \cdot (|D| - w + 1) \cdot w$. Tedy pro náš konkrétní příklad se výraz rovná hodnotě $5 \cdot 5 \cdot 11$, tedy 275. Je to již mnohem lepší výsledek, ale stále je jednoznakových porovnání více než za použití dynamického programování. Je to dáno tím, že přestože se sekvence porovnávají po slovech, v rámci vzájemného porovnávání jednotlivých slov probíhá opět porovnávání po znacích. Tím se algoritmus stává stále pomalejší než dynamické programování.

Hlavní myšlenka algoritmu BLAST je právě v tom, že neprohledává celý prostor, pouze zkouší nalézt větší podobné úseky, jejichž skóre je větší než práh *T*. A jenom s těmito úseky pracuje v dalších fázích. To vedlo k následující úpravě vlastního algoritmu. Sekvence budeme porovnávat po slovech o délce *w*, ovšem v sekvenci *D* se nebudeme posouvat po jednom znaku ale po celém slově. Je tak mírně snížena citlivost algoritmu. Ovšem pokud se porovnávají dvě podobné sekvence alespoň

o délce 2*w*, lze předpokládat, že se podaří najít zásah. Počet jednoznakových porovnání je nyní možné vyjádřit vztahem $(|Q| - w + 1) \cdot floor(|D|/w) \cdot w$, tedy $5 \cdot 1 \cdot 11$, což je 55.

	A	**G**	**C**	**A**	**G**	**T**	**T**	**A**	**G**	**C**	**A**	**T**	**C**	**A**	**C**	**G**	**T**	**G**	**A**	**C**	**G**	**T**
A	+											-										
G	-	+										-	-									
T	-	-	-									**+**	-	-								
A	+	-	-	+								-	**-**	+	-							
G	-	+	-	-	+							-	-	**-**	-	+						
C	-	-	+	-	-	-						-	+	-	**+**	-	-					
A	+	-	-	+	-	-	-					-	-	+	-	**-**	-	-				
T	-	-	-	-	-	+	+	-				**+**	-	-	-	-	**+**	-	-			
G	-	+	-	-	+	-	-	-	+			-	**-**	-	-	+	-	**+**	-	-		
A	+	-	-	+	-	-	-	+	-	-		-	-	**+**	-	-	-	-	**+**	-	-	
C	-	-	+	-	-	-	-	-	-	+	-	-	+	-	**+**	-	-	-	-	**+**	-	-
G	-	+	-	-	+	-	-	-	+	-	-	-	-	-	-	**+**	-	+	-	-	**+**	-
T	-	-	-	-	-	+	+	-	-	-	-	+	-	-	-	-	**+**	-	-	-	-	**+**
G	-	+	-	-	+	-	-	-	+	-	-	-	-	-	-	+	-	**+**	-	-	+	-
C	-	-	+	-	-	-	-	-	-	+	-	-	+	-	+	-	-	-	**-**	+	-	-
C	-	-	+	-	-	-	-	-	-	+	-	-	+	-	+	-	-	-	-	**+**	-	-
G	-	+	-	-	+	-	-	-	+	-	-	-	-	-	-	+	-	+	-	-	**+**	-
T	-	-	-	-	-	+	+	-	-	-	-	+	-	-	-	-	+	-	-	-	-	**+**

Tabulka 8.:Hledání zásahů po slovech o délce 11 znaků; + značí shodu, - neshodu, modře jsou zvýrazněny nalezené zásahy splňující práh T = 8

Mezi další výhody tohoto postupu lze zařadit i to, že nalezené zásahy se navzájem nepřekrývají. Prodlužováním překrývajících se zásahů dostaneme zpravidla stejný úsek podobnosti, což je zbytečně další čas navíc ve fázi prodlužování. Podobně i při prodlužování sousedních zásahů dostáváme zpravidla stejné úseky podobnosti a proto se již při hledání zásahů sousední zásahy spojují do jednoho zásahu. Tím se místo několika sousedních zásahů ve fázi prodlužování prodlužuje pouze jeden zásah.

Rozšiřování zásahů

Do této fáze vstupují všechny nalezené zásahy z předešlého kroku. Tyto zásahy se nyní podle původního algoritmu BLAST mají oboustranně bezmezerově rozšiřovat, dokud skóre rozšiřovaného zásahu neklesne o hodnotu *X* oproti nejlepšímu dosaženému skóre během rozšiřování. Současné rozšiřování zásahu na obě strany se však zdá být poměrně nepraktické. Zásadní problém je v tom, že pokud je rozšiřování na jednu stranu úspěšné a zvedá se skóre a rozšiřování na druhou stranu neúspěšné a skóre klesá, nastává situace, kdy celkové skóre se prakticky nemění, tedy neklesá ani nestoupá a do zásahu se započítávají i znaky, které do něj nepatří. Z tohoto důvodu vlastní algoritmus

bezmezerově rozšiřuje zásah na obě strany, ale postupně, tedy prvně na jednu stranu a poté na stranu druhou.

Tím, že je zásah rozšiřován bezmezerově, je algoritmus schopen se při rozšiřování vypořádat pouze s mutací nukleotidů. V případě že se uprostřed sekvence objeví inserce nebo delece obě sekvence se v daném místě vůči sobě posunou o mezeru, což zapříčiní, že se od místa inserce nebo delece sekvence jeví algoritmu jako výrazně rozdílné a rozšiřování se zastaví. Současné verze algoritmu BLAST již používají rozšiřování s mezerami, které spouští prakticky hned za rozšiřováním bez mezer. Využívají k tomu různých kombinací technik dynamického programování. Do implementace tohoto algoritmu však mezerové rozšiřování není zahrnuto.

Po skončení rozšiřování zásahu se vezme takové jeho rozšíření, které v průběhu rozšiřování dosáhlo nejlepšího skóre a pokud je toto skóre větší než práh *S*, označí se rozšířený zásah jako HSP. V průběhu rozšiřování všech zásahů se postupně uchovává informace o rozšíření s dosud nejlepší hodnotou skóre. Na úplném konci rozšiřování se rozšíření s nejlepším skóre označí jako MSP.

Spojování zásahů

Do tohoto kroku vstupují pouze HSP a MSP z předešlého kroku. Přestože se během algoritmu mnoho falešných zásahů odfiltrovalo, může mezi HSP stále existovat falešný úsek shody. Toto je relativně nepříjemný problém. Zvláště u sekvencí DNA, které mohou obsahovat dlouhé úseky opakujících se motivů. Mohou to být například úseky oddělující geny, nebo nekódující úseky aj.

Konkrétní postup, jak získané úseky shod pospojovat, se v dostupné literatuře neuvádí. Vytvořený algoritmus postupuje následovně. Vezme všechny HSP a vloží je do struktury TreeMap. Struktura automaticky zařídí, že jsou po vložení všech prvků prvky seřazeny, v našem případě podle hodnoty skóre. Spojování začíná od úseku nejlépe ohodnoceného HSP tedy MSP. Při tom se zjistí, na jaké diagonále se HSP nachází. Poté se od této diagonály stanoví na každou stranu určitý počet dalších diagonál, pro které je ještě zarovnání optimální a postupně se od nejvyššího skóre přidávají úseky HSP z těchto diagonál do výsledného řetězce zarovnání

7.2 Výsledky

Pro objektivní posouzení výsledků, byly zároveň do výsledné aplikace zařazeny algoritmy využívající techniky dynamického programování. Jsou také naimplementovány v jazyce Java s využitím tříd projektu BioJava [13].

Vlastní algoritmus BLAST pracoval při prvním testování, kdy ve fázi hledání zásahů postupoval po jednom znaku v obou sekvencích, několikanásobně pomaleji než algoritmy dynamického programování. Po úpravě, která zavedla posouvání se po sekvencích databáze o celé slovo, se čas zarovnání sekvencí výrazně zrychlil. Po přidání dalších úprav, jako spojování

sousedních zásahů a po celkovém doladění algoritmu, dosáhl vlastní algoritmus téměř dvakrát větší rychlosti oproti algoritmům dynamického porovnání, přitom využívá daleko méně paměti.

Algoritmy dynamického programování využijí pro zarovnání dvou sekvencí Q a D paměťový prostor minimálně o velikosti $|Q| \cdot |D| \cdot 8$ bytů. Pro výpočet se využívá datový typ double. V případech, kdy se rozlišuje začínající mezera od mezery prodlužovací se tato velikost ještě násobí třemi. Na osobním počítači, tak velmi rychle dochází k vyčerpání paměťových zdrojů, což se u algoritmu BLAST nestává. Ten pro svou potřebu vyžaduje pole ukazatelů na seznamy segmentů HSP o velikosti $|Q| + |D| \cdot 4$ bytů plus to, co zaberou samotné struktury seznamů nalezených zásahů.

Z pohledu časové a paměťové náročnosti to tedy není špatný výsledek. Zaměříme-li se ale na kvalitu porovnání, je na první pohled vidět, že vlastní implementace nedosahuje takové kvality zarovnání, jako dosahují algoritmy dynamického porovnání nebo profesionální algoritmy BLAST. Je to dáno tím, že ve fázi rozšiřování se zásahy rozšiřují pouze bezmezerově, a tudíž v případech, kdy se v jedné ze sekvencí objevují příliš často inserce nebo delece blízko za sebou, nejsou podobné úseky sekvencí z důvodu různého posunutí k sobě navzájem zarovnány. Tento nedostatek je možné zmírnit úpravou vstupních parametrů tak, aby do prodlužovací fáze prošlo více zásahů, tedy snížit parametr S, popřípadě i T a W, a zvýšit parametr X. Tím se ovšem algoritmus o něco zpomalí.

Reálné algoritmy BLAST, které jsou nasazovány pro prohledávání obrovských databází, jsou několikanásobně rychlejší než tato implementace algoritmu BLAST. Je to způsobeno zřejmě tím, že profesionální programy z rodiny BLAST pracují s databázemi, které jsou speciálně indexované, a porovnání se neprovádí znakově ale binárně. Využívají výkon několika stovek počítačů a mohou před vlastním vyhledáváním provést předzpracování. Vše, co jde předpočítat a co bude algoritmus využívat, umisťují do paměti v podobě různých hash tabulek nebo lookup tabulek. Spotřeba paměťových zdrojů se pak sice pohybuje v gigabytech, ale sám algoritmus je pak několikanásobně rychlejší.

Tyto algoritmy jsou také mimoto daleko lépe vyladěny pro konkrétní případy, se kterými je možné se u jednotlivých sekvencí DNA nebo proteinů setkat. Podle potřeb biologů, vznikly algoritmy právě jako IgBLAST apod.

8 Závěr

Cílem této práce bylo zorientovat se na poli bioinformatiky, prozkoumat možnosti porovnávání sekvencí a vyhledávání jejich podobností a pokusit se navrhnout možný postup, jak takové porovnání sekvencí provést. Výsledný algoritmus implementuje postup vycházející ze známých metod původního algoritmu BLAST. Bylo nutné se vypořádat s nemožností provedení předzpracování a s porovnáním řetězců ve znakové podobě. Na druhou stranu bylo možné využívat šikovné knihovny Javy a BioJavy, které práci často významně usnadnily.

Přínos této práce pro obor bioinformatiky je jistě zanedbatelný. Největší přínos je zřejmě čistě osobní. Přesto je možné spatřit přínos pro okolí v přiblížení některých technik a jazyků, které lze pro stanovení homologie sekvencí použít, a v rozboru některých klasických problémů, z čehož se dá čerpat pro další zkoumání a zlepšování popsaných technik.

Z této práce jednoznačně vyplývá, pokud se srovnají výsledky algoritmů založených na dynamickém programování a algoritmů založených na heuristice, že algoritmy s heuristikami jsou daleko rychlejší, více flexibilní a je možné je lépe přizpůsobovat daným požadavkům biologů. Na druhou stranu, při špatně zvolené heuristice nebo špatném nastavení skórovacího systému, je možné obdržet chybné výsledky, respektive výsledky, které chybně reflektují biologický význam. Dokonce, i když je vše nastaveno v pořádku, nemůžeme si být jisti optimálností výsledku. Je to způsobeno právě heuristikami a optimalizacemi, které sice zvyšují rychlost, ale snižují citlivost algoritmu. Může se tedy stát, že skóre, které je získáno zarovnáním dvou sekvencí, nemusí být vždy nejlepší možné. U algoritmů jako Smith-Waterman nebo Needleman-Wunch dostáváme za výsledek, vždy optimální zarovnání dvou sekvencí.

Pojmy „nejlepší“ nebo „optimální“ silně závisí na skórovacím systému, který je ke stanovení homologie využit. Pouze v případě, kdy se skórovací systém snaží modelovat nějaké pojmy biologické podobnosti, lze očekávat, že i zarovnání mající nejlepší skóre bude více biologicky významné, než kdyby skórovací systém vůbec biologické zákony nezohledňoval. Přesto žádná biologická dedukce (jako homologie), která se provádí přímo a výhradně z interpretace zarovnávacího skóre, nespoléhá pouze na takto vzniklou hypotézu. Je jasné, že není pouze jedna ideální volba skórovacích parametrů, která by modelovala všechny biologické pohledy, jestli vůbec nějaká existuje. Z těchto důvodů se výsledky zarovnání sekvencí vždy reprodukují s ohledem na biologický kontext, s ohledem na nějakou vnější znalost. To už musí posoudit nějaký biolog. Informatika zajímá pouze to, zdali je nebo není získaný výsledek statisticky významný.

Naimplementovaný algoritmus tvoří pouze základ pro další rozšiřování a zlepšování. Největší nedostatek je zřejmě absence mezerového rozšiřování. To v určitých případech, kdy se v sekvencích vyskytuje několik insercí nebo delecí za sebou, způsobuje, že se sekvence v těchto místech k sobě nezarovnají. Slabým místem celého algoritmu je také hledání zásahů, které trvá většinu (přibližně

75%) času běhu programu. Kvalitní hledání zásahů je důležité, optimální stav je, když se najdou všechny žádoucí zásahy a žádný nežádoucí. Mělo by se tedy této fázi věnovat dostatek času, ale přiměřeně k celému algoritmu. Řešení, které je v algoritmu použito, se jeví jako relativně pomalé. Zlepšení by jistě přineslo porovnávání slov binárně a ne znakově.

Literatura

[1] Krane K. D., Raymer M. L.: Fundamental Concepts of Bioinformatics, Benjamin Cummings, 2003, ISBN: 0-8053-4633-3

[2] Fenstermacher D.: Introduction to Bioinformatics, University of Pennsylvania, (7 stran)

[3] Rudolfová I.: Databáze biologických dat, VUT Brno, 2005, semestrální práce, (20 stran)

[4] Frédérique, G.: The Fasta and Blast programs, 2000, (17 stran)

[5] Kent, J., W.: BLAT – The BLAST-Like Alignment Tool, University of California, (9 stran)

[6] Bordoli, L.: Similarity Searches on Sequence Databases: BLAST, FASTA, Swiss Institute of Bioinformatics, 2003, (59 stran)

[7] Madden, T.: The BLAST Sequence Analysis Tool, NCBI, 2003, (17 stran)

[8] Wang H., Ong T.,Ooi B. Ch., Tan K.: BLAST++: A Tool for BLASTing Queries in Batches, Dept. of Computer Science, National University of Singapore,Singapore (8 stran)

[9] Susie S., Shiby T.: ODM BLAST: Sequence Homology Search in the RDBMS, Oracle Corporation, (4 strany)

[10] Stránky věnované algoritmu WU-BLAST [online], Dostupné z: <http://blast.wustl.edu/blast>

[11] Stránky organizace NCBI (National Center for Biotechnology Information) [online], Dostupné z: < http://www.ncbi.nlm.nih.gov>

[12] Wikipedia.org: elektronické stránky otevřené encyklopedie Wikipedie [online], Dostupné z: <http://wikipedia.org> (květen 2007).

[13] BioJava.org: elektronické stránky věnované projektu BioJava [online], Dostupné z: <http://biojava.org/wiki> (květen 2007).

[14] BioPerl.org: elektronické stránky věnované projektu BioPerl [online], Dostupné z: <http://bioperl.org/wiki> (květen 2007).

[15] BioPython.org: elektronické stránky věnované projektu BioPython [online], Dostupné z: <http://biopython.org/wiki/> (květen 2007).

[16] Statistická teorie využívaná algoritmy BLAST [online] Dostupné z: <http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html>

Seznam příloh

Příloha 1. DVD (manuál, zdrojové texty)