

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTRONIKE I RAČUNARSTVA

SEMINAR

Algoritmi za poravnavanje dviju sekvenci

Igor Čanadi

Voditelj: *Mile Šikić*

Zagreb, travanj 2009.

SADRŽAJ

Sadržaj.....	1
1. Uvod.....	2
2. Ocjenjivanje sličnosti.....	3
2.1 Supstitucijske matrice.....	4
2.1.1 PAM supstitucijska matrica	4
2.1.2 BLOSUM supstitucijska matrica	4
2.2 Ocjena procjepa	5
3. Algoritmi temeljeni na dinamičkom programiranju.....	6
3.1 Uvod u dinamičko programiranje.....	6
3.2 Needleman-Wunsch algoritam	7
3.3 Smith-Waterman algoritam	10
4. Heuristički algoritmi za pretraživanje baza bioloških sekvenci	11
4.1 FASTA algoritam	11
4.2 BLAST algoritam	12
4.2.1 Poboľjšani BLAST.....	14
5. Zaključak.....	15
6. Literatura.....	16
7. Sažetak	18

1. UVOD

Središnja dogma molekularne biologije je da unutar svake stanice DNA stvara RNA koji stvara proteine koji reguliraju sve biološke procese unutar organizma. Ljudsko tijelo je sastavljeno od otprilike 10^{12} stanica, od kojih se u svakoj nalazi po 23 para kromosoma koji sadrže svaki po 30.000 gena odnosno 3 milijarde nukleotida DNA.^[1] Biolozi su razvojem računala dobili mogućnost da tu ogromnu količinu informacija sakupe, pohrane i obrade. Tako je nastala znanstvena disciplina bioinformatika.

Jedno od važnijih područja bioinformatike je poravnavanje sekvenci. Cilj poravnavanja sekvenci je algoritamsko pronalaženje i ocjenjivanje sličnosti između sekvenci DNA, RNA i proteina koje bi mogle biti posljedica funkcionalne, strukturalne ili evolucijske veze između tih sekvenci.

Neke od primjena poravnavanja sekvenci:

1. **Gradnja evolucijskog stabla:** u procesu evolucije diversifikacija vrsta je nastala kao posljedica genetskih mutacija i rekombinacija. Znajući sličnost između DNA sekvencije dvije vrste možemo procijeniti prije koliko vremena je živio njihov zajednički predak, onaj iz kojeg su se obje vrste razvile.
2. **Procjena funkcije i strukture proteina:** ukoliko znanstvenik otkrije novi protein te pronade protein slične građe aminokiselina koji ima poznatu funkciju i strukturu, lakše će odrediti funkciju i strukturu novootkrivenog proteina.
3. **Identifikacija intergenskih područja koja se transkribiraju u ncRNA:** u DNA postoje intergenska područja, područja koja transkripcijom ne grade mRNA, pa tako ni proteine. Donedavno se mislilo da su intergenska područja nevažna, no otkrilo se da se dijelovi intergenskih područja ipak transkribiraju, no ne grade proteine. RNA koji nastane transkripcijom dijelova intergenskih područja zovemo ncRNA. Te dijelove nije jednostavno pronaći, budući da ne počinju sa start i ne završavaju sa stop kodonom, kao što je to slučaj kod gena. Jedna od metoda identificiranja dijelova intergenskih područja koji se transkribiraju u ncRNA je poravnavanje sekvenci. Ukoliko pronađemo istu sekvencu u više vrsta na intergenskom području DNA, velika je vjerojatnost da igra važnu ulogu u stanici zbog koje je ostala sačuvana kroz evoluciju.^[2]
4. **Nebiološka primjena:** poravnavanje sekvenci je pronašlo svoju primjenu i na nebiološkom području, u obradi prirodnih jezika i nekim društvenim znanostima.^[3]

Poravnavanje sekvenci dijeli se na:

1. poravnavanje dviju sekvenci i
2. poravnavanje tri i više sekvenci.

U ovom seminarском radu predstaviti ću algoritme za poravnavanje dviju sekvenci.

2. OCJENJIVANJE SLIČNOSTI

Postoji više vrsta mutacija u evolucijskom procesu.^[4]

1. zamjena,
2. ispuštanje,
3. umetanje,
4. udvostručavanje i
5. inverzija.

U obzir ćemo uzeti samo zamjenu, ispuštanje i umetanje, jer su udvostručavanje i inverzija rijetke mutacije i mogu se predstaviti kao ubacivanje odnosno zamjena.

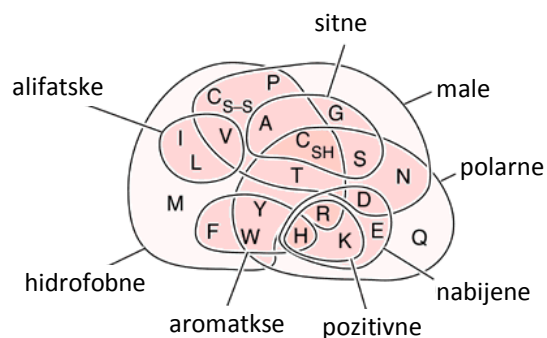
Poravnanje označuje postavljanje dva niza jednog iznad drugog tako da je svakom znaku u jednom nizu pridružen jedan znak ili jedan razmak u drugom nizu.^[4] U takvom poravnanju lako je izračunati potreban broj i vrstu mutacija. Najjednostavniji način za računanje ocjene sličnosti između dvije poravnate sekvence je da za svaki pogodak (jednaki znak u gornjoj i donjem nizu) ukupnom zbroju dodamo bodove, a za svaku mutaciju oduzmemo određeni broj bodova, ovisno o mutaciji.

Primjer 2.1 nam pokazuje poravnanje dvije DNA sekvence. Zelenom bojom označena su jednaka slova, ljubičastom mutacije ispuštanja, crvenom mutacije zamjena i plavom mutacije umetanja. Za pogodak dodjeljujemo 1 bod, za ispuštanje -3 boda, za zamjenu -5 boda i za umetanje -4 bodova. Ocjena sličnosti poravnatih sekvenci je -19.

	T	T	A	C	G	T	A	C	A	-	A	T	T	A	
	T	-	-	G	G	A	A	C	A	G	-	-	T	A	
ocjena:	+1	-3	-3	-5	+1	-5	+1	+1	+1	-4	-3	-3	+1	+1	= -19

PRIMJER 2.1

Međutim, nisu sve mutacije zamjene evolucijski jednako vjerojatne. Isto tako, ukoliko ocjenjujemo strukturalnu ili funkcijsku sličnost proteina, zamjena dviju aminokiselina sličnih kemijskih i fizikalnih svojstava nam nije toliko strašna koliko zamjena nekih drugih aminokiselina. Slika 2.1^[4] prikazuje svojstva aminokiselina i na njoj se jasno vidi da su neke aminokiseline međusobno više slične od ostalih. Hidrofobna aminokiselina će se najvjerojatnije zamijeniti s nekom drugom hidrofobnom aminokiselinom i slično. Zbog toga su razvijene metode ocjenjivanja sličnosti koje se temelje na takozvanim supstitucijskim matricama.



SLIKA 2.1 – AMINOKISELINE I NJIHOVA SVOJSTVA

2.1 SUPSTITUCIJSKE MATRICE

Supstitucijske matrice opisuju vjerojatnost da će se jedan znak u sekvenci promijeniti u neki drugi i služe nam za računanje sličnosti između dviju sekvenci. Na mjestu (i, j) u supstitucijskom matrici nalazi se vjerojatnost da će se znak i nakon nekog evolucijskog vremena zamijeniti sa znakom j . Vjerojatnost zamjene označavamo s takozvanim logaritamsko-vjerojatnosnim ocjenama koje izvodimo iz empirijskih istraživanja. Njih definiramo na sljedeći način:

$$S_{i,j} = \log \frac{p_i \cdot M_{i,j}}{p_i \cdot p_j} = \log \frac{M_{i,j}}{p_j} = \log \frac{\text{izmjerena frekvencija}}{\text{očekivana frekvencija}}$$

gdje je $M_{i,j}$ vjerojatnost da se znak i zamjeni sa znakom j , a p_i i p_j su frekvencije znakova i odnosno j .^[5] Baza logaritma nije bitna. Logaritam se koristi jer nam nezgodnu operaciju množenja vjerojatnosti svodi na zbrajanje s kojim puno lakše rukujemo.

Postoji više načina za računanje supstitucijskih matrica, koje generiraju različite supstitucijske matrice. Dvije najpoznatije i najviše korištene matrice za supstituciju aminokiselina su **PAM** i **BLOSUM** matrice.

2.1.1 PAM SUPSTITUCIJSKA MATRICA

Jednu od prvih supstitucijskih matrica, PAM (engl. Point Accepted Mutation) matricu, razvila je 1978. Margaret Dayhoff. Matrica se temelji na računanju razlika između srodnih proteina. PAM1 matrica procjenjuje kolika bi bila vjerojatnost supstitucije kad bi se u proteinu promijenilo 1% aminokiselina i izvedena je empirijski promatranjem 1572 mutacije na srodnim proteinima. Pomoću PAM1 matrice možemo izračunati matrice za veći postotak promijenjenih aminokiselina, ako pretpostavimo da se supstitucija događa po istom uzorku kao u PAM1 i da se na istom mjestu može desiti više supstitucija. Najveća matrica koju je Dayhoff izvela bila je PAM250. Najčešće se koriste PAM30 i PAM70.

2.1.2 BLOSUM SUPSTITUCIJSKA MATRICA

Pokazalo se da metoda promatranja mutacije srodnih proteina koja je korištena prilikom konstrukcije PAM matrica nije pogodna za poravnanje evolucijski dalekih sekvenci. Naime, promjene na sekvencama kroz duže evolucijsko razdoblje ne ponašaju se kao veliki broj malih promjena izvedenih na kraćem razdoblju.

Steven Henikoff and Jorja G. Henikoff su 1992. riješili taj problem konstrukcijom BLOSUM (engl. BLOck SUBstitution Matrix) supstitucijskih matrica promatrajući evolucijsko daleke sekvence^[25]. U evolucijski dalekim sekvencama tražili su i uzimali u obzir samo konzervirane blokove. Konzervirani blokovi su dobro očuvani dijelovi sekvenca pronađeni poravnanjem više sekvenci i za njih se pretpostavlja velika funkcijska važnosti, budući da su kroz evoluciju ostali više-manje očuvani. Blokove su u idućem koraku konstrukcije BLOSUM matrica podijelili u segmente te su zanemarili sve segmente koji su bili sličniji od određenog praga. U najčešće korištenoj BLOSUM62 matricu, taj prag iznosi 62%. Učestalost pojedinih supstitucija je tada prebrojana u ostalim segmentima, dakle, samo u onima kojima je ocjena sličnosti ispod tog praga. BLOSUM matrice s većim brojem koriste se u poravnanju bliskih sekvenci, dok se one s manjim brojevima koriste za biološki udaljenije sekvence.

Slika 2.2 prikazuje BLOSUM62 matricu u usporedbi s drugom supstitucijskom matricom imena CCF53.^[6]

BLOSUM62

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	
A	4	5	-2	-2	-2	0	0	0	0	-2	-2	-3	-2	-1	-2	0	0	0	-2	-3	0
R	-1	5	-2	-3	-3	0	-1	-2	0	-3	-4	1	-3	-3	-2	-2	0	0	-3	-4	-5
N	-2	0	6	5	-4	0	1	-1	0	-5	-6	-3	-4	-4	0	-2	-2	-2	-2	-5	-5
D	-2	-2	1	6	8	-2	-3	-1	-1	0	-2	-3	0	-1	-1	1	0	0	-2	0	0
C	0	-3	-3	-3	9	5	2	0	0	-2	-4	0	-2	-3	0	0	0	0	-2	-3	0
Q	-1	1	0	0	-3	5	5	0	0	-3	-4	0	-3	-3	0	0	0	-2	-3	-3	-3
E	-1	0	0	2	-4	2	5	6	0	-4	-5	-2	-3	-2	-2	0	0	0	-2	-3	-3
G	0	-2	0	-1	-3	-2	-2	6	6	-3	-4	0	-2	0	0	0	0	0	0	2	-2
H	-2	0	1	-1	-3	0	0	-2	8	4	0	-3	2	0	-2	-3	0	0	-3	2	1
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	4	0	0	-3	-4	-3	0	-4	0	-4	0
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	4	-2	-4	-1	-2	0	0	-3	-4	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	6	0	-3	-3	-2	0	-3	2	2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	6	-3	-2	-2	2	2	0	0
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	7	0	0	-2	-3	0	0
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	4	2	-2	-2	-3	-3
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	5	-1	-3	0	0
T	0	-1	0	-1	-1	-1	-1	-2	-1	-1	-1	-1	-2	-1	1	5	9	2	-1	-1	-1
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	7	-3	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-2	-2	2	7	4	4	4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	4

CCF53P62

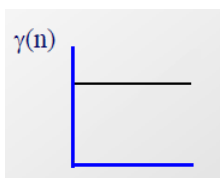
SLIKA 2.2 – BLOSUM62 I CCF53P62 SUPSTITUCIJSKE MATRICE

2.2 Ocjena procjepa

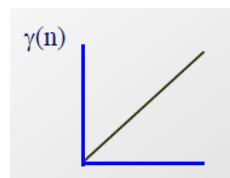
U dosadašnjem modelu smo procjep veličine n ocjenjivali kao $n \cdot k$, gdje je k broj bodova za mutaciju ispuštanja. Međutim, biološki je vjerojatnija mutacija proširenja procjepa (dvije ili više operacije ispuštanja) od mutacije stvaranja procjepa. Zbog toga uvodimo nove modele ocjene procjepa, koji aproksimiraju idealnu vjerojatnost stvaranja procjepa veličine n .

Modeli ocjena procjepa:^[8]

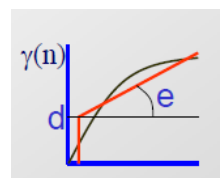
1. **Konstantna ocjena procjepa (Slika 2.3):** broj bodova za procjep, neovisno o njegovoj veličini, je uvijek konstantan.
2. **Linearna ocjena procjepa (Slika 2.4):** ako je k broj bodova za mutaciju ispuštanja, procjep veličine n ocjenjuje se kao $n \cdot k$.
3. **Afina ocjena procjepa (Slika 2.5):** uvode se dva parametra, d kao ocjena otvaranja procjepa i e kao ocjena proširenja procjepa (za svaku novu operaciju ispuštanja). Ocjena procjepa je $d + (n - 1) \cdot e$.
4. **Idealna ocjena procjepa (Slika 2.6):** konkavna funkcija izvedena empirijski; ne koristi se u praktičnoj primjeni zbog složene i spore implementacije. Aproksimira se dobro s jednom od prethodna tri modela ocjene procjepa.



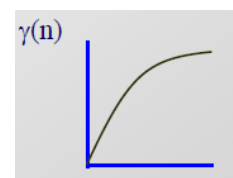
SLIKA 2.3 – KONSTANTNA Ocjena PROCJEPa



SLIKA 2.4 – LINEARNA Ocjena PROCJEPa



SLIKA 2.5 – AFINA Ocjena PROCJEPa



SLIKA 2.6 – IDEALNA Ocjena PROCJEPa

3. ALGORITMI TEMELJENI NA DINAMIČKOM PROGRAMIRANJU

Postoje dva algoritma za poravnanje sekvenci koji se temelje na dinamičkom programiranju:

1. **Needleman-Wunsch:** globalno poravnanje – traži optimalno poravnanje dviju cijelih sekvenci.
2. **Smith-Waterman:** lokalno poravnanje – traži optimalno poravnanje dva podniza sekvenci.

Za te algoritme definiramo:

- **Ulaz:** dvije sekvence i model za ocjenu sličnosti poravnatih sekvenci.
- **Izlaz:** optimalno poravnanje sekvenci prema modelu dobivenom u ulazu. Algoritmi će uvijek pronaći optimalno poravnanje.

3.1 UVOD U DINAMIČKO PROGRAMIRANJE

Dinamičko programiranje je metoda rješavanja (najčešće optimizacijskih) problema koji imaju sljedeća svojstva:^[10]

1. **Optimalna podstruktura:** optimalno rješenje podproblema se može iskoristiti za pronalaženje optimalnog rješenja cijelog problema.
2. **Preklapajući podproblemi:** rješenje jednog podproblema se koristi za rješavanje više većih podproblema.

Problem se, dakle, mora podijeliti na podprobleme pa rješenje problema možemo konstruirati pomoću tri komponente:

1. **Definicija podproblema:** skup podataka koji jednoznačno određuje podproblem i značenje podproblema.
2. **Rekurzivna relacija:** metoda odnosno formula po kojoj se određuje rješenje većeg podproblema na temelju manjih.
3. **Rješenje najmanjih podproblema:** budući da najmanje probleme ne možemo računati pomoću još manjih, morat ćemo ih posebno definirati.

Pokažimo navedeno na primjeru računanja n -tog Fibonaccijevog broja metodom dinamičkog programiranja:

1. **Podproblem** će nam biti računanje i -tog Fibonaccijevog broja. Neka $F[i]$ označava tu vrijednost.
2. **Rekurzivna relacija** slijedi iz definicije Fibonaccijevih brojeva: $F[i] = F[i - 1] + F[i - 2], i \geq 2$
3. **Rješenje najmanjih podproblema** također slijedi iz definicije Fibonaccijevih brojeva: $F[0] = 1, F[1] = 1$.

Implementacija dinamičkog rješenja se uglavnom izvodi na sljedeća dva načina:^[10]

1. **Top-down pristup:** Krećemo s rješavanjem velikog problema, kojeg razbijamo na podprobleme. Nakon što riješimo određeni podproblem zapamtimo njegovo rješenje, tako da ga ne moramo ponovo računati kad nam opet zatreba kod računanja nekog većeg podproblema.
2. **Bottom-up pristup:** Unaprijed rješavamo i pamtimo rješenja manjih podproblema koji će nam trebati kod računanja većih.

3.2 NEEDLEMAN-WUNSCH ALGORITAM

Needleman-Wunsch algoritam je povijesno prva primjena dinamičkog programiranja na usporedbu bioloških sekvenci.^[11] Konstruirali su ga i objavili 1970. godine Saul Needleman i Christian Wunsch.^[12] Koristi se za globalno poravnanje sekvenci.

Neka su A i B dvije biološke sekvence duljina n i m . S je supstitucijska matrica, $S(a, b)$ označava cijenu zamjene znaka a sa znakom b . Koristit ćemo linearni model ocjene procjepa, $S(a, -) = S(-, a) = d$, gdje je d cijena mutacije ispuštanja odnosno umetanja. Poravnanjem sekvenci A i B nazivamo dvije nove biološke sekvence jednake duljine, A' i B' nastale tako da su u sekvenci A i B uneseni razmaci (procjepi) na određenim mjestima. Vrijednost poravnanja A' i B' definiramo kao:

$$F(A', B') = \sum_{i=1}^{|A'|=|B'|} S(A'[i], B'[i])$$

Naš zadatak je pronaći optimalno poravnanje, ono s najvećom vrijednosti. Konstruirajmo rješenje pomoću dinamičkog programiranja:^[11]

1. **Podproblem** ćemo definirati kao optimalno poravnanje prvih nekoliko znakova sekvenci A i B . Konkretno, neka nam $P(i, j)$ označuje optimalnu vrijednost poravnanja sekvenci $A[1 \dots i]$ i $B[1 \dots j]$, dakle prvih i znakova sekvence A i prvih j znakova sekvence B .
2. Za definiciju **rekurzivne relacije** moramo pronaći vezu između većih i manjih podproblema. Razmotrimo kako možemo konstruirati optimalno poravnanje (i, j) pomoću manjih optimalnih poravnanja:
 - a. Uzeti optimalno poravnanje $(i, j - 1)$, dodati na prvi niz procjep, a na drugi $B[j]$. Vrijednost takvog poravnanja je $P(i, j - 1) + S(-, B[j])$.
 - b. Uzeti optimalno poravnanje $(i - 1, j)$, dodati na prvi niz $A[i]$, a na drugi procjep. Vrijednost takvog poravnanja je $P(i - 1, j) + S(A[i], -)$.
 - c. Uzeti optimalno poravnanje $(i - 1, j - 1)$, dodati na prvi niz $A[i]$, a na drugi niz $B[j]$. Vrijednost takvog poravnanja je $P(i - 1, j - 1) + S(A[i], B[j])$.

Optimalno poravnanje je najveća vrijednost od tri moguća načina konstrukcije. Zapisano matematički, rekurzivna relacija je:

$$P(i, j) = \max \begin{cases} P(i, j - 1) + S(-, B[j]) \\ P(i - 1, j) + S(A[i], -) \\ P(i - 1, j - 1) + S(A[i], B[j]) \end{cases}, i, j \geq 1$$

$$\begin{aligned} P(0, j) &= P(0, j - 1) + S(-, B[j]), \quad j \geq 1 \\ P(i, 0) &= P(i - 1, 0) + S(A[i], -), \quad i \geq 1 \end{aligned}$$

3. **Rješenje najmanjeg podproblema:** $P(0, 0) = 0$.

Implementacija Needleman-Wunsch algoritma je vrlo jednostavna. Pseudokod top-down pristupa:

```

poravnanje (i, j){
    ako sam već prije izračunao (i, j) → vrati izračunato
    ako je (i, j) = 0 → P = 0
    ako je i = 0 i j >= 1 → P = poravnanje(0, j-1) + S(-, B[j])
    ako je i >= 1 i j = 0 → P = poravnanje(i-1, 0) + S(A[i], -)
    ako je i >= 1 i j <= 1 →
        P = max( poravnanje(i, j-1) + S(-, B[j]),
                poravnanje(i-1, j) + S(A[i], -),
                poravnanje(i-1, j-1) + S(A[i], B[j]) )
    zapamti P kao rješenje od (i, j)
    vrati P
}
    
```

Pogledajmo kako algoritam radi na primjeru dvije DNA sekvence, ACTA i ATATA. Koristit ćemo jednostavnu supstitucijsku matricu (Tablica 3.1).

	A	G	C	T	-
A	3	-1	-1	-1	-2
G	-1	3	-1	-1	-2
C	-1	-1	3	-1	-2
T	-1	-1	-1	3	-2
-	-2	-2	-2	-2	X

TABLICA 3.1 – SUPSTITUCIJSKA TABLICA

Nakon izvršenja algoritma na ACTA i ATATA, matrica rješenja će izgledati ovako:

		A	C	T	A
	0	-2	-4	-6	-8
A	-2	3	1	-1	-3
T	-4	1	2	4	2
A	-6	-1	0	2	7
T	-8	-3	-2	3	5
A	-12	-5	-4	1	6

TABLICA 3.2 – MATRICA RJEŠENJA

Vidimo da je vrijednost optimalnog poravnanja ove dvije sekvence 6. Postavlja se pitanje koje je to poravnanje? Drugim riječima, kako napraviti rekonstrukciju rješenja? Algoritam koji smo do sad definirali ne odgovara na to pitanje, ne podržava rekonstrukciju rješenja. U svrhu rekonstrukcije moramo za pojedini podproblem osim optimalne vrijednosti poravnanja pamtili i kako smo došli do te vrijednosti. Strelice u Tablica 3.2 oslikavaju tu ideju. Vidimo da smo do krajnje vrijednosti 6, došli tako da smo na optimalno poravnanje sekvenci ACT i ATAT, vrijednosti 3, dodali na kraj obje poravnate sekvence slovo A, što ima vrijednost $S(A, A) = 3$.

Slijedeći strelice od krajnje vrijednosti do početka, možemo rekonstruirati optimalno poravnanje. Postoje dva poravnanja koja daju vrijednost 6, što možemo očitovati promatrajući narančaste strelice. Naime, do vrijednosti 0 u podproblemu (AC, ATA) možemo doći na dva načina. Optimalna poravnanja dana su u Primjeru 3.1 i 3.2.

	A	C	-	T	A	
	A	T	A	T	A	
ocjena:	+3	-1	-2	+3	+3	= 6

PRIMJER 3.1

	A	-	C	T	A	
	A	T	A	T	A	
ocjena:	+3	-2	-1	+3	+3	= 6

PRIMJER 3.2

Vremenska složenost Needleman-Wunsch algoritma je $O(n \cdot m)$, gdje su n i m dužine sekvenci. Naime, složenost računanja određenog podproblema je $O(1)$, svaki podproblem računamo najviše jednom i imamo ukupno $n \cdot m$ podproblema. Prostorna složenost je $O(n \cdot m)$; za svaki podproblem pamtimo jednu ili dvije vrijednosti. Međutim, postoji verzija Needleman-Wunsch algoritma čija prostorna složenost je $O(\min\{n, m\})$. Ta verzija zove se Hirschbergov algoritam i temelji se na divide and conquer strategiji.^[13]

Opisani algoritam podržava isključivo linearan model ocjene procjepa. Uz neke modifikacije, algoritam se može koristiti i za ostale modele ocjene procjepa. Rezultat tih modifikacija je i veća vremenska složenost.^[8]

3.3 SMITH-WATERMAN ALGORITAM

Smith-Waterman algoritam je deterministički algoritam za lokalno poravnanje sekvenci; dakle, traži optimalno poravnanje podnizova dvije sekvence. Objavili su ga 1981. godine Temple Smith i Michael Waterman.^[14]

Lokalno poravnanje koristimo npr. u ocjenama sekvenci koje nisu biološki blisko povezane i koje su evolucijski prošle kroz mnoge mutacije. Veliki broj mutacija onemogućuje poravnanje cijelih sekvenci, pa tražimo samo dijelove, odnosno podnizove tih sekvenci koji su slični.

Algoritam se definira na sljedeći način (oznake su iste kao i u prethodnom poglavlju):

1. **Podproblem** definiramo slično kao i u algoritmu Needleman-Wunsch: $P(i, j)$ nam označuje optimalnu vrijednost poravnanja sekvenci $A[k \dots i]$ i $B[l \dots j]$, za bilo koje k i l za koje vrijedi $k \leq i$ te $l \leq j$. Drugim riječima optimalno poravnanje podnizova koji završavaju na pozicijama i i j .
2. Jedina razlika kod **rekurzivne relacije** s obzirom na Needleman-Wunsch algoritam je ta da negativno rješenje možemo postaviti na 0, i to onda gledamo kao mogući početak poravnanja:

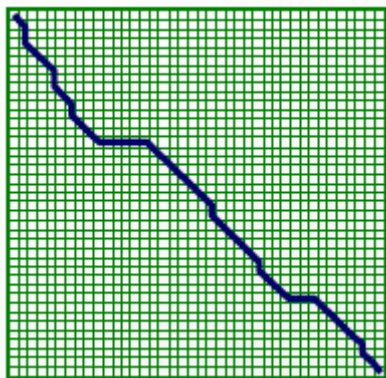
$$P(i, j) = \max \begin{cases} P(i, j-1) + S(-, B[j]) \\ P(i-1, j) + S(A[i], -) \\ P(i-1, j-1) + S(A[i], B[j]) \\ 0 \end{cases}, i, j \geq 1$$

$$P(0, j) = P(0, j-1) + S(-, B[j]), j \geq 1$$

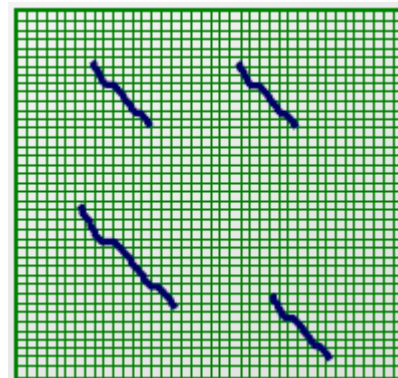
$$P(i, 0) = P(i-1, 0) + S(A[i], -), i \geq 1$$

3. **Rješenje najmanjeg podproblema:** $P(0,0) = 0$.

Slika 3.1 i Slika 3.2 prikazuju razliku između matrice rješenja Needleman-Wunsch i Smith-Waterman algoritma.^[8]



SLIKA 3.1 – NEEDLEMAN-WUNSCH



SLIKA 3.2 – SMITH-WATERMAN

Optimalno poravnanje u Smith-Watermanu je najveća vrijednost u matrici rješenja, ma gdje god se ona nalazila. Rekonstrukcija rješenja se vrši do pozicije na kojoj je vrijednost 0, ona nam označava početak podnizova.

Vremenska i prostorna složenost identične su kao i kod algoritma Needleman-Wunsch: obje iznose $O(n \cdot m)$, gdje su n i m dužine sekvenci. Međutim, Hirschbergovim algoritmom prostorna složenost se može izvesti u $O(\min \{n, m\})$.^[13]

4. HEURISTIČKI ALGORITMI ZA PRETRAŽIVANJE BAZA BIOLOŠKIH SEKVENCI

Opisani dinamički algoritmi su prilično neefikasni kod problema identifikacije novosekvencioniranog gena. Naime, taj gen se mora usporediti sa nizom bioloških sekvenci čija ukupna veličina dostiže redove veličine 10^7 .^[15] U tu svrhu razvijeni su heuristički algoritmi koji za ulazni niz pronalaze slične podnizove unutar velike baze podataka bioloških sekvenci.

Za razliku od algoritama koji se temelje na dinamičkom programiranju, heuristički algoritmi ne garantiraju da će uvijek pronaći niz s najvećom sličnošću. Tri parametra prema kojima se ocjenjuju heuristički algoritmi za pretraživanje baza bioloških sekvenci su:^[16]

1. **Brzina**
2. **Osjetljivost** – sposobnost pronalazjenja evolucijski povezanih sekvenci usprkos slabih sličnosti uslijed veće evolucijske udaljenosti. Veća osjetljivost smanjuje broj lažnih negativnih rezultata.
3. **Selektivnost** – sposobnost isključenja sličnosti koje su posljedica slučajnosti. Veća selektivnost smanjuje broj lažnih pozitivnih rezultata.

4.1 FASTA ALGORITAM

FASTA algoritam konstruirali su 1985. David J. Lipman i William R. Pearson^[17], te su ga dodatno poboljšali 1988.^[18]

Na svakoj sekvenci iz baze izvodi se sljedeći algoritam:^{[15][19]}

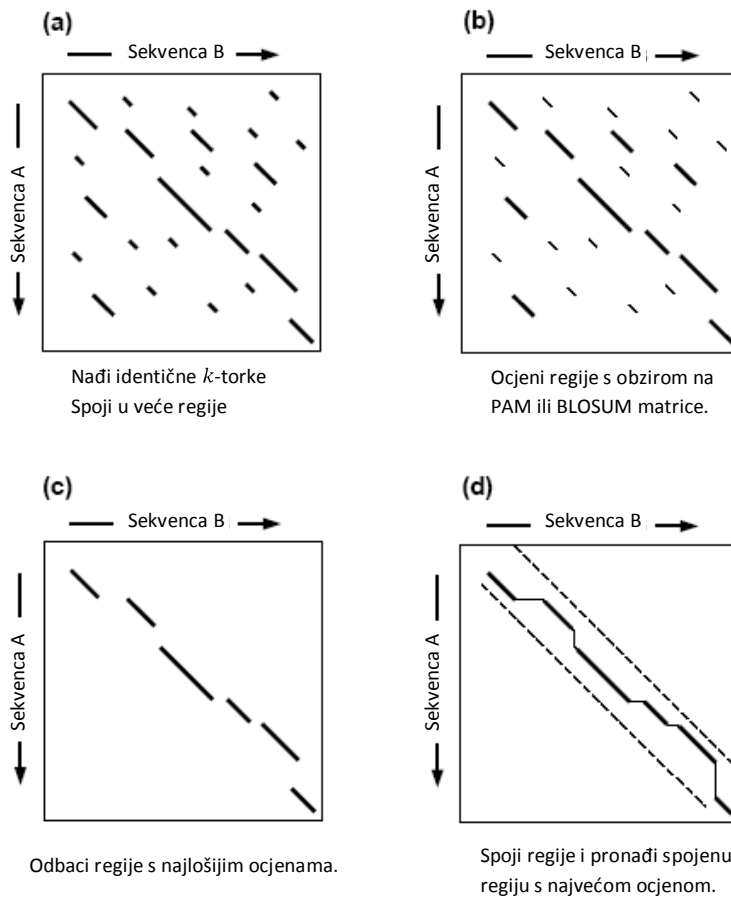
1. Iz ulazne sekvence izdvoje se svi podnizovi duljine k , k -torke. Parametar k je 1-2 za sekvence aminokiselina, a 4-6 za sekvence nukleotida. Za svaku k -torku iz ulaznog niza, traže se i označuju sva pojavljivanja u bazi sekvenci. Traženje se efikasno implementira pomoću hash funkcije i lookup tablica. Ako je udaljenost nekih k -torka u ulaznom nizu i njihovog pojavljivanja u bazi sekvenci jednaka, spajaju se, zajedno s elementima između njih, u veće regije. Te regije zbog jednake udaljenosti između k -torki nemaju procjepe. Za svaku identificiranu regiju definira se i računa gustoća kao omjer između broja jednakih elemenata u regiji i ukupnog broja elemenata.
2. U drugom koraku, uzima se 10 regija s najvećom gustoćom. Za tih 10 regija računa se njihova ocjena prema PAM ili BLOSUM supstitucijskim matricama. Regija s najvećom ocjenom zove se $init_1$ regija. Osim biranja $init_1$ regije, u ovom se koraku i odbacuju regije s malim ocjenama.
3. Preostale regije iz koraka 2. spajaju se u veće regije. Ocjena većih regija je zbroj ocjena spojenih inicijalnih regija minus konstantna kazna za svaki procjep koji se treba uvesti da bi se regije spojile. Algoritam kojim se pronalazi spojena regija s najvećom ocjenom temelji se na teoriji grafova. Najbolja regija iz ovog koraka zove se $init_n$ regija.
4. Izvodi se ograničen Smith-Waterman algoritam s ishodištem u $init_1$ regiji. Ograničen znači da ne smije uvesti više od određenog broja procjepa, s obzirom na ishodište. Broj procjepa koji se smije uvesti ovisi o parametru k , te se računa određenim statističkim metodama. Poravnanje pronađeno u ovom koraku naziva se *opt* poravnanje.

Slika 4.1 grafički prikazuje prva tri koraka.^[20]

Na najboljih nekoliko sekvenci iz baze s obzirom na ocjenu $init_n$ i *opt* poravnanja pokreće se potpuni Smith-Waterman algoritam, koji pronalazi sekvence iz baze s najvećom sličnosti s ulaznom sekvencom. To je konačni izlaz FASTA algoritma.

Iako je FASTA heuristički algoritam i postoje slučajevi kad ne daje optimalno rješenje, tvrdi se i empirički je provjereno da je izlaz iz FASTA algoritma uvijek vrlo blizu optimalnog izlaza koji bi se dobio primjenom mnogo sporijeg determinističkog Smith-Waterman algoritma.^[15]

FASTA algoritam



SLIKA 4.1 - GRAFIČKI PRIKAZ PRVA TRI KORAKA FASTA ALGORITMA

4.2 BLAST ALGORITAM

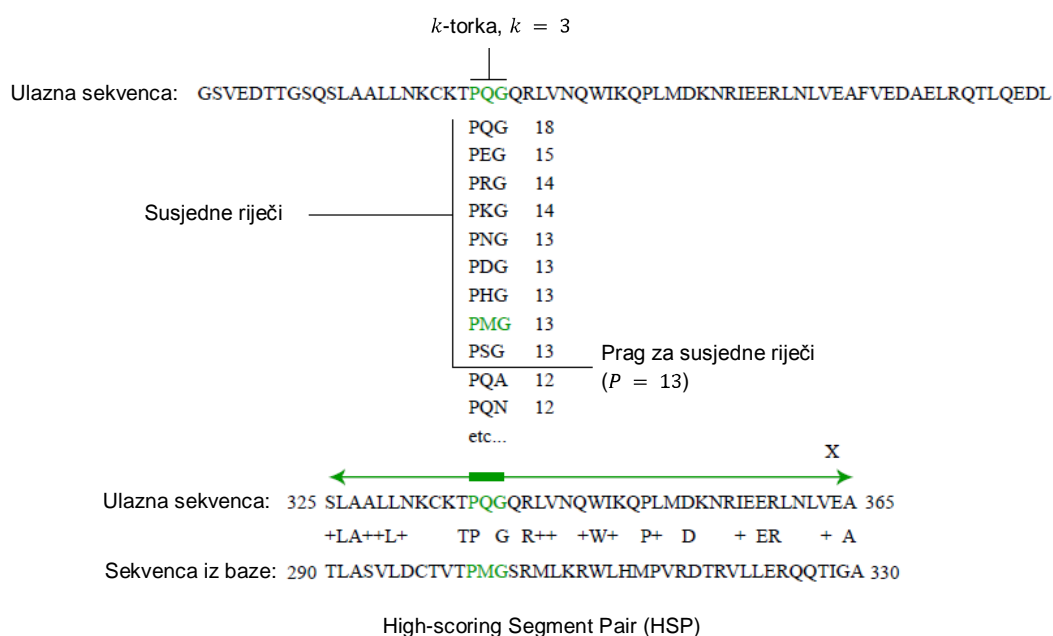
BLAST algoritam su 1990. konstruirali Eugene Myers, Stephen Altschul, Warren Gish, David J. Lipman i Webb Miller na National Institute of Health.^[21] Njihova želja bila je ubrzati FASTA algoritam smanjenjem broja k -torki i povećanjem njihove relevantnosti. Ideja je bila integrirati supstitucijsku matricu u prvi korak traženja k -torki.^[15] Osnovni BLAST algoritam traži poravnanja bez procjepa.

Definirajmo prvo ključan pojam BLAST algoritma: **High-scoring segment pair (HSP)** se sastoji od podniza (iz ulazne i sekvence iz baze) jednake duljine, takva da je ocjena njihova poravnanja bez razmaka veća od unaprijed zadanog praga. Ocjena poravnanja mora biti lokalno maksimalna, tj. produljenje poravnanja u bilo kojem smjeru smanjuje ocjenu.^[16]

Osnovni BLAST se svodi na pronalaženje HSP-ova.

Osnovni BLAST algoritam:^[16]

1. Generiraju se sve k -torke iz ulazne sekvence. Parametar k je obično 3 za aminokiseline, a 11 za nukleotide. Za neku k -torku, definiramo *susjedne riječi* kao nizove duljine k koji u poravnanju s k -torkom daju ocjenu veću od određenog praga. Susjedne riječi svih k -torki iz ulazne sekvence zovemo *ciljne riječi*.
2. Podniz sekvence iz baze podataka koji je jednak nekoj ciljnoj riječi zovemo *hit*. Hitovi se efikasno pronalaze korištenjem automata koji prihvaća ciljne riječi.
3. Svaki pronađen hit se proširuje na lijevu i desnu stranu bez razmaka tako dugo dok ukupna ocjena ne padne previše u odnosu na najveću pronađenu. Ako ocjena proširenog hita pređe unaprijed zadani prag za HSP, dobili smo HSP. (Slika 4.2 grafički prikazuje prva tri koraka^[9])
4. Za svaki pronađeni HSP računa se statistička važnost dobivenog rezultata preko Gumbelove distribucije ekstremnih vrijednosti (EVD). Naime, rezultat Smith-Waterman algoritma na dvije slučajne sekvence slijedi Gumbelovu EVD (neovisno o tome radi li s procjepima ili ne). Mjera statističke važnosti HSP-a je E koji označava očekivani broj poravnanja slučajne sekvence s ocjenom većom ili jednakom od ocjene tog HSP-a u korištenoj bazi sekvenci.^[22] E nam zapravo govori koliko je moguće da se HSP desio slučajno i da su sekvence evolucijski i funkcijski nepovezane.



SLIKA 4.2 – GRAFIČKI PRIKAZ PRVA TRI KORAKA BLAST ALGORITMA

BLAST je oko 50 puta brži od Smith-Watermana. Također, brži je i od FASTA-e, a njihova osjetljivost je usporediva, te je zbog toga dominantan u praktičnoj uporabi.^[22]

4.2.1 POBOLJŠANI BLAST

Konstruktori BLAST-a nastavili su razvijati algoritam i poboljšavati ga s obzirom na brzinu i osjetljivost. 1997. godine objavili su poboljšani BLAST algoritam.^[23]

Dva najvažnija poboljšanja koja su implementirali:^[24]

1. Umjesto traženja i razmatranja jednog hit-a, u poboljšanoj verziji traže se dva jednako udaljena hit-a (na istoj dijagonali). Razmatraju se samo hit-ovi koji su blizu jedan drugome. Takav način znatno povećava brzinu, ali i smanjuje osjetljivost. Da bi se osjetljivost povećala nauštrb brzine (engl. trade-off), smanjuje se prag kod računanja susjednih riječi tako da raspolažemo s većim brojem ciljnih riječi.
2. Drugo poboljšanje fokusiralo se na traženje poravnanja s razmacima. Rješenje je izvedeno slično kao i u FASTA algoritmu: Najbolji HSP pronađen u prvoj fazi BLAST algoritma koristi se kao ishodište ograničenog Smith-Waterman algoritma. Smith-Waterman se od HSP-a širi lijevo i desno tako dugo dok ocjena ne padne za određeni postotak ispod najbolje do tad pronađene ocjene.

BLAST algoritam se nastavio razvijati u više smjerova pa danas postoji mnogo verzija koji se vrte oko dosad opisanih principa, uz neke specifične preinake i poboljšanja. Glavna vanjska razlika između njih je konfiguracija brzine, osjetljivosti i selektivnosti te su kao takve svaka pogodna za određenu primjenu.

5. ZAKLJUČAK

Iz osnovnih tehnika računarne znanosti uz teoriju vjerojatnosti i empirijske podatke i biološke činjenice, razvili su se algoritmi za poravnanje sekvenci:

- 1970. – Needleman-Wunsch
- 1981. – Smith-Waterman
- 1985. – FASTA
- 1990. – BLAST

Needleman-Wunsch i Smith-Waterman koriste tehniku dinamičkog programiranja koju je već 1940ih opisao Richard Bellman.

FASTA i BLAST su heuristički algoritmi koji ne garantiraju optimalno rješenje, ali su dovoljno točni za sve praktične primjene i puno brži od determinističkih algoritama.

Njihove implementacije su danas besplatno dostupne na internetu te omogućavaju svim biološkim istraživačima da u kratkom vremenu dobiju odgovor na važna i kompleksna pitanja.

Neka od važnih pitanja na koja direktno ili indirektno odgovaraju algoritmi za poravnanje sekvenci:

1. Koliko su dvije vrste genetski slične, odnosno koliko je udaljen njihov zajednički predak?
2. Postoje li podaci o proteinu sličnom nepoznatom proteinu definiranom nizom aminokiselina?
3. Koji dijelovi DNA ili proteina igraju biološki važniju ulogu od drugih?

Bioinformatika, multidisciplinarnost biologije i računarne znanosti samo je jedna od mnogih priča iz zadnjih desetak godina o korištenju znanja iz jedne znanosti u korist razvoja druge. Sve se češće događaju znanstveni proboji u području dvije ili čak više znanosti, te se u znanosti povećava potreba za multidisciplinarnim stručnjacima.

Taj fenomen posebno je istaknut baš u primjeru računarne znanosti koja je u zadnjih nekoliko desetaka godina ubrzanog razvoja za sobom povukla i većinu ostalih znanosti, npr.:

- Arheologija: neka stara pisma danas dešifriraju računalni programi.
- Lingvistika: postoje algoritmi koji služe za analizu prirodnih jezika, pa čak se i algoritmi za poravnanje sekvenci koriste u lingvistici.
- Psihologija: analiza sociograma, mreže ljudskih kontakata izvodi se pomoću algoritama iz područja kompleksnih mreža.
- Ekonomija: algoritmi analiziraju ponašanja financijskih tržišta te makro i mikroekonomska kretanja.
- Meteorologija: algoritmi za simulacije omogućuju simuliranje meteoroloških procesa i upravljanje s velikom količinom podataka.

6. LITERATURA

- [1] Richon A. B. *A Short History of Bioinformatics*. <http://www.netsci.org/Science/Bioinform/feature06.html> (3.5.2009.)
- [2] Vaidyanathan P.P. (2004.) *Genomics and Proteomics: A Signal Processor's Tour*. IEEE Circuits and Systems Magazine, 4 (4), pp. 6-29. ISSN 1531-636X
- [3] *Sequence Alignment*. http://en.wikipedia.org/wiki/Sequence_alignment (3.5.2009.)
- [4] *Usporedba sljedova*. <http://www.irb.hr/hr/home/ristov/predavanja/Poravnanja07.ppt> (3.5.2009.)
- [5] *Substitution matrix*. http://en.wikipedia.org/wiki/Substitution_matrix (3.5.2009.)
- [6] Brick K., Pizzi E. (2008.) *A novel series of compositionally biased substitution matrices for comparing Plasmodium proteins*. BMC Bioinformatics 2008, 9:236. doi:10.1186/1471-2105-9-236
- [7] *Sequence Alignment and Dynamic Programming* (2005.) <http://ocw.mit.edu/NR/ronlyres/Electrical-Engineering-and-Computer-Science/6-895Fall-2005/43EB71AB-9BDB-4E42-A9E9-8425665A91D8/0/lecture2.pdf> (3.5.2009.)
- [8] *Sequence Alignment (part 2)* (2005.) <http://ocw.mit.edu/NR/ronlyres/Electrical-Engineering-and-Computer-Science/6-895Fall-2005/2ECE6A11-805E-4163-904D-06554E829DF0/0/lecture3.pdf> (3.5.2009.)
- [9] *Database Search* (2005.) <http://ocw.mit.edu/NR/ronlyres/Electrical-Engineering-and-Computer-Science/6-895Fall-2005/0A869B43-37A7-4985-B599-9A2752EDD186/0/lecture5.pdf> (3.5.2009.)
- [10] *Dynamic programming* http://en.wikipedia.org/wiki/Dynamic_programming (3.5.2009.)
- [11] *Needleman-Wunsch algorithm*. http://en.wikipedia.org/wiki/Needleman-Wunsch_algorithm (3.5.2009.)
- [12] Needleman S. B., Wunsch C. D. (1970.) *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. J Mol Biol 48 (3): 443-53. doi:10.1016/0022-2836(70)90057-4
- [13] *Hirschberg's algorithm*. http://en.wikipedia.org/wiki/Hirschberg's_algorithm (3.5.2009.)
- [14] Smith T., Waterman M. (1981.) *Identification of Common Molecular Subsequences*. Journal of Molecular Biology 147: 195-197. doi:10.1016/0022-2836(81)90087-5
- [15] Shamir R. (1998.) *Lecture 3: December 27, 1998*. www.cs.tau.ac.il/~rshamir/algmb/98/scribe/pdf/lec03.pdf (3.5.2009.)
- [16] *Postupci za pretraživanje baza bioloških sekvenci*. <http://www.irb.hr/hr/home/ristov/predavanja/TreciTjedan07.ppt> (3.5.2009.)
- [17] Lipman D.J., Pearson W.R. (1985.) *Rapid and sensitive protein similarity searches*. Science 227 (4693): 1435-41
- [18] Lipman D.J., Pearson W.R. (1988.) *Improved tools for biological sequence comparison*. Proc Natl Acad Sci U.S.A. 85(8): 2444-8
- [19] *FASTA and BLAST* (2007.) www.ccl.rutgers.edu/~ouyang/5020/FASTA-BLAST.ppt (3.5.2009.)
- [20] Barton G. (1996.) *Protein Structure prediction – a practical approach*. Oxford University Press. ISBN 0-19-963496-3. http://upload.wikimedia.org/wikipedia/en/c/cd/Document_html_47f1ed1b.gif (3.5.2009.)

- [21] Altschul S.F., Gish W., Miller W., Myers E.W., Lipman D.J. (1990.) *Basic local alignment search tool*. J Mol Biol 215 (3): 403-410. doi:10.1006/jmbi.1990.9999
- [22] BLAST. <http://en.wikipedia.org/wiki/BLAST> (3.5.2009.)
- [23] Altschul S.F., Madden T.L., Schaffer A.A., Zhang Z., Miller W., Lipman D.J. (1997.) *Gapped blast and psi-blast: a new generation of protein database search programs*. Nucleic Acids Res., 25 (17):3389-402
- [24] Liljas L. (2000.) *Lecture about BLAST and FASTA*. <http://www.bioinfo.se/kurser/swell/blasta-fasta.shtml> (3.5.2009.)
- [25] Henikoff S., Henikoff J. (1992.) *Amino acid substitution matrices from protein blocks*. Proc. Natl. Acad. Sci. USA. 89(biochemistry): 10915 - 10919.

7. SAŽETAK

Bioinformatika je znanstvena disciplina koja primjenjuje informacijsku tehnologiju u području molekularne biologije. Jedno od važnijih područja bioinformatike je poravnavanje sekvenci. Cilj poravnavanja sekvenci je algoritamsko pronalaženje i ocjenjivanje sličnosti između sekvenci DNA, RNA i proteina koje bi mogle biti posljedica funkcionalne, strukturalne ili evolucijske veze između tih sekvenci. Poravnavanje sekvenci koristi se prilikom gradnje evolucijskog stabla, usporedbe vrsta, identifikacije proteina i ostale biološke, pa čak i nebiološke primjene.

U ovom seminaru prikazana su četiri osnovna algoritma za poravnavanje sekvenci:

1. Needleman-Wunsch,
2. Smith-Waterman,
3. FASTA algoritam i
4. BLAST algoritam.

Implementacije ovih algoritama, zajedno s bazama bioloških sekvenci, besplatno su dostupne široj javnosti putem interneta.