

Obrada informacija

BIOINFORMATIKA

Doc. dr. sc. Krešimir Križanović

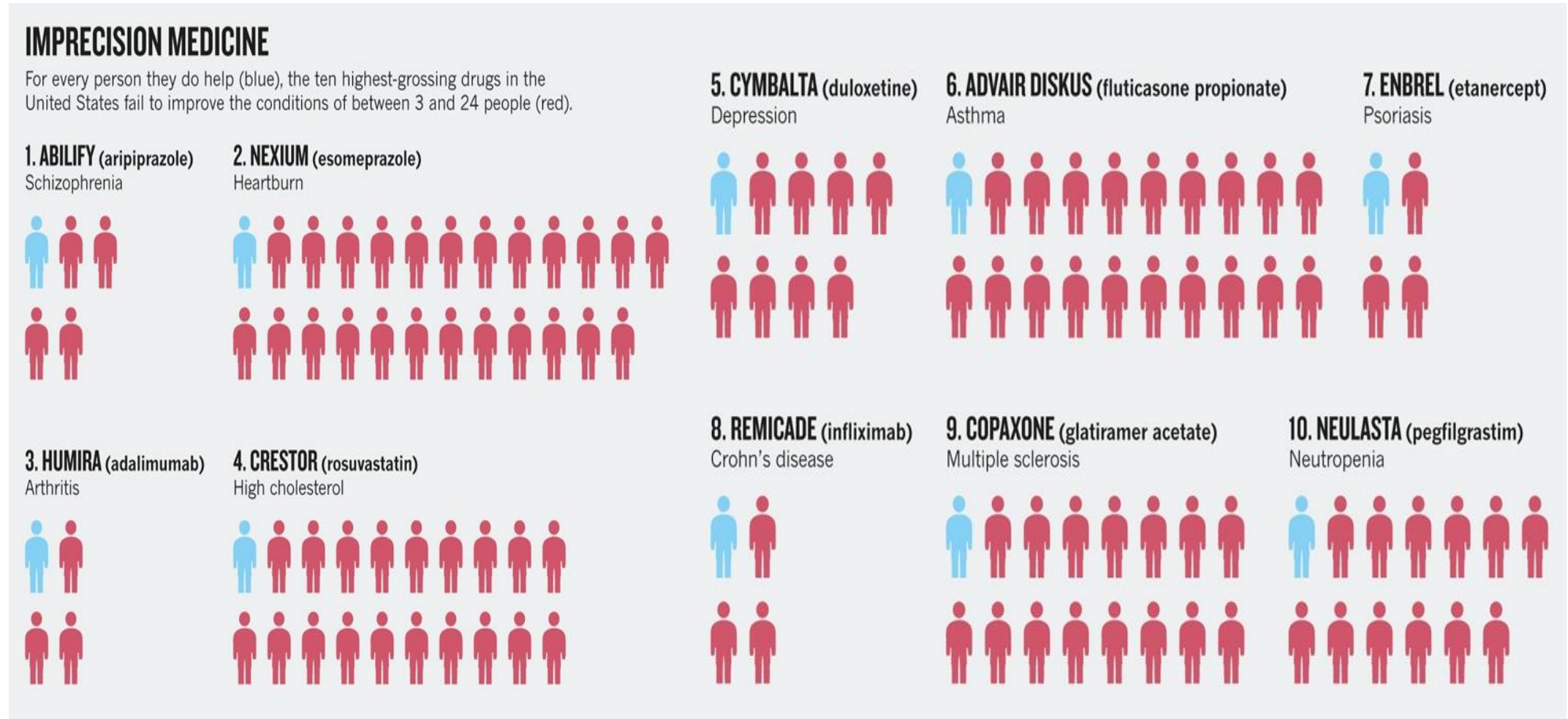
Sadržaj

- Uvod u Bioinformatiku
- Sekvenciranje
- Pojam poravnanja i poravnanja više sekvenci
- Signal u bioinformatici
- Brza Fourierova transformacija
- MAFFT algoritam – poravnanje više sekvenci pomoću brze Fourierove transformacije

Bioinformatika

- Korištenje računala za analizu bioloških podataka, u užem smislu za analizu DNK, RNK i proteina.
- Merriam Webster dictionary:
 - „the collection, classification, storage, and analysis of biochemical and biological information using computers especially as applied to molecular genetics and genomics”
- Osnovni zadaci
 - Sastavljanje genoma
 - Određivanje varijacija u genomima
 - Ekspresija gena
 - Određivanje kompozicije uzoraka
- Primjena u medicini, farmaciji, biologiji, poljoprivredi ...

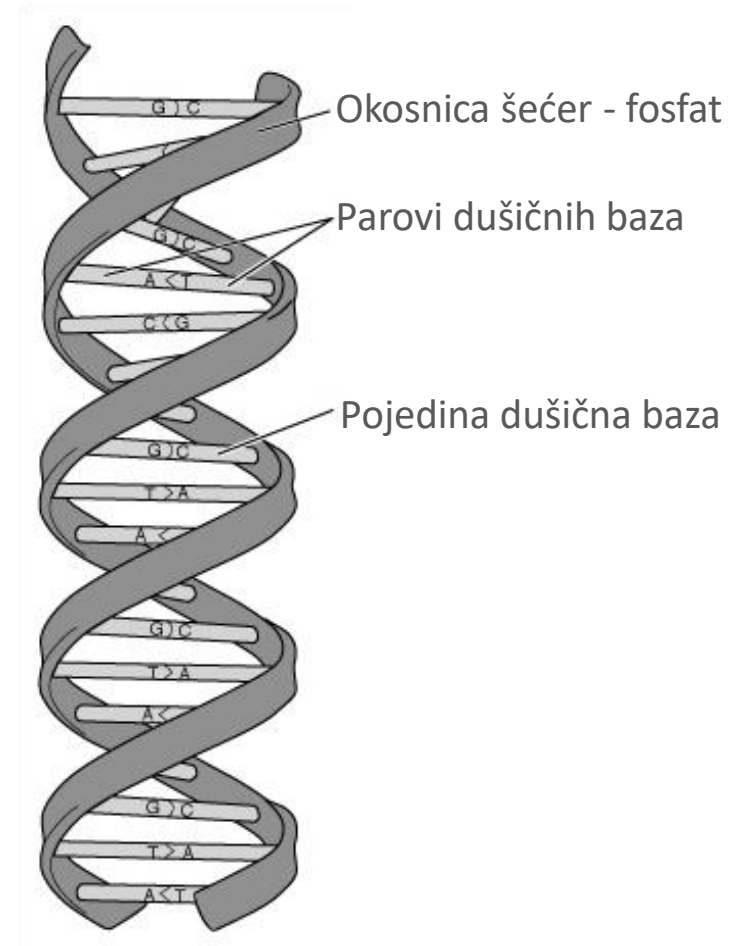
Bioinformatika



Nicholas J. Schork Personalized medicine: *Time for one-person trials*, *Nature*, 2015

Bioinformatika - DNA

- Genetski materijal
 - DNA – eukarioti i prokarioti
 - DNA ili RNA - virusi
- DNA ima dva lanca tvorenih od nukleotida koji se sastoje od šećera deoksiriboze, fosfatne grupe i baze (**A**denin, **T**imin, **G**vanin i **C**itozin)
- Sekvenca DNA nosi genetsku informaciju



Bioinformatika - Centralna dogma molekularne biologije

- Opisuje tijek genetskih informacija unutar bioloških sustava
- Pojednostavljeno:
 - RNA se radi na temelju DNA, Proteini se rade na temelju RNA
 - DNA → RNA → Protein
- „once "information" has passed into protein it cannot get out again”

OPĆI	POSEBNI	NEPOZNATI
DNA → DNA	RNA → DNA	Protein → DNA
DNA → RNA	RNA → RNA	Protein → RNA
RNA → Protein	DNA → Protein	Protein → Protein

Bioinformatika – DNA, RNA i proteini

- DNA – cijeli genetski materijal organizma, sastoji se od kromosoma
- RNA – manji dio genetskog materijala koji se prenosi od jezgre (DNA) prema ribosomima radi izrade proteina (ima i druge funkcije)
- Transkripcija: sinteza RNA (prepisivanjem dijela DNA)
 - Događa se u jezgri (kod eukariota)
- Translacija: sinteza proteina (prevođenje RNA)
 - Sinteza proteinskog lanca koristeći genetski kod RNA molekule kao uputu.
 - Događa se u ribosomima

Bioinformatika - DNA, RNA i proteini

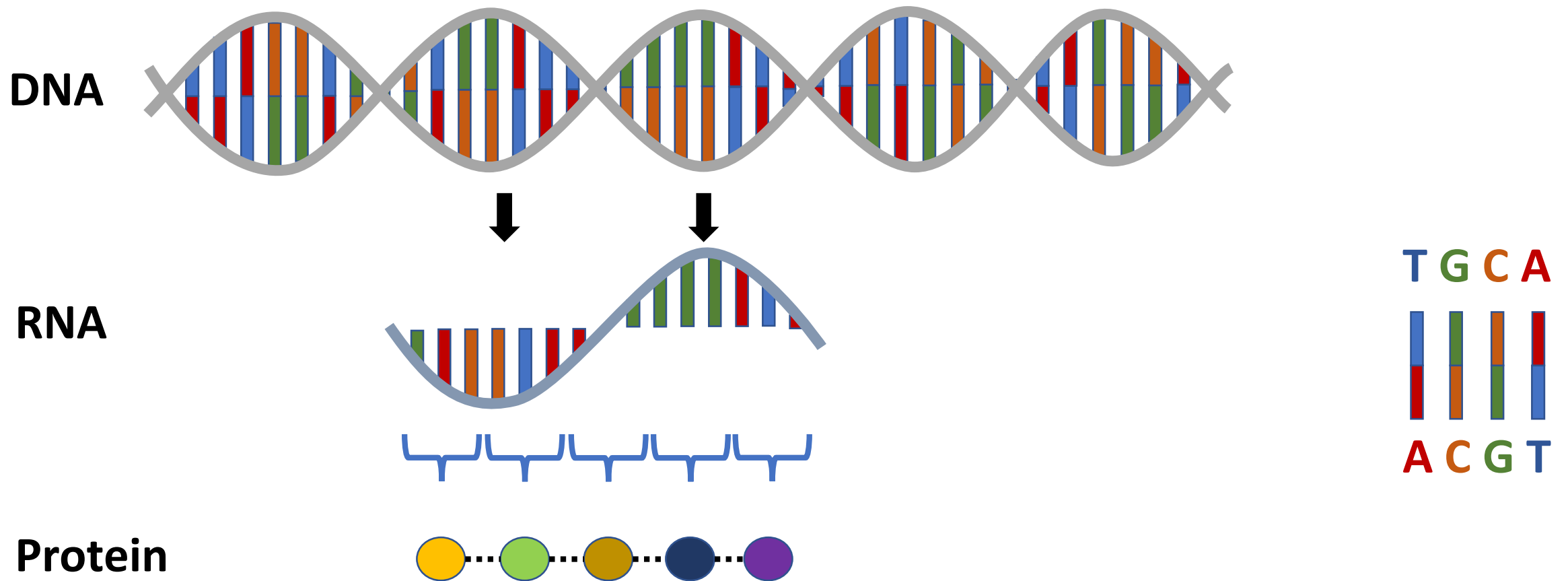
- Geni su specifične sekvence nukleotida koje prenose osobine s roditelja na potomstvo
- Geni su upravo oni dijelovi DNA koji se prepisuju u RNA

Bioinformatika - DNA, RNA i proteini

- Translacija: sinteza proteina (prevođenje RNA)
 - Sinteza proteinskog lanca koristeći genetski kod RNA molekule kao uputu.
 - Molekule DNA i RNA sagrađene su od **4 različita nukleotida**
 - Proteini su sagrađeni od **20 različitih aminokiselina**
 - Kod translacije 3 nukleotida određuju (kodiraju) jednu aminokiselinu
 - Takva tri nukleotida nazivaju se kodon

prva baza kodona (5')	druga baza kodona				treća baza kodona (3')
	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Bioinformatika - DNA, RNA i proteini



Sekvenciranje

- Sekvenciranje je određivanje strukture ili slijeda velikih bioloških (linearnih) molekula
- Sekvenciranje DNA i RNA – određivanje slijeda nukleotidnih baza
- Sekvenciranje proteina – određivanje slijeda aminokiselina
- Postojeće tehnologije za sekvenciranje nisu u stanju pročitati cijele DNA molekule

Sekvenciranje

- Današnja tehnologija ne omogućuje precizno „čitanje” cijelog genoma odjednom
- Veličina genoma:
 - Virus ~ 50.000 baza
 - Bakterija ~ 5.000.000 baza
 - Kvasac ~ 10.000.000 baza
 - Ptice ~ 1.000.000.000 baza
 - Čovjek ~ 3.000.000.000 baza
 - Jedan kromosom ~ 50.000.000 – 250.000.000 baza
 - Neke biljke čak preko 100.000.000.000 baza

Sekvenciranje

- Današnja tehnologija ne omogućuje precizno „čitanje” cijelog genoma odjednom
- Genom se umnaža (50x), nasumično razbija na manje fragmente koje se može „pročitati” – *shotgun sequencing*
 - Fragmente nazivamo očitanjima
- Kako provjeriti je li rezultat dobar?
- Prve metode sekvenciranja: označavanje pojedinih dijelova kromosoma, sekvenciranje i sastavljanje.
- Prvi sastavljeni ljudski genom koštao je oko 3 milijarde \$

Sekvenciranje

Veličina genoma:

Virus ~ 50.000 baza

Bakterija ~ 5.000.000 baza

Kvasac ~ 10.000.000 baza

Ptice ~ 1.000.000.000 baza

Čovjek ~ 3.000.000.000 baza

Jedan kromosom ~ 50.000.000 – 250.000.000 baza

Neke biljke čak preko 100.000.000.000 baza

- Prva generacija
 - Očitavanja srednje duljine (oko **1000** baza), precizno (~ 0,1% pogreške)
 - Jako sporo
- Druga generacija (NGS – *next generation sequencing*)
 - Očitavanja male duljine, tipično **100-200** baza, precizno (~ 1-3% pogreške)
 - Brzo i jeftino
- Treća generacija
 - Očitavanja velike duljine (**deseci tisuća baza**) s relativno velikom greškom (~ 5-15% pogreške)
 - Brzo i jeftino?

Sekvenciranje

- Uređaji nekad



- Uređaji danas



Sekvenciranje – što dalje

- Uređaji za sekvenciranje proizvode skup pročitanih fragmenata koja nazivamo očitavanja.
- Očitavanja kao i poznate sekvence (genomi, geni, proteini ...) zapisuju se u tekstualnom formatu:
- FASTA format (poznate sekvence):

```
>CP027599.1 Escherichia coli strain 97-3250 chromosome, complete genome
ATCCCGGCCCCCGGCAGAACCGACCTATCGTTCTAACGTAAACGTCAAACACACGTTTGATAACTTCGTTG
AAGGTAAATCTAACCAACTGGCGCGCGCGGGCGGCTCGCCAGGTGGCGGATAACCCTGGCGGTGCCTATAA
```

- FASTQ format (očitanja):

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((( (***) ) %%%++ ) (%%%) .1***-+*'' ) **55CCF>>>>>CCCCCCC65
```


Sekvenciranje – što dalje

- Uređaji za sekvenciranje proizvode skup pročitanih fragmenata koja nazivamo očitavanja.
Što možemo napraviti s njima?
- Ovisi o tome što se želi postići!
- Sastavljanje genoma – određivanje do sada nepoznatih sekvenci
- Određivanje varijanti – određivanje razlika između sekvenciranog genoma i sličnog predloška (poznatog genoma)
- Ekspresija gena – koji geni su aktivni u sekvenciranom organizmu
- Metagenomska analiza – određivanja sastava uzorka koji sadrži više organizama

Sekvenciranje – što dalje

- Uređaji za sekvenciranje proizvode skup pročitanih fragmenata koja nazivamo očitavanja.
Što možemo napraviti s njima?

- Ovisi o tome što se želi postići!

- Sastavljanje genoma

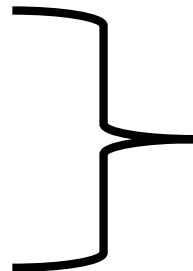


Određivanje **preklapanja**
među očitanjima

- Određivanje varijanti

- Ekspresija gena

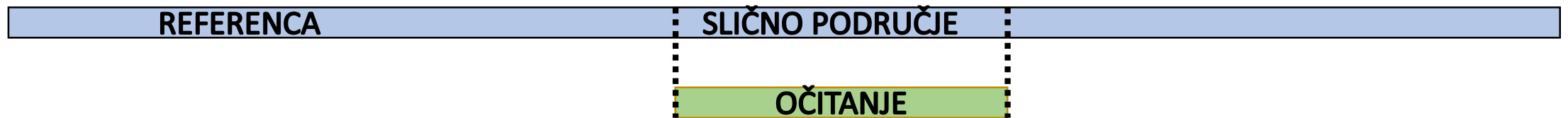
- Metagenomska analiza



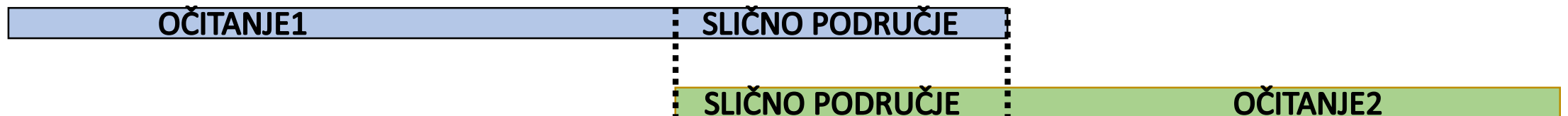
Poravnanje očitavanja na
jedan ili više poznatih
referentnih genoma

Poravnanje

- Poravnanje očitavanja na referencu – pronalaženje pozicije na referenci na kojem je referenca „najsličnija”



- Preklapanje očitavanja – pronalaženje takvih očitavanja kod kojih je kraj jednog očitavanja „sličan” početku drugog



- Pitanje: Kako odrediti „slična” područja na referenci i očitanjima?

Poravnanje

- Referenca i očitavanja predstavljeni su kao nizovi znakova (**A C G T**)
- Algoritmi za preklapanje i poravnanje temelje se na sličnosti nizova znakova.
- Udaljenost uređivanja nizova (Levenshtein-ova odaljenost)
 - Minimalan broj **zamjena, umetanja i brisanja** potrebnih za transformaciju jednog niza u drugi
 - To su upravo vrste pogrešaka koje proizvode uređaji za sekvenciranje

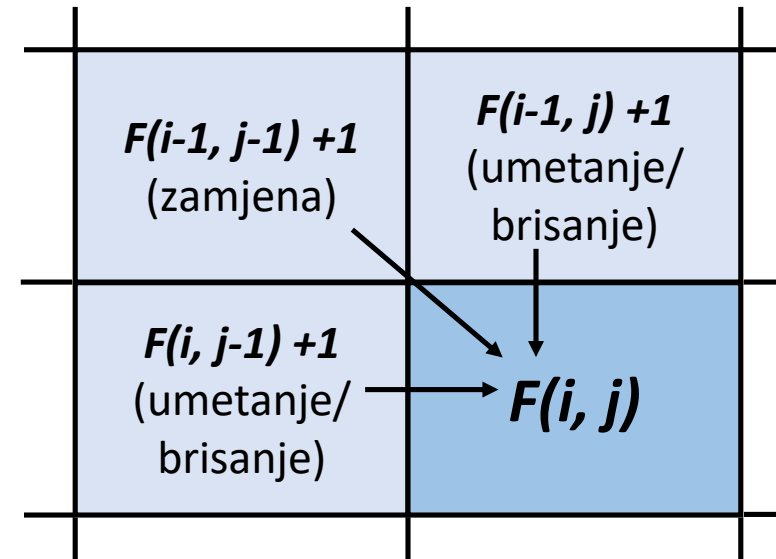
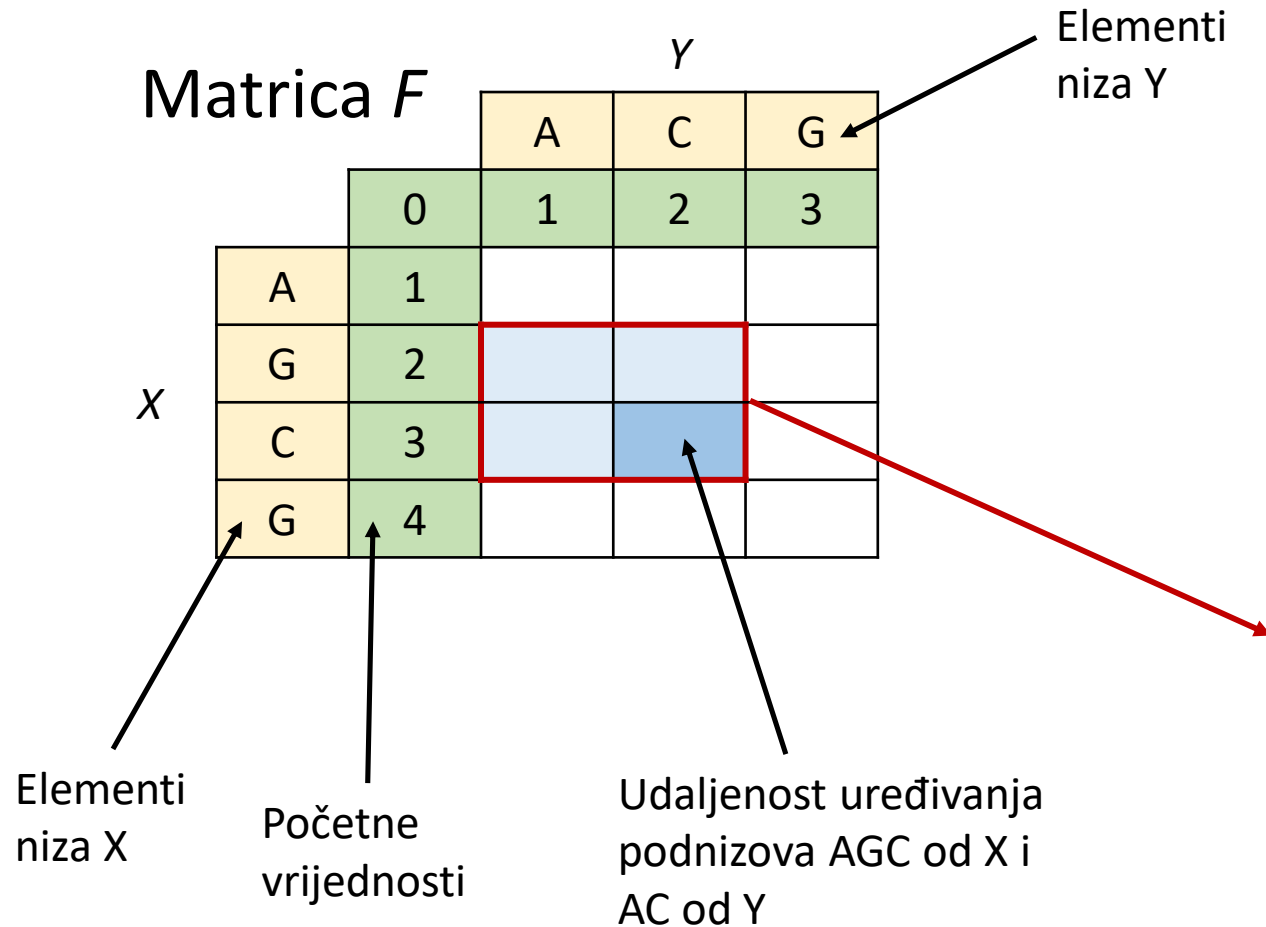
GAT T ACA	-ATTACC C
X	X X X
GAT C ACA	GATTAC- A

- $\text{Min(udaljenost uređivanja)} \sim \text{Max(sličnost)}$

Poravnanje – dinamičko programiranje

- Želimo izračunati udaljenost uređivanja za nizove X i Y
- Neka je duljina niza $X = n$, a duljina niza $Y = m$
- Konstruiramo matricu dimenzija $(n+1) \times (m+1)$
- $F(i, j)$ = udaljenost uređivanja najboljeg poravnanja podnizova od prvih i i j znakova od X i Y
 - $X_1 \dots X_i$ poravnato s $Y_1 \dots Y_j$
- $F(n, m)$ će sadržavati konačnu udaljenost uređivanja
- Kako izračunati $F(i, j)$?

Poravnanje – dinamičko programiranje



Poravnanje – dinamičko programiranje

- Želimo izračunati udaljenost uređivanja za nizove X i Y
- Trebamo odrediti operacije zamjene, umetanja i brisanja kojima ćemo od niza X dobiti niz Y
- Gledamo li usporedbu podnizova AGC i AC ($F(3,2)$), udaljenost uređivanja može biti jedna od tri vrijednosti:
 1. Udaljenost AG i A + trošak zamjene(X_3, Y_2)
 2. Udaljenost AGC i A + trošak praznine (umetanje u Y)
 3. Udaljenost AG i AC + trošak praznine (umetanje u X)
 - Trošak zamjene(X_3, Y_2) je 0 jer su znakovi jednaki (C)
 - Umetanje i brisanje zajedno se nazivaju prazninom
 - Zamjena i praznina u nekim varijantama algoritma mogu imati vrijednosti različite od 1

Matrica F

		Y			
		A	C	G	
X		0	1	2	3
	A	1			
	G	2	1.	3.	
	C	3	2.		
	G	4			

The diagram illustrates edit operations between sequences. A blue-shaded cell at row G, column C contains red numbers 1. and 2. Two arrows originate from this cell: one points diagonally down-right to the cell at row C, column C, and another points horizontally right to the cell at row C, column G. The target cells at (C, C) and (C, G) are also highlighted in blue.

Poravnanje – dinamičko programiranje

- Gledamo li usporedbu podnizova AGC i AC ($F(3,2)$), udaljenost uređivanja može biti jedna od tri vrijednosti:
 1. Udaljenost AG i A + trošak zamjene(X_3, Y_2)
 2. Udaljenost AGC i A + trošak praznine (umetane u Y)
 3. Udaljenost AG i AC + trošak praznine (umetanje u X)
- Prava udaljenost uređivanja je najmanja od te tri vrijednosti!

$$F(i, j) = \min \begin{cases} F(i-1, j-1) + \text{zamjena}(x_i, y_j) \\ F(i-1, j) + \text{praznina}() \\ F(i, j-1) + \text{praznina}() \end{cases}$$

Matrica F

		Y			
		A	C	G	
X		0	1	2	3
	A	1			
	G	2	1.	3.	
	C	3	2.		
	G	4			

Poravnanje – dinamičko programiranje

1. Inicijalizirati prvi redak i stupac matrice
 - Oni predstavljaju usporedbu s praznim podnizom: $i \times \text{praznina}()$
2. Popuniti ostatak matrice s lijeva na desno, od vrha do dna
3. Za svako polje matrice spremiti „od kuda se došlo”
4. $F(n,m)$ sadrži konačnu udaljenost uređivanja
5. Put od $F(n,m)$ natrag do $F(0,0)$ dat će nam koje je operacije (zamjena, umetanje i brisanje) potrebno obaviti da se ostvari najmanja udaljenost uređivanja
6. Wagner-Fisher algoritam za određivanje udaljenost uređivanja (*engl. edit distance*)

Poravnanje – dinamičko programiranje

- Primjer poravnanja nizova $X = \text{AGCG}$ i $Y = \text{ACG}$

Inicijalizacija:

Matrica F

		Y			
		A	C	G	
		0	1	2	3
X	A	1			
	G	2			
	C	3			
	G	4			

Poravnanje – dinamičko programiranje

- Primjer poravnanja nizova $X = \text{AGCG}$ i $Y = \text{ACG}$

Računanje
elemenata
matrice:

Matrica F

		Y			
		A	C	G	
X		0	1	2	3
	A	1	0	1	2
	G	2	1	1	1
	C	3	2	1	2
	G	4	3	2	1

Poravnanje – dinamičko programiranje

- Primjer poravnanja nizova $X = \text{AGCG}$ i $Y = \text{ACG}$

Udaljenost
uređivanja: 1

Poravnanje:
AGCG
A-CG

Matrica F

		Y			
		A	C	G	
X		0	1	2	3
	A	1	0	1	2
	G	2	1	1	1
	C	3	2	1	2
	G	4	3	2	1

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$

Globalno poravnanje (inicijalizacija):

Matrica F

		Y									
		G 1	A 2	T 3	A 4	C 5	G 6	T 7	T 8	A 9	
		0	1	2	3	4	5	6	7	8	9
X	G 1	1									
	T 2	2									
	A 3	3									
	C 4	4									
	C 5	5									

Vrijednosti u „nultom”
retku i stupcu postavljamo
na $(i \times \text{trošak praznine})$
odnosno na $(i \times 1)$.

Te vrijednosti predstavljaju
usporedbu s praznim podnizom.

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$

Globalno poravnanje (inicijalizacija):

Matrica F

		Y								
		G	A	T	A	C	G	T	T	A
		1	2	3	4	5	6	7	8	9
X	G	1								
	T	2								
	A	3								
	C	4								
	C	5								

Vrijednosti u „nultom” retku i stupcu postavljamo na $(i \times \text{trošak praznine})$ odnosno na $(i \times 1)$

Te vrijednosti predstavljaju usporedbu s praznim podnizom. Svaka vrijednost dobivena je na temelju prethodne u početnom retku ili stupcu – **crvene strelice**

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$
Globalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		G	A	T	A	C	G	T	T	A
		1	2	3	4	5	6	7	8	9
X	G	1								
	T	2								
	A	3								
	C	4								
	C	5								

Vrijedost u svakom polju izračunamo na temelju tri prethodno izračunate vrijednost

$F(1,1)$ računamo kao minimum:

1. $F(0,0) + \text{zamjena}(G,G) = 0+0 = \mathbf{0}$
2. $F(1,0) + \text{praznina}() = 1+1 = 2$
3. $F(0,1) + \text{praznina}() = 1+1 = 2$

Vrijednost polja $F(1,1)$ je 0

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$

Globalno poravnanje (popunjavanje matrice):

Matrica F

		Y									
		G 1	A 2	T 3	A 4	C 5	G 6	T 7	T 8	A 9	
		0	1	2	3	4	5	6	7	8	9
X	G 1	1	0								
	T 2	2									
	A 3	3									
	C 4	4									
	C 5	5									

Vrijednost polja $F(1,1)$ je 0, te je izračunata iz polja $F(0,0)$ – **crvena strelica**

Izračunajmo još nekoliko vrijednosti!

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$
Globalno poravnanje (popunjavanje matrice):

Matrica F

		Y									
		G 1	A 2	T 3	A 4	C 5	G 6	T 7	T 8	A 9	
X		0	1	2	3	4	5	6	7	8	9
	G 1	1	0	1	2	3	4	5	6	7	8
	T 2	2	1								
	A 3	3									
	C 4	4									
	C 5	5									

Izračunajmo još nekoliko vrijednosti!

$F(2,2)$ računamo kao minimum:

1. $F(1,1) + \text{zamjena}(A,T) = 0+1 = \mathbf{1}$
2. $F(2,1) + \text{praznina}() = 1+1 = 2$
3. $F(1,2) + \text{praznina}() = 1+1 = 2$

Vrijednost polja $F(2,2)$ je 1 i izračunata je na temelju polja $F(1,1)$

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$
Globalno poravnanje (popunjavanje matrice):

Matrica F

		Y									
		G 1	A 2	T 3	A 4	C 5	G 6	T 7	T 8	A 9	
X		0	1	2	3	4	5	6	7	8	9
	G 1	1	0	1	2	3	4	5	6	7	8
	T 2	2	1	1							
	A 3	3									
	C 4	4									
	C 5	5									

Izračunajmo još nekoliko vrijednosti!

$F(2,3)$ računamo kao minimum:

1. $F(1,2) + \text{zamjena}(T,T) = 1+0 = \mathbf{1}$
2. $F(2,2) + \text{praznina}() = 1+1 = 2$
3. $F(1,3) + \text{praznina}() = 2+1 = 3$

Vrijednost polja $F(2,3)$ je 1 i izračunata je na temelju polja $F(1,2)$

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$
Globalno poravnanje (popunjavanje matrice):

Matrica F

		Y									
		G	A	T	A	C	G	T	T	A	
		1	2	3	4	5	6	7	8	9	
X		0	1	2	3	4	5	6	7	8	9
	G	1	0	1	2	3	4	5	6	7	8
	T	2	1	1	1	2	3	4			
	A	3									
	C	4									
	C	5									

Izračunajmo još nekoliko vrijednosti!

$F(2,7)$ računamo kao minimum:

1. $F(1,6) + \text{zamjena}(T,T) = 5+0 = \mathbf{5}$
2. $F(2,6) + \text{praznina}() = 4+1 = \mathbf{5}$
3. $F(1,7) + \text{praznina}() = 6+1 = 7$

Vrijednost polja $F(2,7)$ je 5, ali može biti izračunata je na temelju polja $F(1,6)$ i na temelju polja $F(2,6)$.

Više slijedova operacije koje vode do iste udaljenosti uređivanja – odaberemo bilo koju.

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$
Globalno poravnanje (popunjavanje matrice):

Matrica F

		Y									
		G 1	A 2	T 3	A 4	C 5	G 6	T 7	T 8	A 9	
X		0	1	2	3	4	5	6	7	8	9
	G 1	1	0	1	2	3	4	5	6	7	8
	T 2	2	1	1	1	2	3	4	5		
	A 3	3									
	C 4	4									
	C 5	5									

Izračunajmo još nekoliko vrijednosti!

$F(2,7)$ računamo kao minimum:

1. $F(1,6) + \text{zamjena}(T,T) = 5+0 = \mathbf{5}$
2. $F(2,6) + \text{praznina}() = 4+1 = \mathbf{5}$
3. $F(1,7) + \text{praznina}() = 6+1 = 7$

Vrijednost polja $F(2,7)$ je 5, ali može biti izračunata je na temelju polja $F(1,6)$ i na temelju polja $F(2,6)$.

Više slijedova operacije koje vode do iste udaljenosti uređivanja – odaberemo bilo koju.

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$

Globalno poravnanje (konačno rješenje):

Matrica F

		Y									
		G 1	A 2	T 3	A 4	C 5	G 6	T 7	T 8	A 9	
X		0	1	2	3	4	5	6	7	8	9
	G 1	1	0	1	2	3	4	5	6	7	8
	T 2	2	1	1	1	2	3	4	5	6	7
	A 3	3	2	1	2	1	2	3	4	5	6
	C 4	4	3	2	2	2	1	2	3	4	5
	C 5	5	4	3	3	3	2	2	3	4	5

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$
Globalno poravnanje (konačno rješenje):

Matrica F

		Y									
		G	A	T	A	C	G	T	T	A	
		1	2	3	4	5	6	7	8	9	
X		0	1	2	3	4	5	6	7	8	9
	G	1	0	1	2	3	4	5	6	7	8
	T	2	1	1	1	2	3	4	5	6	7
	A	3	2	1	2	1	2	3	4	5	6
	C	4	3	2	2	2	1	2	3	4	5
C	5	4	3	3	3	2	2	3	4	5	

Konačno rješenje, odnosno minimalnu udaljenost možemo očitati u zadnjem retku i zadnjem stupcu, u polju $F(5,9)$.
 $F(5,9) = 5$, dakle udaljenost uređivanja nizova GTACC i GATACGTTA je 5.

Poravnanje – primjer

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{GATACGTTA}$
Globalno poravnanje (konačno rješenje):

Matrica F

		Y									
		G 1	A 2	T 3	A 4	C 5	G 6	T 7	T 8	A 9	
X		0	1	2	3	4	5	6	7	8	9
	G 1	1	0	1	2	3	4	5	6	7	8
	T 2	2	1	1	1	2	3	4	5	6	7
	A 3	3	2	1	2	1	2	3	4	5	6
	C 4	4	3	2	2	2	1	2	3	4	5
	C 5	5	4	3	3	3	2	2	3	4	5

Poravnanje nizova GTACC i GATACGTTA
možemo odrediti vraćajući se unatrag
pomoću strelica.

Poravnanje koje nam daje najmanju
udaljenost uređivanja je:

G-TACC---
GATACGTTA

Poravnanje – dinamičko programiranje

- Ovako dobiveno poravnanje naziva se **GLOBALNIM** poravnanjem. Postoje i druge vrste poravnanja.

NAPOMENE:

- Trošak zamjene i praznine može se mijenjati ovisno o namjeni algoritma
- Umjesto udaljenosti uređivanja , može se gledati sličnost. U tom slučaju se prilikom računanja elemenata matrice gleda maksimum od tri vrijednosti, trošak praznine ima negativnu vrijednost, dok zamjena ima pozitivnu vrijednost ako su znakovi jednaki, a negativnu ako su različiti.

Poravnanje – preklapanje

- Imamo dvije sekvence $X = \text{CAGCACTTGGATTCTCGG}$ i $Y = \text{CAGCGTGG}$
 - Kako izgleda njihovo globalno poravnanje ?

CAGCACTTGGATTCTCGG

CAGC-----G--T-----GG

- Ono što želimo postići je:

CAGCA-CTTGGATTCTCGG

---CAGCGTGG-----

- Preklapanje je poravnanje u kojem su praznine na početku i kraju zanemarene.

Poravnanje – preklapanje

1. Inicijalizirati prvi redak i stupac matrice
 - Postavljaju se na 0 jer poravnanje može početi bilo gdje
2. Popuniti ostatak matrice s lijeva na desno, od vrha do dna
 - Koristimo sličnost umjesto udaljenosti
3. Za svako polje matrice spremiti „od kuda se došlo”
4. Konačno rješenje je maksimum u zadnjem retku i stupcu
5. Put od rješenja natrag do $F(0,0)$ dat će nam koje je operacije potrebno obaviti da se ostvari najmanja udaljenost uređivanja
6. Needleman-Wunsch algoritam za polu-globalno poravnanje

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$ pomoću **polu-globalnog poravnanja**.

Pri tome ćemo koristiti sličnost umjesto udaljenosti, na sljedeći način:

- Podudaranje znakova povećava sličnost za jedan (zamjena)
- Različiti znakovi smanjuju sličnost za jedan (zamjena)
- Umetanja i brisanja smanjuju sličnost za jedan (praznina)

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (inicijalizacija):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0								
	T ₂	0								
	A ₃	0								
	C ₄	0								
	C ₅	0								

Vrijednosti u „nultom” retku i stupcu postavljamo na 0 jer preklapanje može početi nizom praznina. Bez obzira gdje počinje preklapanje, **sličnost** je na početku nula!

Možemo dodati i strelice prema prethodnim poljima, radi lakšeg računanja poravnanja na kraju.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0								
	T ₂	0								
	A ₃	0								
	C ₄	0								
	C ₅	0								

Računamo vrijednost u polje $F(1,1)$ na temelju tri poznate vrijednosti.

$F(1,1)$ računamo kao **maksimum**:

1. $F(0,0) + \text{zamjena}(G,T) = 0 + (-1) = \textbf{-1}$
2. $F(1,0) + \text{praznina}() = 0 + (-1) = \textbf{-1}$
3. $F(0,1) + \text{praznina}() = 0 + (-1) = \textbf{-1}$

Vrijednost polja $F(1,1)$ je -1

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1							
	T ₂	0								
	A ₃	0								
	C ₄	0								
	C ₅	0								

$F(1,1)$ računamo kao **maksimum**:

1. $F(0,0) + \text{zamjena}(G,T) = 0 + (-1) = -1$
2. $F(1,0) + \text{praznina}() = 0 + (-1) = -1$
3. $F(0,1) + \text{praznina}() = 0 + (-1) = -1$

Vrijednost polja $F(1,1)$ je -1, i izračunata je na temelju bilo kojeg od tri poznata polja, odaberimo $F(0,0)$.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1							
	T ₂	0								
	A ₃	0								
	C ₄	0								
	C ₅	0								

Računamo vrijednost u polje $F(1,2)$.

$F(1,2)$ računamo kao **maksimum**:

1. $F(0,1) + \text{zamjena}(G,T) = 0 + (-1) = -1$
2. $F(1,1) + \text{praznina}() = -1 + (-1) = -2$
3. $F(0,2) + \text{praznina}() = 0 + (-1) = -1$

Vrijednost polja $F(1,1)$ je -1, i izračunata je na temelju polja $F(0,1)$ – tako odabiremo.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1				
	T ₂	0								
	A ₃	0								
	C ₄	0								
	C ₅	0								

Računamo vrijednost u polje $F(1,6)$.

$F(1,6)$ računamo kao **maksimum**:

1. $F(0,5) + \text{zamjena}(G,G) = 0 + 1 = \mathbf{1}$
2. $F(1,5) + \text{praznina}() = -1 + (-1) = -2$
3. $F(0,6) + \text{praznina}() = 0 + (-1) = -1$

Vrijednost polja $F(1,6)$ je +1, i izračunata je na temelju polja $F(0,5)$ – ovaj put ništa ne biraemo jer je najbolji rezultat jedinstven.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1			
	T ₂	0								
	A ₃	0								
	C ₄	0								
	C ₅	0								

Računamo vrijednost u polje $F(1,6)$.

Vrijednost polja $F(1,6)$ je +1, i izračunata je na temelju polja $F(0,5)$ – ovaj put ništa ne biraмо jer je najbolji rezultat jednoznačan.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1	0	-1	-1
	T ₂	0								
	A ₃	0								
	C ₄	0								
	C ₅	0								

Računamo vrijednost u polje $F(2,1)$.

$F(2,1)$ računamo kao **maksimum**:

1. $F(1,0) + \text{zamjena}(T,T) = 0 + 1 = \mathbf{1}$
2. $F(2,0) + \text{praznina}() = 0 + (-1) = -1$
3. $F(1,1) + \text{praznina}() = -1 + (-1) = -2$

Vrijednost polja $F(2,1)$ je +1, i izračunata je na temelju polja $F(1,0)$.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	-1	+1	0	-1
	T ₂	0	+1							
	A ₃	0								
	C ₄	0								
	C ₅	0								

Računamo vrijednost u polje $F(2,1)$.

$F(2,1)$ računamo kao **maksimum**:

1. $F(1,0) + \text{zamjena}(T,T) = 0 + 1 = \mathbf{1}$
2. $F(2,0) + \text{praznina}() = 0 + (-1) = -1$
3. $F(1,1) + \text{praznina}() = -1 + (-1) = -2$

Vrijednost polja $F(2,1)$ je +1, i izračunata je na temelju polja $F(1,0)$.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1	0	-1	-1
	T ₂	0	+1							
	A ₃	0								
	C ₄	0								
	C ₅	0								

Diagram showing red arrows indicating the path for calculating $F(2,2)$ from $F(1,1)$, $F(2,1)$, and $F(1,2)$. A black arrow points from the text on the right to the cell $F(2,2)$.

Računamo vrijednost u polje $F(2,2)$.

$F(2,2)$ računamo kao **maksimum**:

1. $F(1,1) + \text{zamjena}(T,T) = -1 + 1 = \mathbf{0}$
2. $F(2,1) + \text{praznina}() = 1 + (-1) = \mathbf{0}$
3. $F(1,2) + \text{praznina}() = -1 + (-1) = -2$

Vrijednost polja $F(2,2)$ je 0, i izračunata je na temelju polja $F(1,1)$ – odabirom.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1	0	-1	-1
	T ₂	0	+1	0						
	A ₃	0								
	C ₄	0								
	C ₅	0								

Računamo vrijednost u polje $F(2,2)$.

$F(2,2)$ računamo kao **maksimum**:

1. $F(1,1) + \text{zamjena}(T,T) = -1 + 1 = \mathbf{0}$
2. $F(2,1) + \text{praznina}() = 1 + (-1) = \mathbf{0}$
3. $F(1,2) + \text{praznina}() = -1 + (-1) = -2$

Vrijednost polja $F(2,2)$ je 0, i izračunata je na temelju polja $F(1,1)$ – odabirom.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1	0	-1	-1
	T ₂	0	+1	0	-1	-2	0	+2	+1	0
	A ₃	0	0	0	-1	0	-1	+1	+1	+2
	C ₄	0	-1	-1						
	C ₅	0								

Računamo vrijednost u polje $F(4,3)$.

$F(4,3)$ računamo kao **maksimum**:

1. $F(3,2) + \text{zamjena}(\text{C}, \text{C}) = 0 + 1 = \mathbf{1}$
2. $F(4,2) + \text{praznina}() = -1 + (-1) = -2$
3. $F(3,3) + \text{praznina}() = -1 + (-1) = -2$

Vrijednost polja $F(4,3)$ je 1, i izračunata je na temelju polja $F(3,2)$.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (popunjavanje matrice):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1	0	-1	-1
	T ₂	0	+1	0	-1	-2	0	+2	+1	0
	A ₃	0	0	0	-1	0	-1	+1	+1	+2
	C ₄	0	-1	-1	+1					
	C ₅	0								

Računamo vrijednost u polje $F(4,3)$.

$F(4,3)$ računamo kao **maksimum**:

1. $F(3,2) + \text{zamjena}(\text{C}, \text{C}) = 0 + 1 = \mathbf{1}$
2. $F(4,2) + \text{praznina}() = -1 + (-1) = -2$
3. $F(3,3) + \text{praznina}() = -1 + (-1) = -2$

Vrijednost polja $F(4,3)$ je 1, i izračunata je na temelju polja $F(3,2)$.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (konačno rješenje):

Matrica F

		Y									
		T 1	T 2	C 3	A 4	C 5	G 6	T 7	T 8	A 9	
X		0	0	0	0	0	0	0	0	0	
	G 1	0	-1	-1	-1	-1	-1	+1	0	-1	-1
	T 2	0	+1	0	-1	-2	-2	0	+2	+1	0
	A 3	0	0	0	-1	0	-1	-1	+1	+1	+2
	C 4	0	-1	-1	+1	0	+1	0	0	0	+1
	C 5	0	-1	-2	0	0	+1	0	-1	-1	0

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (konačno rješenje):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1	0	-1	-1
	T ₂	0	+1	0	-1	-2	0	+2	+1	0
	A ₃	0	0	0	-1	0	-1	+1	+1	+2
	C ₄	0	-1	-1	+1	0	+1	0	0	+1
	C ₅	0	-1	-2	0	0	+1	0	-1	-1

Najveću sličnost za **polu-globalno** poravnanje tražimo u **zadnjem retku i zadnjem stupcu**.

Poravnanje – preklapanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GTACC}$ i $Y = \text{TTCACGTTA}$

Poluglobalno poravnanje (konačno rješenje):

Matrica F

		Y								
		T ₁	T ₂	C ₃	A ₄	C ₅	G ₆	T ₇	T ₈	A ₉
X		0	0	0	0	0	0	0	0	0
	G ₁	0	-1	-1	-1	-1	+1	0	-1	-1
	T ₂	0	+1	0	-1	-2	0	+2	+1	0
	A ₃	0	0	0	-1	0	-1	+1	+1	+2
	C ₄	0	-1	-1	+1	0	+1	0	0	+1
	C ₅	0	-1	-2	0	0	+1	0	-1	-1

Najveću sličnost za **polu-globalno** poravnanje tražimo u **zadnjem retku i zadnjem stupcu**.

Poravnanje možemo rekonstruirati vraćanjem unatrag do polja $F(0,0)$ i produživanjem do polja $F(n,m)$, duž retka ili stupca.

Poravnanje:

-----G-TACC
TTCACGTTA--

Poravnanje – lokalno poravnanje

- Čest problem od biološkog interesa je usporedba dva dugačka niza i traženje lokalnih područja sličnosti.
 - Npr. sačuvanost gena između vrsta
 - Prepoznavanje kodirajućih regija u genomu

Lokalno poravnanje:

1. Inicijalizirati prvi redak i stupac matrice

- Postavljaju se na 0 jer poravnanje može početi s prazninama

2. Popuniti ostatak matrice s lijeva na desno, od vrha do dna računajući sličnost nizova. U svakom trenutku ako sličnost postane manja od 0, postavljamo je na 0.

4. Za konačno rješenje tražimo maksimalnu vrijednost u bilo kojem polju matrice

- Poravnanje je lokalno, tražimo dijelove nizova koji su slični

6. Smith-Waterman algoritam za LOKALNOG poravnanja

Poravnanje – lokalno poravnanje

PRIMJER: Želimo pronaći poravnanje nizova $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću **lokalnog poravnanja**.

Pri tome ćemo koristiti sličnost umjesto udaljenosti, na sljedeći način:

- Podudaranje znakova povećava sličnost za **dva** (zamjena)
- Različiti znakovi smanjuju sličnost za **jedan** (zamjena)
- Umetanja i brisanja smanjuju sličnost za **dva** (praznina)

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
X		0	0	0	0	0	0	0	0
	G ₁	0							
	A ₂	0							
	T ₃	0							
	C ₄	0							
	A ₅	0							
	T ₆	0							
	A ₇	0							
	T ₈	0							
	T ₉	0							

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
poravnanja (inicijalizacija):

Vrijednosti u „nultom” retku i stupcu postavljamo na 0 kao i kod polu-globalnog poravnanja. Bez obzira gdje počinje preklapanje, **sličnost** je na početku nula!

Strelice prema prethodnim poljima u nultom retku i stupcu nisu potrebne jer je poravnanje lokalno.

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
X	G ₁	0							
	A ₂	0							
	T ₃	0							
	C ₄	0							
	A ₅	0							
	T ₆	0							
	A ₇	0							
	T ₈	0							
	T ₉	0							

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(1,1)$ na temelju tri
 poznate vrijednosti.

$F(1,1)$ računamo kao **maksimum**:

1. $F(0,0) + \text{zamjena}(G,T) = 0 + (-1) = -1$

2. $F(1,0) + \text{praznina}() = 0 + (-2) = -2$

3. $F(0,1) + \text{praznina}() = 0 + (-2) = -2$

Praznina
 smanjuje
 sličnost za 2!

*Najveća izračunata vrijednost je -1, koja je
 manja od 0. Stoga vrijednost polja $F(1,1)$
 postavljamo na 0 i ne postavljamo strelicu.*

To znači da sa računanjem sličnosti
 počinjemo iz početka – do ovog trenutka
 nizovi nisu slični

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
X		0	0	0	0	0	0	0	0
	G ₁	0	0						
	A ₂	0							
	T ₃	0							
	C ₄	0							
	A ₅	0							
	T ₆	0							
	A ₇	0							
	T ₈	0							
	T ₉	0							

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(1,2)$ na temelju tri poznate vrijednosti.

$F(1,2)$ računamo kao **maksimum**:

1. $F(0,1) + \text{zamjena}(G,C) = 0 + (-1) = -1$

2. $F(1,1) + \text{praznina}() = 0 + (-2) = -2$

3. $F(0,2) + \text{praznina}() = 0 + (-2) = -2$

Praznina smanjuje sličnost za 2!

Najveća izračunata vrijednost je -1, koja je manja od 0. Stoga vrijednost polja $F(1,2)$ postavljamo na 0 i **ne postavljamo strelicu**.

To znači da sa računanjem sličnosti počinjemo iz početka – do ovog trenutka nizovi nisu slični

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
X	G ₁	0	0	0					
	A ₂	0							
	T ₃	0							
	C ₄	0							
	A ₅	0							
	T ₆	0							
	A ₇	0							
	T ₈	0							
	T ₉	0							

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(1,3)$ na temelju tri
 poznate vrijednosti.

$F(1,3)$ računamo kao **maksimum**:

1. $F(0,2) + \text{zamjena}(G,G) = 0 + 2 = 2$
2. $F(1,2) + \text{praznina}() = 0 + (-2) = -2$
3. $F(0,3) + \text{praznina}() = 0 + (-2) = -2$

Podudaranje
 povećava
 sličnost za 2!

*Najveća izračunata vrijednost je 2, koja je
 veća od 0.*

*Stoga vrijednost polja $F(1,3)$ postavljamo na
 2 i postavljamo strelicu na polje $F(0,2)$.*

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
G ₁	0	0	0	2					
A ₂	0								
T ₃	0								
C ₄	0								
A ₅	0								
T ₆	0								
A ₇	0								
T ₈	0								
T ₉	0								

X

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(1,3)$ na temelju tri
 poznate vrijednosti.

$F(1,3)$ računamo kao **maksimum**:

1. $F(0,2) + \text{zamjena}(G,G) = 0 + 2 = 2$
2. $F(1,2) + \text{praznina}() = 0 + (-2) = -2$
3. $F(0,3) + \text{praznina}() = 0 + (-2) = -2$

Podudaranje
 povećava
 sličnost za 2!

*Najveća izračunata vrijednost je 2, koja je
 veća od 0.*

*Stoga vrijednost polja $F(1,3)$ postavljamo na
 2 i postavljamo strelicu na polje $F(0,2)$.*

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
X	G ₁	0	0	0	2	0	0	2	0
	A ₂	0	0	0	0				
	T ₃	0							
	C ₄	0							
	A ₅	0							
	T ₆	0							
	A ₇	0							
	T ₈	0							
	T ₉	0							

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(2,4)$ na temelju tri
 poznate vrijednosti.

$F(2,4)$ računamo kao **maksimum**:

1. $F(1,3) + \text{zamjena}(A,T) = 2 + (-1) = \mathbf{1}$
2. $F(2,3) + \text{praznina}() = 0 + (-2) = -2$
3. $F(1,4) + \text{praznina}() = 0 + (-2) = -2$

Nepodudaranje
 smanjuje
 sličnost za 1!

Najveća izračunata vrijednost je 1, koja je
 veća od 0.

Stoga vrijednost polja $F(2,4)$ postavljamo na
 1 i postavljamo strelicu na polje $F(1,3)$.

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
G ₁	0	0	0	2	0	0	2	0	2
A ₂	0	0	0	0	1				
T ₃	0								
C ₄	0								
A ₅	0								
T ₆	0								
A ₇	0								
T ₈	0								
T ₉	0								

X

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(2,4)$ na temelju tri
 poznate vrijednosti.

$F(2,4)$ računamo kao **maksimum**:

1. $F(1,3) + \text{zamjena}(A,T) = 2 + (-1) = \mathbf{1}$
2. $F(2,3) + \text{praznina}() = 0 + (-2) = -2$
3. $F(1,4) + \text{praznina}() = 0 + (-2) = -2$

Nepodudaranje
 smanjuje
 sličnost za 1!

Najveća izračunata vrijednost je 1, koja je
 veća od 0.

Stoga vrijednost polja $F(2,4)$ postavljamo na
 1 i postavljamo strelicu na polje $F(1,3)$.

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
G ₁	0	0	0	2	0	0	2	0	2
A ₂	0	0	0	0	1	2	0	1	0
T ₃	0	2	0	0	2	0	1	0	0
C ₄	0	0	4	2	0	1	0	3	1
A ₅	0	0	2	3	1	2	0	1	2
T ₆	0	2	0						
A ₇	0								
T ₈	0								
T ₉	0								

X

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(6,3)$ na temelju tri
 poznate vrijednosti.

$F(6,3)$ računamo kao **maksimum**:

1. $F(5,2) + \text{zamjena}(T,G) = 2 + (-1) = \mathbf{1}$
2. $F(6,2) + \text{praznina}() = 0 + (-2) = -2$
3. $F(5,3) + \text{praznina}() = 3 + (-2) = \mathbf{1}$

*Najveća izračunata vrijednost je 1, koja je
 veća od 0.*

*Stoga vrijednost polja $F(6,3)$ postavljamo na
 1 i postavljamo strelicu na polje $F(5,2)$ –
 odabirom.*

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
G ₁	0	0	0	2	0	0	2	0	2
A ₂	0	0	0	0	1	2	0	1	0
T ₃	0	2	0	0	2	0	1	0	0
C ₄	0	0	4	2	0	1	0	3	1
A ₅	0	0	2	3	1	2	0	1	2
T ₆	0	2	0	1					
A ₇	0								
T ₈	0								
T ₉	0								

X

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(6,3)$ na temelju tri
 poznate vrijednosti.

$F(6,3)$ računamo kao **maksimum**:

1. $F(5,2) + \text{zamjena}(T,G) = 2 + (-1) = \mathbf{1}$
2. $F(6,2) + \text{praznina}() = 0 + (-2) = -2$
3. $F(5,3) + \text{praznina}() = 3 + (-2) = \mathbf{1}$

*Najveća izračunata vrijednost je 1, koja je
 veća od 0.*

*Stoga vrijednost polja $F(6,3)$ postavljamo na
 1 i postavljamo strelicu na polje $F(5,2)$ –
 odabirom.*

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
G ₁	0	0	0	2	0	0	2	0	2
A ₂	0	0	0	0	1	2	0	1	0
T ₃	0	2	0	0	2	0	1	0	0
C ₄	0	0	4	2	0	1	0	3	1
A ₅	0	0	2	3	1	2	0	1	2
T ₆	0	2	0	1					
A ₇	0								
T ₈	0								
T ₉	0								

X

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(6,4)$ na temelju tri
 poznate vrijednosti.

$F(6,3)$ računamo kao **maksimum**:

1. $F(5,3) + \text{zamjena}(T,T) = 3 + 2 = \mathbf{5}$
2. $F(6,3) + \text{praznina}() = 1 + (-2) = -1$
3. $F(5,4) + \text{praznina}() = 1 + (-2) = -1$

Najveća izračunata vrijednost je 5, koja je
 veća od 0.

Stoga vrijednost polja $F(6,4)$ postavljamo na
 5 i postavljamo strelicu na polje $F(5,3)$.

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
G ₁	0	0	0	2	0	0	2	0	2
A ₂	0	0	0	0	1	2	0	1	0
T ₃	0	2	0	0	2	0	1	0	0
C ₄	0	0	4	2	0	1	0	3	1
A ₅	0	0	2	3	1	2	0	1	2
T ₆	0	2	0	1	5				
A ₇	0								
T ₈	0								
T ₉	0								

X

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (popunjavanje matrice):

Računamo vrijednost u polje $F(6,4)$ na temelju tri
 poznate vrijednosti.

$F(6,3)$ računamo kao **maksimum**:

1. $F(5,3) + \text{zamjena}(T,T) = 3 + 2 = \mathbf{5}$
2. $F(6,3) + \text{praznina}() = 1 + (-2) = -1$
3. $F(5,4) + \text{praznina}() = 1 + (-2) = -1$

*Najveća izračunata vrijednost je 5, koja je
 veća od 0.*

*Stoga vrijednost polja $F(6,4)$ postavljamo na
 5 i postavljamo strelicu na polje $F(5,3)$.*

Poravnanje – lokalno poravnanje

Matrica F

Y

		T 1	C 2	G 3	T 4	A 5	G 6	C 7	G 8
		0	0	0	0	0	0	0	0
G 1	0	0	0	2	0	0	2	0	2
A 2	0	0	0	0	1	2	0	1	0
T 3	0	2	0	0	2	0	1	0	0
C 4	0	0	4	2	0	1	0	3	1
A 5	0	0	2	3	1	2	0	1	2
T 6	0	2	0	1	5	3	1	0	0
A 7	0	0	1	0	3	7	5	3	1
T 8	0	2	0	0	2	5	6	4	2
T 9	0	2	1	0	2	3	4	5	3

X

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
poravnanja (popunjavanje matrice):

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
X	G ₁	0	0	0	2	0	0	2	0
	A ₂	0	0	0	0	1	2	0	1
	T ₃	0	2	0	0	2	0	1	0
	C ₄	0	0	4	2	0	1	0	3
	A ₅	0	0	2	3	1	2	0	1
	T ₆	0	2	0	1	5	3	1	0
	A ₇	0	0	1	0	3	7	5	3
	T ₈	0	2	0	0	2	5	6	4
	T ₉	0	2	1	0	2	3	4	5

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (konačno rješenje):

Kod lokalnog poravnanja konačno rješenje
 može se nalaziti u bilo kojem polju matrice.
 Tražimo maksimalnu vrijednost u cijeloj
 matrici.

U ovom slučaju maksimalna sličnost je 7 i
 nalazi se u polju $F(7,5)$

Poravnanje – lokalno poravnanje

Matrica F

		Y							
		T ₁	C ₂	G ₃	T ₄	A ₅	G ₆	C ₇	G ₈
		0	0	0	0	0	0	0	0
X	G ₁	0	0	0	2	0	0	2	0
	A ₂	0	0	0	0	1	2	0	1
	T ₃	0	2	0	0	2	0	1	0
	C ₄	0	0	4	2	0	1	0	3
	A ₅	0	0	2	3	1	2	0	1
	T ₆	0	2	0	1	5	3	1	0
	A ₇	0	0	1	0	3	7	5	3
	T ₈	0	2	0	0	2	5	6	4
	T ₉	0	2	1	0	2	3	4	5

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (konačno rješenje):

Optimalno poravnanje računamo kretanjem
 prema natrag sve dok sličnost ne padne na
 nula.

Zasivljena polja nisu dio poravnanja jer
 se njima ukupna vrijednost sličnosti
 samo smanjuje!

Poravnanje – lokalno poravnanje

Matrica F

		Y								
		T 1	C 2	G 3	T 4	A 5	G 6	C 7	G 8	
		0	0	0	0	0	0	0	0	
X	G 1	0	0	0	2	0	0	2	0	2
	A 2	0	0	0	0	1	2	0	1	0
	T 3	0	2	0	0	2	0	1	0	0
	C 4	0	0	4	2	0	1	0	3	1
	A 5	0	0	2	3	1	2	0	1	2
	T 6	0	2	0	1	5	3	1	0	0
	A 7	0	0	1	0	3	7	5	3	1
	T 8	0	2	0	0	2	5	6	4	2
	T 9	0	2	1	0	2	3	4	5	3

PRIMJER: Želimo pronaći poravnanje nizova
 $X = \text{GATCATATT}$ i $Y = \text{TCGTAGCG}$ pomoću lokalnog
 poravnanja (konačno rješenje):

Optimalno poravnanje računamo kretanjem
 prema natrag sve dok sličnost ne padne na
 nula.

Optimalno poravnanje u ovom slučaju:

TCATA

TCGTA

Ili sa svim znakovima, ali su oni koji ne
 sudjeluju u poravnanju su zasivljeni:

GAT**TCATA**TT

TCGTAGCG

Ovdje nema umetanja i brisanja jer
 znakovi GA nisu dio poravnanja!

Poravnanje - složenost

- Algoritmi poravnanja imaju u osnovnom obliku imaju memorijsku i vremensku složenost proporcionalnu umnošku duljine nizova $\sim O(n \times m)$
- Ako promotrimo slučaj poravnanja očitavanja treće generacije od 10.000 nukleotida na bakterijski genom *Echerichia coli* koji ima 4.650.000 nukleotida, potrebno bi bilo popuniti tablicu od **46,5 milijardi polja**! Osim trajanja, ovo predstavlja nepremostiv memorijski problem – za veće genome.
- Iako postoje varijante algoritama koje smanjuju vremensku ili memorijsku složenost, i u takvom obliku algoritmi poravnanja nisu pogodni za poravnanje dugih očitavanja na referencu.

Veličina genoma:

Virus ~ 50.000 baza

Bakterija $\sim 5.000.000$ baza

Kvasac $\sim 10.000.000$ baza

Ptice $\sim 1.000.000.000$ baza

Čovjek $\sim 3.000.000.000$ baza

Jedan kromosom $\sim 50.000.000 - 250.000.000$ baza

Neke biljke čak preko 100.000.000.000 baza

Duljina očitavanja:

Prva generacija

Očitavanja srednje duljine (oko **1000** baza)

Druga generacija (NGS – *next generation sequencing*)

Očitavanja male duljine, tipično **100-600** baza

Treća generacija

Očitavanja velike duljine (**deseci tisuća baza**), najveće očitavanje 1M baza

Poravnanje - složenost

- Umjesto izravnog korištenja egzaktnih algoritama poravnanja, koriste se aproksimativne metode – u nekim situacijama nije potrebno znati točno poravnanja već samo poziciju gdje se ono nalazi
- Ako je poravnanje potrebno, onda se u prvom koraku pomoću aproksimativnih metoda pronalaze kandidatne lokacije na referenci, te se egzaktno poravnanje računa samo za te lokacije.

Veličina genoma:

Virus ~ 50.000 baza

Bakterija ~ 5.000.000 baza

Kvasac ~ 10.000.000 baza

Ptice ~ 1.000.000.000 baza

Čovjek ~ 3.000.000.000 baza

Jedan kromosom ~ 50.000.000 – 250.000.000 baza

Neke biljke čak preko 100.000.000.000 baza

Duljina očitavanja:

Prva generacija

Očitavanja srednje duljine (oko **1000** baza)

Druga generacija (NGS – *next generation sequencing*)

Očitavanja male duljine, tipično **100-600** baza

Treća generacija

Očitavanja velike duljine (**deseci tisuća baza**), najveće očitavanje 1M baza

Poravnanje više sekvenci

- Poravnanje više sekvenci (engl. *multiple sequence alignment*)
- Do sada smo poravnavali dvije sekvence – očitanje i referencu ili dva očitavanja.
- Međutim, u različitim primjenama potrebno je poravnati više sekvenci sve zajedno
- Naivno:
 - Poravnamo prve dvije sekvence
 - Dodajemo im jednu po jednu sekvencu
- Ovisi o redoslijedu dodavanja sekvenci
 - Jedan od popularnih pristupa **prvo računa sličnost sekvenci bržim algoritmom**, a zatim „spaja” sekvence po sličnosti – *progressive MSA*

Poravnanje više sekvenci

Multiple sequence alignment

>16WBS:00354:00627

GAT-CCTCTCTC-TGT-AGCACA-TTTCCTGCTGTATACTACGAGCGAGTGTCA-TTTCTCCAACGGGGAC-GC-AGCGGGT-GGGGTTCC-TGGACAGATACTTCTATAACGG

>16WBS:00296:01071

GAT-CCTCTCTC-TGC-AGCACA-TTTCCTGCTGTATACTACGAGCGAGTGTCA-TTTCTCCAACGGGGAC-GC-AGCGGGTGGGGTTCC-TGGACAGATACTTCTATAACGG

>16WBS:00303:00966

GAT-CCTCTCTC-TGC-AGCACA-TTTCCTGCTGTATACTACGAGCGAGTGTCA-TTTCTCCAAC-GGGAC-GC-AGCGGGT-GGGGTTCC-TGGACAGATACTTCTATAACGG

>16WBS:00459:01208

GAT-CCTCTCTC-TGC-AGCACA-TTTCCTGCTGTATACTACGAGCGAGTGTCA-TTTCTCCAAC-GGGAC-GC-AGCGGGT-GGGGTTCC-TGGACAGATACTTCTATAACGG

>16WBS:00525:00224

GAT-CCTCTCTC-TGC-AGCACA-TTTCCTGCTGTATACTACGAGCGAGTGTCA-TTTCTCCAAC-GGGAC-GC-AGCGGGT-GGGGTTCC-TGGACAGATACTTCTATAACGG

>16WBS:00263:01002

GAT-CCTCTCTC-TGC-AGCACA-TTTCCTGCTGTATACTACGAGCGAGTGTCA-TTTCTCCAAC-GGGAC-GC-AGCGGGT-GGGGTTCC-TGGACAGACACTTCTATAACGG

>16WBS:00234:00085

GAT-CCTCTCTC-TGC-AGCACA-TTTCCTGCTGTA~~A~~ACTACGAGCGAGTGTCA-TTTCTCCAAC-GGGAC-GC-AGCGGGT-GGGGTTCC-TGGACAGATACTTCTATAACGG

>16WBS:00077:00008

GAT-CCTCTCTC-TGC-AGCACA-TTTCCTGCTGTATACTACGAGCGAGTGTCA-TTTCTCCAAC-GGGAC-GC-AGCGGCT-GGGGTTCC-TGGACAGATACTTCTATAACGG

...

Sekvenciranje i signali

- Uređaji za sekvenciranje proizvode skup pročitanih fragmenata koja nazivamo očitavanja.
 - FASTA format (poznate sekvence):

```
>CP027599.1 Escherichia coli strain 97-3250 chromosome, complete genome
ATCCCGGCCCCGGCAGAACCGACCTATCGTTCTAACGTAAACGTCAAACACACGTTTGATAACTTCGTTG
AAGGTAAATCTAACCAACTGGCGCGCGCGGCGGCTCGCCAGGTGGCGGATAACCCTGGCGGTGCCTATAA
```
 - FASTQ format (očitanja):

```
@SEQ_ID
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!' '*((( (**+)) %%%++) (%%%) .1***-+*' ')) **55CCF>>>>>CCCCCCC65
```
- Gdje je tu signal !?

Sekvenciranje i signali

- Gdje je tu signal !?
- Iako pojedini nukleotidi u sekvenci (ili amino-kiseline u proteinima) nemaju oznaku vremena, možemo ih promatrati kao signal u prostoru – tako da im dodijelimo redni broj

Seq	=	G	A	T	T	T	G	G	G	G	T	T	C	A	A	A	G	C	A	G	T	...
X(t)	=	G	A	T	T	T	G	G	G	G	T	T	C	A	A	A	G	C	A	G	T	...
t	=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...

Sekvenciranje i signali

- Gdje je tu signal !?

Seq	=	GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT																				
X(t)	=	G	A	T	T	T	G	G	G	G	T	T	C	A	A	A	G	C	A	G	T	...
t	=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...

- Vrijednosti nukleotida zamijenimo brojevima

Seq	=	GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT																				
X(t)	=	G	A	T	T	T	G	G	G	G	T	T	C	A	A	A	G	C	A	G	T	...
X'(t)	=	1	2	3	3	3	1	1	1	1	3	3	4	2	2	2	1	4	2	1	3	...
t	=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...

Sekvenciranje i signali

- Gdje je tu signal !?

Seq	=	G	A	T	T	T	G	G	G	G	T	T	C	A	A	A	G	C	A	G	T	...
X(t)	=	G	A	T	T	T	G	G	G	G	T	T	C	A	A	A	G	C	A	G	T	...
X'(t)	=	1	2	3	3	3	1	1	1	1	3	3	4	2	2	2	1	4	2	1	3	...
t	=	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...

- Kod proteinskih sekvenci, pojedina amino-kiselina se opisuje sa dvije vrijednosti: veličinom (*engl. volume*) i polaritetom (*engl. polarity*)
- **IDEJA:** Umjesto izravnog uspoređivanja znakova pomoću dinamičkog programiranja, možemo li iskoristiti frekvencijsku domenu!?

Fourierova transformacija – podsjetnik

- Fourierova transformacija vremenski diskretnih signala (DTFT)

$$X(\Omega) = \sum_{n=-\infty}^{\infty} x(n) \cdot e^{-j\Omega n}$$

- Pretpostavljamo jednolike frekvencije $\Omega = \frac{2\pi}{N} k$

$$X_k = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k}$$

- Inverzna Fourierova transformacija vremenski diskretnih signala (IDTFT)

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{j\frac{2\pi}{N} \cdot n \cdot k}$$

Fourierova transformacija - implementacija

$$X_k = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k}$$

- X – možemo promatrati kao vektor od N elemenata
- x – također vektor od N elemenata
- $e^{-j\frac{2\pi}{N} \cdot n \cdot k}$ – koeficijenti matrice M

$$X = M \cdot x$$

Fourierova transformacija - implementacija

```
import numpy as np
```

```
def DFT(x):
```

```
    """Računamo diskretnu Fourierovu transformaciju 1D polja x"""
```

```
    x = np.asarray(x, dtype=float)
```

```
    N = x.shape[0]
```

```
    n = np.arange(N)
```

```
    k = n.reshape((N, 1))
```

```
    M = np.exp(-2j * np.pi * k * n / N)
```

```
    return np.dot(M, x)
```

$$X_k = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k}$$

→ pretvaramo x u numpy array

→ dohvaćamo dimenziju x

→ punimo vektor n brojevima 0 ... N-1

→ punimo vektor k brojevima 0 ... N-1 (okomito)

→ punimo matricu M faktorima za računanje DFT

→ rezultat računamo kao umnožak matrice M i vektora x

Fourierova transformacija - složenost

- X – vektor od N elemenata
- x – vektor od N elemenata
- $e^{-j\frac{2\pi}{N} \cdot n \cdot k}$ – koeficijenti matrice M

$$X = M \cdot x$$

- Složenost izravnog računanja DFT je $O(N^2)$

Brza Fourierova transformacija

- Brza Fourierova transformacija (engl. Fast Fourier Transform - FFT)
 - Nije nova transformacija već algoritam za računanje DFT (ili IDFT)
- Izravno računanje DFT $\sim O(N^2)$
- FFT $\sim O(N \log N)$
- Prvi put se „nešto slično” spominje u Gaussovom tekstu još 1805. godine, pokušao ju je iskoristiti za interpolaciju orbite asteroida Pallas i Juno na temelju opažanja (na kraju je ipak koristio druge metode)
- Više nezavisnih spominjanja u raznim radovima
- 1965, nezavisno jedan od drugoga James Cooley i John Tukey opisuju općenitu verziju algoritma

Brza Fourierova transformacija

- Brza Fourierova transformacija – Cooley & Tukey algoritam
- Za početak pogledajmo koja je vrijednost od X_{N+k}

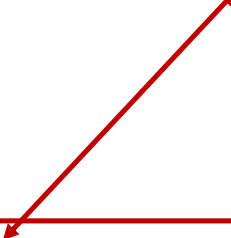
$$X_{N+k} = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot (N+k)} = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi n} \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k}$$

$$X_{N+k} = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi n} \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k}$$

Brza Fourierova transformacija

- Brza Fourierova transformacija – Cooley & Tukey algoritam
- Za početak pogledajmo koja je vrijednost od X_{N+k}

$$X_{N+k} = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi n} \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k}$$


$$e^{-j\omega_0 t} = \cos(\omega_0 t) - j \cdot \sin(\omega_0 t) = \cos(2\pi n) - j \cdot \sin(2\pi n) = 1$$

Brza Fourierova transformacija

- Brza Fourierova transformacija – Cooley & Tukey algoritam
- Za početak pogledajmo koja je vrijednost od X_{N+k}

$$X_{N+k} = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi n} \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k}$$

$$X_{N+k} = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k} = X_k$$

Brza Fourierova transformacija

- Brza Fourierova transformacija – Cooley & Tukey algoritam
- Za početak pogledajmo koja je vrijednost od X_{N+k}

$$X_{N+k} = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N} \cdot n \cdot k} = X_k$$

- Također vrijedi:

$$X_{k+i \cdot N} = X_k$$

Brza Fourierova transformacija

- Cooley i Tukey su pokazali da se DFT duljine N (gdje je N paran broj) može prikazati kao zbroj dvije DFT od kojih je svaka duljine $N/2$.

$$X_k = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}nk}$$

$$X_k = \sum_{m=0}^{N/2-1} x(2m) \cdot e^{-j\frac{2\pi}{N}(2m)k} + \sum_{m=0}^{N/2-1} x(2m+1) \cdot e^{-j\frac{2\pi}{N}(2m+1)k}$$

$$X_k = \sum_{m=0}^{N/2-1} x(2m) \cdot e^{-j\frac{2\pi}{N/2}mk} + e^{-j\frac{2\pi}{N}k} \cdot \sum_{m=0}^{N/2-1} x(2m+1) \cdot e^{-j\frac{2\pi}{N/2}mk}$$

Brza Fourierova transformacija

- Cooley i Tukey su pokazali da se DFT duljine N (gdje je N paran broj) može prikazati kao zbroj dvije DFT od kojih je svaka duljine $N/2$.
- Prva se sastoji od parnih elemenata niza, a druga od neparnih

$$X_k = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}nk}$$
$$X_k = \sum_{m=0}^{N/2-1} x(2m) \cdot e^{-j\frac{2\pi}{N/2}mk} + e^{-j\frac{2\pi}{N}k} \cdot \sum_{m=0}^{N/2-1} x(2m+1) \cdot e^{-j\frac{2\pi}{N/2}mk}$$

Parni elementi

Konstanta za X_k

DFT duljine $N/2$

Neparni elementi

Brza Fourierova transformacija

- Cooley i Tukey su pokazali da se DFT duljine N (gdje je N paran broj) može prikazati kao zbroj dvije DFT od kojih je svaka duljine N/2.
- Prva se sastoji od parnih elemenata niza, a druga od neparnih

$$X_k = \sum_{m=0}^{N/2-1} x(2m) \cdot e^{-j\frac{2\pi}{N/2}mk} + e^{-j\frac{2\pi}{N}k} \cdot \sum_{m=0}^{N/2-1} x(2m+1) \cdot e^{-j\frac{2\pi}{N/2}mk}$$

DFT Parnih elemenata – DFT_P_k

DFT neparnih elemenata – DFT_N_k

$$X_k = DFT_P_k + e^{-j\frac{2\pi}{N}k} \cdot DFT_N_k$$

Brza Fourierova transformacija

- Pogledajmo kako možemo izračunati $X_{k+N/2}$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-j\frac{2\pi}{N}m(k+\frac{N}{2})} + e^{-j\frac{2\pi}{N}(k+\frac{N}{2})} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-j\frac{2\pi}{N}m(k+\frac{N}{2})}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-j\frac{2\pi}{N}mk} \cdot \underbrace{e^{-j2\pi m}}_{\substack{\text{red circle} \\ \text{arrow to } 1}} + e^{-j\frac{2\pi}{N}k} \cdot \underbrace{e^{-j\pi}}_{\substack{\text{red circle} \\ \text{arrow to } -1}} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-j\frac{2\pi}{N}mk} \cdot \underbrace{e^{-j2\pi m}}_{\substack{\text{red circle} \\ \text{arrow to } 1}}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-j\frac{2\pi}{N}mk} - e^{-j\frac{2\pi}{N}k} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-j\frac{2\pi}{N}mk}$$

Brza Fourierova transformacija

- Pogledajmo kako možemo izračunati $X_{k+N/2}$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{N/2-1} x_{2m} \cdot e^{-j\frac{2\pi}{N}mk} - e^{-j\frac{2\pi}{N}k} \cdot \sum_{m=0}^{N/2-1} x_{2m+1} \cdot e^{-j\frac{2\pi}{N}mk}$$

DFT Parnih elemenata – DFT_P_k

DFT neparnih elemenata – DFT_N_k

$$X_{k+\frac{N}{2}} = DFT_P_k - e^{-j\frac{2\pi}{N}k} \cdot DFT_N_k$$

Brza Fourierova transformacija

$$X_k = DFT_P_k + e^{-j\frac{2\pi}{N}k} \cdot DFT_N_k$$

$$X_{k+\frac{N}{2}} = DFT_P_k - e^{-j\frac{2\pi}{N}k} \cdot DFT_N_k$$

- Početni problem od $O(N^2)$ pretvorili smo u dva problema od $O(M^2)$, gdje je $M=N/2$
- Taj princip možemo rekurzivno primjenjivati što nam u konačnici daje složenost $\sim O(N\log N)$

Brza Fourierova transformacija

- Početni problem od $O(N^2)$ pretvorili smo u dva problema od $O(M^2)$, gdje je $M=N/2$
- Taj princip možemo rekurzivno primjenjivati što nam u konačnici daje složenost $\sim O(N \log N)$
- Ovaj algoritam pretpostavlja da je N potencija od 2 (da bi se rekurzija mogla primijeniti do kraja)
- U principu se ne koriste rekurzivne implementacije, već se stablo izračuna obilazi *breadth first* metodom

<https://www.karlsims.com/fft.html>

Brza Fourierova transformacija

$$X_k = DFT_P_k + e^{-j\frac{2\pi}{N}k} \cdot DFT_N_k$$
$$X_{k+\frac{N}{2}} = DFT_P_k - e^{-j\frac{2\pi}{N}k} \cdot DFT_N_k$$

```
def FFT(x):  
    """Rekurzivna implementacija Cooley-Tukey algoritma"""  
    x = np.asarray(x, dtype=float)  
    N = x.shape[0]  
  
    if N % 2 > 0:  
        raise ValueError("size of x must be a power of 2")  
    elif N <= 32: # proizvoljni cutoff (slično quicksortu - ASP)  
        return DFT(x)  
    else:  
        X_even = FFT(x[::2])  
        X_odd = FFT(x[1::2])  
        factor = np.exp(-2j * np.pi * np.arange(N) / N)  
        return np.concatenate([X_even + factor[:N / 2] * X_odd,  
                                X_even + factor[N / 2:] * X_odd])
```

* preuzeto s <https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/>

MAFFT algoritam

- MAFFT – Multiple sequence Alignment with Fast Fourier Transform
- Promatra sekvence (DNA, RNA ili proteina) kao signal
- **Sličnost dvije sekvence opisuje kao korelaciju signala**
- Grupira slične sekvence da bi pomoću dinamičkog programiranja izračunao poravnanje više sekvenci progresivnom metodom
 - *Engl. progressive technique, hierarchical method, tree method*
 - Prvo računa poravnanje međusobno najsličnijih sekvenci, koja kasnije postupno spaja u poravnanje više i više sekvenci
 - Ne garantira globalni optimum
 - **Izvan domene ovog predmeta 😊**

Ovime ćemo
se baviti

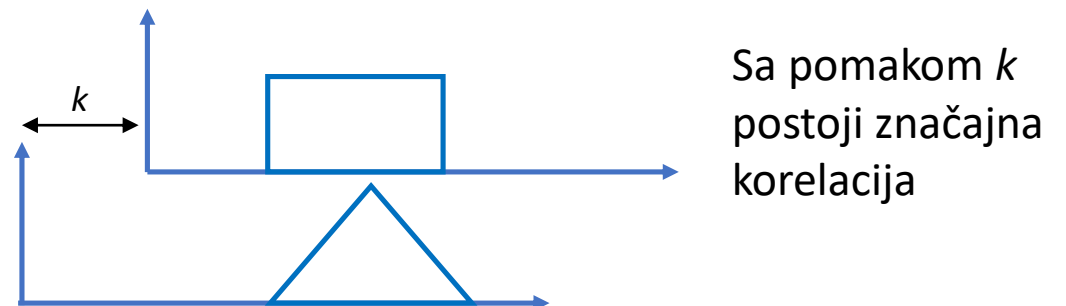
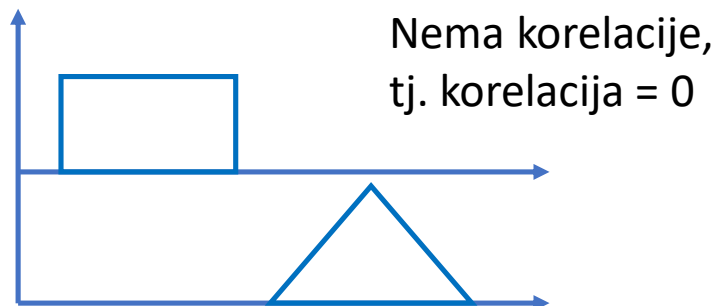


MAFFT algoritam

- Sličnost dvije sekvence opisuje kao korelaciju signala
- Korelacija dva signala je mjera njihove sličnosti kao funkcija pomaka između njih, može se opisati kao:

$$Cor(k) = \sum_{\substack{1 \leq n \leq N, \\ 1 \leq n+k \leq M}} S_1(n) \cdot S_2(n+k)$$

- Korelacija će biti najveća kada se signali najviše podudaraju



MAFFT algoritam

- Korelacija dva signala je mjera njihove sličnosti kao funkcija pomaka između njih, može se opisati kao:

$$Cor(k) = \sum_{\substack{1 \leq n \leq N, \\ 1 \leq n+k \leq M}} S_1(n) \cdot S_2(n+k)$$

- Složenost računanja korelacije je $O(NM)$, tj. ako su N i M slične veličine, složenost je $O(N^2)$
- Računanje korelacije možemo ubrzati pomoću FFT!

MAFFT algoritam

- Računanje korelacije možemo ubrzati pomoću FFT!

- Teorem o korelaciji kaže:

$$s_1(n) \cdot s_2(n) \Leftrightarrow S_1^*(k) \cdot S_2(k)$$

- Odnosno, korelaciju dva signala možemo dobiti tako da izračunamo DFT oba signala (prebacimo ih u frekvencijsku domenu), jedan DFT konjugiramo, pomnožimo ih, te rezultat vratimo u originalnu domenu (vremensku, prostornu ...) pomoću IDFT.
- Ako pri tome koristimo FFT za sve tri operacije (računamo DFT oba signala te za rezultat računamo IDFT), složenost postupka je $O(N \log N)$

MAFFT algoritam – primjer 1

- Neka imamo dva signala:

$$a = [1, 2, 3, 4]$$

$$b = [4, 3, 2, 1]$$

- Korelaciju možemo izračunati za vrijednosti $k = -3, -2, -1, 0, +1, +2, +3$

$$k = -3$$

$$a = [0 \ 0 \ 0 \ \mathbf{1} \ 2 \ 3 \ 4]$$

$$b = [4 \ 3 \ 2 \ \mathbf{1} \ 0 \ 0 \ 0]$$

$$\text{Cor}(-3) = \mathbf{1}$$

$$k = -2$$

$$a = [0 \ 0 \ \mathbf{1} \ \mathbf{2} \ 3 \ 4]$$

$$b = [4 \ 3 \ \mathbf{2} \ \mathbf{1} \ 0 \ 0]$$

$$\text{Cor}(-2) = 2+2 = \mathbf{4}$$

$$k = -1$$

$$a = [0 \ \mathbf{1} \ \mathbf{2} \ \mathbf{3} \ 4]$$

$$b = [4 \ \mathbf{3} \ \mathbf{2} \ \mathbf{1} \ 0]$$

$$\text{Cor}(-1) = 3+4+3 = \mathbf{10}$$

$$k = 0$$

$$a = [\mathbf{1} \ \mathbf{2} \ \mathbf{3} \ \mathbf{4}]$$

$$b = [\mathbf{4} \ \mathbf{3} \ \mathbf{2} \ \mathbf{1}]$$

$$\text{Cor}(0) = 4+6+6+4 = \mathbf{20}$$

MAFFT algoritam – primjer 1

- Neka imamo dva signala:

$$a = [1, 2, 3, 4]$$

$$b = [4, 3, 2, 1]$$

- Korelaciju možemo izračunati za vrijednosti $k = -3, -2, -1, 0, -1, -2, -3$

Najveća korelacija izračunata je za $k=1$ i iznosi 25.
Dakle signali a i b su najsličniji s pomakom $k=1$.

$k = 1$

$$a = [1 \text{ } \color{red}{2} \text{ } \color{red}{3} \text{ } \color{red}{4} \text{ } 0]$$

$$b = [0 \text{ } \color{red}{4} \text{ } \color{red}{3} \text{ } \color{red}{2} \text{ } 1]$$

$$\text{Cor}(-3) = 8+9+8 = \mathbf{25}$$

$k = 2$

$$a = [1 \text{ } 2 \text{ } \color{red}{3} \text{ } \color{red}{4} \text{ } 0 \text{ } 0]$$

$$b = [0 \text{ } 0 \text{ } \color{red}{4} \text{ } \color{red}{3} \text{ } 2 \text{ } 1]$$

$$\text{Cor}(-3) = 12+12 = \mathbf{24}$$

$k = 3$

$$a = [1 \text{ } 2 \text{ } 3 \text{ } \color{red}{4} \text{ } 0 \text{ } 0 \text{ } 0]$$

$$b = [0 \text{ } 0 \text{ } 0 \text{ } \color{red}{4} \text{ } 3 \text{ } 2 \text{ } 1]$$

$$\text{Cor}(-3) = \mathbf{16}$$

MAFFT algoritam – primjer 1

- Računamo korelaciju u pythonu

```
# pomoću ugrađene funkcije
import numpy as np

a = [1,2,3,4]
b = [4,3,2,1]

C = np.correlate(a,b,'full')
print(c)
```

>>>

1	4	10	20	25	24	16
-3	-2	-1	0	1	2	3
						k

```
# pomoću FFT
import numpy as np
a = [1,2,3,4]
b = [4,3,2,1]
A = np.fft.fft([0,0,0]+a)
B = np.fft.fft(b+[0,0,0])

C = np.conjugate(B)*A
C = np.fft.ifft(C)
print(c)
```

>>>

1	4	10	20	25	24	16
-3	-2	-1	0	1	2	3
						k

Punimo
nulama da bi
dobili punu
korelaciju

MAFFT algoritam – primjer 2

- Imamo dva niza:
 seq1 = GATTGGG
 seq2 = CCGATCTA
- Pretpostavimo sljedeće vrijednosti za pretvorbu u signal:
 A – 1
 C – 2
 G – 3
 T – 4
- Pokušajmo pronaći postoje li slične regije na dva niza koristeći korelaciju kao mjeru sličnosti i računanje pomoću FFT

MAFFT algoritam – primjer 2

```
# Primjer za MAFFT algoritam na nukleotidima
import numpy as np
```

```
nuc2num = {}
nuc2num['A'] = 1
nuc2num['C'] = 2
nuc2num['G'] = 3
nuc2num['T'] = 4
```

```
seq1 = 'GATTTGGG'
seq2 = 'CCGATCTA'
```

```
x1 = [nuc2num[s] for s in seq1]
x2 = [nuc2num[s] for s in seq2]
```

```
len1 = len(x1)
len2 = len(x2)
k_arr = range(-len2+1, len1)
```

Definiramo rječnik (mapu) za pretvorbu nukleotida (slova) u brojčane vrijednosti

Sekvence nukleotida koje uspoređujemo

Sekvence nukleotida pretvaramo u sekvence brojeva tj. u signal

Računamo veličine sekvenci, te vrijednosti koje će poprimiti parametar k (pohranjujemo u polje)

MAFFT algoritam – primjer 2

```
...  
avg = np.average(nuc2num.values())  
std = np.std(nuc2num.values())  
x1 = [(x-avg)/std for x in x1]  
x2 = [(x-avg)/std for x in x2]
```

Da bi korelacija imala smisla, normaliziramo signal.

```
padding1 = [0]*(len2-1)  
padding2 = [0]*(len1-1)
```

Nule koje dodajemo na početak x1 i x2, da bi dobili punu korelaciju

```
X1 = np.fft.fft(padding1+x1)  
X2 = np.fft.fft(x2+padding2)  
Cor = np.conjugate(X2)*X1  
cor = np.fft.ifft(Cor)
```

Računamo korelaciju pomoću FFT

```
k = k_arr[cor.argmax()]  
print("Correlation by FFT:")  
print(cor)  
print("k={}".format(k))
```

Ispisujemo korelaciju i k za koji je korelacija najveća

MAFFT algoritam – primjer 2

```
>>>  
Correlation by FFT:  
[-0.6  2.4 -3.8  1.2 -3.  4.4 -1.4  0.8 -0.2 -0.8 -1.  -1.2 -0.2 -0.4  
 -0.2]  
k=-2
```

Budući da k poprima vrijednosti od -7 do 7, ovo je vrijednost za k=-2

Za k=-2 nizovi su poravnati na sljedeći način:

seq1: --GATTGGG

seq2: CCGATCTA--



Crveno su označeni detektirani slični dijelovi nizova, međutim to ne možemo vidjeti iz korelacije. Korelacija nam daje smo pomak između nizova (k) i sličnost (vrijednost korelacije). Da bismo odredili slične dijelove nizova i njihovu točnu lokaciju, potrebno je koristiti druge metode.

MAFFT algoritam koristi prozor od 30 elemenata kojim se kreće po nizovima da bi usporedbom pronašao slične regije.

MAFFT algoritam – primjer 2

```
>>>  
Correlation by FFT:  
[-0.6 2.4 -3.8 1.2 -3. 4.4 -1.4 0.8 -0.2 -0.8 -1. -1.2 -0.2 -0.4  
 -0.2]  
k=-2
```



Možemo primijetiti još jednu pozitivnu vrijednost korelacije koja može biti značajna za $k=-6$. Ona označava sljedeće poravnanje:

```
seq1:  -----GATTGGG  
seq2:  CCGATCTA-----
```

Očito je da ovo poravnanje nije značajno te će ono biti izbačeno u daljnjoj analizi (nije dio ovog predavanja)

MAFFT algoritam – primjer 3

- Primijenimo isti algoritam na sljedeća dva niza:
 seq1 = GATTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTGTTACTCACAGTTT
 seq2 = TTGGCGTT
- Ako usporedimo rezultate dobivene metodom *np.correlate()* i pomoću FFT možemo vidjeti neke razlike uzrokovane numeričkim pogreškama.

Riječ je o
jako malim
brojevima
blizu nule

```
>>>
Correlation by np.correlate:
...
1.11022302e-16 -2.40000000e+00 -2.80000000e+00 -6.80000000e+00
...
k=3
Correlation by FFT:
...
1.27261386e-15 -2.40000000e+00 -2.80000000e+00 -6.80000000e+00
k=41
```

MAFFT algoritam – primjer 3

- Primijenimo isti algoritam na sljedeća dva niza:
 seq1 = GATTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTGTTACTCACAGTTT
 seq2 = TTGGCGTT
- Ako usporedimo rezultate dobivene metodom *np.correlate()* i pomoću FFT možemo vidjeti neke razlike uzrokovane numeričkim pogreškama.

Metoda *argmax()* vraća različite vrijednosti. Objе vrijednosti korelacije su 7.6. Vidjet ćemo kasnije da vrijednosti ipak nisu jednake.

```
>>>
Correlation by np.correlate:
...
1.11022302e-16 -2.40000000e+00 -2.80000000e+00 -6.80000000e+00
...
k=3
Correlation by FFT:
...
1.27261386e-15 -2.40000000e+00 -2.80000000e+00 -6.80000000e+00
...
k=41
```

MAFFT algoritam – primjer 3

- Primjenimo isti algoritam na sljedeća dva niza:

seq1 = GATTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTGTTACTCACAGTTT

seq2 = TTGGCGTT

- Međutim, umjesto samo najveće korelacije, želimo saznati sve značajne vrijednosti. Pretpostavimo da su značajne vrijednosti korelacije one koje su veće od polovice maksimuma – nazovimo ih vrhovima korelacije (engl. *peaks ili spikes*).

```
...  
X1 = np.fft.fft(padding1+x1)  
X2 = np.fft.fft(x2+padding2)  
Cor = np.conjugate(X2)*X1  
cor = np.fft.ifft(Cor)  
  
maxc = np.max(cor)  
spikes = np.where(cor > maxc/2)  
spikes2 = [(k_arr[s], cor2[s]) for s in spikes]  
print(spikes2)
```

Računamo korelaciju pomoću FFT

Računamo maksimalnu vrijednost korelacije, te izdvajamo indekse polja koje imaju vrijednost veću od polovice.

Ispisujemo k i korelaciju za svaki *spike*.

MAFFT algoritam – primjer 3

```
>>>
(-3, 4.9999999999999997)
(2, 6.7999999999999999)
(3, 7.5999999999999997)
(4, 4.4000000000000001)
(36, 4.3999999999999995)
(40, 3.9999999999999998)
(41, 7.6)
(42, 4.8000000000000002)
(45, 5.1999999999999999)
(51, 4.8)
(57, 4.1999999999999999)
```

k	korelacija
---	------------

- Imamo jedanaest značajnih vrijednosti korelacije.
- Možemo vidjeti da je u izračunu pomoću FFT, vrijednost za k=3 neznatno manja od 7.6. Zbog manje preciznosti, metoda *numpy.correlate()* je zaokružila je vrijednost na 7.6.
- Pogledajmo pojedine slučajeve.

MAFFT algoritam – primjer 3

---GATTGGGGTTCAAAGCAGTATTCAAATAGTAAATCCATT...
|XX||
TTGGCGTT-----...

(-3, 4.9999999999999997)

k korelacija

GATTGGGGTTCAAAGCAGTATTCAAATAGTAAATCCATTTGT...
||X|X|X|
--TTGGCGTT-----...

(2, 6.7999999999999999)

k korelacija

GATTGGGGTTCAAAGCAGTATTCAAATAGTAAATCCATTTGT...
||||X|||
---TTGGCGTT-----...

(3, 7.5999999999999997)

k korelacija

GATTGGGGTTCAAAGCAGTATTCAAATAGTAAATCCATTTGT...
|X||XX|
----TTGGCGTT-----...

(4, 4.4000000000000001)

k korelacija

MAFFT algoritam – primjer 3

...ATCGATCAAATAGTAAATCCATTTGTGTTACTCACAGTTT
 |XXXX|
...-----TGGCGTT-----

(36, 4.3999999999999995)

k korelacija

...ATCGATCAAATAGTAAATCCATTTGTGTTACTCACAGTTT
 |XXXXX|
...-----TGGCGTT-----

(40, 3.9999999999999982)

k korelacija

...ATCGATCAAATAGTAAATCCATTTGTGTTACTCACAGTTT
 ||X|X||
...-----TTGGCGTT-----

(41, 7.6)

k korelacija

...ATCGATCAAATAGTAAATCCATTTGTGTTACTCACAGTTT
 |||XXX||
...-----TTGGCGTT-----

(42, 4.8000000000000002)

k korelacija

MAFFT algoritam – primjer 3

...ATCGATCAAATAGTAAATCCATTGTGTTACTCACAGTTT
 |XXXXX|
...-----TTGGCGTT-----

(45, 5.1999999999999999)

k korelacija

...ATCGATCAAATAGTAAATCCATTGTGTTACTCACAGTTT
 |XXXX|||
...-----TTGGCGTT-----

(51, 4.8)

k korelacija

...ATCGATCAAATAGTAAATCCATTGTGTTACTCACAGTTT-----
 ||
...-----TTGGCGTT-----

(57, 4.1999999999999999)

k korelacija

MAFFT algoritam – zaključak

- MAFFT – primjer kako se tehnike obrade signala mogu koristiti u raznim područjima primjene (naizgled nepovezanim)
- Međutim, ostalo je dosta neodgovorenih pitanja – FFT nije dovoljan već je potrebno koristiti i druge algoritme
 - Kako odrediti područja sličnosti nizova
 - Kako ih grupirati
 - Kako izračunati poravnanje više sekvenci
 - Koje vrijednosti dodijeliti nukleotidima i amino-kiselinama za najbolje rezultate
 - Kako odrediti prag korelacije za koji se smatra da su nizovi dovoljno slični
- Time se nećemo baviti na ovim predavanjima!

MAFFT algoritam - dodatno

- Nukleotide klasificiramo kao purine (adenin i gvanin) ili pirimidine (citozin i timin)
- Tranzicija purin -> purin ili pirimidin -> pirimidin
- Transverzija purin -> pirimidin ili pirimidin -> purin
- Tranzicije češće nego transverzije
 - Dodijelimo slične vrijednosti za A i G, te za C i T
- Kod proteinskih sekvenci, pojedina amino-kiselina se opisuje sa dvije vrijednosti: veličinom (*engl. volume*) i polaritetom (*engl. polarity*)
 - Veća je vjerojatnost transformacije između sličnih amino-kiselina