

Bioinformatika

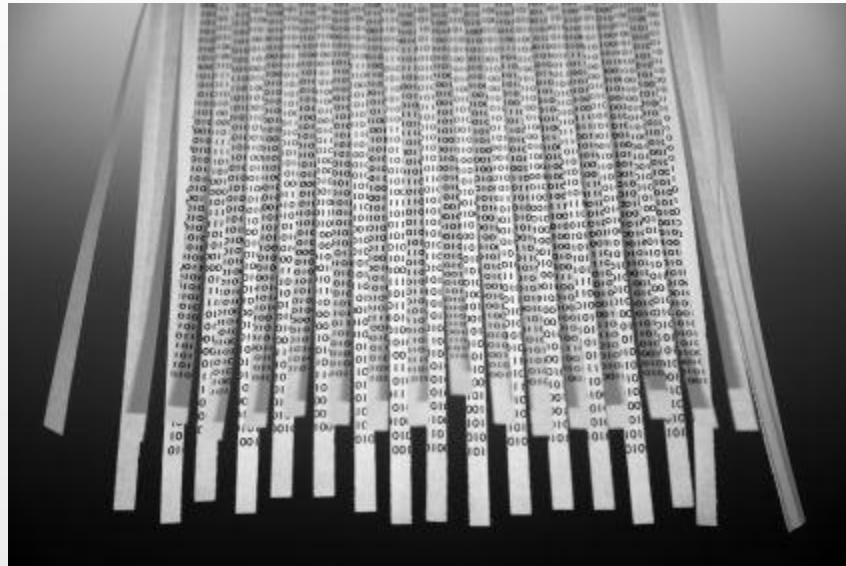
2020/2021

Sekvenciranje i sastavljanje genoma

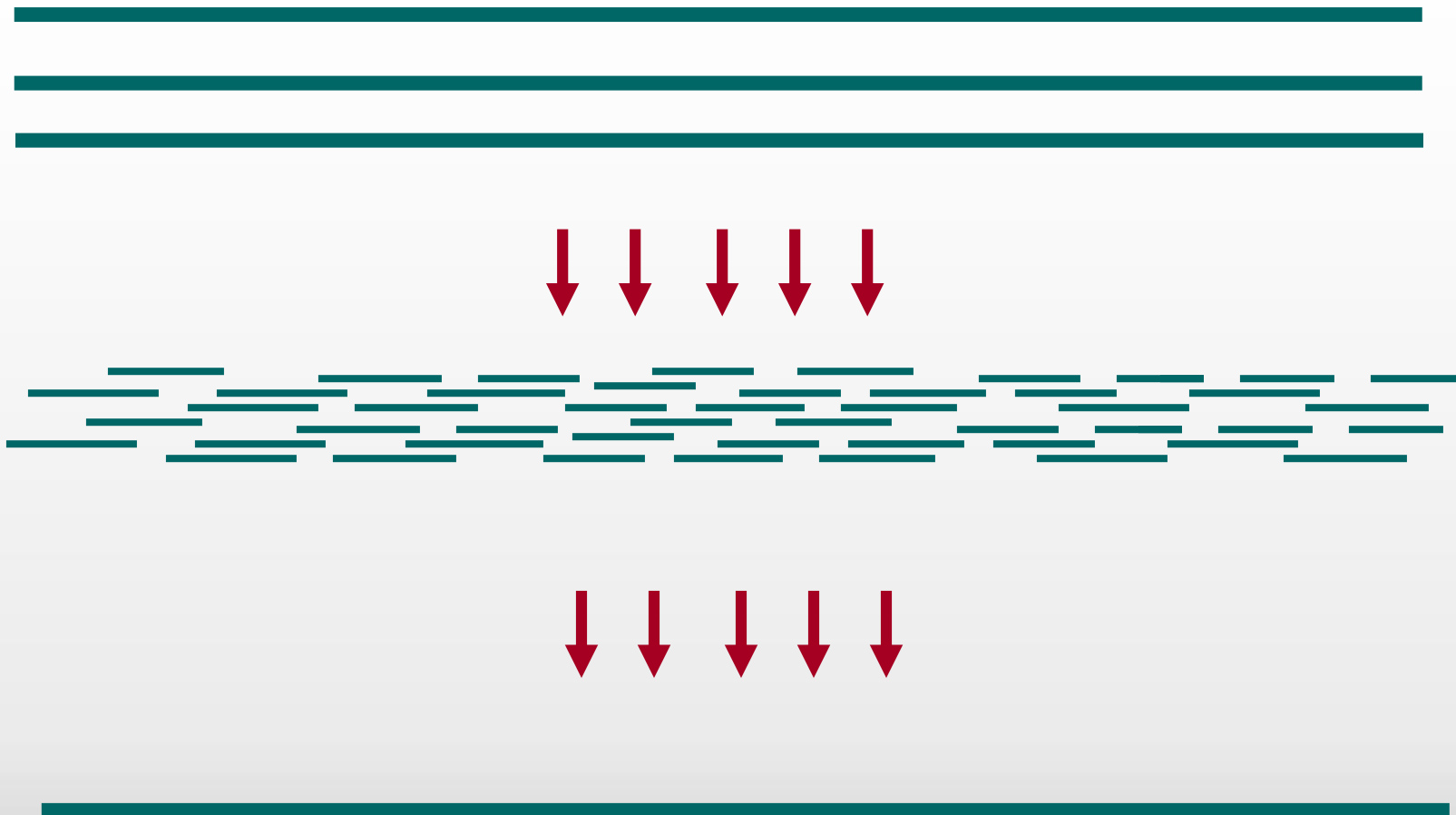
DNA sekvenciranje



DNA sekvenciranje



Sastavljanje



Metode sekvenciranja

- Prva generacija:
 - Sangerovo sekvenciranje (duljina očitavanja 400 – 900 bp)
 - Točnost: 99,9 %
 - Cijena za 1 mil. baza: 2400 USD
 - Skupo i nepraktično za velike projekte

Prva generacija



Izvor: Wikipedia

Metode sekvenciranja

- Druga generacija (engl. *Next-generation*)
 - Illumina
 - HiSeq: 35 – 150 bp
 - MiSeq: 300 bp
 - Roche 454: 200 400 bp
 - Applied Biosystems SOLiD: 50 bp
 - Life Technologies Ion semiconductor: 100 – 400 bp (2012)
 - Točnost: 98 – 99,9 %
 - Cijena za 1 mi. baza: 0,05 – 10 USD

Nova generacija



Izvor: Illumina

Metode sekvenciranja

- Treća generacije
 - PacBio
 - 10 – 15 kbp (srednja vrijednost duljine)
 - Max 40 kb
 - Točnost: 87 %
 - Cijena za 1 mil. baza: 0,33 – 1 USD
 - HiFi – 10-15kbp, greška < 1% - jako skupo
 - Oxford Nanopore
 - Duljina očitavanja teoretski do 1Mb
 - Greška 5 – 20 %

Nova generacija



Izvor: PacBio

Na koja pitanja odgovaramo

- Kako izgleda vaša genom?
- Kakav je vaš genom u usporedbi s mojim?
- Gdje se nalaze geni i kako se aktiviraju?
- Kako se genska aktivnost mijenja tijekom razvoja?
- Kako epigenetske promjene utječu na vas (metilacija)?
- Kako se proteini vežu i reguliraju gene?
- Koji virusi i mikrobi žive unutar vas?
- Kako bolesti mutiraju vaš genom?
- Koje vam lijekove treba dati?

Koncepti - preklapanje

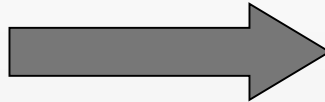
GCATACTACTAGGA**C**TA
ACTAGGAGTATGCTAT

- Poluglobalno poravnanje
- Može li brže?

Koncepti – poravnanje više sekvenci

- Pronaći maksimalno poravnanje između fragmenata

ACCGT
CGTGC
TTAC
TACCGT



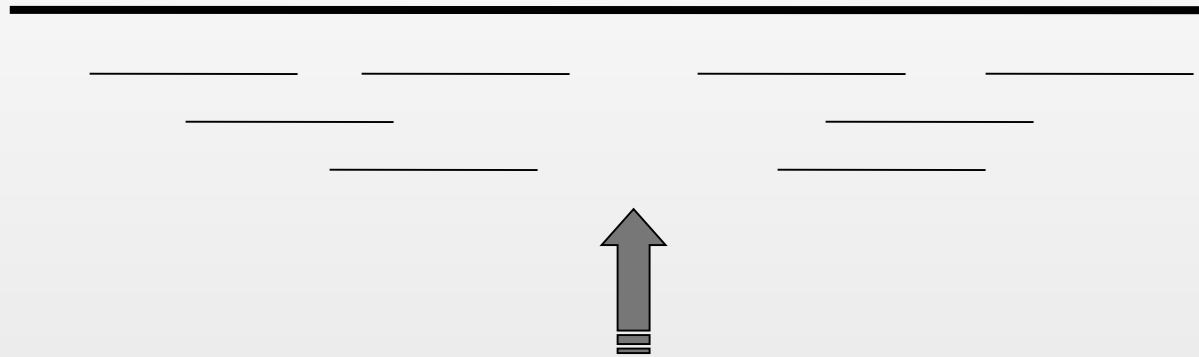
--	A	C	C	G	T	--			
---		C	G	T	G	C			
T	T	A	C	---	---	---			
-	T	A	C	C	G	T	-		
<hr/>									
	T	T	A	C	C	G	T	G	C

*Konsenzus sekvenca
određena glasanjem*

Metrike kvalitete

- Pokrivenost (engl. *Coverage*). Pokrivenost na poziciji i je broj fragmenata koji se preklapaju na toj poziciji

Originalna sekvenca:



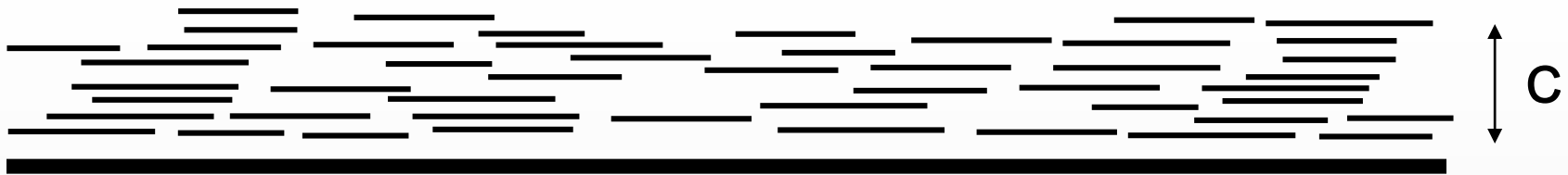
Bez pokrivanja

Metrike kvalitete - pokrivenost

ATACTACTAGGA
CTACTAGGACTA
TACTAG
TACTAGGACTAGCTA
GCATACTACTAGGACTA

5

Metrike kvalitete - pokrivenost



Duljina segmenta genoma: **G**

Broj očitavanja: **N**

Duljina svakoga očitavanja: **L**

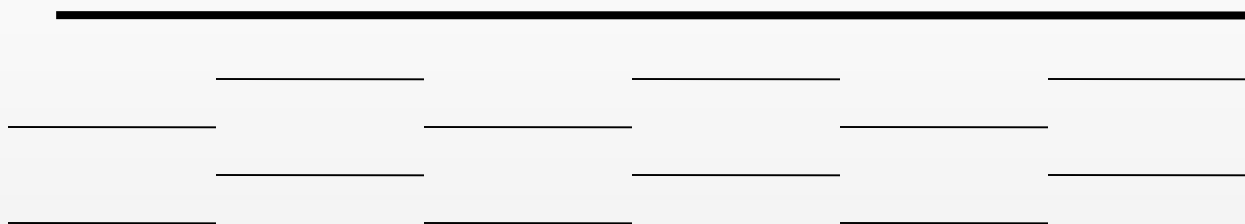
Definicija: Ukupna (prosječna) pokrivenost **$C = N L / G$**

Pokrivenost ne mora biti ravnomjerna duž cijelog genoma – teže je sastaviti slabije pokrivene djelove.

Metrike kvalitete - povezivanje

- Povezivanje (*engl. Linkage*) – stupanj preklapanja između fragmenata

Originalna sekvenca:



*Savršeno poklapanje, ravnomjerna pokrivenost,
slabo prosječno i minimalno povezivanje*

- Kako odrediti/procijeniti povezivanje?

Stvarne poteškoće

- Greške određivanja pojedinih baza
- Kimerni fragment, kontaminacija

```
--ACCGT--
---CGTGC
TTAC-----
-TGCCGT-
TTACCGTGC
```

Pogreška određivanja
baze

```
--ACC-GT--
----CAGTGC
TTAC-----
-TACC-GT-
TTACCGTGC
```

Pogreška umetanja

```
--ACCGT--
---CGTGC
TTAC-----
-TAC-GT-
TTACCGTGC
```

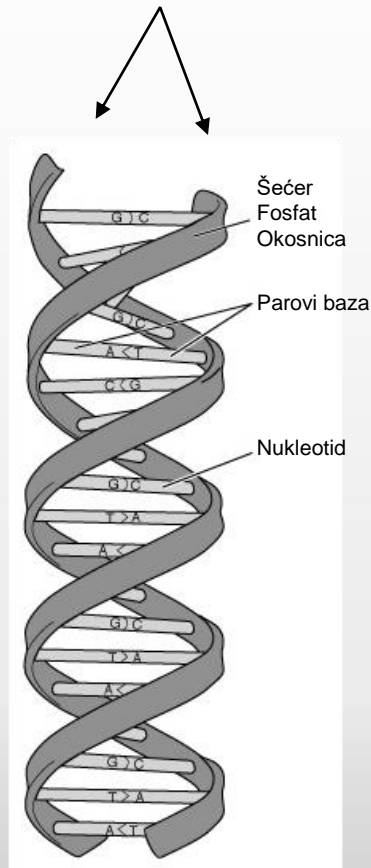
Pogreška brisanja

Stvarne poteškoće

- Greške određivanja pojedinih baza
 - Algoritmi određivanja preklapanja
 - Faza konsenzusa – popravljjanje pogrešaka (engl. per-base accuracy)
- Kimerni fragment, kontaminacija
 - Nužno ih je otkriti i odstraniti

Nepoznata orijentacija

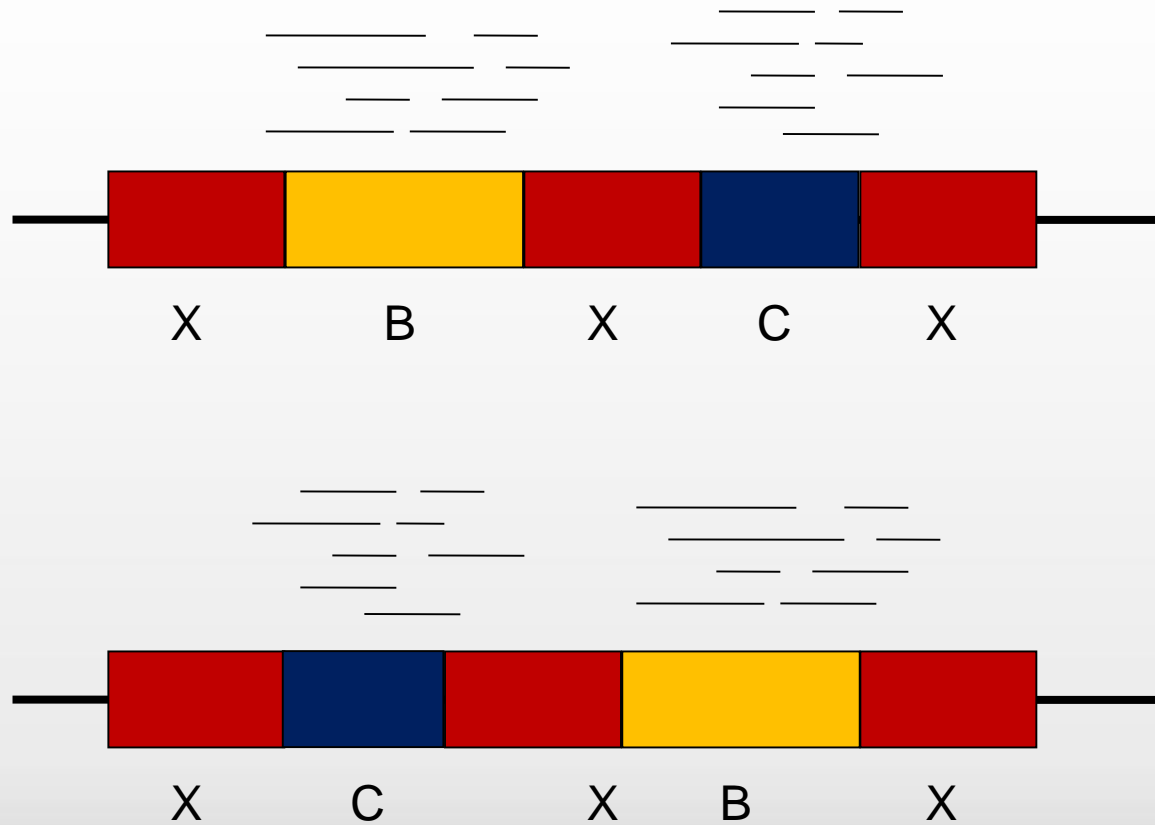
Fragment može doći s oba lanca



CACGT	→	CACGT
ACGT	→	-ACGT
ACTACG	←	--CGTAGT
GTACT	←	-----AGTAC
ACTGA	→	-----ACTGA
CTGA	→	-----CTGA

Ponavljjanja

- Direktna ponavljanja



Učestalost ponavljanja

- Duljina 100 – 1000000 bp


- Učestalost

- Bakterije

- 5 % sekvence

- Sisavci

- 50 % sekvence



Kako ovo
riješiti?

Sastavljanje genoma

- Mapiranje na postojeći referentni genom
 - Referentni genom
 - obično dobiven Sangerovom metodom
 - Od više donora
 - Npr. GRCh37 dobiven od 13 anonimnih ljudi iz Buffala, New York, US
 - Sličnost dva ljudska genoma 99.5 – 99.9 %
- *De novo* sastavljanje
 - Referentni genom nepoznat

Najkraći zajednički nadniz

- (engl. *Shortest common superstring SCS*):
 - Za dani niz znakova S , pronaći $SCS(S)$ - najkraći niz takav da sadrži nizove iz S kao podnizove
 - ne postoji garancija da SCS mora biti jednak stvarnom početnom nizu
- **Primjer** (ACA, ATA, ATT, CAT, TAC, TAG, TAT, TTA)
 - Jednostavno rješenje dobijemo spajanjem (nije najkraće)
ACAATAATTCATTACTAGTATTTA
 - Originalni niz
TATACATTAG

Iscrpno pretraživanje

- Poredati na neki način očitavanja
- Npr. ACA, ATA, ATT, CAT, TAC, TAG, TAT, TTA
- Konstruirati nadniz (spojiti najdulja preklapanja susjeda)
 - ACATA (ACA i ATA)
 - ACATATT (dodamo ATT)
 - ACATATTCAT (CAT nema preklapanja)
 - ACATATTCATAC
 - ACATATTCATACTAGTATTA

Iscrupno pretraživanje

- Poredani na drugačiji način
TAT, ATA, TAC, ACA, CAT, ATT, TTA,
TAG
- Nadniz
 - TATA
 - TATAC
 - TATACA
 - TATACAT
 - ...
 - TATACATTAG

Iscrpno pretraživanje

- $N!$ permutacija
- Vrlo težak problem (NP-hard) – nije poznat egzaktni algoritam koji ga rješava u polinomnom vremenu
- Koristimo heurističke metode

Pohlepni algoritam

- Heuristička metoda za rješavanje SCS-a:
Pohlepno ujedinjujemo dva niza s
maksimalnim preklapanjem
 - Npr. TAG se može nastavljati na ATA, zato što dva
zadnja znaka od ATA su ista kao prva dva od TAG

Pohlepni algoritam

- A**CA**, A**TA**, ATT, **CA**T, **TAC**, **TAG**, **TAT**, T**TA**
 - AC**AT**, **AT**A, ATT, TAC, TAG, TAT, TTA
 - ACAT**A**, ATT, **TAC**, TAG, TAT, TTA
 - ACATAC, **AT**T, TAG, T**AT**, TTA
 - ACATAC, TA**TT**, TAG, **TTA**
 - ACATAC, TAT**TA**, **TAG**
 - ACATAC, TATTAG
 - ACATACTATTAG
- Rekonstruirani niz različit i dulji od stvarnoga (TATACATTAG)
- Nije jednoznačno, u svakom koraku odabiremo spajanje između više kandidata

Pohlepno sestavljanje 6-torki niza a_long_long_long_time

ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long g_time ng_tim
ng_time ng_lon _long_ a_long long_l ong_ti ong_lo long_t g_long
ng_time g_long_ ng_lon a_long long_l ong_ti ong_lo long_t
ng_time long_ti g_long_ ng_lon a_long long_l ong_lo
ng_time ong_lon long_ti g_long_ a_long long_l
ong_lon long_time g_long_ a_long long_l
long_lon long_time g_long_ a_long
long_lon g_long_time a_long
long_long_time a_long
a_long_long_time

← nadniz

a_long_long_long_time

↓
a_long_long_time

**Napomena: ne
koristimo sve 6-torke**

Sastavljanje u slučaju ponavljanja

```
a_long_long_time
a_long long_t
_long_ong_ti
_long_l ng_tim
long_l g_time
ong_lo
ong_lo
ng_lon
ng_lon
g_long
g_long
_long_
_long_
```

```
a_long_long_long_time
a_long long_l ng_tim
_long_ong_lo g_time
_long_l ng_lon
ong_lo g_long
ng_lon_long_
g_long_long_t
_long_ong_ti
```

Sastavljanje u slučaju ponavljanja

```
a_long_long_long_long_long_time
a_long long_l g_long ng_tim
_long ng_lon long_l g_time
ong_lo _long_ ng_lon
g_long ong_lo _long_
long_t
ong_ti
```

Suvremeni pristupi za sastavljanje genoma

- Preklapanje/razmještaj/konsenzus (engl. *Overlap/layout/consensus – OLC*):
 - *de Bruijn* graf
- Oba pristupa utemeljena na grafovima

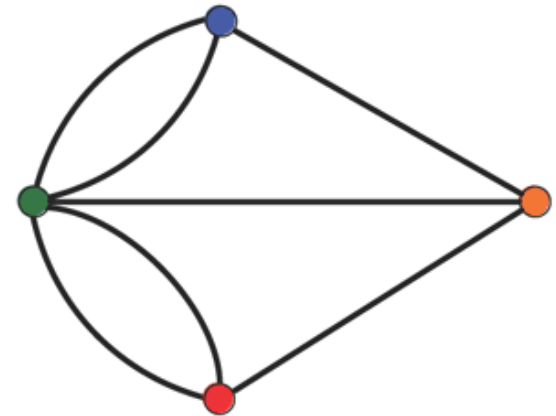
Teorija grafova

- 7 mostova Königsberga (danas Kaljningrad)
- dva otoka na rijeci Pregel
- kako obići grad da se prijeđe svaki most samo jedanput
- riješio Euler

a

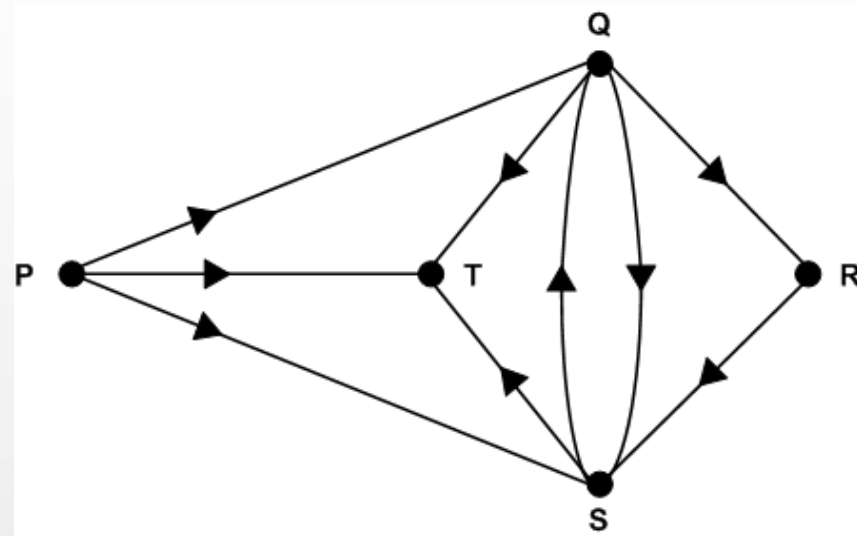


b



Definicija grafa

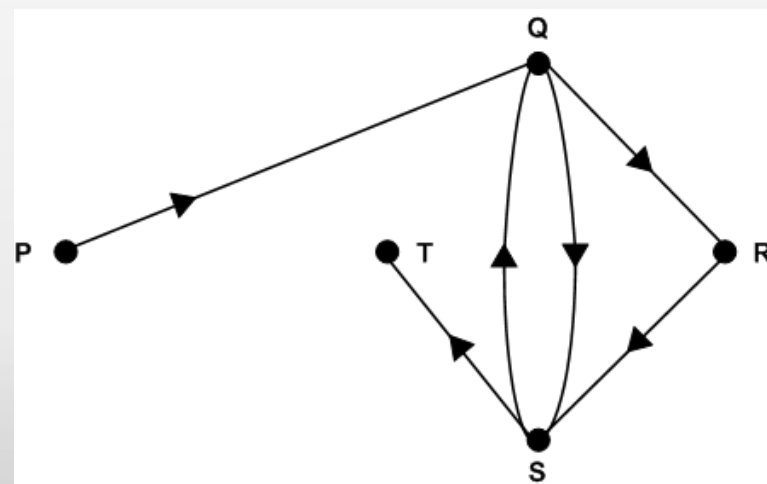
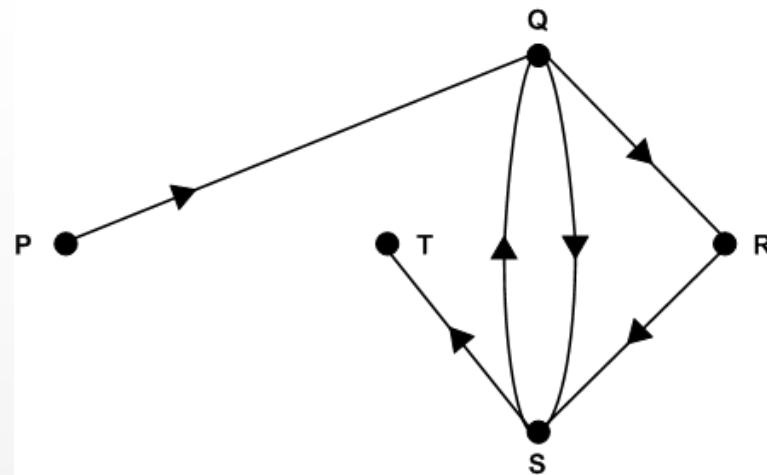
- Graf g s vrhovima (čvorovima)
 $V: \{P, T, Q, S, R\}$
- Skup bridova (veza između čvorova)
 $E: \{(P, T), (P, Q), (P, S), (Q, T), (S, T), (Q, S), (S, Q), (Q, R), (R, S)\}$
 - Usmjereni i neusmjereni
 - Svakom bridu može biti pridružena težina
- Šetnja od P do R : $(P, Q), (Q, R)$
- Šetnja od R do T : $(R, S), (S, Q), (Q, T)$
ili $(R, S), (S, T)$
- Šetnja od R do P : nije moguća



Preuzeto iz: *Introduction to Graph Theory*
Robert J. Wilson

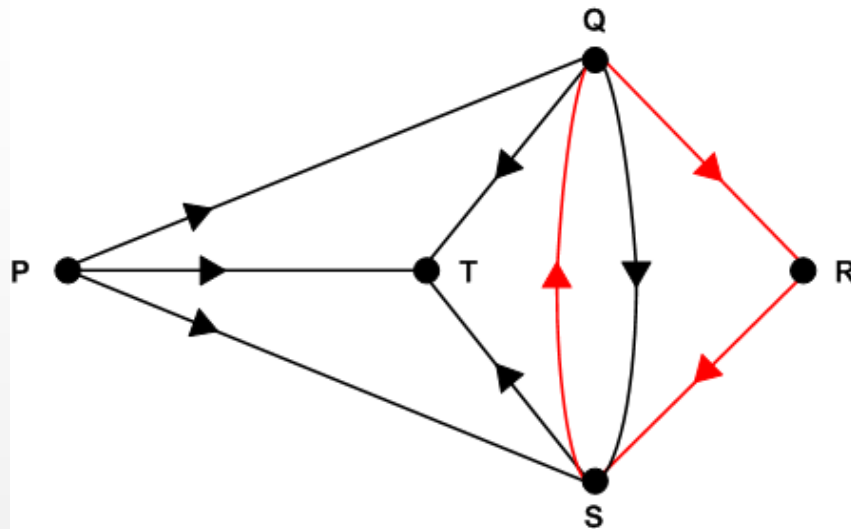
Definicija grafa

- Staza: šetnja grafom pri čemu je svaki brid posjećen jedanput.
- Primjer staze: $(P,Q), (Q,R), (R,S), (S,Q), (Q,S), (S,T)$
- Put: šetnja gdje je svaki vrh posjećen jedanput
- Primjer puta: $(P,Q), (Q,R), (R,S), (S,T)$



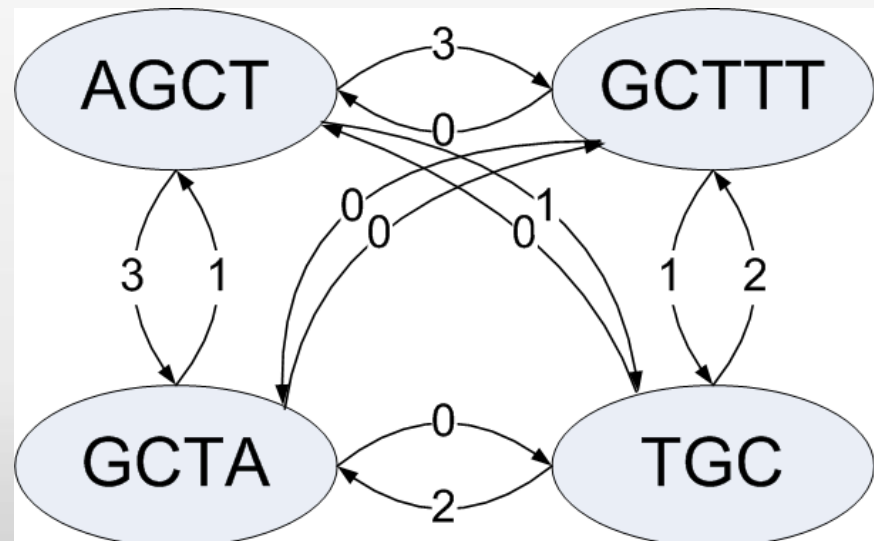
Definicija grafa

- Šetnja je zatvorena ako su početni i krajnji vrh isti
- Ciklus – zatvorena staza čiji se vrhovi ne ponavljaju (osim krajeva)
- Primjer ciklusa $(Q,R),(R,S),(S,Q)$



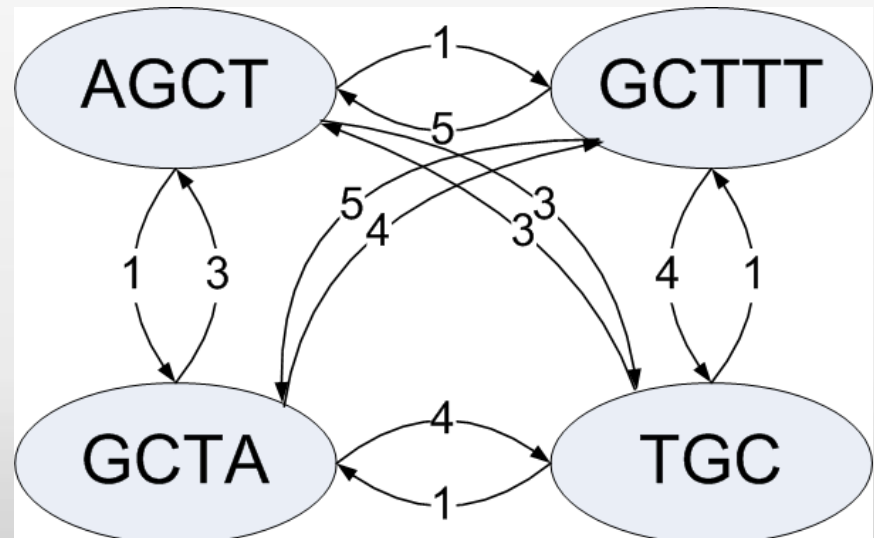
Graf preklapanja

- Vrhovi – nizovi
- Bridovi – preklapanja, težina – duljina preklapanja
- Bridovi kojima je polazište i ishodište u istom vrhu nisu prikazani



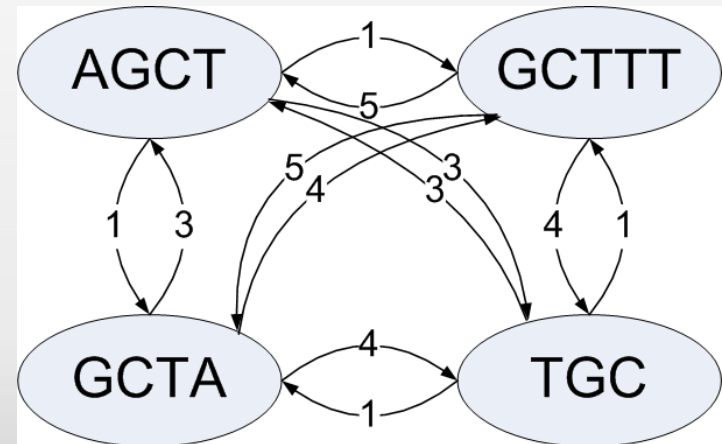
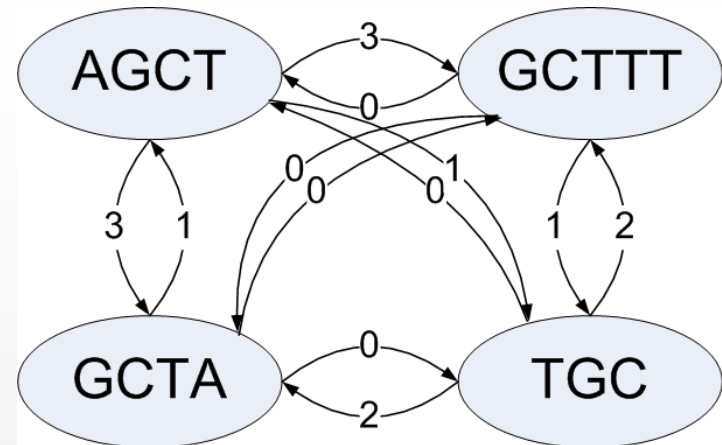
Graf udaljenosti

- Vrhovi – nizovi
- Bridovi – preklapanja, težina – duljina prefiksa prvoga niza koji nije u preklapanju
- Bridovi kojima je polazište i ishodište u istom vrhu nisu prikazani

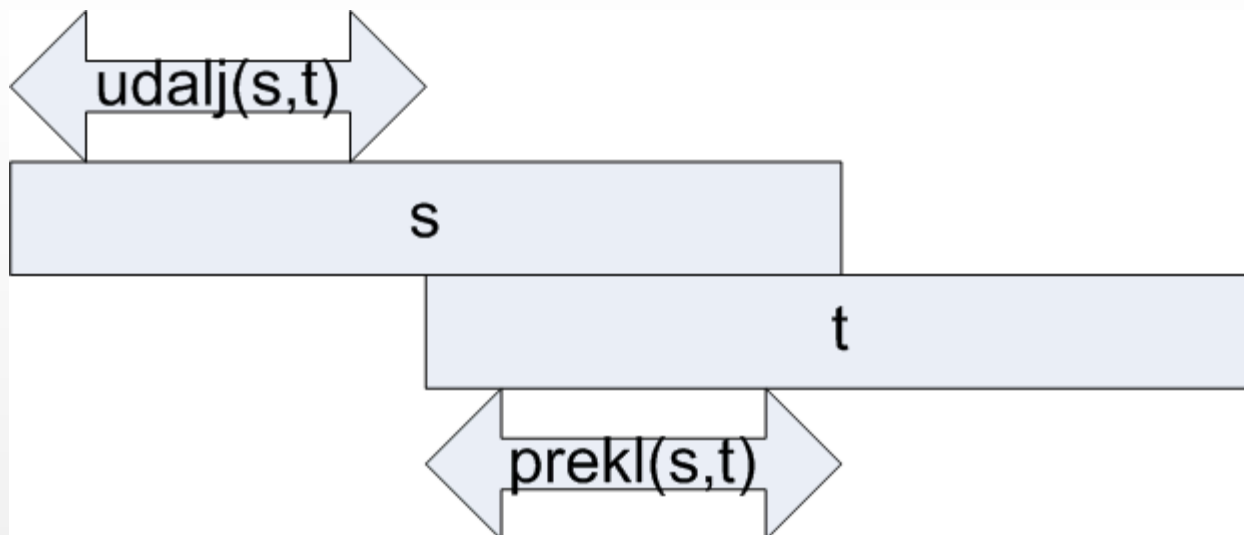


Grafovi

- Graf preklapanja
 - Težina puta – suma težina bridova+prefix prvog čvora koji nije u preklapanju
 - Maksimizacija težine puta
- Graf udaljenosti
 - Suma težina bridova + duljina niza u zadnjem čvoru
 - Minimizacija težine puta



Relacija između udaljenosti i preklapanja



- $udalj(s,t) + prekl(s,t) = duljina(s)$

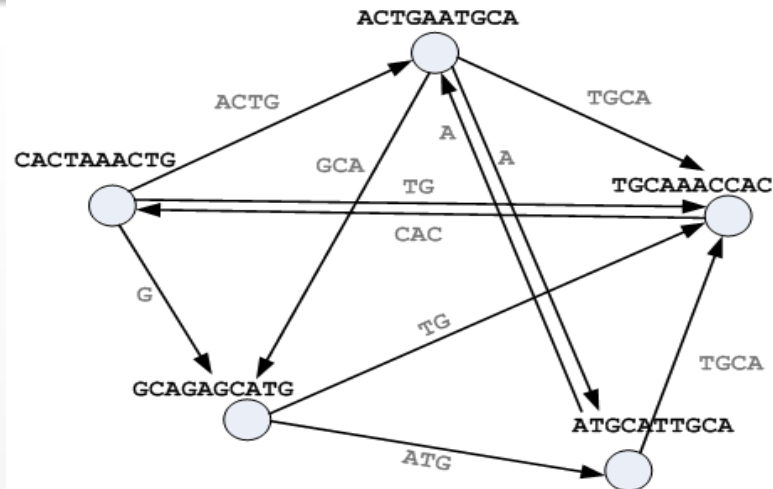
OLC pristup

- Pronaći **preklapanja** (*engl. Overlap*) poravnanjem sekvenci očitavanja
- **Razmjestiti** (*engl. Layout*) očitavanja na temelju poravnanja
- Postići **konsenzus** spajanjem svih sekvenci očitavanja ujedinjavanjem preklapanja
- Voditi računa da uređaji za sekvenciranje očitavaju s lijeva na desno i s desna na lijevo na slučajan način

OLC pristup

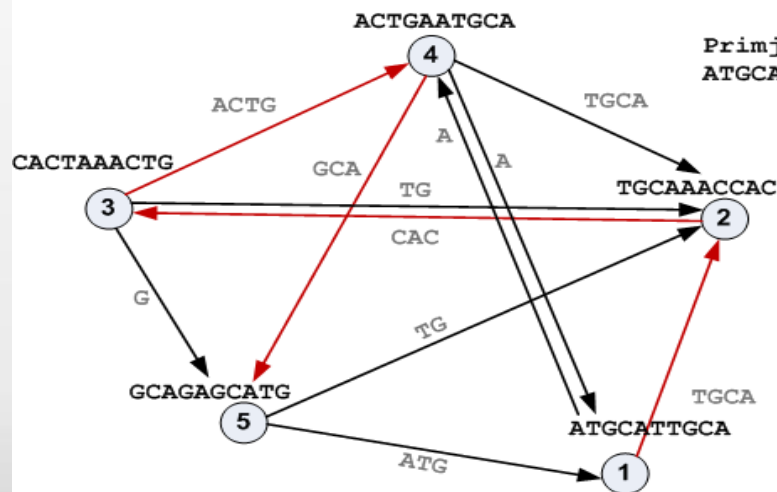
- Definicija grafa:
 - Svaki čvor je očitavanje
 - Postoji brid od čvora A prema čvoru B ako se sufiks od čvora A **značajno** preklapa s prefiksom čvora B
 - Cilj : Pronaći put koji posjećuje svaki čvor točno jedanput
 - Problem pronalaska Hamiltonovog puta

OLC pristup



Očitanja:
 ACTGAATGCA
 CACTAAACTG
 GCAGAGCATG
 ATGCATTGCA
 TGCAAACCAC

PREKLAPANJE



Primjer Hamiltonovog puta:
 ATGCATTGCA
 TGCAAACCAC

CACTAAACTG
 ACTGAATGCA
 GCAGAGCATG

RAZMJJEŠTAJ

Sastavljeni genom:
 ATGCATTGCAAACCACTAAACTGAATGCAGAGCATG

KONSENZUS

Preklapanje

- Poravnanje očitaje:
 - Računalno najzahtjevniji korak
 - Potrebno je istražiti sve međusobne parove poravnanja:
 - Dinamičko programiranje – preklapanje
 - Sufiksna stabla
 - Sufiksna polja
 - FM indeks
 - **Minimizeri** - približno

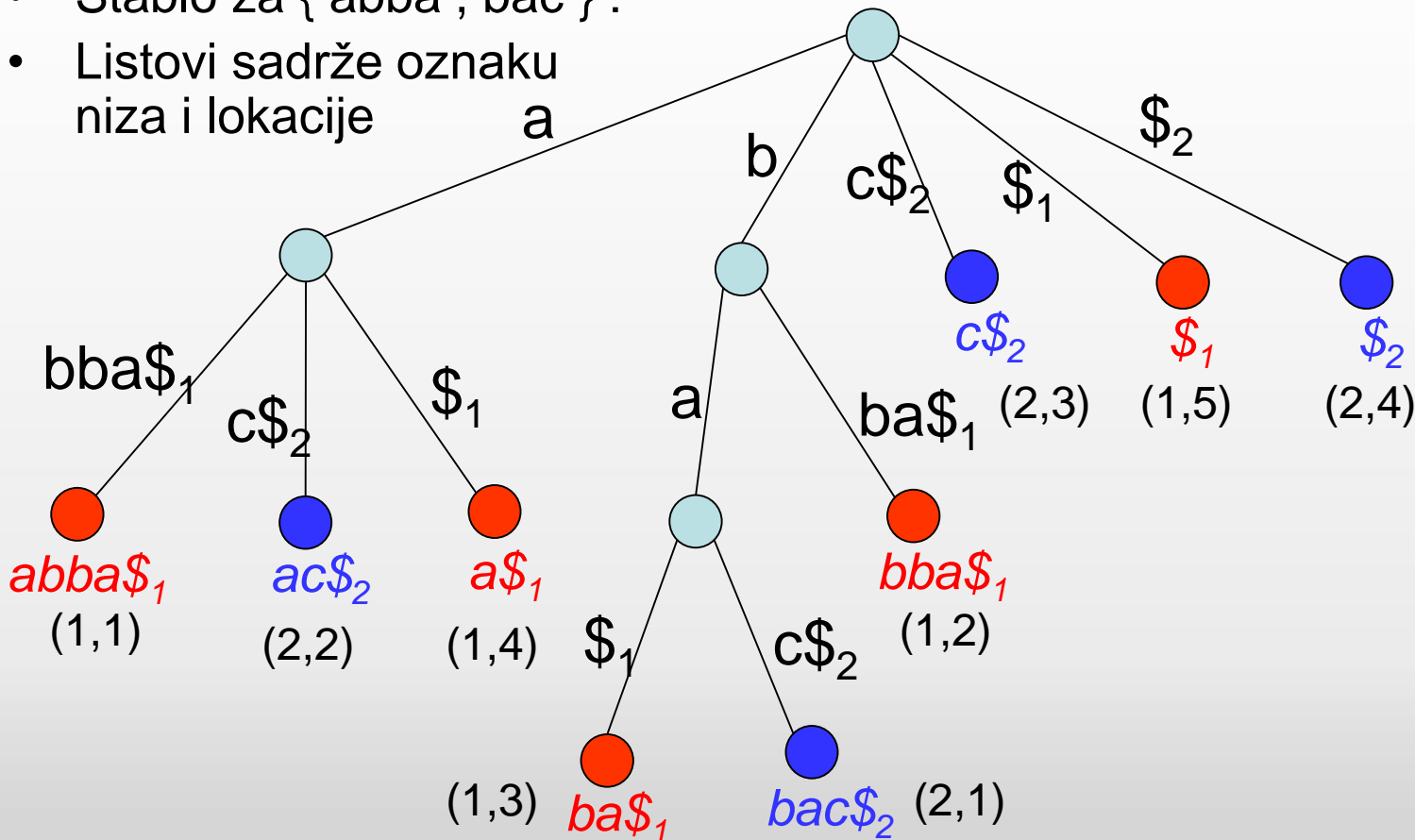
Dinamičko programiranje

- Iterirati preko svih parova nizova (s, t)
- Provjeriti sve moguće vrijednosti preklapanje duljine o
- Provjeriti da li je $s_{|s|-o+1, |s|} = t_{1, o}$
- Vremenska složenost

$$O(\sum_{i,j} l_i l_j) = O(\sum_i l_i \sum_j l_j) = O(\sum_i l_i L) = O(L^2)$$

Poopćeno sufiksno stablo GST

- Ulaz: skup nizova $\{T_1, \dots, T_k\}$
- Poopćeno sufiksno stablo: sadrži sve sufikse svih nizova
- Stablo za $\{ abba, bac \}$:
- Listovi sadrže oznaku niza i lokacije

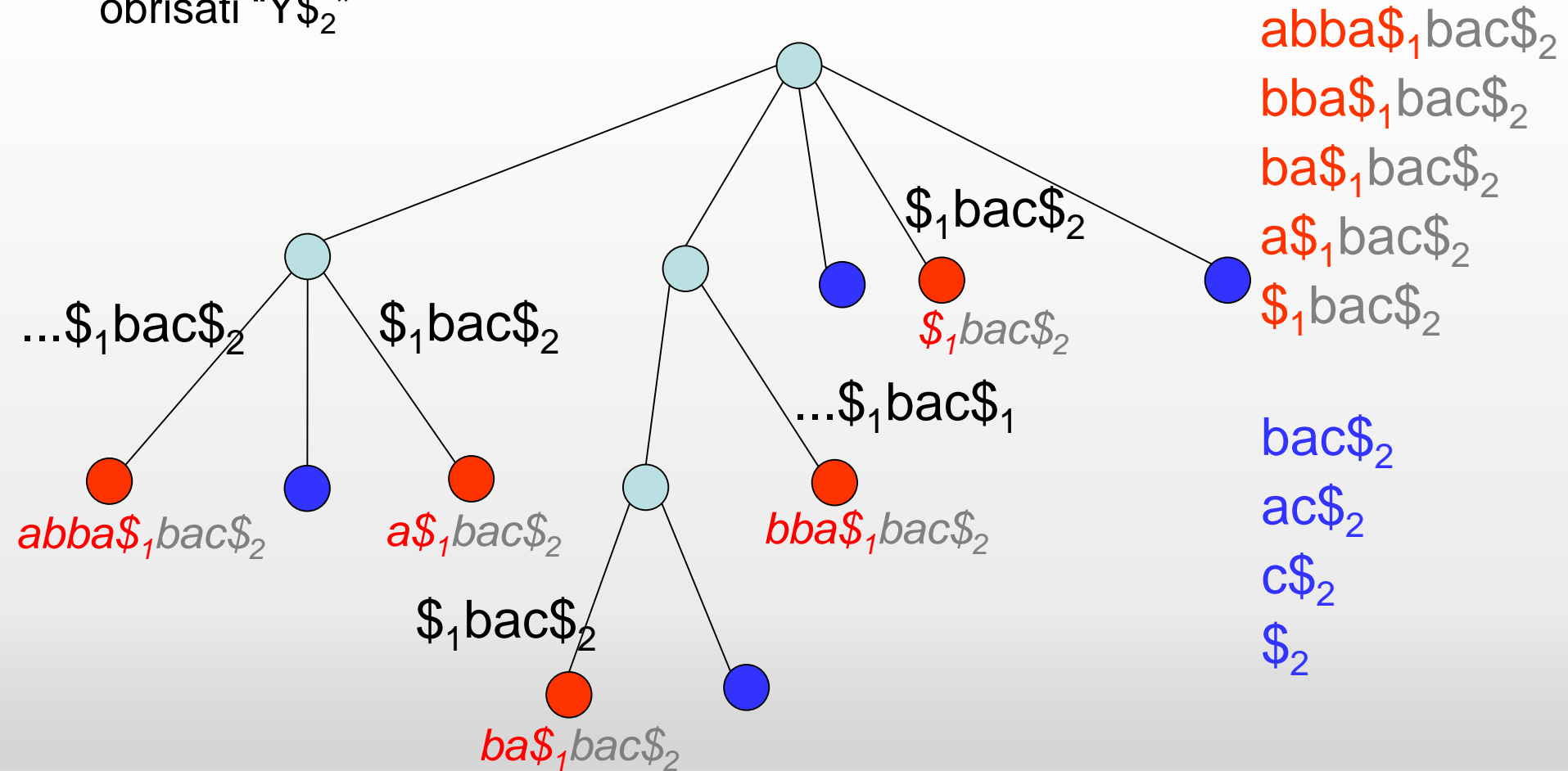


abba\$₁
 bba\$₁
 ba\$₁
 a\$₁
 \$₁

bac\$₂
 ac\$₂
 c\$₂
 \$₂

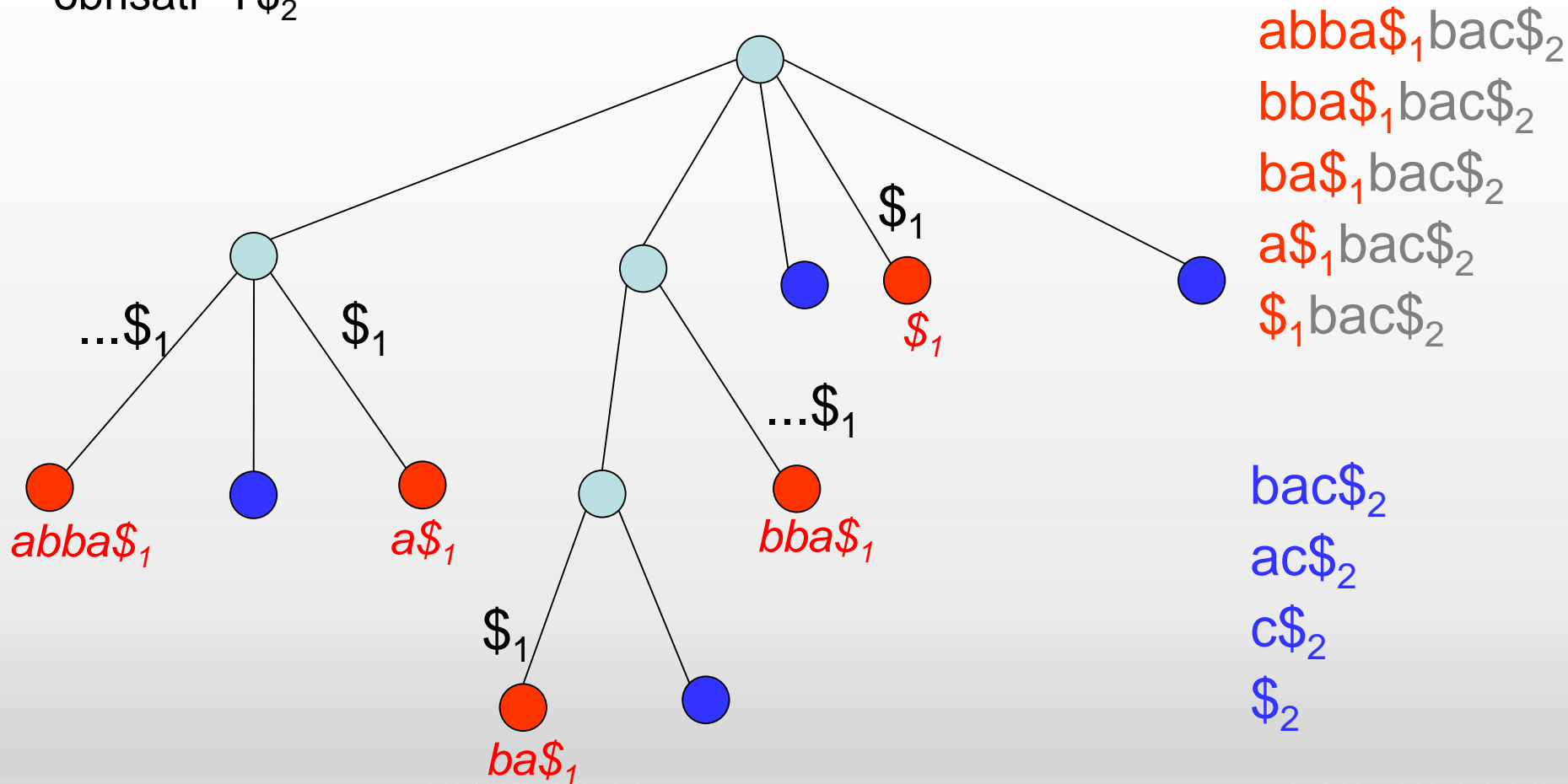
Izgradnja poopćenog sufiksnog stabla za $\{X, Y\}$

- Izgradnja sufiksnog stabla za $X\$_1Y\$_2$
- Bridovi prema crvenim listovima su označeni kao “ $\dots\$_1Y\$_2$ ”; obrisati “ $Y\$_2$ ”



Izgradnja poopćenog sufiksnog stabla za $\{X, Y\}$

- Izgradnja sufiksnog stabla za $X\$_1Y\$_2$
- Bridovi prema crvenim listovima su označeni kao “ $\dots\$_1Y\$_2$ ”; obrisati “ $Y\$_2$ ”



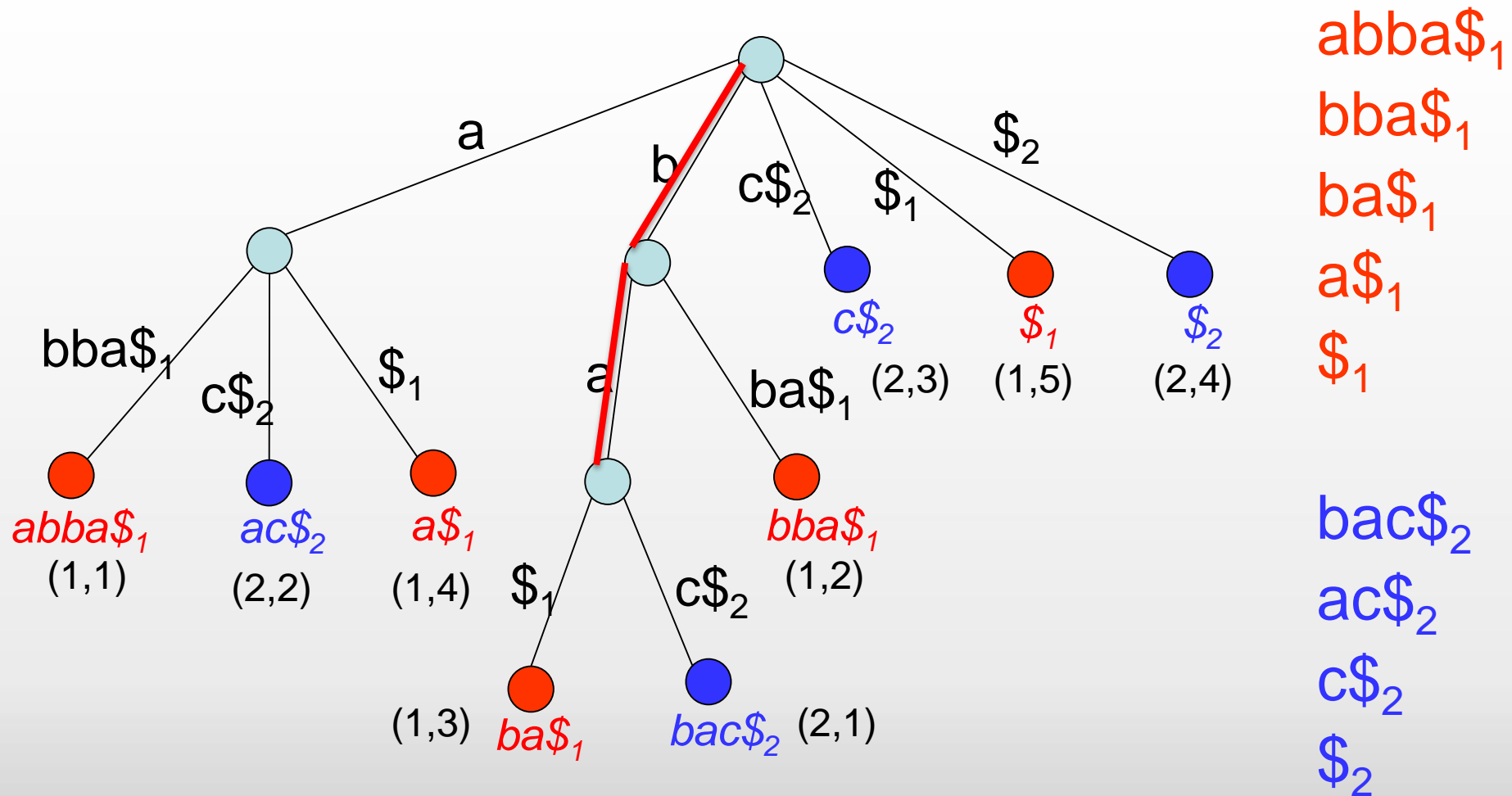
Izgradnja poopćenoga sufiksnoga stabla $\{T_1, \dots, T_k\}$

- Možemo koristiti isti trik
Konstruiramo sufiksno stablo za $T_1\$_1 \dots T_k\$_k$
- Linearna složenost: $O(|T_1| + \dots + |T_k|)$
- Jedan od načina je i da radimo stablo za niz po niz
 - Napravimo stablo za prvi niz
 - Na to stablo nastavimo stablo za drugi, itd.
- U praksi, koristimo modificirani osnovni algoritam za $T_1\$_1 \dots T_k\$_k$ (npr. Ukkonenov alg.) kojim dobijemo dodatno ubrzanje

Sufix-prefix preklapanje

- Skup nizova $T = \{T_1, \dots, T_k\}$ ukupne duljine N
- Kreiramo sufiksno stablo za $T_1\$_1 \dots T_k\$_k$
- Gledamo najdulji sufiks α od T_i koji je prefiks od T_j
- α je početni dio puta od korijena stabla do lista označenog $(j, 1)$ koji završava u nekom unutarnjem čvoru koji sadrži brid s oznakom $\$_i$
- To mora biti najdublji čvor na navedenom putu
- Duljina pripadajućeg sufiks-prefix preklapanja je dana dubinom unutarnjeg čvora

Sufiks – prefiks preklapanje



Sufiks – prefiks preklapanje

- Za skup nizova $T = \{T_1, \dots, T_k\}$ ukupne duljine N najdulja preklapanja mogu biti riješena u $O(N + k^2)$ vremena.
- Ako je $k' \leq k^2$ broj uređenih parova nizova koji imaju poravnanja veća od nula, onda možemo naći ta poravnanja u $O(N + k')$

Preklapanje koristeći sufiksna polja

- Konstruirati sufiksno polje od niza $T_1\$T_2\$...T_k\$$
- Pronaći pojavljivanje svakoga T_i procesiranjem slovo po slovo
- Traženje početka i kraja polja za zadani znak koristeći binarna pretraživanja:
 - Lijeva granica – prva pojava traženoga znaka
 - Desna granica – zadnja pojava traženoga znaka

Primjer

- Nizovi AGT, GTC, TCA
- AGT\$GTC\$TCA\$

Primjer

- Sufiksi:

- AGT\$GCT\$TCA\$
- GT\$GTC\$TCA\$
- T\$GTC\$TCA\$
- \$GTC\$TCA\$
- GTC\$TCA\$
- TC\$TCA\$
- C\$TCA\$
- \$TCA\$
- TCA\$
- CA\$
- A\$
- \$

AGT
GTC
TCA

Primjer

AGT
GTC
TCA

- Sortirani sufiksi:
 - \$
 - \$GTC\$TCA\$
 - \$TCA\$
 - A\$
 - AGT\$GTC\$TCA\$
 - C\$TCA\$
 - CA\$
 - GT\$GTC\$TCA\$
 - GTC\$TCA\$
 - T\$GTC\$TCA\$
 - TC\$TCA\$
 - TCA\$

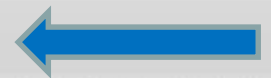
Primjer

- Pretraga za preklapanjima s AGT

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$

AGT
GTC
TCA



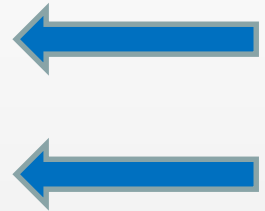
Primjer

- Pretraga za preklapanjima s AGT
- A

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$

AGT
GTC
TCA



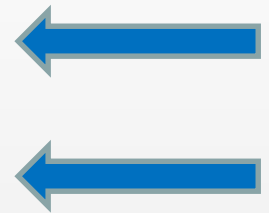
Primjer

- Pretraga za preklapanjima s AGT
- Pokušamo završiti preklapanje
- A\$

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$

AGT
GTC
TCA



Primjer

- Pretraga za preklapanjima s AGT
- A\$
- Preklapanje pronađeno:
 - TCA AGT

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$



AGT
GTC
TCA

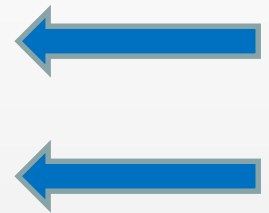
Primjer

- Pretraga za preklapanjima s AGT
- Uklanjamo dollar znak i dodamo slijedeći znak
- AG

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$

AGT
GTC
TCA



Primjer

- Pretraga za preklapanjima s AGT
- Uklanjammo dollar znak i dodamo slijedeći znak
- AG

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$

AGT
GTC
TCA



Primjer

- Pretraga za preklapanjima s AGT
- Pokušavamo završiti preklapanje
- AG\$ - nema podudaranja

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$



AGT
GTC
TCA

Primjer

- Pretraga za preklapanjima s AGT
- Uklanjamo dollar znak i dodamo slijedeći znak
- AGT

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$

AGT
GTC
TCA



Primjer

AGT
GTC
TCA

- Pretraga za preklapanjima s AGT
- Pokušavamo završiti preklapanje
- AGT\$
- Preklapanje pronađeno
 - **AGT AGT** – samopreklapanje

- Sortirani sufiksi:

- \$
- \$GTC\$TCA\$
- \$TCA\$
- A\$
- AGT\$GTC\$TCA\$
- C\$TCA\$
- CA\$
- GT\$GTC\$TCA\$
- GTC\$TCA\$
- T\$GTC\$TCA\$
- TC\$TCA\$
- TCA\$



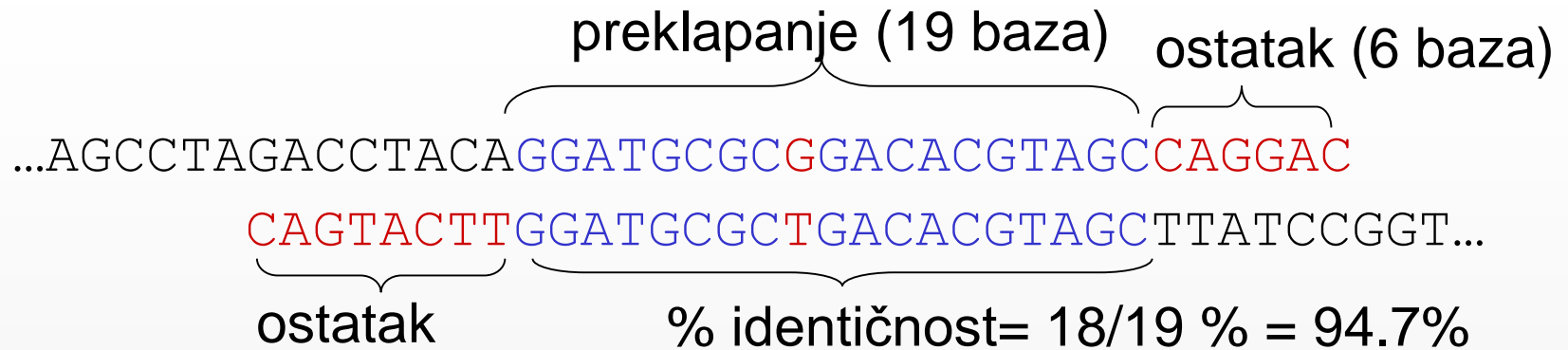
Vremenska složenost

- $O(N \log N + k')$ gdje je k' ukupan broj preklapanja
- Sufiksno stablo = sufiksno polje + LCP
- Koristeći sufiksno polje, LCP i još par pomoćnih struktura moguće je postići složenost $O(N + k')$

FM indeks

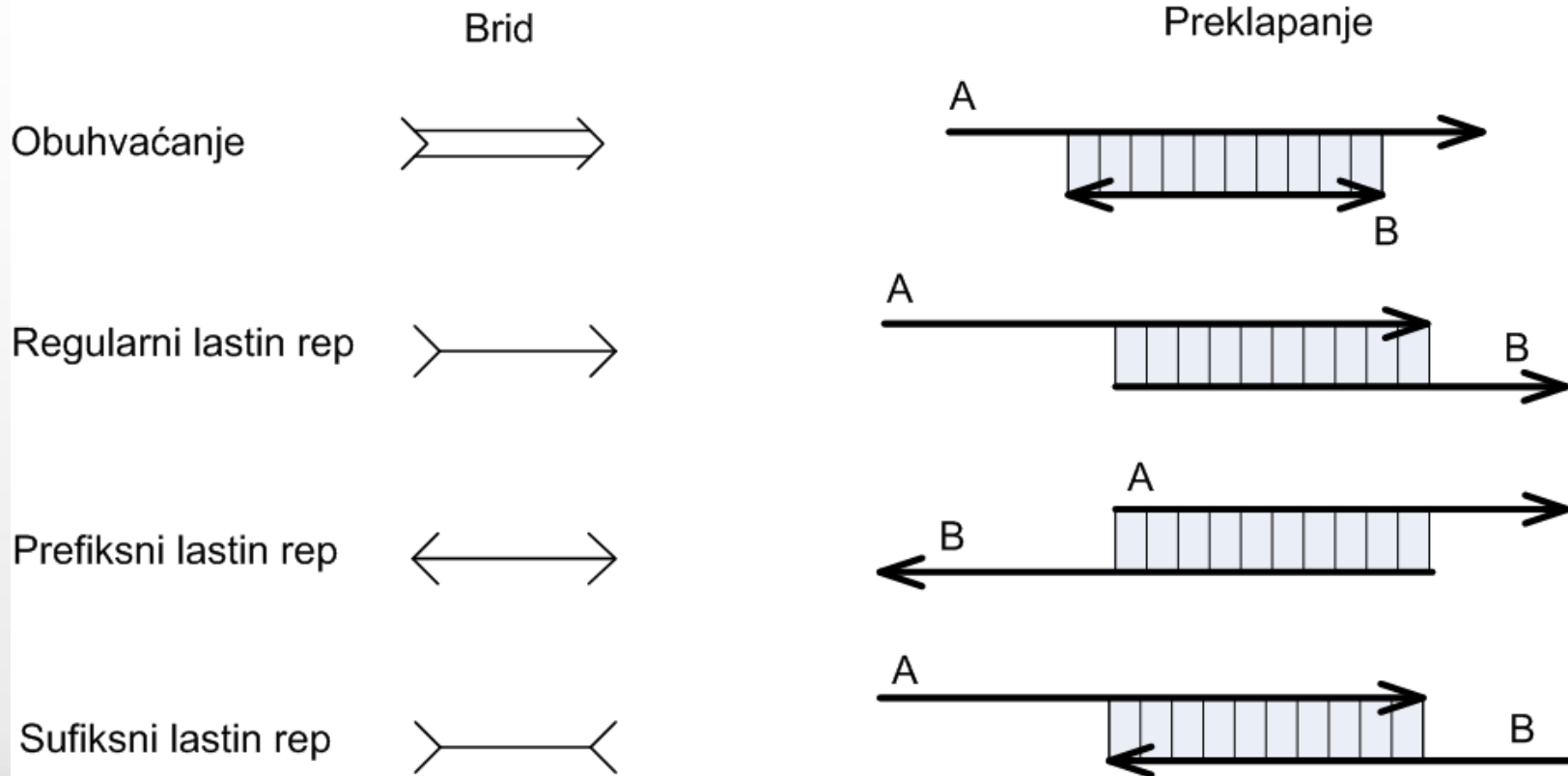
- Korištenjem FM indexa isto tako se može postići linearna složenost pretraživanja
- Velik napor je usmjeren na razvijanje algoritama kada postoji do k razlika između sufiksa i prefiksa – kritično za podatke s velikom greškom (npr. PacBio)

Preklapanje između dvije sekvence



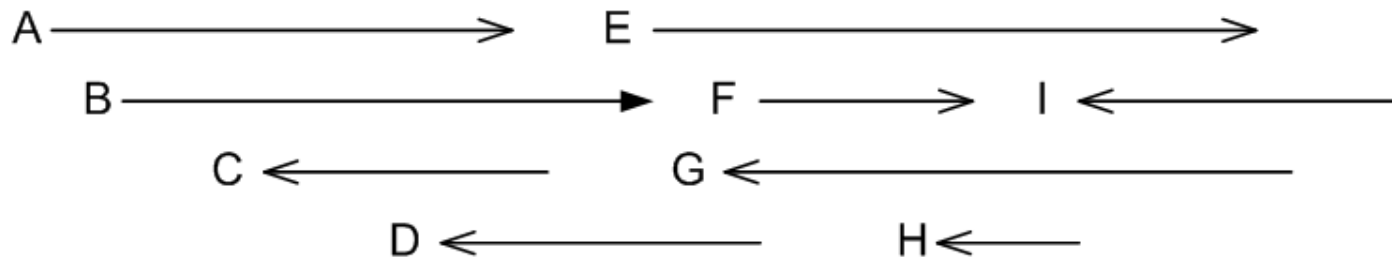
- **preklapanje** – regija sličnosti između sekvenci
- **ostatak** – neporavnati krajevi sekvenci
- Odluka o spajanju ovisi o:
 - duljini preklapanja
 - % identičnosti preklapljenih regija
 - maksimalnom ostatku

Taksonomija preklapanja

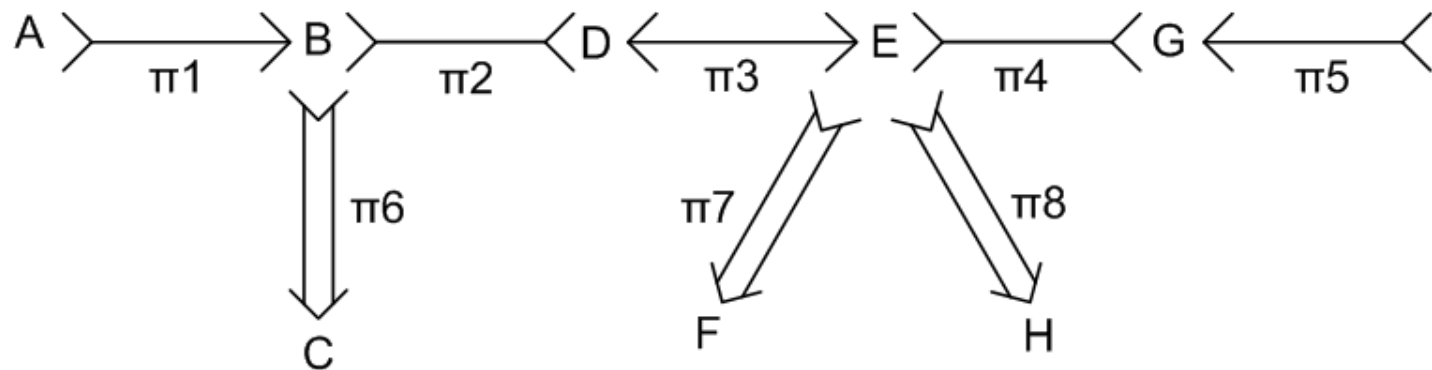


Razmještanje

Razmještaj:

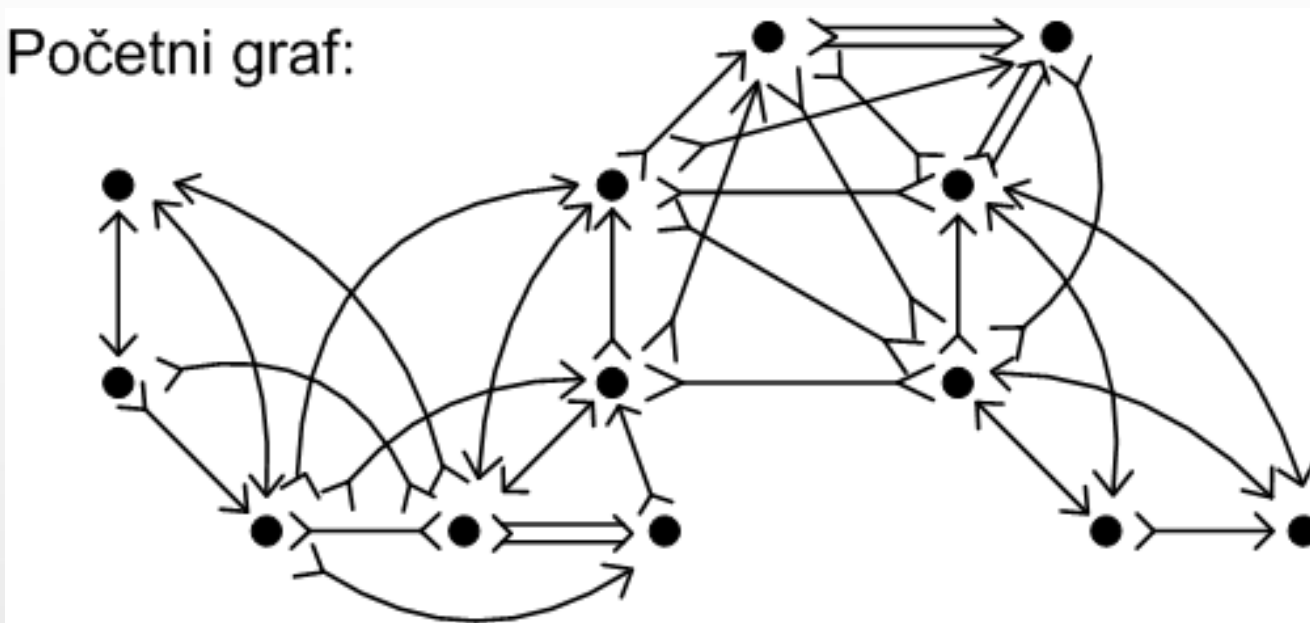


Graf:



Primjer početnoga grafa

Početni graf:



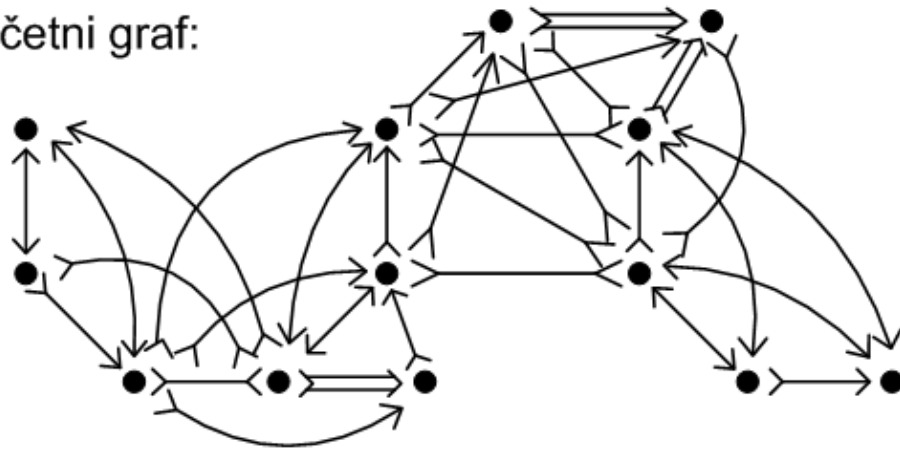
Pojednostavljenje grafa

1. Uklanjanje obuhvaćenih očitavanja
2. Uklanjanje tranzitivnih bridova
3. Ujedinjavanje očitavanja

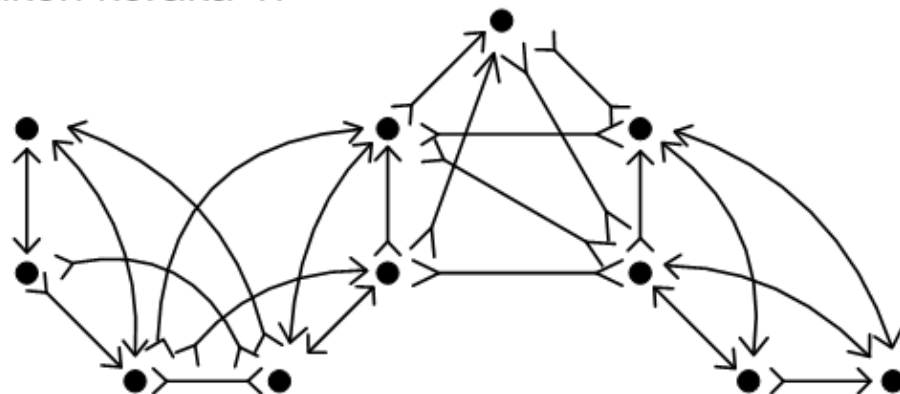
1. Uklanjanje obuhvaćenih očitavanja

- Uklanjanje svih obuhvaćenih očitavanja (čvorova) i svih bridova povezanih s njima

Početni graf:

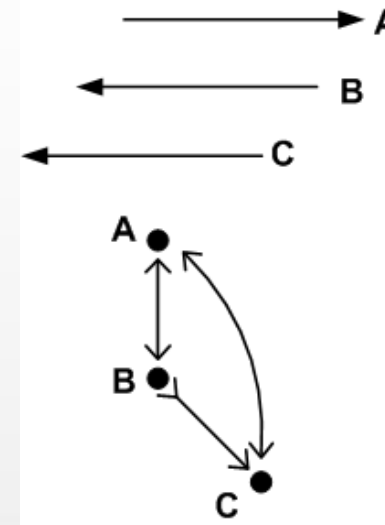


Nakon koraka 1:



2. Uklanjanje tranzitivnih bridova

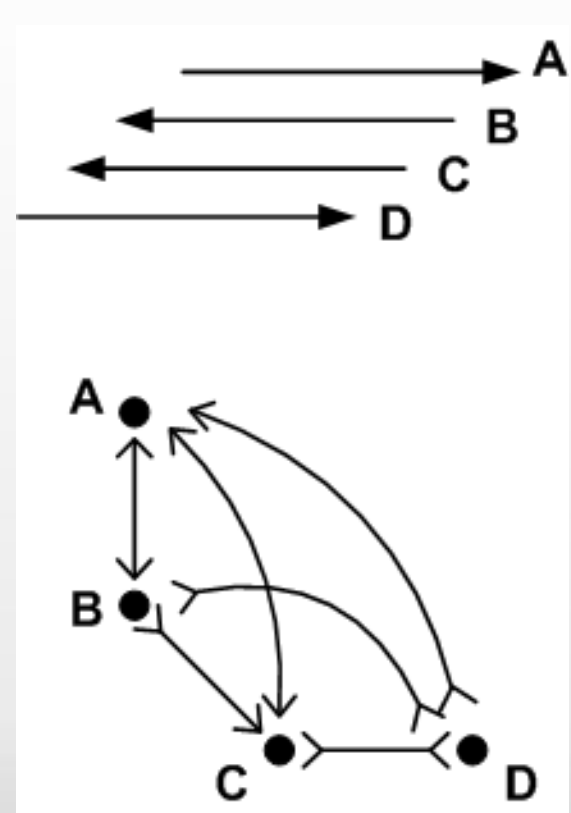
- Tranzitivan brid/preklapanje
 - preklapanje između A i C sadržano u preklapanjima između A i B te B i C do na razinu neke pogreške ε
 - strelice u vrhovima A i C jednake
 - strelice u vrhu B različite
- Uklanjamo tranzitivni brid (A \rightarrow C)



A = TGAAC
B = CCTGA
C = ACCTG

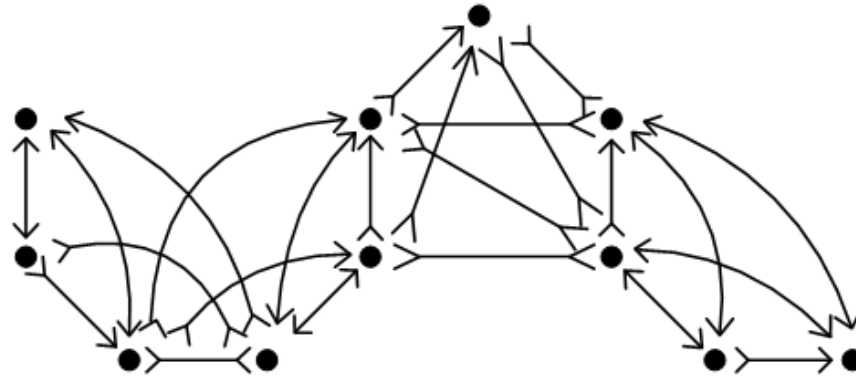
2. Uklanjanje tranzitivnih bridova

- Prilikom uklanjanja tranzitivnih bridova voditi računa da uklonjeni bridovi neki drugi brid čine tranzitivnim
 - npr. $A \rightarrow C$ zajedno sa $C \rightarrow D$, čini brid $A \rightarrow D$ tranzitivnim
- Pamtiti uklonjene bridove

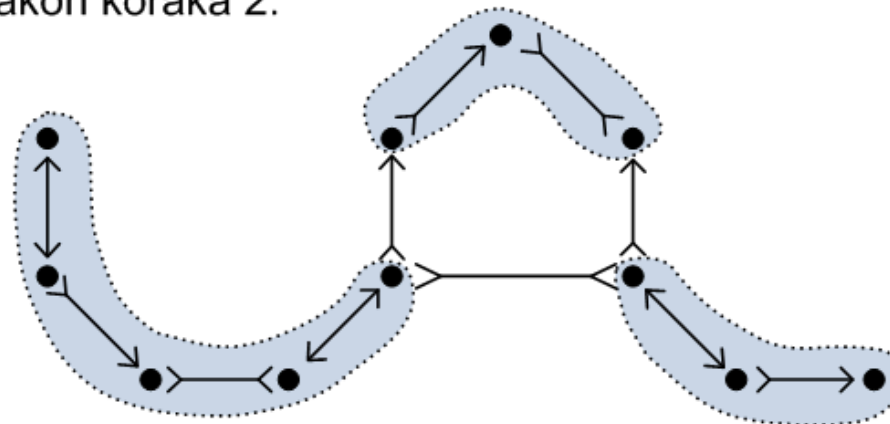


OLC graf nakon drugog koraka

Nakon koraka 1:



Nakon koraka 2:

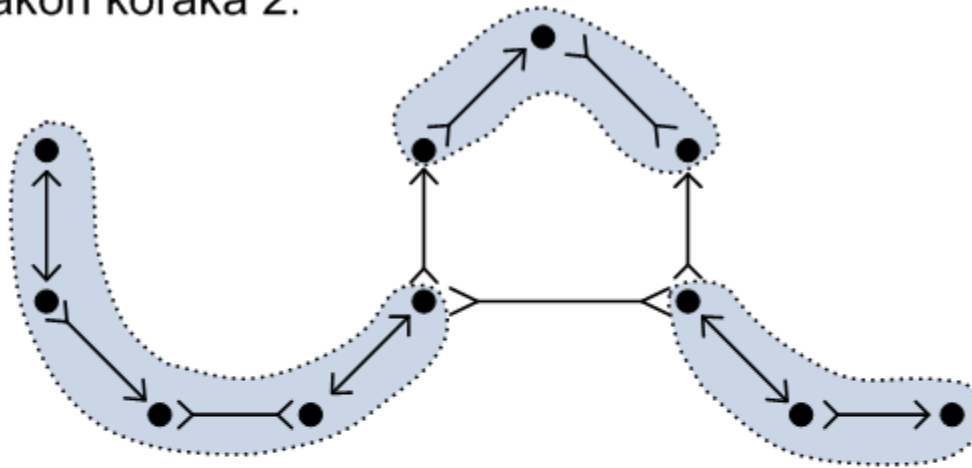


3. Ujedinjavanje

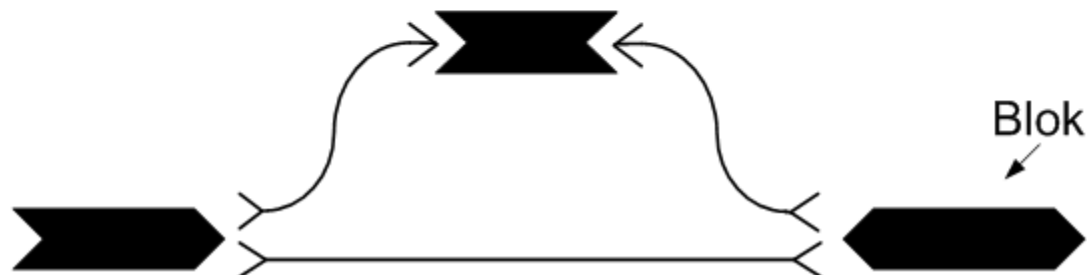
- Ujedinjavanje:
 - Ako imamo vrhove a i b i brid preklapanja između njih π
 - Uvjeti ujedinjavanja a i b
 - Smjerovi svih drugih vrhova strelica spojenih u a okrenuti su u drugom smjeru od brida π
 - Smjerovi svih drugih vrhova strelica spojenih u b okrenuti su u drugom smjeru od brida π
 - Tada ujedinjavamo vrhove a i b u jedan blok

3. Ujedinjavanje

Nakon koraka 2:



Nakon koraka 3:



Konsenzus

- Konsenzus sekvenca se određuje iz sastavljenih fragmenata
- Potreban je dovoljan broj očitavanja da se osigura statistički značajan konsenzus
- Greške očitavanja se korigiraju u ovoj fazi
- Za uspješan konsenzus potrebno je sačuvati sva očitavanja iz faze razmještanja

Određivanje konsenzus sekvence

```
TAGATTACACAGATTACTGA TTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAACTA
TAG TTACACAGATTATTGACTTCATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
TAGATTACACAGATTACTGACTTGATGGGGTAA CTA
```



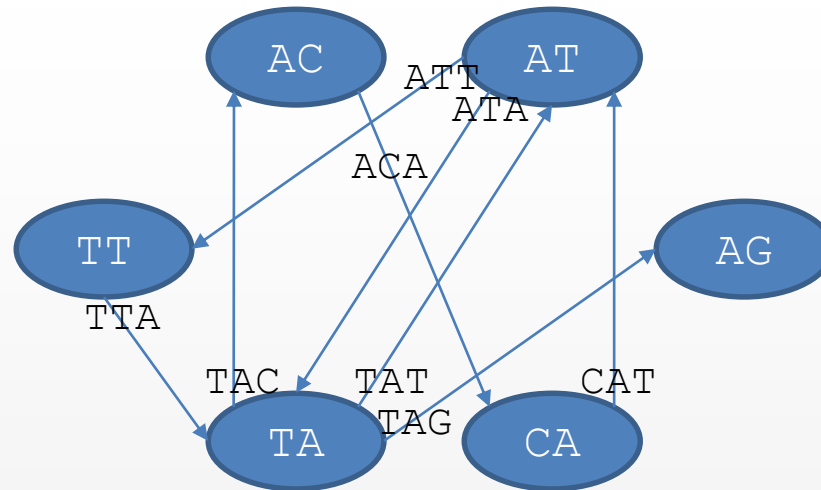
```
TAGATTACACAGATTACTGACTTGATGGCGTAA CTA
```

- Određivanje višestrukoga poravnanja očitavanja
- Određivanje konsenzusa za svaku bazu težinskim glasanjem

Drugačija definicija grafa

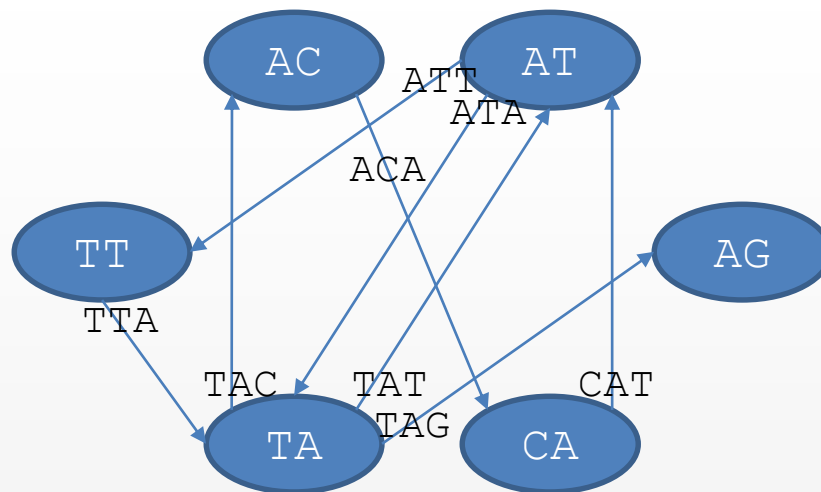
- Svaki **brid** je očitavanje
- Dva vrha povezana bridom su izvedeni od prefiksa i sufiksa pripadajućeg očitavanja
- Različita očitavanja mogu dijeliti čvorove:
- Cilj: Pronaći put koji obilazi svaki brid točno jedanput – Problem pronalaska Eulerovog puta

Formulacija Eulerove staze



- Očitavanja: ACA, ATA, ATT, CAT, TAC, TAG , TAT, TTA
- Duljina očitavanja: $k = 3$
- Pretpostavimo da je svako očitavanje sadržano u dva vrha, jedan vrh sadrži prefiks očitavanja duljine 2 (općenito $k - 1$), a drugi sufiks očitavanja duljine 2 (općenito $k - 1$). Postoji brid usmjeren od prvoga k drugom, a na bridu se nalazi samo očitavanje.

Pronalazak Eulerove staze

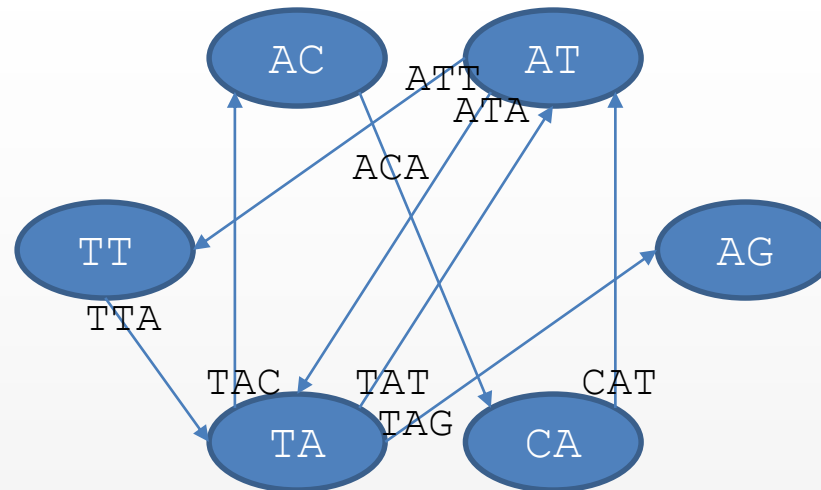


- Postojanje Eulerove staze:

- Ulazni stupanj vrha je broj bridova koji ulaze u njega
- Izlazni stupanj vrha je broj bridova koji iz njega izlaze
- Da bi povezani usmjereni graf imao Eulerovu stazu potrebno je ispuniti slijedeće nužne i dovoljne uvjete

- Najviše jedan vrh ima $(\text{izlazni stupanj} - \text{ulazni stupanj}) = 1$
- Najviše jedan vrh ima $(\text{izlazni stupanj} - \text{ulazni stupanj}) = -1$
- Svi ostali vrhovi imaju $(\text{izlazni stupanj} - \text{ulazni stupanj}) = 0$

Pronalazak Eulerove staze

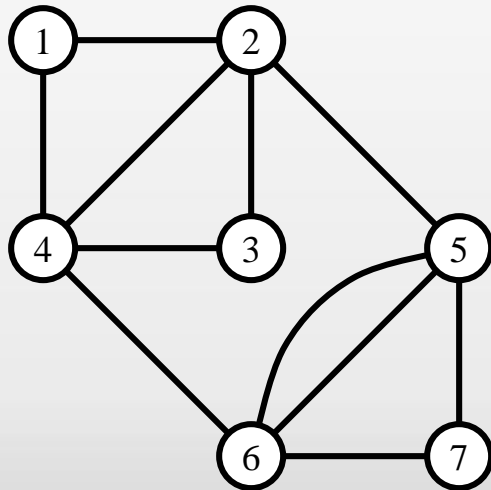


- (Hierholzerov algoritam):

- Krenuti s vrhom kojem je izlazni stupanj za jedan veći od ulaznoga stupnja (ako takav ne postoji, krenuti od bilo kojeg)
- Slijediti bilo koji nekorišteni brid za posjetu drugim vrhovima dok se ne zaglavimo
 - Zaglaviti se možemo samo u početnom vrhu ili vrhu koji ima za jedan veći ulazni stupanj od izlaznoga stupnja
- Ako bilo koji već posjećen vrh ima nekorištenih bridova, ponoviti gornju proceduru s time vrhom kao početnim. Put mora završiti u istom vrhu. Ujediniti novi put sa starim.

Traženje Eulerove staze

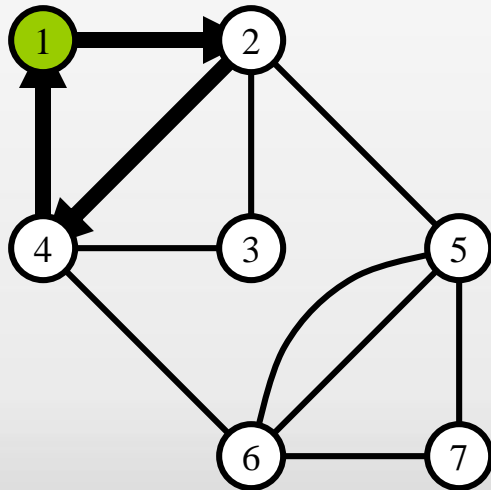
- Primjer



Traženje Eulerove staze

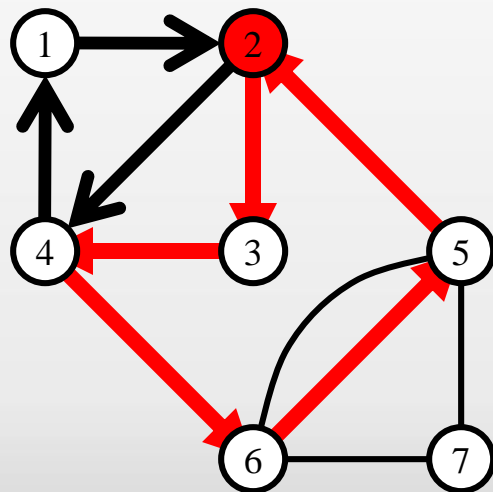
- Primjer

Korak 1: 1-2-4-1



Traženje Eulerove staze

- Primjer



Korak 1: 1-2-4-1

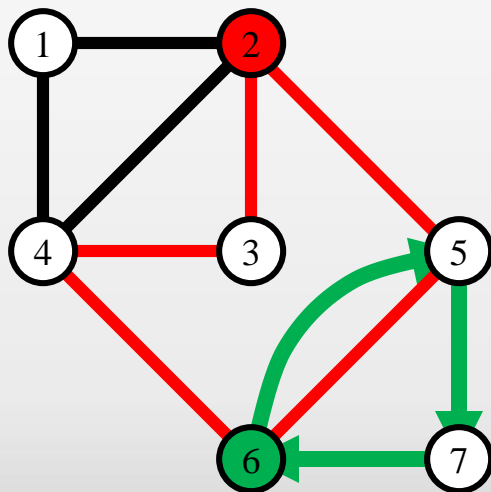
Korak 2: $v = 2$

Put: 2-3-4-6-5-2

Ujedinimo ih

1-2-3-4-6-5-2-4-1

Traženje Eulerove staze



Korak 1: 1-2-4-1

Korak 2: $v = 2$

Put: 2-3-4-6-5-2

Ujedinimo ih

1-2-3-4-6-5-2-4-1

Korak 3: $v = 6$

Put: 6-5-7-6

Ujedinimo ih

1-2-3-4-6-5-7-6-5-2-4-1

Svi bridovi posjećeni. Kraj.

De Bruijn graf

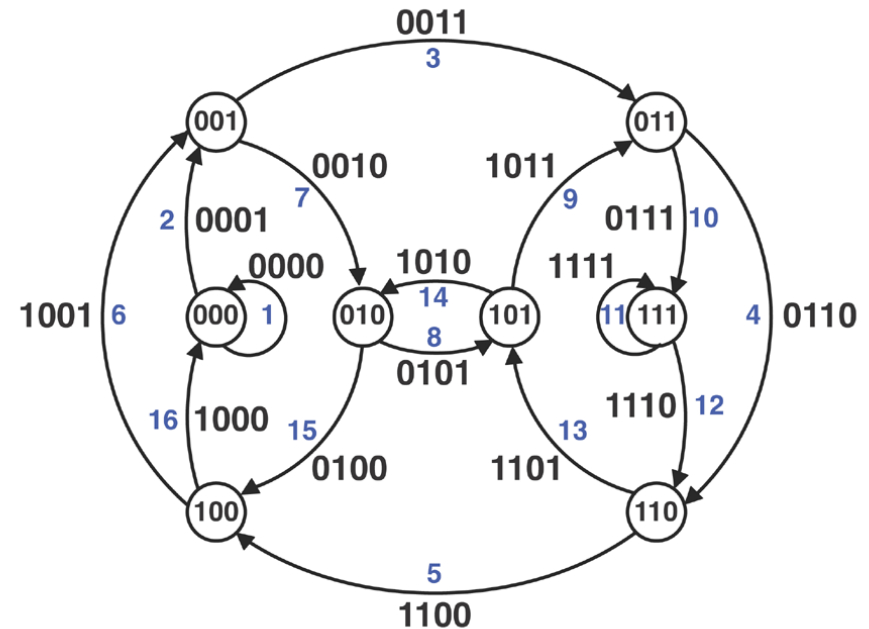
- U originalnoj definiciji predložio ih je nizozemski matematičar Nicolaas de Bruijn 1946.:
 - cilj – pronaći najkraći cirkularni nadniz koji sadrži sve moguće podnizove duljine k (k -torke) dane abecede
 - postoji n^k k -torki gdje je n duljina abecede
 - npr. za abecedu $\{0,1\}$ sve moguće trojke su: 000, 001, 010, 011, 100, 101, 110, 111.
 - cirkularni nadniz je 0001110100
 - kako ga pronaći za bilo koji k i bilo koju abecedu?

De Bruijn graf

- de Bruijn graf je graf za koga je svaka $(k-1)$ - torka određene abecede predstavljena kao vrh.
- Povezati dva vrha bridom koji predstavlja k -torku ukoliko jedan vrh predstavlja $k-1$ prefiks, a drugi $k-1$ sufiks te k -torke.

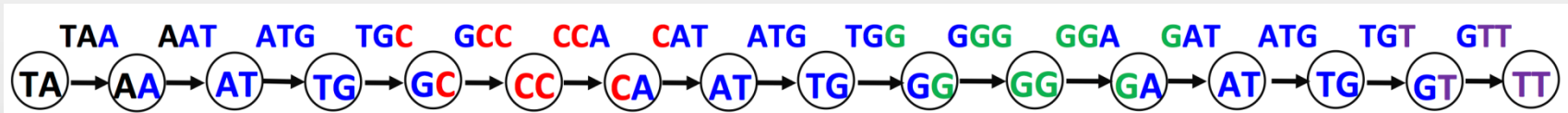
Primjer de Bruijn grafa

- De Bruijn graph B za $k = 4$ i $\{0,1\}$ abecedu
- Graf ima Eulerov ciklus (označen plavim brojevima) zato što je svaki izlazni i ulazni stupanj jednak 2
- Zapisujući samo prvi znak svakoga brida dobijemo ciklički nadniz

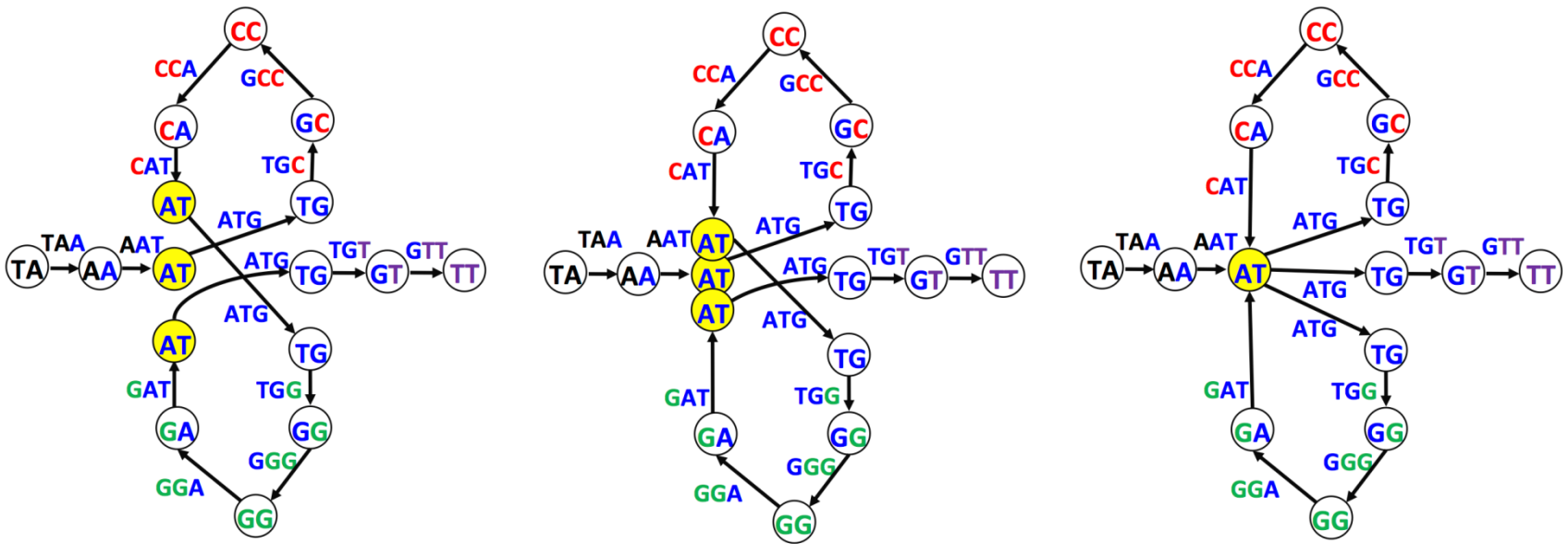


Izrada grafa

- Početna sekvenca:
TAATGCCATGGGATGTT
- Očitavanja:
TAA AAT ATG TGC GCC CCA CAT ATG
TGG GGG GGA GAT ATG TGT GTT
- Idealni graf

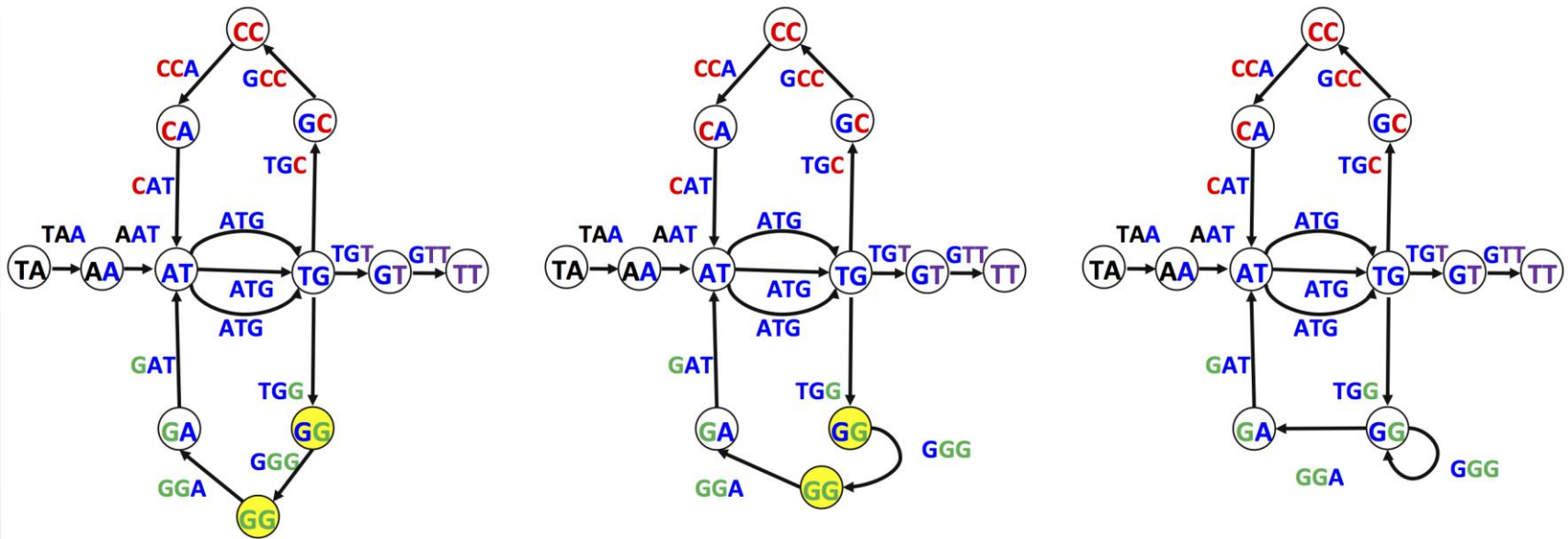


Izrada grafa



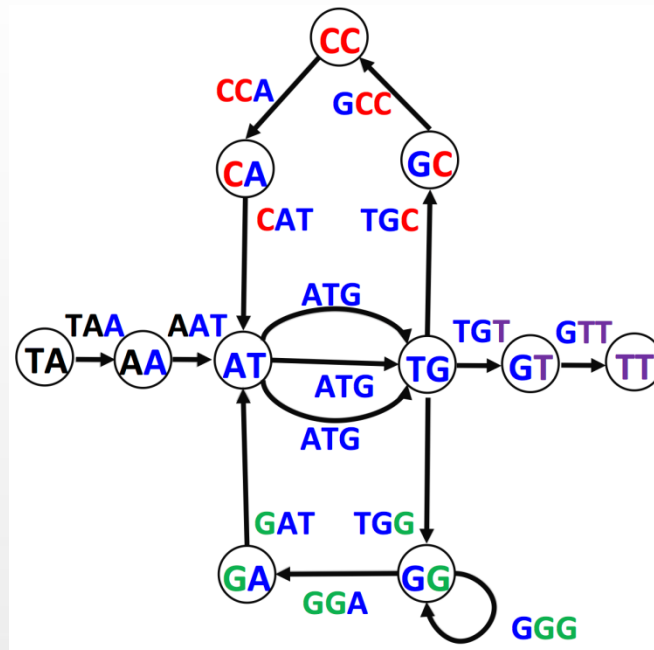
- Spajanje čvorova koji imaju istu oznaku:
 - spojimo prvo npr. Sve AT čvorove zadržavši njihove bridove
 - nakon toga spajamo TG čvorove

Izrada grafa



- Spajanje čvorova koji imaju istu oznaku:
 - Čvorovi GG imaju između sebe brid
 - Taj brid se onda zatvara sam u sebe

Konačan graf



- Rješavanje grafa na način da nađemo Eulerovu stazu
- U ovom slučaju imamo nekoliko rješenja (TAATGCCATGGGATGTT i TAATGGGATGCCATGTT)
- Ispisujemo prvo slovo svakoga brida, osim zadnjeg brida kojeg ispišemo cijeloga

de Bruijnov graf za sastavljanje sekvence

- Korišćenje de Bruijnovog grafa za sastavljanje sekvence :
 - Razmatramo jedino k-torke koje su podsekvence nekog očitavanja
 - Obično uzimamo duljine k do 20
 - U praksi, samo mali postotak od 4^k mogućih k-torki se pojavljuje u očitanjima

Izrada de Bruijn grafa

- Lomljenje očitavanja
 - lijeva očitavanja ne sadrže sve 10–torke
 - ako ih razlomimo na 5-torke (desno) dobijemo savršenu pokrivenost
- Kompromisan odabir duljine
 - kraća očitavanja – veća vjerojatnost potpune pokrivenosti, no graf je zapetljaniji i teže je pronaći pravu Eulerovu stazu
 - dulja očitavanja – jednostavniji graf, no manja pokrivenost

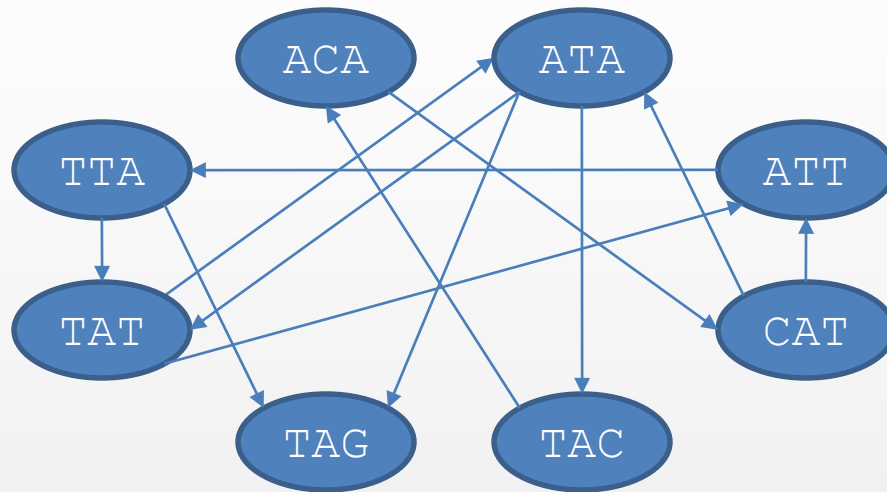
atgccgtatggacaacgact
atgccgtatg
gccgtatgga
gtatggacaa
gacaacgact

atgccgtatggacaacgact
atgcc
tgccg
gccgt
ccgta
cgtat
gtatg
tatgg
atgga
tggac
ggaca
gacaa
acaac
caacg
aacga
acgac
cgact

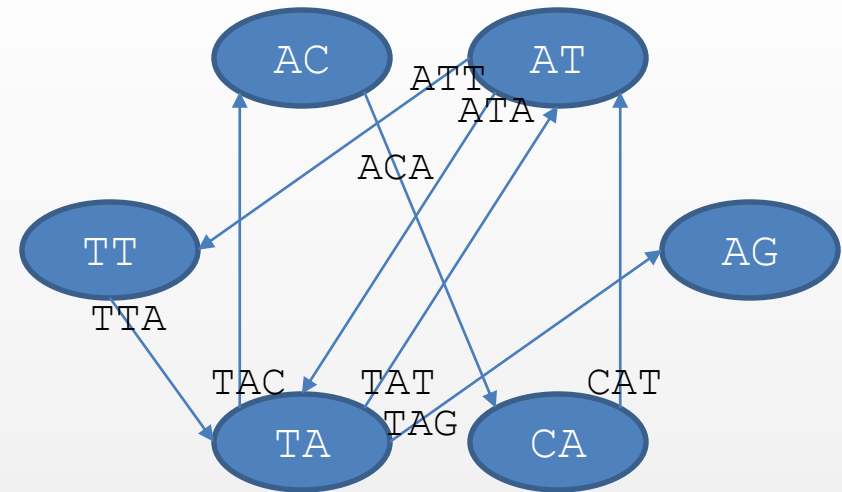
Usporedba dviju formulacija

Stvarna DNA sekvenca (nepoznata na početku): TATACATTAG

Formulacija Hamiltonovog puta



Formulacija Eulerove staze



- Hierholzerov algoritam se izvršava u realnom vremenu

Zbog čega je velika razlika?

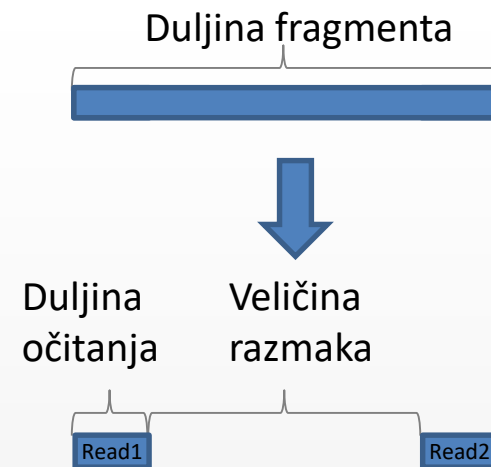
- Zbog čega je problem pronalaska Eulerove staze (puta) mnogo lakši od općeg problema pronalaska Hamiltonovog puta?
 - Izravni nužni i dovoljni uvjeti za postojanje Eulerovog puta
 - Uvjeti su jednostavni: Mogu biti efikasno provjereni
 - za Eulerov put, rješenja podproblema mogu uvijek pridonijeti rješenju originalnoga problema

OLC nasuprot de Bruijn

- OLC
 - Tri faze – lakše ga je modularno implementirati
 - Preklapanja mogu varirati u duljini
 - Pogodniji za dulja očitavanja
- De Bruijn
 - Vrlo brzo računa parove preklapanja
 - Puno manje je zahtjevno pronaći Eulovu stazu od Hamiltonovog puta
 - Vrlo osjetljiv na ponavljanja zbog kratkih k-torki
 - Vrlo osjetljiv na pogreške u očitanjima
 - Pogodniji za kraća očitavanja

Sekvenciranje parova krajeva

- Sekvenciranje oba kraja fragmenta
- Dva očitavanja se nazivaju upareni krajevi
- Veličina razmaka nije fiksna zbog slučajne fragmentacije
 - jedno očitavanje je vjerojatno unutar određene udaljenosti od drugoga
 - ako je pozicija jednoga očitavanja nejasna, može se koristiti drugo kao pomoć
- U praksi je teže zbog neprecizne veličine razmaka



ATA . . . TTA
TAC . . . TAG
TAG . . . CTT

Osnovni koncepti sastavljanja

- Ključni pojmovi:
 - Kontig (*eng. Contig*): Djelomično sastavljen niz od nekoliko očitavanja
 - Skafold (*engl. Scaffold*): niz kontiga za koje su određeni međusobna pozicija i udaljenost
- Obično krajnji rezultat ne sadrži jedan niz, nego nekoliko skafolda



Deskriptivne statistike sastavljanja

- Duljina najduljeg kontiga
- Prosječna duljina kontiga
- Ukupna duljina kontiga
- N50: Takva duljina kontiga da on i dulji kontinzi čine 50% ili više ukupne duljine svih kontiga
 - Ako su duljine (u proizvoljnom dijelu) 10, 8, 6, 5, 3, 3, 2, 1, 1, 1, onda N50 value je 6, zbog $(10+8+6) = 24$, što je više od 50% od sume $(10+8+6+5+3+3+2+1+1+1) = 40$