# **Bioinformatika**
# Heurističke metode za pretraživanje baza sekvenci

Prof. G. Pavlović-Lažetić,
Matematički fakultet,
Beogradski univerzitet,
šk.2012/2013. g.
gordana@matf.bg.ac.rs
http://alas.matf.bg.ac.rs/~bioinformatika

# Motivacija za heurističko poravnanje

- Ranije izloženi algoritmi poravnanja imaju polinomijalnu složenost, ali su neprimenljivi za velike baze. O(mn) suviše sporo za velike baze podataka sa intenzivnim upitima
- Heuristički algoritmi brži, mada ne garantuju optimalna rešenja. Heurističke metode proizvode brze aproksimacije dinamičkog programiranja
- Ideja je da se pretražuje što je moguće manji deo ćelija u matrici dinamičkog programiranja a da se pri tome ipak obrade sva poravnanja visokog skora
- Suštinski dva programska sistema za pretraživanja baza podataka koji daju prihvatljiva rešenja
  - FASTA [Pearson & Lipman, 1988]
  - BLAST [Altschul *et al*., 1990; Altschul et al., *Nucleic Acids Research* 1997]

# Motivacija za heurističko poravnanje

- Posmatrajmo zadatak pretraživanja SWISS-PROT baze za neku upitnu sekvencu
  - Neka je upitna sekvenca duga 362 aa
  - SWISS-PROT verzija 38 sadrži 29,085,265 aa
  - Nalaženje lokalnog poravnanja dinamičkim programiranjem uzelo bi $O(10^{10})$ matričnih operacija

- Mnogi serveri obrađuju hiljade takvih upita dnevno (NCBI > 100 000)

# Pregled BLAST-a

- **BLAST: B**asic **L**ocal **A**lignment **S**earch **T**ool
- BLAST heuristički traži lokalna poravnanja visokog skora
- Obično se koristi za traženje upitne sekvence u bazi sekvenci
- DNK ili proteinske sekvence
- Osnovna nagodba: osetljivost (odziv, engl. *sensitivity*) vs. brzina

- $$\text{sensitivity} = \frac{\#\ \textbf{\textit{otkrivenih}}\ \text{značajnih poklapanja}}{\#\ \text{značajnih poklapanja u bazi podataka}}$$

# Ideja BLAST algoritma

- Pretpostavka da pravi “pogoci” (engl. *matches*) pri poravnanju vrlo verovatno sadrže unutar sebe kratke sekvence visokog skora.

- Te kratke sekvence mogu da se koriste kao “seme” (engl. *seed*) od kojih može da se pođe u traženju dobrog dužeg poravnanja

- Kada su semene sekvence kratke, može da se pre-procesira upitna sekvenca i da se izgradi tabela svih mogućih semena sa njihovim početnim pozicijama

# Pregled BLAST algoritma

- Neka je data upitna sekvenca *q*, dužina reči *w* (reči mogu da se izdvoje iz upitne sekvence), prag skora reči *T*, i prag skora segmenta *S*
  - Sastaviti listu "upitnih reči" (dužine *w*) sa skorom većim ili jednakim *T* kada se uporede sa rečima iz *q* (na osnovu neke matrice skora)
  - Pretražiti bazu podataka na poklapanja sa rečima iz liste
  - Proširiti sva poklapanja u traženju poravnanja visokog skora (engl. *hit extension*), kao poravnanja bez praznina na obe strane
- Vratiti: poravnanja sa skorom većim ili jednakim *S*

# Određivanje upitnih reči

- Neka su dati
  - Upitna sekvenca *q* = QLNFSAGW
  - Dužina reči *w*=2 (za proteine se obično uzima *w*=3, za DNK - 11)
  - Prag skora reči *T*=9

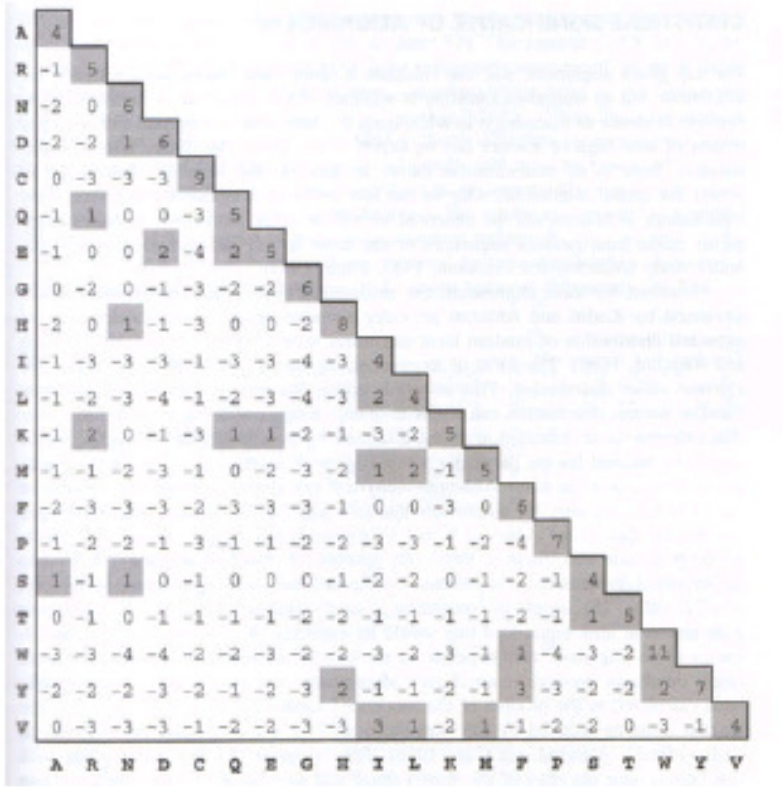
- Korak 1: odrediti sve reči dužine *w* u upitnoj sekvenci:

  QL LN NF FS SA AG GW

- Korak 2: odrediti sve upitne reči, tj. reči dužine *w* koje daju skor veći ili jednak *T* kada se uporede sa nekom rečju u upitnoj sekvenci (u odnosu na neku matricu skora)

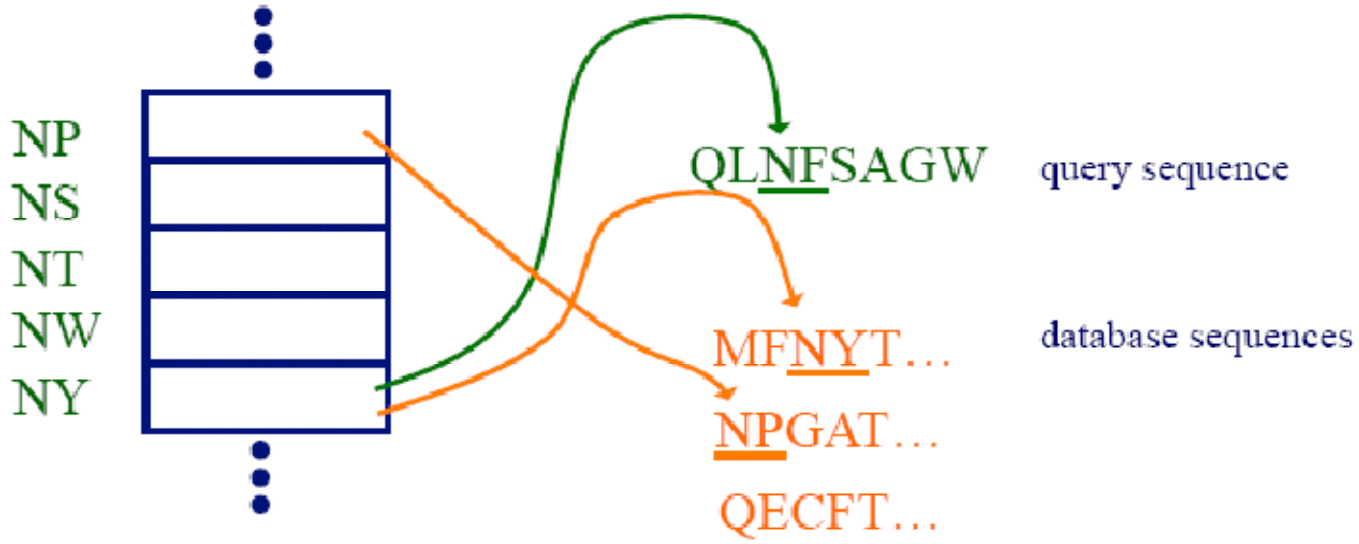
# Određivanje upitnih reči

| words from sequence | query words w/ T=9 |
|---|---|
| QL | QL=9 |
| LN | LN=10 |
| NF | NF=12, NY=9 |
| … | |
| SA | none |
| … | |



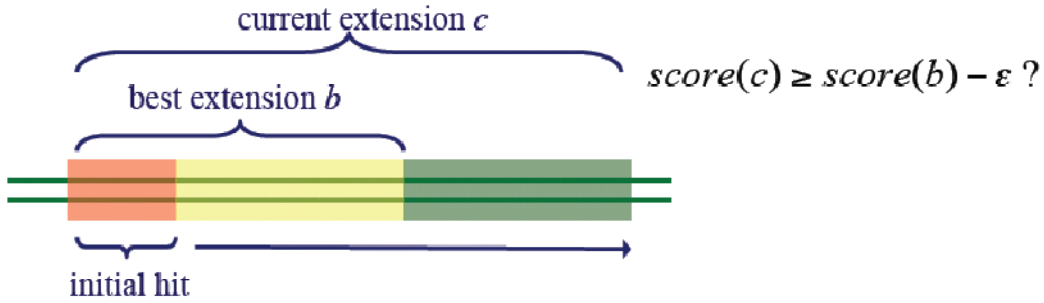
# Skeniranje  baze podataka

- Tražiti u bazi podataka sva pojavljivanja upitnih reči
- Pristup:
  - Indeksirati sekvence baze podataka kroz tabelu reči (prethodno izračunatu)
  - Indeksirati upitne reči kroz tabelu (u vreme izvršavanja)



# Proširenje pogotka

- Proširuje pogodak u oba pravca bez praznina
- Prekida proširenje u jednom pravcu kada skor padne za određeni iznos ispod najboljeg skora za kraća proširenja




- Vraća parove segmenata sa skorom ≥ *S*

# Osetljivost vs. vreme

- Glavni parametar koji kontroliše odnos osetljivost vs. vreme izvršavanja jeste prag *T* kao kriterijum šta jeste upitna reč
  - Malo *T*: veća osetljivost, više pogodaka za proširenje
  - Veliko *T*: niža osetljivost, manje pogodaka za proširenje

# BLAST

- Heuristička tehnika: može da propusti neka dobra poravnanja
- Brza: empirijski, 10-15 puta brža od Smith-Waterman
- Veliki uticaj
  - NCBI BLAST server obradi više od 100 000 upita dnevno
  - Najkorišćeniji bioinformatički program na svetu

# FASTA [Pearson & Lipman, 1988]

- Program FASTA (FAST All) je naslednik programa FASTP (FAST Protein)
- Neprekidno je u razvoju
- Osnovni princip: obrazac – upit nad bazom – redom se poredi sa svim sekvencama u bazi podataka
- Koristi višeetapni pristup u pronalaženju lokalnih poravnanja visokog skora
  - polazeći od tačnih (*exact*) sravnjivanja kratkih reči
  - kroz proširenja maksimalnog skora bez praznina
  - da bi se na kraju identifikovala poravnanja sa prazninama

# FASTA

- Poređenje obrasca sa jednom sekvencom iz baze odvija se u četiri koraka:
  - *1. korak:* bira se parametar *k* i traže se sva tačna poklapanja dužine *k* između obrasca i sekvence iz baze – "vruća tačka" (engl. "hot spot")
    - jednoznačno se opisuje parom (start. poz. u obrascu, start. poz. u sekvenci)
    - tipična vrednost za k je *k*=6 za DNK sekvence i *k*=2 za proteinske sekvence
    - za ova kratka poređenja dovoljno je imati preprocesiranu tabelu ("lookup" tabelu) koja sadrži ove parove za sve moguće "vruće tačke".
    - Na primer (k=1),

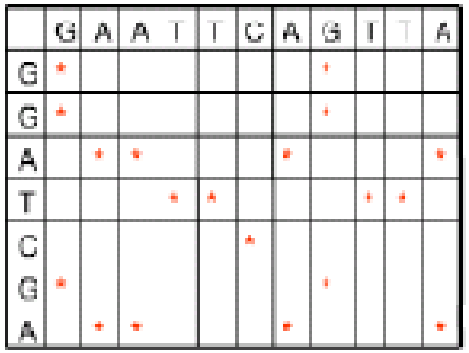



**Figure 4**: shows exact matches between query and database [8].

    - Zatim se traže dijagonale sa puno uzajamnog poklapanja reči;
- Brza operacija, može da se izvede sortiranjem poklapanja po razlici indeksa (i-j).

# FASTA

- *2. korak*: pokušava se sa grupisanjem (klasterovanjem) vrućih tačaka
  - Razmatra se matrica M nalik matrici sličnosti u algoritmu DP za globalno poravnanje
  - Vrste matrice odgovaraju simbolima iz obrasca p, kolone – simbolima iz sekvence t baze podataka
  - Element matrice na poziciji (i,j) je 1 ako je $p_i=t_j$ i 0 inače
  - Svaka vruća tačka odgovara segmentu dijagonale koji počinje na poziciji M(i; j)
  - Nije neophodno da se konstruiše cela matrica M, samo dijagonalni segmenti
  - Za svaki dijagonalni segment se ocenjuje ***skor***
  - Svaka vruća tačka doprinosi pozitivno skoru, svaka praznina između dve susedne vruće tačke doprinosi negativno skoru
  - Bira se 10 dijagonalnih segmenata sa najboljim skorovima
  - Svaki određuje jedno poravnanje podniske obrasca i podniske sekvence
  - Može da sadrži podudaranja i nepodudaranja ali ne i insercije / delecije
  - Najbolje parcijalno poravnanje je ulaz u korak 4

# FASTA

- *3. korak*:
  - Svako od 10 parcijalnih poravnanja se razmatra ako sličnost prelazi neki prag
  - Pokušava se sa spajanjem parcijalnih poravnanja da bi se dobilo duže poravnanje sa boljim skorom
  - Jedno rešenje FASTA algoritma: problem teorije grafova
    - Parcijalna poravnanja su temena usmerenog grafa obeležena skorovima
    - Neka su u, v dva parcijalna poravnanja sa krajem u (i,j), odnosno početkom u (i',j'), redom
    - Graf sadrži usmerenu granu (u,v) ako i samo ako je i<i' i j<j'
    - Tada dva parcijalna poravnanja mogu da se povežu tako da daju duže poravnanje
    - Grana ima negativnu težinu u zavisnosti od udaljenja pozicija (i,j) i (i', j')
    - Optimalno poravnanje koje se sastoji od parcijalnih poravnanja računa se kao putanja sa najboljim skorom u grafu

# FASTA

- U *4. koraku*, algoritam izračunava alternativno rešenje bazirano na optimalnom parcijalnom poravnanju iz koraka 2.

- Koristi se DP algoritam za lokalno poravnanje (Smith & Waterman)

- Računanje matrice sličnosti ograničeno je na traku oko dijagonale koja sadrži optimalno parcijalno poravnanje

# Heurističke metode za pretraživanje baza podataka: FASTA - rezime

- Prethodna četiri koraka primenjuju se na svaku sekvencu iz baze podataka
- Zatim FASTA algoritam procenjuje značajnost dobijenih rešenja primenom statističkih metoda
- Princip FASTA algoritma na primeru obrasca TACCGA i sekvence ACTGAC

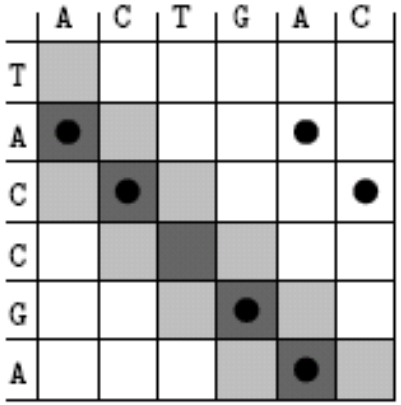



**Fig. 5.5.** The principle of the FASTA heuristic on the strings `TACCGA` and `ACTGAC`. The hot spots for $k = 2$ are shown as pairs of black dots in the matrix; a diagonal run is shaded dark. In this example, the optimal partial alignment coincides with the diagonal run. The lightly shaded area shows a band of width 3 around the partial alignment, within which an optimal local alignment is searched for in step 4

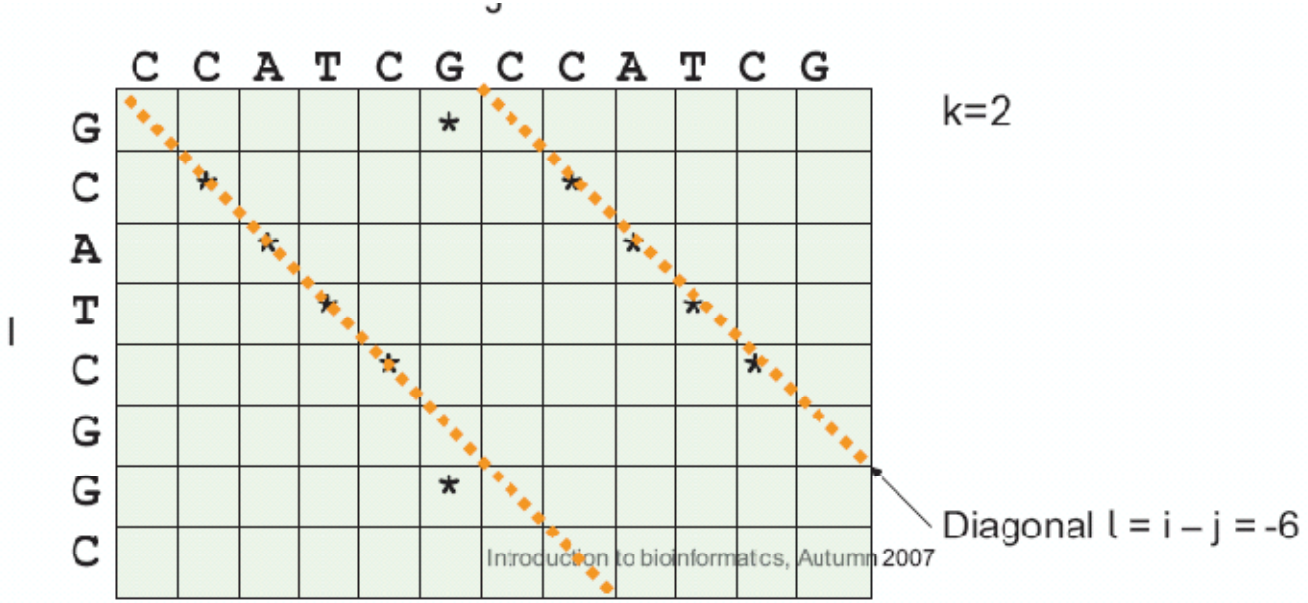
# Nagodbe

- Postoji nagodba između brzine i osetljivosti u izboru parametra *k*: veća vrednost za *k* obezbeđuje brže izvršavanje ali se povećavaju i izgledi da se propuste zaista značajna poklapanja.

- Da bi se postigla osetljivost bliska onoj za puno lokalno dinamičko programiranje proteinskih sekvenci, potrebno je uzeti $k = 1$.

# FASTA primer: izračunavanje dijagonalnih suma (korak 1)

- Želimo da nađemo dijagonale tačkaste matrice sa visokim skorom
- Dijagonale indeksiramo razlikom l=i-j



# Izračunavanje dijagonalnih suma (nast.)

- Na primer, izračunajmo dijagonalne sume za I=GCATCGGC i J=CCATCGCCATCG, k=2
  1. Konstruisati listu k-reči $L_W(J)$
  2. Računaju se dijagonalne sume $S_l$, smeštaju u tabelu i indeksiraju razlikom i-j i inicijalizuju sa 0

| l | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_1$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

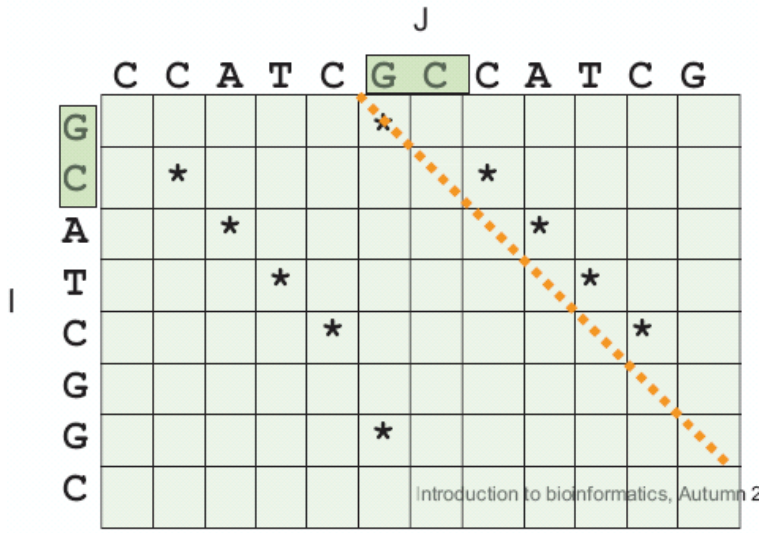
# Izračunavanje dijagonalnih suma (nast.)

3. Ići kroz k-reči od I, tražiti poklapanja u $L_W(J)$ i ažurirati dijagonalne sume

   Za prvu 2-reč u I, GC, $L_{GC}(J)=\{6\}$

   Sada možemo da ažuriramo sumu dijagonale

   I = i-j = 1-6 = -5, tako da postane $S_{-5} = S_{-5}+1 = 0+1 = 1$



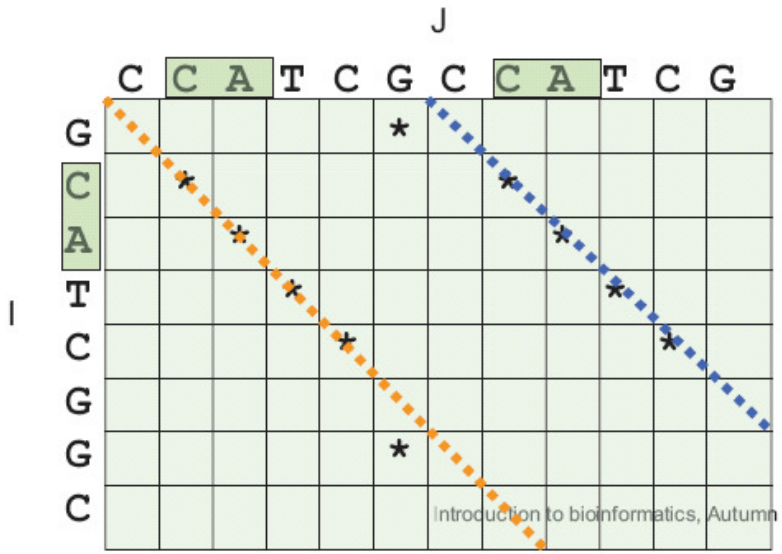
# Izračunavanje dijagonalnih suma (nast.)

3. Ići kroz k-reči od I, tražiti poklapanja u $L_W(J)$ i ažurirati dijagonalne sume

Sledeća 2-reč u I je CA, za koju je $L_{CA}(J)=\{2,8\}$

Ažuriraju se dve dijagonalne sume: $l=i-j=2-2=0$, $S_0 = S_0+1 = 0+1 = 1$, i

$l = i-j = 2-8 = -6$, $S_{-6} = S_{-6}+1 = 0+1 = 1$



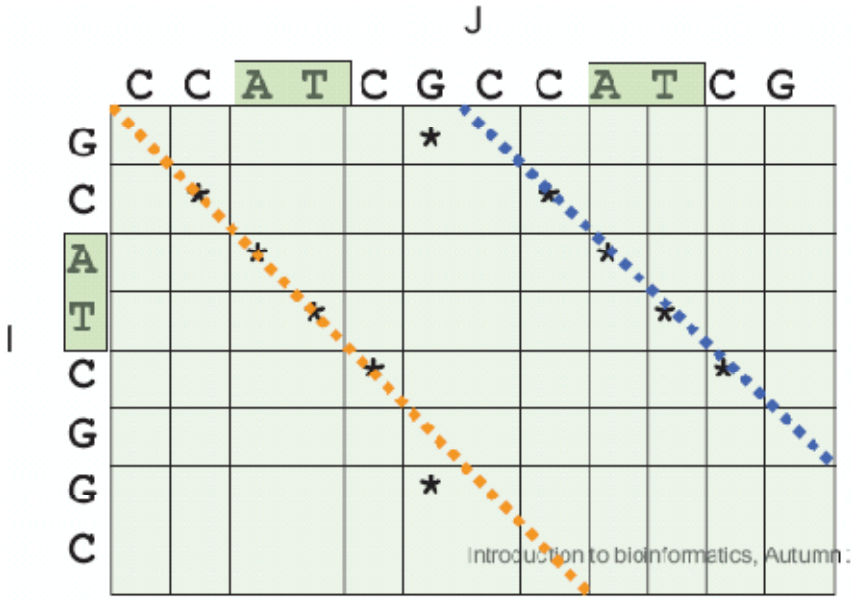
# Izračunavanje dijagonalnih suma (nast.)

3. Ići kroz k-reči od I, tražiti poklapanja u $L_W(J)$ i ažurirati dijagonalne sume

Sledeća 2-reč u I je AT, za koju je $L_{AT}(J)=\{3,9\}$

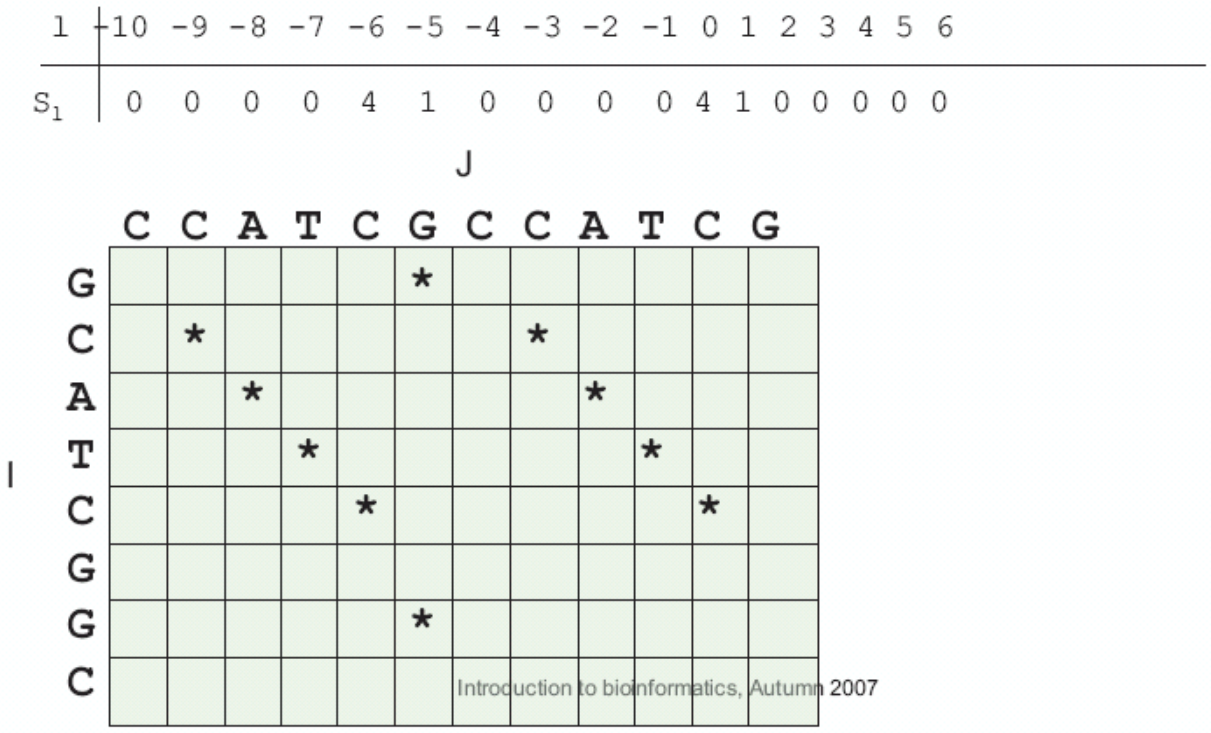
Ažuriraju se dve dijagonalne sume: $l=i-j=3-3=0$, $S_0 = S_0+1 = 1+1 = 2$, i

$l = i-j = 3-9 = -6$, $S_{-6} = S_{-6}+1 = 1+1 = 2$



# Izračunavanje dijagonalnih suma (nast.)

Posle prolaza kroz k-reči iz I, rezultat je



# Algoritam za izračunavanje dijagonalne sume skorova

$S_l = 0$ za sve $1-m \leq l \leq n-1$;

Izračunati $L_W(J)$ za sve reči W;

```
for (i=1; i<=n-k+1; i++)
{
    W=l_i l_{i+1} ... l_{i+k-1};
    for (j∈ L_W(J))
    {
        l=i-j;
        S_l=S_l+1;
    }
}
```

# Spajanje dijagonala

- Dve dijagonale mogu da se spoje razmakom, ako rezultujuće poravnanje ima viši skor
- Koriste se različite “kazne” otvaranja razmaka i proširenja razmaka
- Naći kombinaciju dijagonala sa najboljim skorom