

# Sekų panašumų paieška

Dalinai parengta pagal  
(nukopijuota neatsiklausus)  
Dr. Joanne Fox  
[joanne@bioinformatics.ubc.ca](mailto:joanne@bioinformatics.ubc.ca)



# Panašumų paieška

- Kas yra sekų panašumas
- Kodėl įdomu ir svarbu surasti panašumus
- Kaip atlikti efektyviai panašumų paiešką



# Panašumų paieškos svarba

- Genomų sekoskaita būtų neįmanoma be panašių sekų paieškos
- Genomų palyginimas
  - Genų palyginimas
  - Baltymų palyginimas
  - Genomo fizinės struktūros palyginimas
- Genų funkcijos atspėjimas



## ➤ Prasmė:

- Seka pati iš savęs nėra informatyvi. Tik palyginamojoje analizėje galime iškelti hipotezes apie “giminaičius” ir funkciją



# Panašumų paieškos panaudojimas

- Sekų duomenų bazėse sukaupti milžiniški duomenų kiekiai, terabaitų eilės
- Todėl galima rasti daug naudingos informacijos
- Bet efektyvi paieška labai sudėtinga



# Panašumų paieškos panaudojimas

- Ar ebola bus perduodama oro lašiniu būdu ir kada?
- Kaip detektuoti variabilius virusus ir kurti jiems vakcinas(HIV, Papiloma,Gripas).
- Teismo ekspertizė. Kaip atpažinti ar tai kraujas?Kaip iš kraujo sukurti veido profilį?...nustatyti nusikaltėlio odos spalvą?



# Sekų panašumai

- Sekų panašumai paremti sekų palyginiu (angl. Alignment)

```
GATGCCATAGAGCTGTAGTCGTACCCT  <-  
->CTAGAGAGCGTAGTCAGAGTGTCTTTGAGTTCC  
  
GATGCCATAGAGCTGTAGTCGTACCCT  <-  
-> CTAGAGAGC-GTAGTCAGAGTGTCTTTGAGTTCC
```

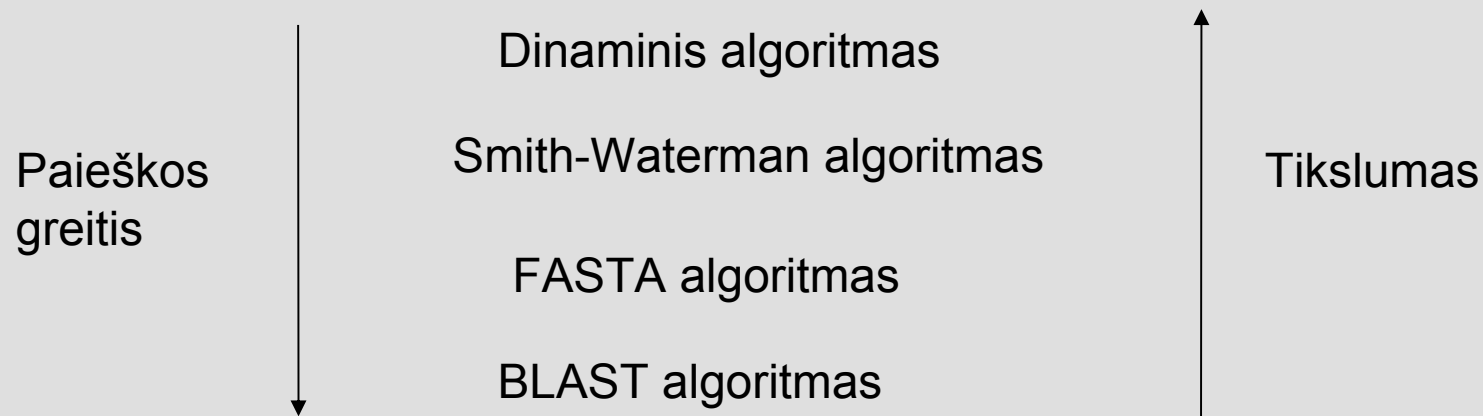
# Panašumų paieškos metodai

- Vizualinis taškinė “matrica” (dotplot)
- Dinaminis algoritmas
  - Needleman S.B. and Wunsch C.D. 1970. J. Mol. Biol. 48: 443-453
- Smith-Waterman algoritmas
  - Smith T.F. and Waterman M.S. 1981. J. Mol. Biol. 147: 195-197
- FASTA algoritmas
- BLAST algoritmas



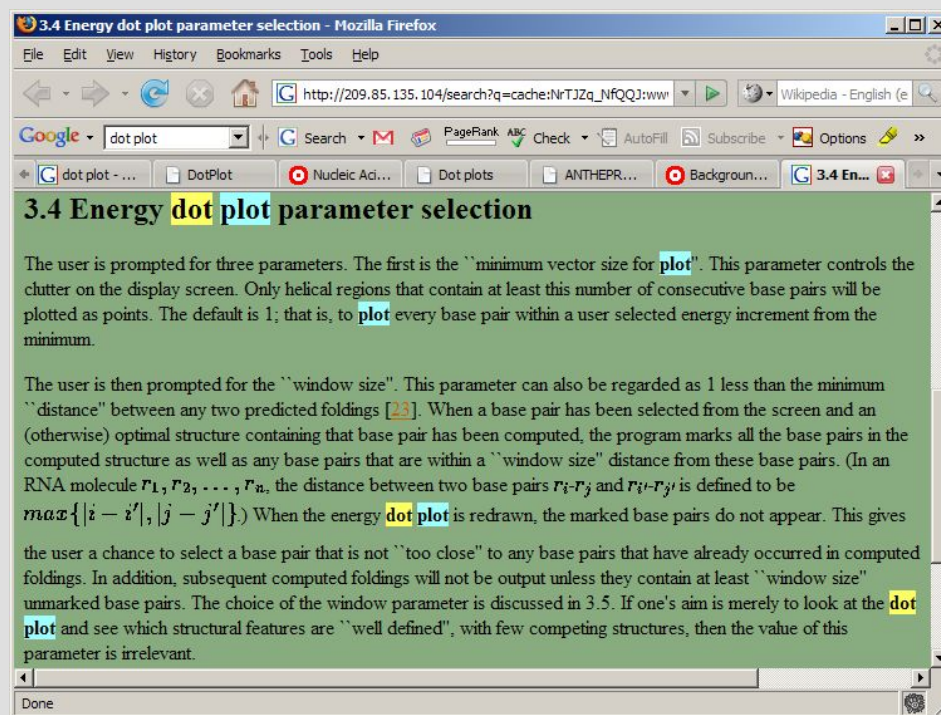


# Panašumų paieškos metodai



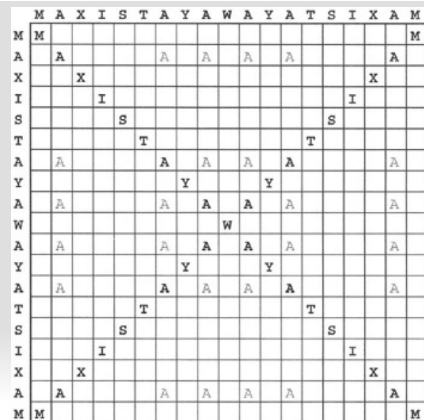
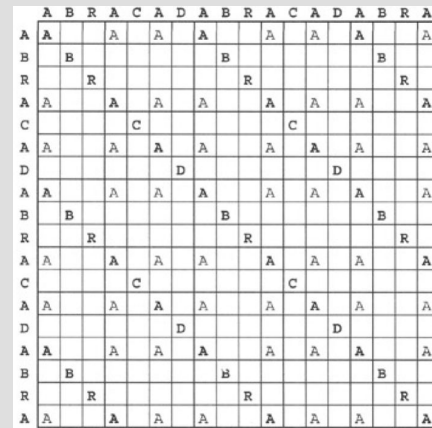
# Panašumų paieška

- Sakoma kad 90 procentų baltųjų turi vizualinę mąstymą
- Net ir paprasti paieškos varikliai turi rezultatų vizualizaciją



# Panašumų paieška

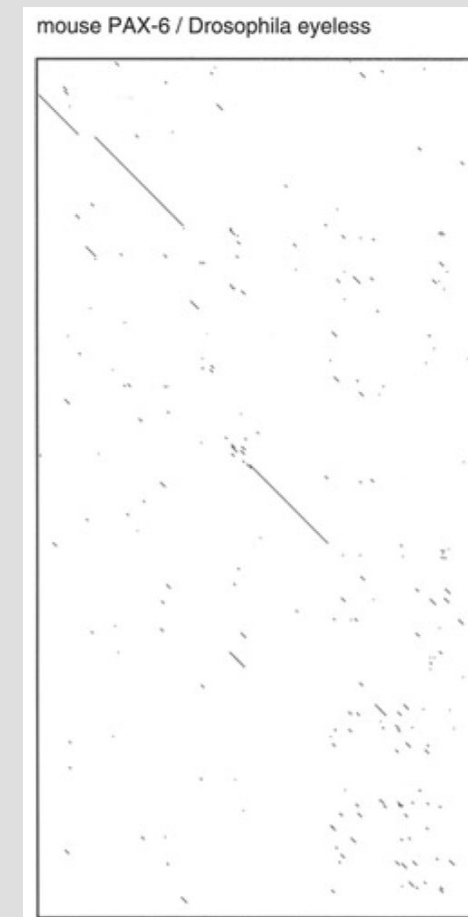
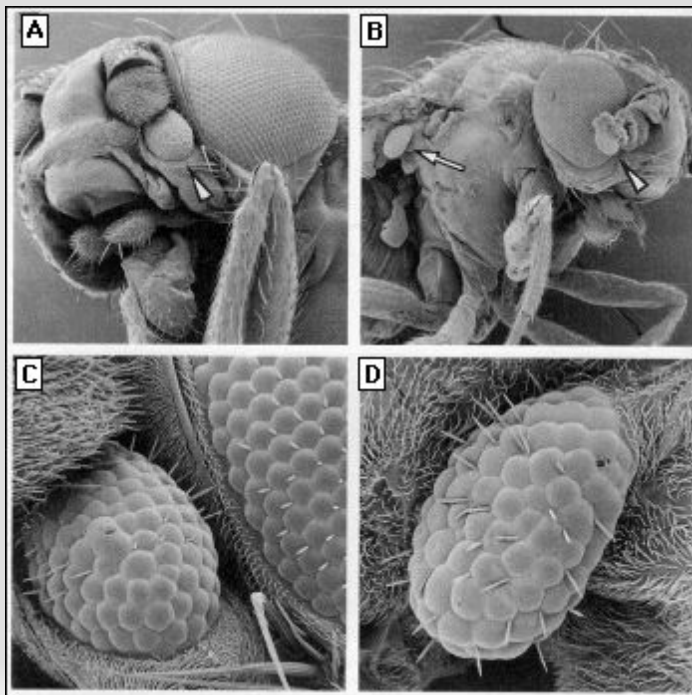
- Pavaizduojama taškinė “matrica” (dotplot)



# Panašumų paieška

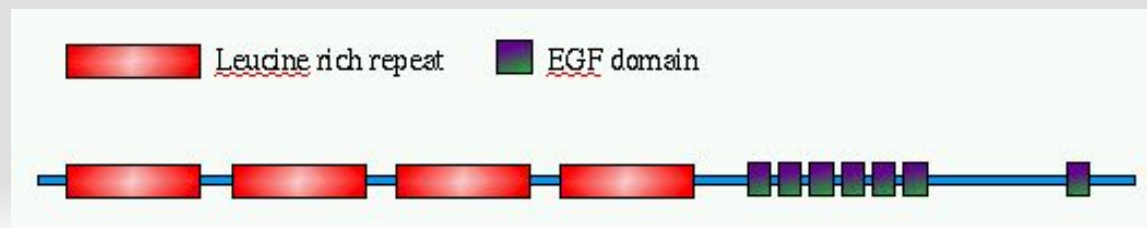
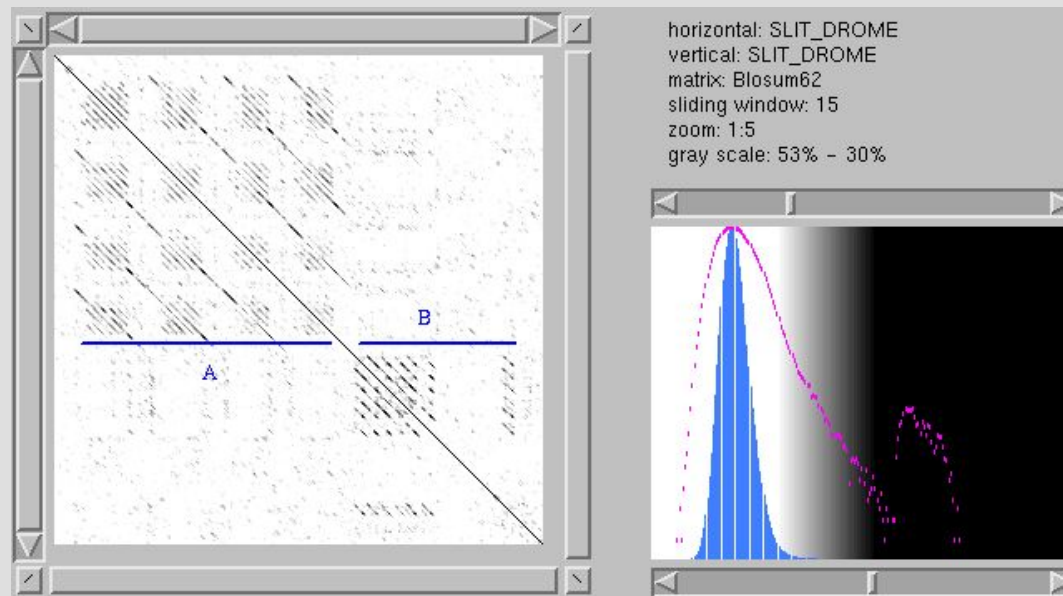
PAX-6 genas susijęs su rainelės neišsivystymu

Drosophila eyless genas susijęs su akių nebuvimu



# Taškinės “matricos”

<http://www.isrec.isb-sib.ch/java/dotlet/Dotlet.html>



# Svarbūs panašumų paieškos terminai su skirtingomis reikšmėmis

## ➤ **Panašumas, Similarity**

- Nukleotidų ar baltymų sekos sąryšio įvertis. Šis įvertis gali būti išreikštas identiškumo arba konservacijos procentu.

## ➤ **Identiškumas, Identity**

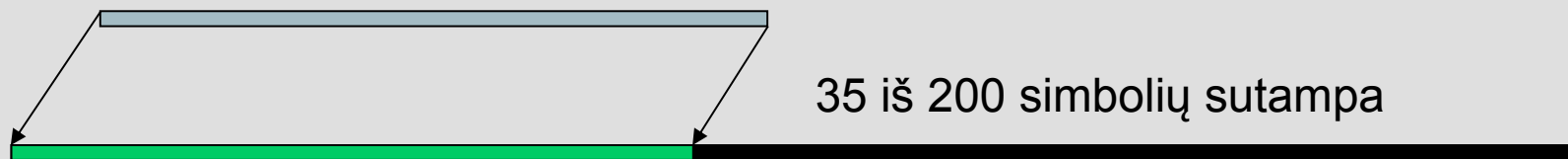
- Dviejų sekų invariantiškumo lygis ☺

## ➤ **Homologija, Homology**

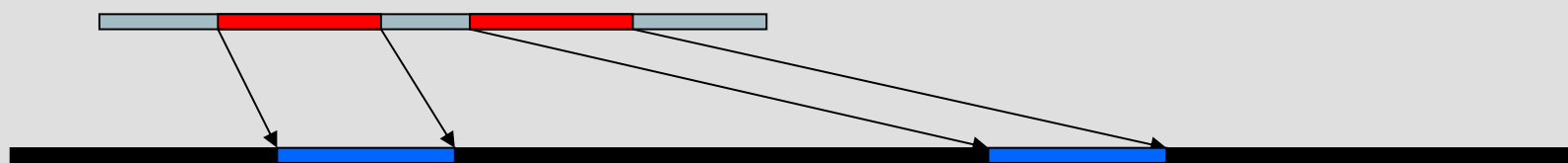
- Panašumas susietas su bendra kilme.
  - Šiai sąvokai netinka procentas

# Sekų panašumai

➤ O kas yra panašumas?

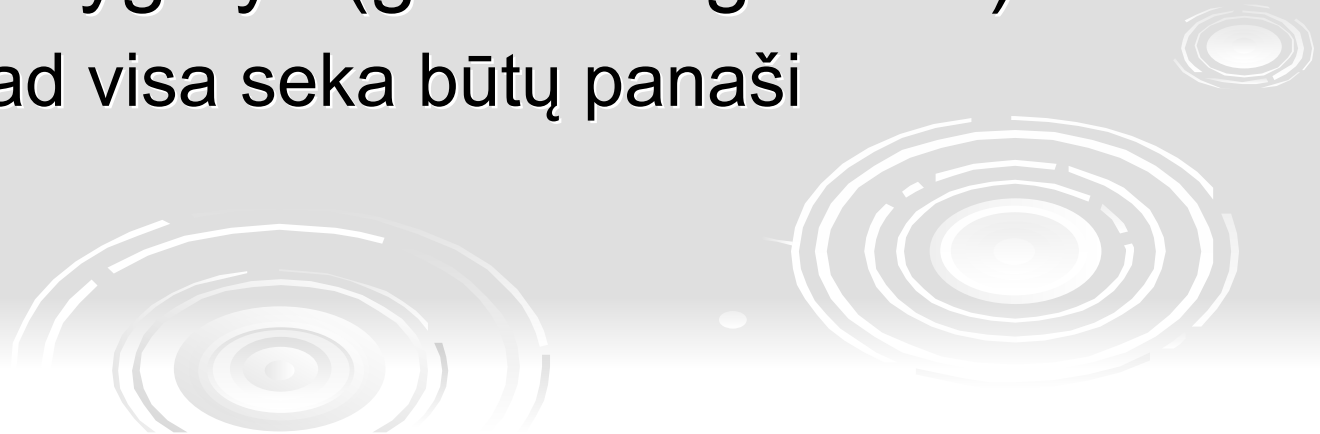


ar



# Sekų panašumai

- Todėl pagal rezultatą išskiriami:
- Lokalus palyginys (local alignment)
  - Svarbu kad atskiri segmentai būtų kuo labiau panašūs vienas į kitą
- Globalus palyginys (global alignment)
  - Svarbu kad visa seka būtų panaši





# Algoritmų taikymo sritis

Taikomi biologinių duomenų paieškai:

Apribota abėcėlė:

Baltymų sekos sudarytos iš 20 amino rūgščių

Nukleotidų sekos turi 4 (5) nukleotidus



# Sekos paieškos rezultatas gali būti:

## Tikslus sutapimas:

CGCATA  
CGTA

CGCATA  
CG--TA

## Netikslus sutapimas:

GARIIPPRST  
GARIISGGSFPGKWT

GARI-----PPRST  
GARIISGGSFPGKWT

Baltymų palyginimuose daug netikslių sutapimų dėl didelės abėcėlės (Kokios?)

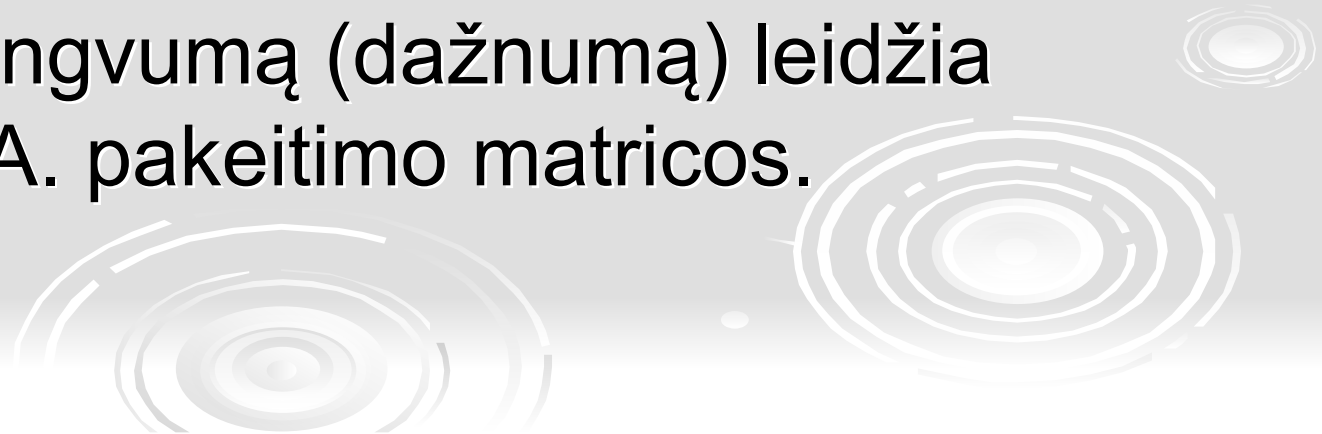
Reikėjo apsispresti kokias "raides" laikyti geriau sutampančiomis, kokias blogiau - įvestos pakeitimo matricos.

# Amino rūgčių pakeitimo matricos

Amino rūgštys pasižymi tam tikromis, savitomis fizikinėmis ir cheminėmis savybėmis.

Ortologiniuose baltymuose amino rūgštys keičiamos viena į kitą skirtingais dažnumais.

Pakeitimo lengvumą (dažnumą) leidžia įvertinti A.A. pakeitimo matricos.



# Kaip skaičiuojamas įvertis, score (S)?

- Palyginio kokybė parodoma įverčiu.
- **Įverčių matricos, Scoring matrices** naudojamos palyginio įverčio skaičiavime.
- **Palyginio įvertis bus visų padėčių įverčių suma**



# Amino rūgščių pakeitimo matricos pavyzdys

|   | A  | R  | N  | D  | C  | Q  | E  | G  | H  | I  | L  | K  | M  | F  | P  | S  | T  | W  | Y  | V  |
|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| A | 13 | 6  | 9  | 9  | 5  | 8  | 9  | 12 | 6  | 8  | 6  | 7  | 7  | 4  | 11 | 11 | 11 | 2  | 4  | 9  |
| R | 3  | 17 | 4  | 3  | 2  | 5  | 3  | 2  | 6  | 3  | 2  | 9  | 4  | 1  | 4  | 4  | 3  | 7  | 2  | 2  |
| N | 4  | 4  | 6  | 7  | 2  | 5  | 6  | 4  | 6  | 3  | 2  | 5  | 3  | 2  | 4  | 5  | 4  | 2  | 3  | 3  |
| D | 5  | 4  | 8  | 11 | 1  | 7  | 10 | 5  | 6  | 3  | 2  | 5  | 3  | 1  | 4  | 5  | 5  | 1  | 2  | 3  |
| C | 2  | 1  | 1  | 1  | 52 | 1  | 1  | 2  | 2  | 2  | 1  | 1  | 1  | 1  | 2  | 3  | 2  | 1  | 4  | 2  |
| Q | 3  | 5  | 5  | 6  | 1  | 10 | 7  | 3  | 7  | 2  | 3  | 5  | 3  | 1  | 4  | 3  | 3  | 1  | 2  | 3  |
| E | 5  | 4  | 7  | 11 | 1  | 9  | 12 | 5  | 6  | 3  | 2  | 5  | 3  | 1  | 4  | 5  | 5  | 1  | 2  | 3  |
| G | 12 | 5  | 10 | 10 | 4  | 7  | 9  | 27 | 5  | 5  | 4  | 6  | 5  | 3  | 8  | 11 | 9  | 2  | 3  | 7  |
| H | 2  | 5  | 5  | 4  | 2  | 7  | 4  | 2  | 15 | 2  | 2  | 3  | 2  | 2  | 3  | 3  | 2  | 2  | 3  | 2  |
| I | 3  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 2  | 10 | 6  | 2  | 6  | 5  | 2  | 3  | 4  | 1  | 3  | 9  |
| L | 6  | 4  | 4  | 3  | 2  | 6  | 4  | 3  | 5  | 15 | 34 | 4  | 20 | 13 | 5  | 4  | 6  | 6  | 7  | 13 |
| K | 6  | 18 | 10 | 8  | 2  | 10 | 8  | 5  | 8  | 5  | 4  | 24 | 9  | 2  | 6  | 8  | 8  | 4  | 3  | 5  |
| M | 1  | 1  | 1  | 1  | 0  | 1  | 1  | 1  | 1  | 2  | 3  | 2  | 6  | 2  | 1  | 1  | 1  | 1  | 1  | 2  |
| F | 2  | 1  | 2  | 1  | 1  | 1  | 1  | 1  | 3  | 5  | 6  | 1  | 4  | 32 | 1  | 2  | 2  | 4  | 20 | 3  |
| P | 7  | 5  | 5  | 4  | 3  | 5  | 4  | 5  | 5  | 3  | 3  | 4  | 3  | 2  | 20 | 6  | 5  | 1  | 2  | 4  |
| S | 9  | 6  | 8  | 7  | 7  | 6  | 7  | 9  | 6  | 5  | 4  | 7  | 5  | 3  | 9  | 10 | 9  | 4  | 4  | 6  |
| T | 8  | 5  | 6  | 6  | 4  | 5  | 5  | 6  | 4  | 6  | 4  | 6  | 5  | 3  | 6  | 8  | 11 | 2  | 3  | 6  |
| W | 0  | 2  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 55 | 1  | 0  |
| Y | 1  | 1  | 2  | 1  | 3  | 1  | 1  | 1  | 3  | 2  | 2  | 1  | 2  | 15 | 1  | 2  | 2  | 3  | 31 | 2  |
| V | 7  | 4  | 4  | 4  | 4  | 4  | 4  | 4  | 5  | 4  | 15 | 10 | 4  | 10 | 5  | 5  | 5  | 72 | 4  | 17 |

# Kas yra įverčių matrica, scoring matrix?

- Amino rūgštims naudojamos pakeitimo matricos.
- DNR naudojamos paprastos matricos
  - Padėtis gauna +1, jei sutampa ir -2 jei ne.

|   | A  | C  | D  | E  | F  | G  | H  |  |
|---|----|----|----|----|----|----|----|--|
| A | 4  | 0  | -2 | -1 | -2 | 0  | -2 |  |
| C | 0  | 9  | -3 | -4 | -2 | -3 | -3 |  |
| D | -2 | -3 | 6  | 2  | -3 | -1 | -1 |  |
| E | -1 | -4 | 2  | 5  | -3 | -2 | 0  |  |
| F | -2 | -2 | -3 | -3 | 6  | -3 | -1 |  |
| G | 0  | -3 | -1 | -2 | -3 |    |    |  |
| H | -2 | -3 | -1 | 0  |    |    |    |  |

BLOSUM 62

# BLOSUM vs. PAM

**BLOSUM 45**

**BLOSUM 62**

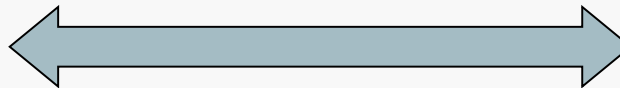
**BLOSUM 90**

**PAM 250**

**PAM 160**

**PAM 100**

*Labiau skirtingi*



*Mažiau skirtingi*

- BLOSUM 62 yra naudojama BLAST 2.0 (dažniausiai naudojamas algoritmas). Ji labiausiai tinka vidutiniškai nutolusių baltymų paieškai. Naudojant kitas matricas galima gauti geresnius rezultatus nutolusiems baltymams.

# Dinaminis programavimas

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.  
RAND corporation (armed forces)

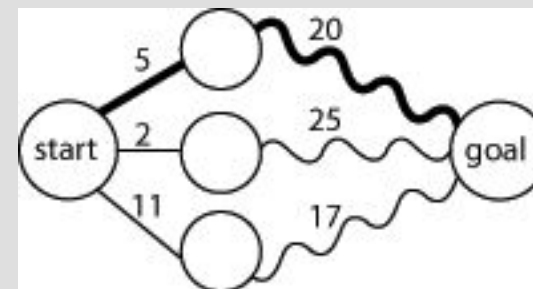
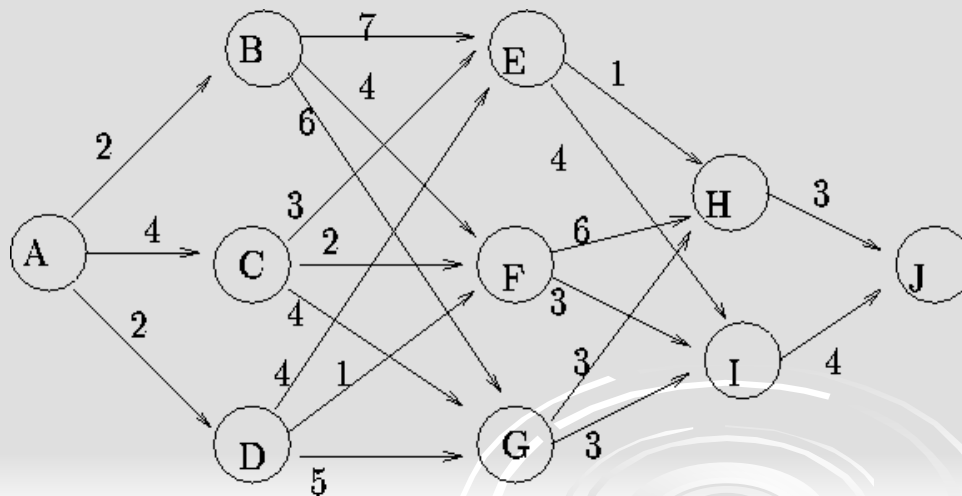




# Dinaminis programavimas

DP yra vienas iš algoritmų, taikomų optimizavimo problemoms spręsti

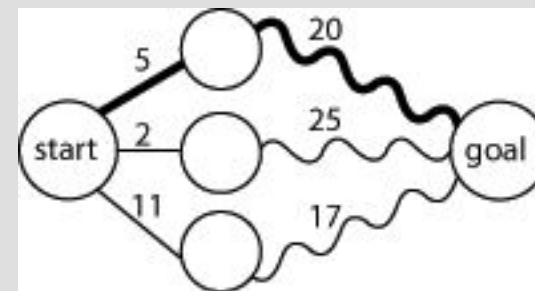
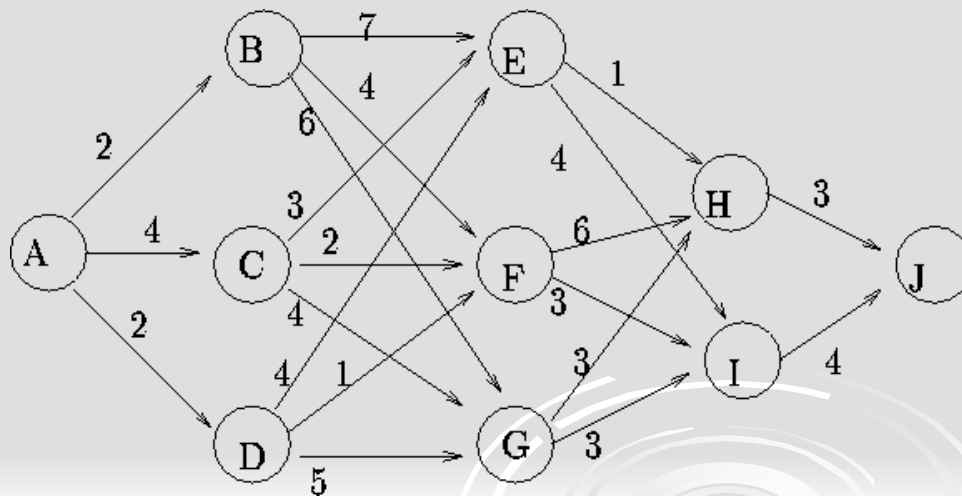
DP veikia skaidant didelę užduotį į mažesnes sub-užduotis.



# Dinaminis programavimas

Kiekviena subužduotis vykdoma tik vieną kartą, o jos rezultatas išsaugomas

DP pasirenka sprendinį su didžiausiu (mažiausiu) įverčiu



# Dynamic Programming Algorithm

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.

Turime kryptinį beciklį grafą, su ne neigiamais svoriais (“kainomis”) priskirtais grafo vektoriams

Turime kryptinio grafo viršūnę  $z$ . Reikia rasti mažiausios “kainos” kelią nuo kiekvienos grafo viršūnės iki viršūnės  $z$ .

Tegul

$C(i,j)$  = kaina einant vektoriumi nuo viršūnės  $i$  iki viršūnės  $j$ .

$J(i)$  = Optimali kaina kelio nuo viršūnės  $i$  iki viršūnės  $z$ .

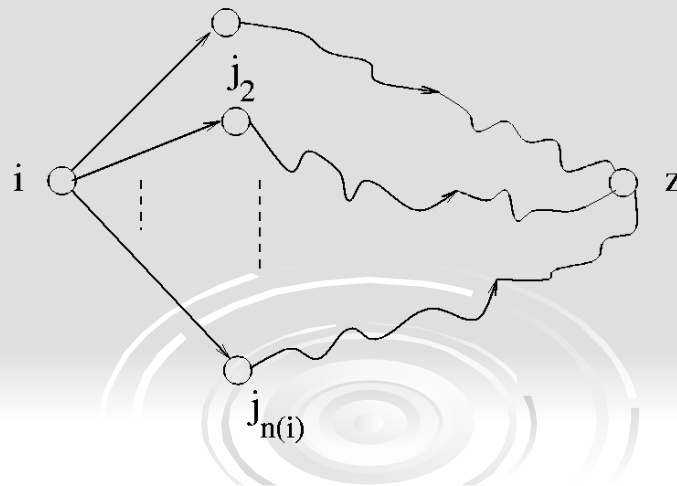
Tada jei optimalus kelias nuo  $i$  iki  $z$  eina per ryšį  $(i,j)$ , tuomet  $J(i)$  :

$$J(z) = 0$$

$$J(i) = C(i, j) + J(j).$$

Tarkime, kad kiekvienai viršūnei  $j$ , kuriai ryšys  $(i,j)$  egzistuoja ir optimalus kelias  $J(j)$  yra žinomas, tuomet :

$$J(i) = \min_j [C(i,j) + J(j)]$$



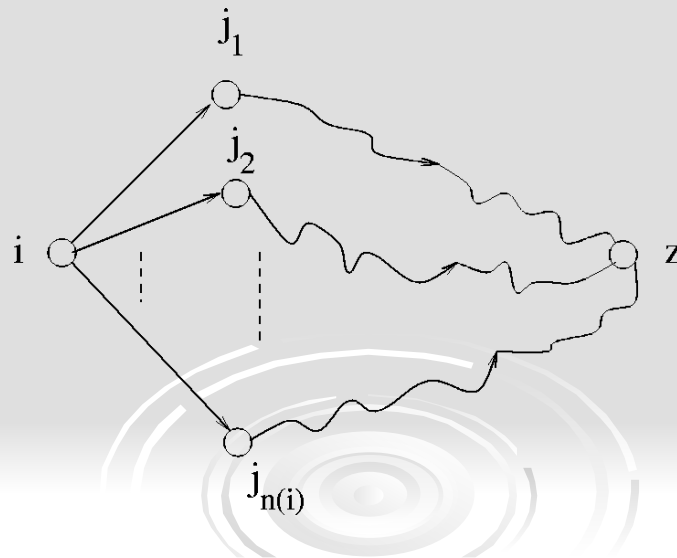
# Dynamic Programming Algorithm

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.

1. Tegul  $J(z) = 0$ . Pradžioje turime tik šią vienintelę viršūnę, kurios kelio iki  $z$  kainą žinome.
2. Laikykime, kad viršūnei  $i$ 
  - $J(i)$ , dar nėra rastas
  - kiekvienai viršūnei  $j$ , kuriai egzistuoja jungtis  $(i,j)$ ,  $J(j)$  yra jau apskaičiuota.
  - Priskirkime  $J(i)$  reikšmę, pagal

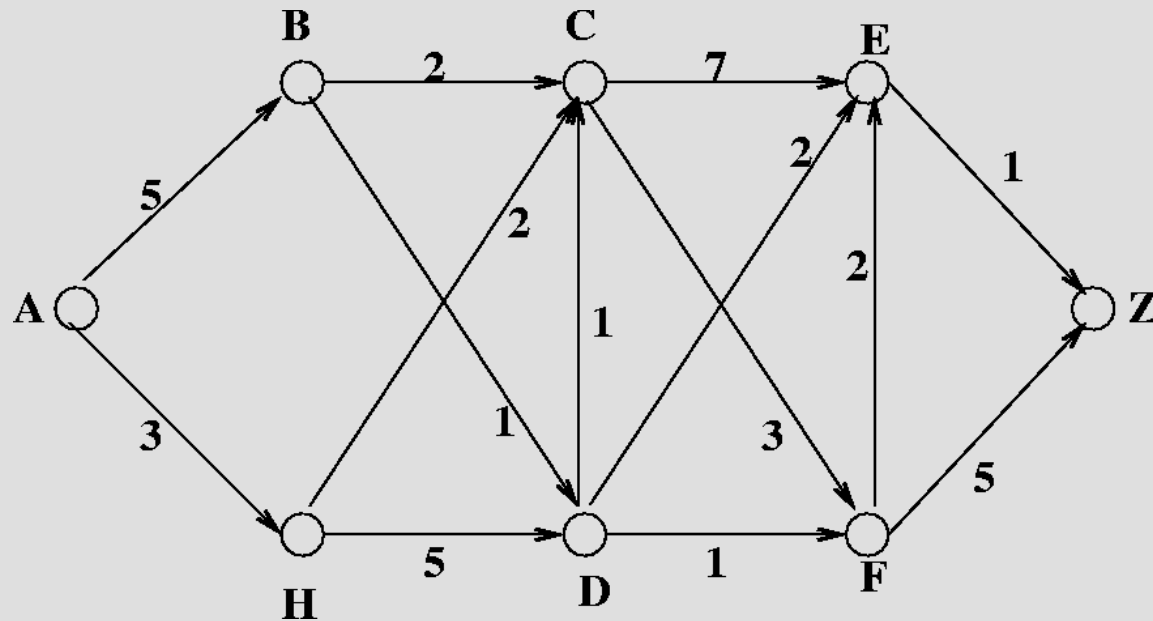
$$J(i) = \min_j [C(i,j) + J(j)]$$

3. Kartokite antrą žingsnį, kol optimalus kelias bus nustatytas visoms viršūnėms.



# Dynamic Programming Algorithm

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.



Žingsnis #1:

Pradžioje  $J(Z) = 0$ .

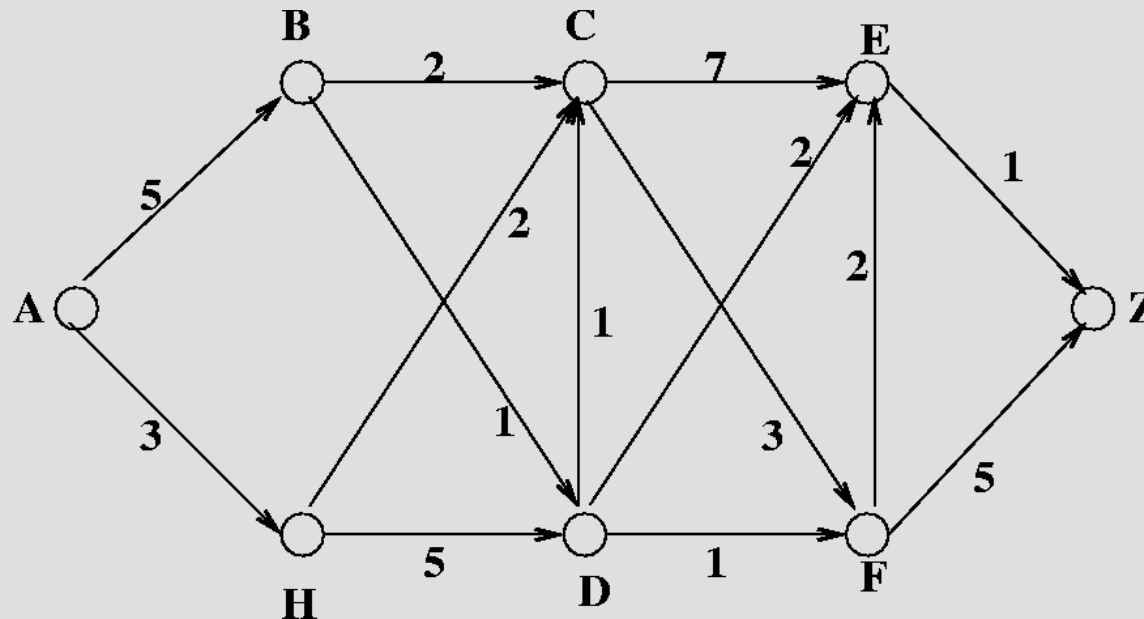
Žingsnis #2:

*E yra vienintelė viršūnė tokia, iš kurios išeina tik vienas vektorius ir jos eina į Z. Taigi*

$$J(E) = C(E, Z) + J(Z) = 1$$

# Dynamic Programming Algorithm

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.



Žingsnis #3:

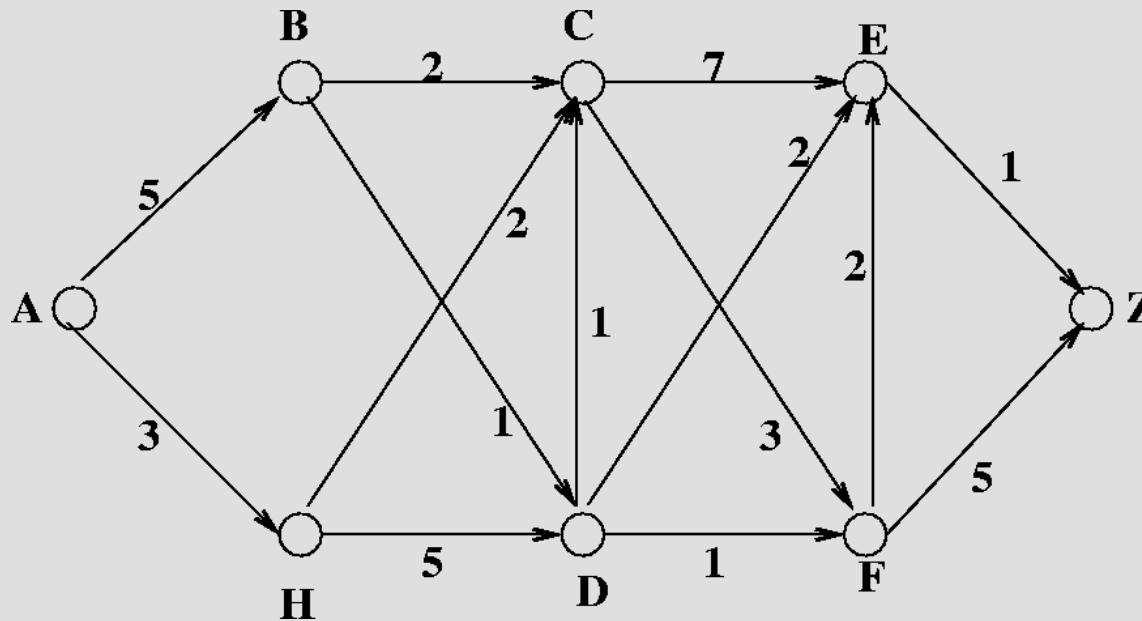
Pasirinkime viršūnę C

$$J(C) = \min \begin{cases} C(C, E) + J(E) = 8 \\ C(C, F) + J(F) = 6 \end{cases}$$
$$= \min(8, 6) = 6$$

Taigi optimalus kelias bus  $F \rightarrow E \rightarrow Z$

# Dynamic Programming Algorithm

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.



Žingsnis #3:

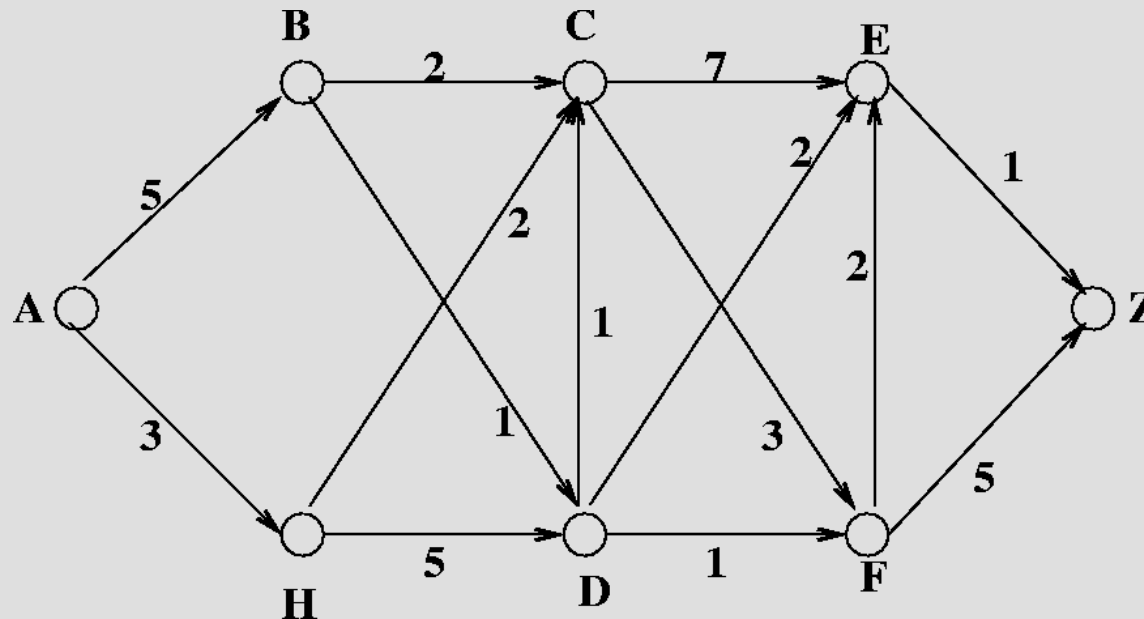
Pasirinkime viršūnę F

$$J(F) = \min \begin{cases} C(F, Z) + J(Z) = 5 \\ C(F, E) + J(E) = 3 \end{cases}$$
$$= \min(5, 3) = 3$$

Taigi optimalus kelias bus  $C \rightarrow F \rightarrow E \rightarrow Z$

# Dynamic Programming Algorithm

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.



Žingsnis #3:

Pasirinkime viršūnę D

$$J(D) = \min \begin{cases} C(D, C) + J(C) = 7 \\ C(D, E) + J(E) = 3 \\ C(D, F) + J(F) = 4 \end{cases} = 3$$

Taigi optimalus kelias bus  $D \rightarrow E \rightarrow Z$  .....ir taip toliau



# Dynamic Programming Algorithm

Richard Bellman. Dynamic Programming. Princeton University Press, 1957.

Limitacijos:

- kompleksiskumas  $O(n^2)$ , kur  $n$  – viršūnių skaičius
- netinka grafams su ciklais.



# Dinaminis programavimas

Pritaikymas sekų lyginimui (Needleman/Wunsch) (1)

Pateikiamas pavyzdys, kaip taikant dinaminį programavimą gautas optimalus sekų palyginys (Needleman/Wunsch) technika.

Ieškosim palyginio šių dviejų sekų:

G A A T T C A G T T A (seka #1)

G G A T C G A (seka #2)

Taigi  $M = 11$  ir  $N = 7$  (atitinkamai ilgis sekos #1 ir sekos #2)

Naudojama parasta įverčio schema

$S_{i,j} = 1$  jei nukleorūgštis sekos #1 pozicijoje  $i$  ir nukleorūgštis sekos #2 pozicijoje  $j$  yra ta pati

$S_{i,j} = -1$  (nesutapimo įvertis)

$w = -2$  (tarpo bausmė)

Trys žingsniai dinaminiame programavime:

- Iniciacija
- Matricos užpildymas
- Gryžimas atgal pagal matricos vertes - palyginio sudarymas

[illegible]



# Dinaminis programavimas

Pritaikymas sekų lyginimui (Needleman/Wunsch) (3)

## Matricos pildymas (2):

Kiekvienam matricos langeliui, priskiriamas maksimalus galimas įvertis  $M_{i,j}$ .

$M_{i,j} = \text{MAXIMUM}[$

$M_{i-1,j-1} + S_{i,j}$  (atitikimo/neatitikimo įvertis. **Raudona rodyklė. Matricoje diagonaliai**),

$M_{i,j-1} + w$  (trūkis sekoje #1, **žalia. Matricoje ėjimas į apačią**),

$M_{i-1,j} + w$  (trūkis sekoje #2, **mėlyna. Matricoje ėjimas į dešinę**)]

Einant žemyn į pirmą stulpelį (1,1) vėl yra sutapimas.

$S_{1,2} = 2$ .  $M_{1,2} = \text{MAX}[M_{0,1} + 2, M_{1,1} - 2, M_{0,2} - 2] = \text{MAX}[0 + 2, 2 - 2, 0 - 2] = \text{MAX}[2, 0, -2] = 2$

|   |   | G | A | A | T | T | C | A | G | T | T | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   |   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 2 |   |   |   |   |   |   |   |   |   |   |
| G | 0 | 2 |   |   |   |   |   |   |   |   |   |   |
| A | 0 |   |   |   |   |   |   |   |   |   |   |   |
| T | 0 |   |   |   |   |   |   |   |   |   |   |   |
| C | 0 |   |   |   |   |   |   |   |   |   |   |   |
| G | 0 |   |   |   |   |   |   |   |   |   |   |   |
| A | 0 |   |   |   |   |   |   |   |   |   |   |   |

! langelį, pagal kurį apskaičiuojame didžiausią reikšmę įsimename (nubrėžiame rodyklę į jį)

# Dinaminis programavimas

Pritaikymas sekų lyginimui (Needleman/Wunsch) (3)

## Matricos pildymas (e):

Kiekvienam matricos langeliui, priskiriamas maksimalus galimas įvertis  $M_{i,j}$ .

$M_{i,j} = \text{MAXIMUM}[$

$M_{i-1,j-1} + S_{i,j}$  (atitikimo/neatitikimo įvertis. **Raudona rodyklė. Matricoje diagonaliai**),

$M_{i,j-1} + w$  (trūkis sekoje #1, **žalia. Matricoje ėjimas į apačią**),

$M_{i-1,j} + w$  (trūkis sekoje #2, **mėlyna. Matricoje ėjimas į dešinę**)]

Einant žemyn į pirmu stulpeliu (1,3) yra nesutapimas.

$S_{1,3} = -1$ .  $M_{1,3} = \text{MAX}[M_{0,2} - 1, M_{1,2} - 2, M_{0,3} - 2] = \text{MAX}[0 - 1, 2 - 2, 0 - 2] = \text{MAX}[-1, 0, -2]$ .

|   | G | A | A | T | T | C | A | G | T | T | A |
|---|---|---|---|---|---|---|---|---|---|---|---|
| G | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 2 |   |   |   |   |   |   |   |   |   |
| A | 0 | 2 | 0 |   |   |   |   |   |   |   |   |
| T | 0 |   |   |   |   |   |   |   |   |   |   |
| C | 0 |   |   |   |   |   |   |   |   |   |   |
| G | 0 |   |   |   |   |   |   |   |   |   |   |
| A | 0 |   |   |   |   |   |   |   |   |   |   |

! langelį, pagal kurį apskaičiuojame didžiausią reikšmę įsimename (nubrėžiame rodyklę į jį)

# Dinaminis programavimas

Pritaikymas sekų lyginimui (Needleman/Wunsch) (4)

Gryžimas atgal pagal vertes ir sekų suliginimo radimas (1):

|   |  | G | A  | A | T  | T  | C  | A  | G  | T  | T  | A  |
|---|--|---|----|---|----|----|----|----|----|----|----|----|
|   |  | 0 | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| G |  | 0 | 2  | 0 | -1 | -1 | -1 | -1 | 2  | 0  | -1 | -1 |
| G |  | 0 | 2  | 1 | -1 | -2 | -2 | -2 | 1  | 1  | -1 | -2 |
| A |  | 0 | 0  | 4 | 3  | 1  | -1 | -3 | 0  | -1 | 0  | 1  |
| T |  | 0 | -1 | 2 | 3  | 5  | 3  | 1  | -1 | -1 | 1  | 2  |
| C |  | 0 | -1 | 0 | 1  | 3  | 4  | 5  | 3  | 1  | -1 | 0  |
| G |  | 0 | 2  | 0 | -1 | 1  | 2  | 3  | 4  | 5  | 3  | 1  |
| A |  | 0 | 0  | 4 | 2  | 0  | 0  | 1  | 5  | 3  | 4  | 2  |

! langelį, pagal kurį apskaičiuojame didžiausią reikšmę įsimename (nubrėžiame rodyklę į jį)

Gryžtam nuo dešinio apatinio langelio iki kairiojo viršutinio sekdami rodyklėmis.

Galima rasti keletą vienodai optimalių kelių...

# Dinaminis programavimas

Pritaikymas sekų lyginimui (Needleman/Wunsch) (4)

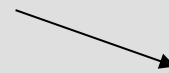
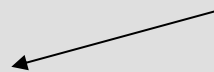
**Gryžimas atgal pagal vertes ir sekų sulyginimo radimas (2):**

|   | G | A  | A | T  | T  | C  | A  | G  | T  | T  | A  |
|---|---|----|---|----|----|----|----|----|----|----|----|
| G | 0 | 0  | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| G | 0 | 2  | 0 | -1 | -1 | -1 | -1 | 2  | 0  | -1 | -1 |
| G | 0 | 2  | 1 | -1 | -2 | -2 | -2 | 1  | 1  | -1 | -1 |
| A | 0 | 0  | 4 | 3  | 1  | -1 | -3 | 0  | -1 | 0  | 0  |
| T | 0 | -1 | 2 | 3  | 5  | 3  | 1  | -1 | -1 | 1  | 2  |
| C | 0 | -1 | 0 | 1  | 3  | 4  | 5  | 3  | 1  | -1 | 0  |
| G | 0 | 2  | 0 | -1 | 1  | 2  | 3  | 4  | 5  | 3  | 1  |
| A |   |    |   |    |    |    |    |    |    |    | 3  |



|   | G | A  | A | T  | T  | C | A | G | T | T | A |
|---|---|----|---|----|----|---|---|---|---|---|---|
| G | 0 | 0  | 0 | 0  | 0  |   |   |   |   |   |   |
| G | 0 | 2  | 0 | -1 | -1 |   |   |   |   |   |   |
| G | 0 | 2  | 1 | -1 | -2 |   |   |   |   |   |   |
| A | 0 | 0  | 4 | 3  | 1  |   |   |   |   |   |   |
| T | 0 | -1 | 2 | 3  | 5  | 3 |   |   |   |   |   |
| C |   |    |   |    |    | 5 | 3 |   |   |   |   |
| G |   |    |   |    |    |   | 5 | 3 | 1 |   |   |
| A |   |    |   |    |    |   |   |   |   |   | 3 |

|   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|
| T | C | A | G | T | T | A |
|   |   |   |   |   |   |   |
| T | C | _ | G | _ | _ | A |



|   | G | A | A | T | T | C | A | G | T | T | A |
|---|---|---|---|---|---|---|---|---|---|---|---|
| G | 0 |   |   |   |   |   |   |   |   |   |   |
| G |   | 2 |   |   |   |   |   |   |   |   |   |
| G |   |   | 1 |   |   |   |   |   |   |   |   |
| A |   |   |   | 3 | 1 |   |   |   |   |   |   |
| T |   |   |   |   | 3 |   |   |   |   |   |   |
| C |   |   |   |   |   | 5 | 3 |   |   |   |   |
| G |   |   |   |   |   |   | 5 | 3 | 1 |   |   |
| A |   |   |   |   |   |   |   |   |   |   | 3 |

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|
| G | A | A | T | T | C | A | G | T | T | A |
|   |   |   |   |   |   |   |   |   |   |   |
| G | G | A | _ | T | C | _ | G | _ | _ | A |

|   | G | A | A | T | T | C | A | G | T | T | A |
|---|---|---|---|---|---|---|---|---|---|---|---|
| G | 0 |   |   |   |   |   |   |   |   |   |   |
| G |   | 2 |   |   |   |   |   |   |   |   |   |
| G |   |   | 1 |   |   |   |   |   |   |   |   |
| A |   |   |   | 3 |   |   |   |   |   |   |   |
| T |   |   |   |   | 3 |   |   |   |   |   |   |
| C |   |   |   |   |   | 5 | 3 |   |   |   |   |
| G |   |   |   |   |   |   | 5 | 3 | 1 |   |   |
| A |   |   |   |   |   |   |   |   |   |   | 3 |

|   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|
| G | A | A | T | T | C | A | G | T | T | A |
|   |   |   |   |   |   |   |   |   |   |   |
| G | G | A | T | _ | C | _ | G | _ | _ | A |



# Dinaminis programavimas

Gali būti taikomas globalių ir lokalių palyginių paieškai.

Palyginių įvertinimui gali būti naudojamos pakeitimo matricos

Reikia įvesti tarpo buvimo palyginyje “baudą”



# Smith-Waterman algoritmas

|   |   | G | C | C | C | T | A | G | C | G |
|---|---|---|---|---|---|---|---|---|---|---|
|   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| G | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| C | 0 | 0 | 2 | 1 | 1 | 0 | 0 | 0 | 2 | 0 |
| G | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 3 |
| C | 0 | 0 | 2 | 1 | 2 | 0 | 0 | 0 | 2 | 1 |
| A | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| A | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 |
| T | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| G | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |

- lokalus o ne globalus
- neigiamos vertės - prilyginamos nuliams
- ieškoma ilgiausios diagonalės iki nulio.

Geriausias lokalus palyginys:

gcg

# Dinaminis programavimas

Garantuoja optimalaus palyginio radimą  
(naudojant tam tikrą įverčių schemą)

Lėtas - sudėtingumas  $O(n^2)$

Kompiuterio atminties reikalavimai auga  
kvadratiškai nuo sekos ilgio

Netinka ilgų sekų palyginimui

