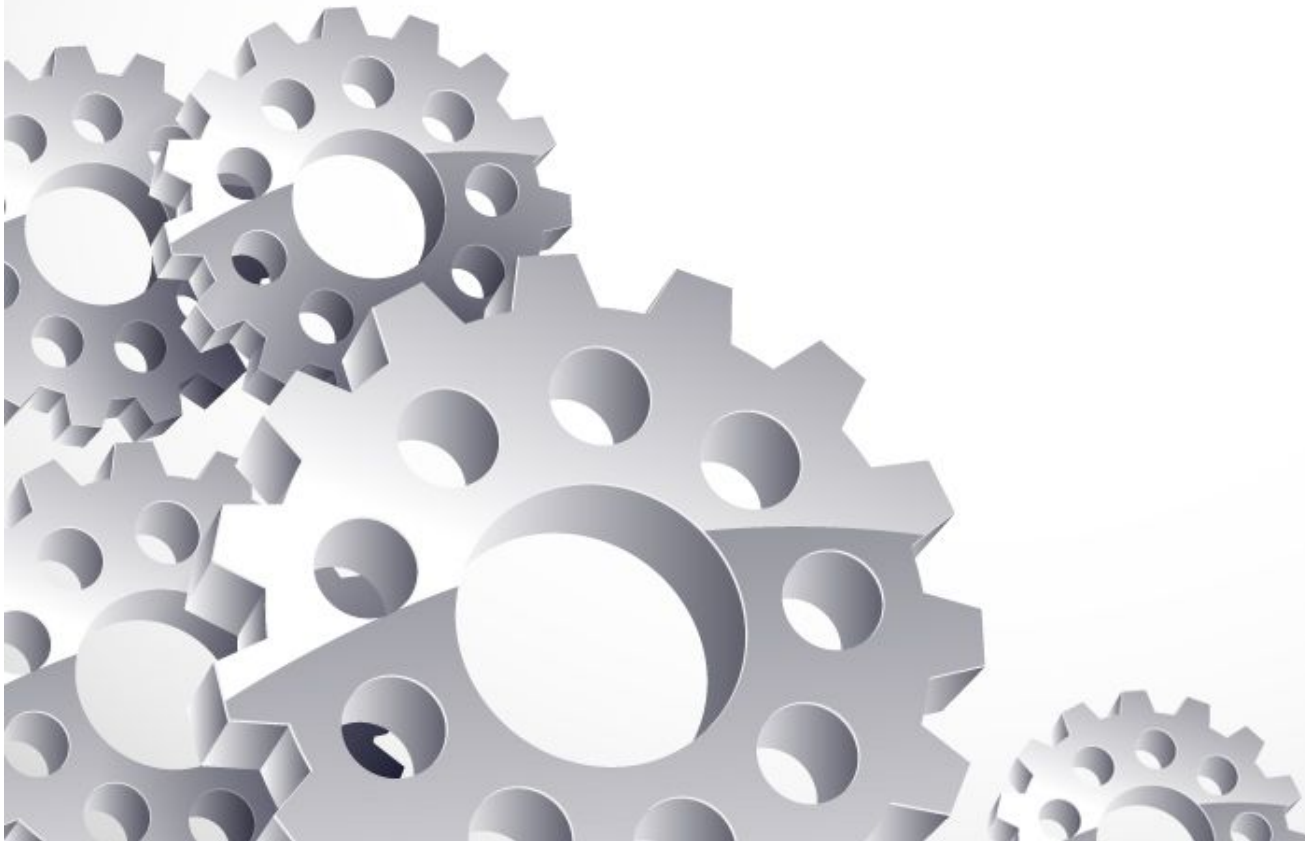


# Euristiniai poriniai sekų palyginimai

## Daugybinis sekų palyginys



# FASTA algoritmas

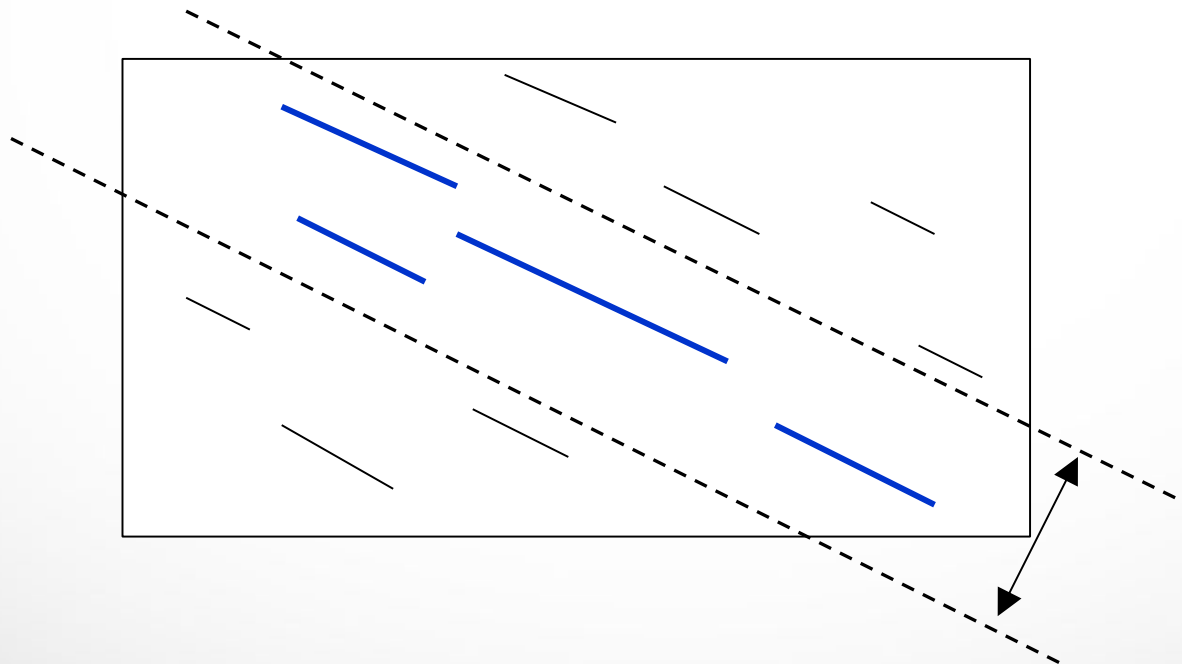
- DP algoritmas atlieka daug skaičiavimų bereikšmėje srityje

	G	A	A	T	T	C	A	G	T	T	A
G	1	1	1	1	1	1	1	1	1	1	1
G	1	1	1	1	1	1	1	2	2	2	2
A	1	2	2	2	2	2	2	2	2	2	2
T	1	2	2	3	3	3	3	3	3	3	3
C	1	2	2	3	3	4	4	4	4	4	4
G	1	2	2	3	3	4	4	5	5	5	5
A	1	2	3	3	3	4	5	5	5	5	6

- FASTA sutelkia paiešką į įstrižainių sritį

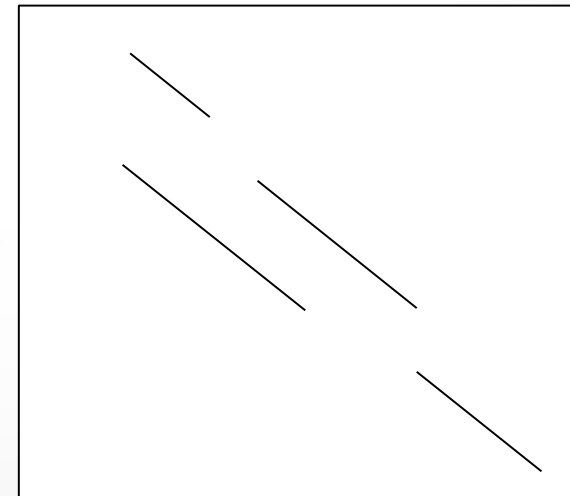
# FASTA

- Naudojami artiniai (“heuristika”):  
geras lokalus palyginys turi tam tikrą  
visišką sutapimo subseką.



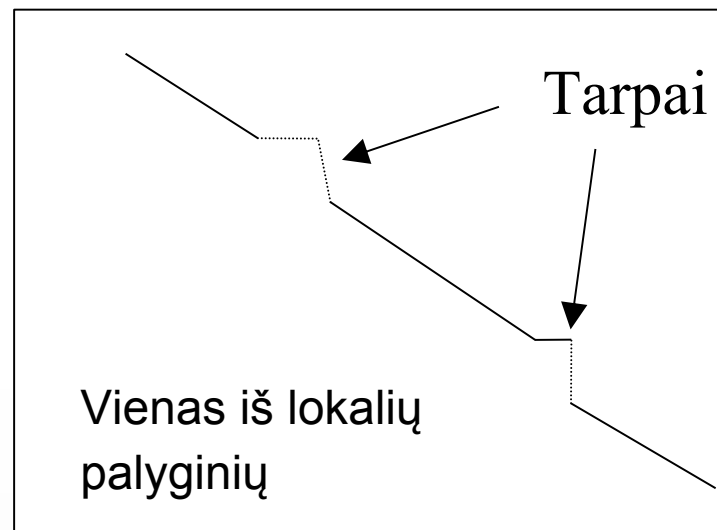
# FASTA algoritmas

- Surasti visus “karštus taškus” (ilgio k sekos, kurios idealiai sutampa)
- Galima naudoti “hash” arba “look-up” lenteles
- Atrinkti N geriausių sekų



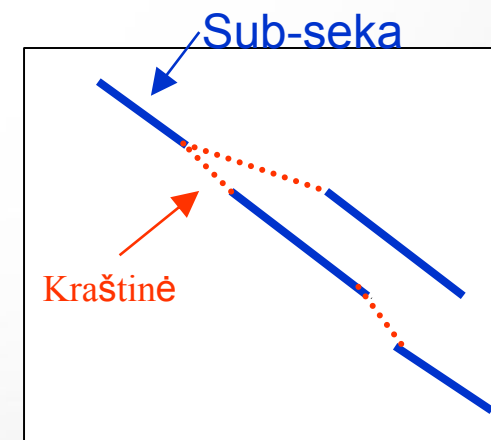
# FASTA algoritmas

- Apjungti sub-palyginius atsižvelgiant į tarpus



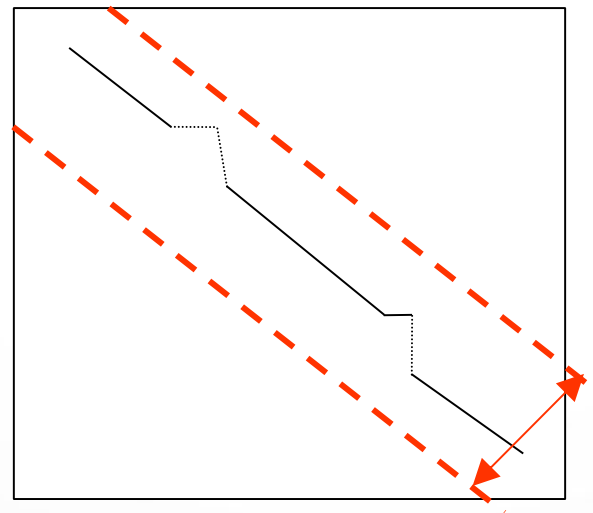
# FASTA algoritmas

- Konstruojamas svorinis kryptinis grafas
- Mazgai yra sub-palygininiai
- Kraštinė  $(u,v)$  egzistuoja, jei  $u$  yra prieš  $v$
- Kiekviena kraštinė turi tarpo baudą (neigiamas svoris)
- Ieškoma maksimalaus svorio kelio



# FASTA algoritmas

- **Apribotoje** srityje naudojamas dinaminio programavimo algoritmas



Juostos plotis  
parametrizuotas

# BLAST algoritmas

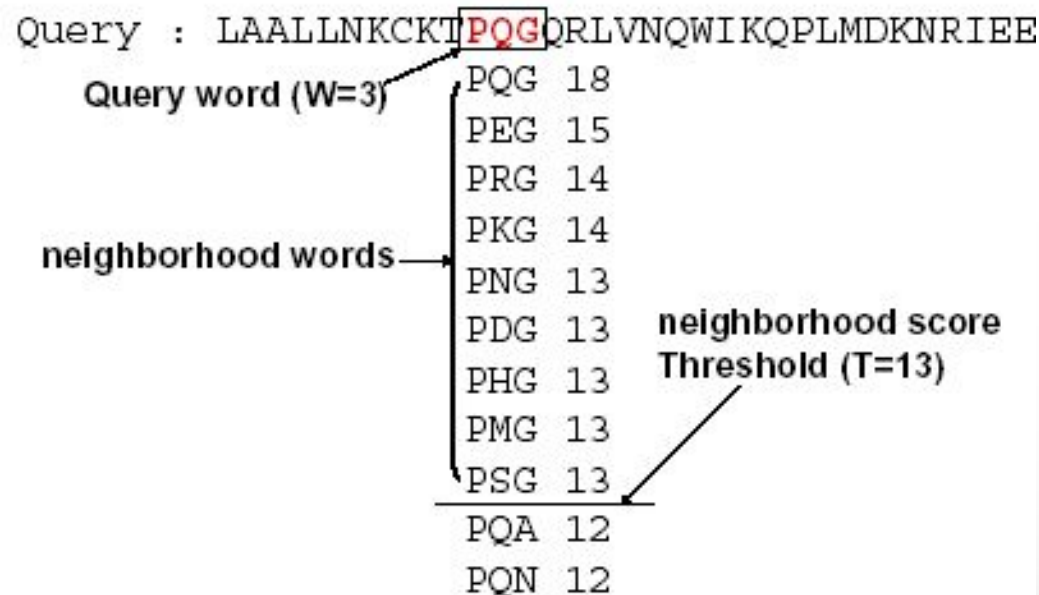


- Kitas heuristinis algoritmas
- Rezultatai įvertinami statistiškai
- Remiasi prielaida, kad homologinės sekos turi trumpų sekų porų su dideliais įverčiais.
- Šiuos trumpus segmentus algoritmas praplečia į abi puses kad būtų gautas optimalus palyginys

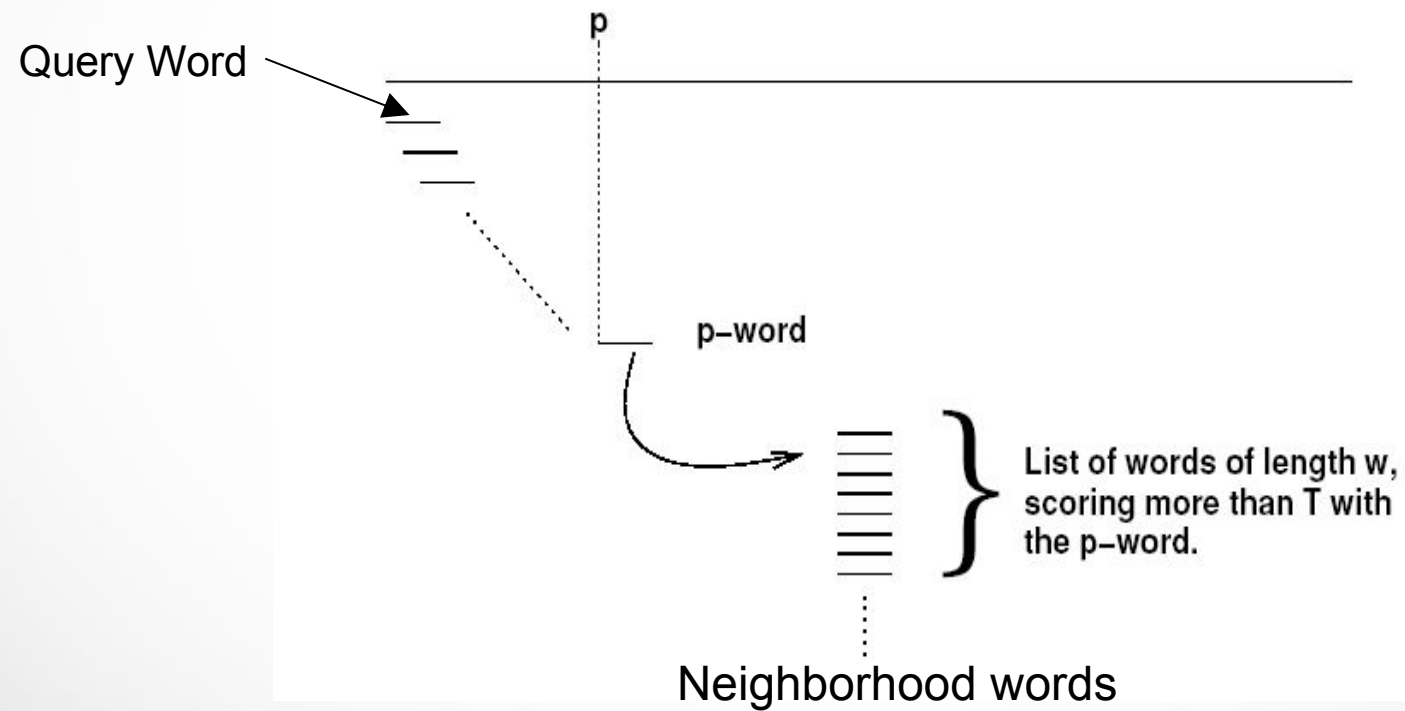


# BLAST algoritmas

- Paruošiamieji darbai:
  - 1 žingsnis – paruošti daugiausiai taškų turinčius žodžius iš užklausos sekos

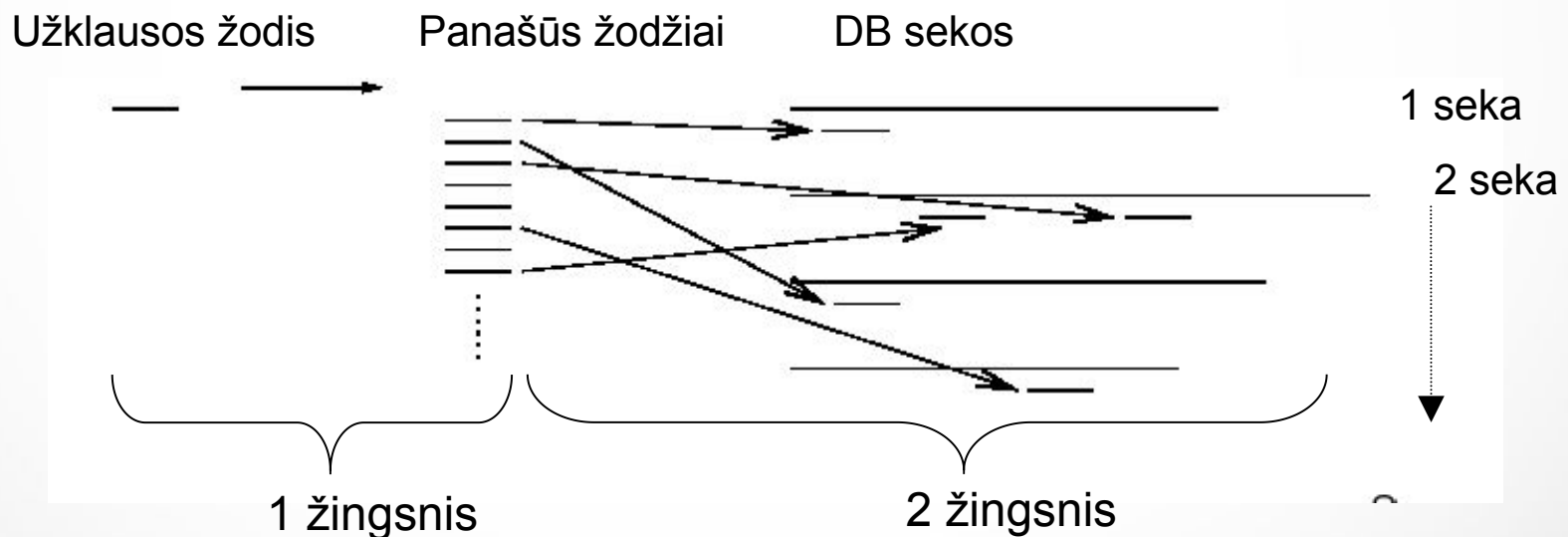


# BLAST algorithms



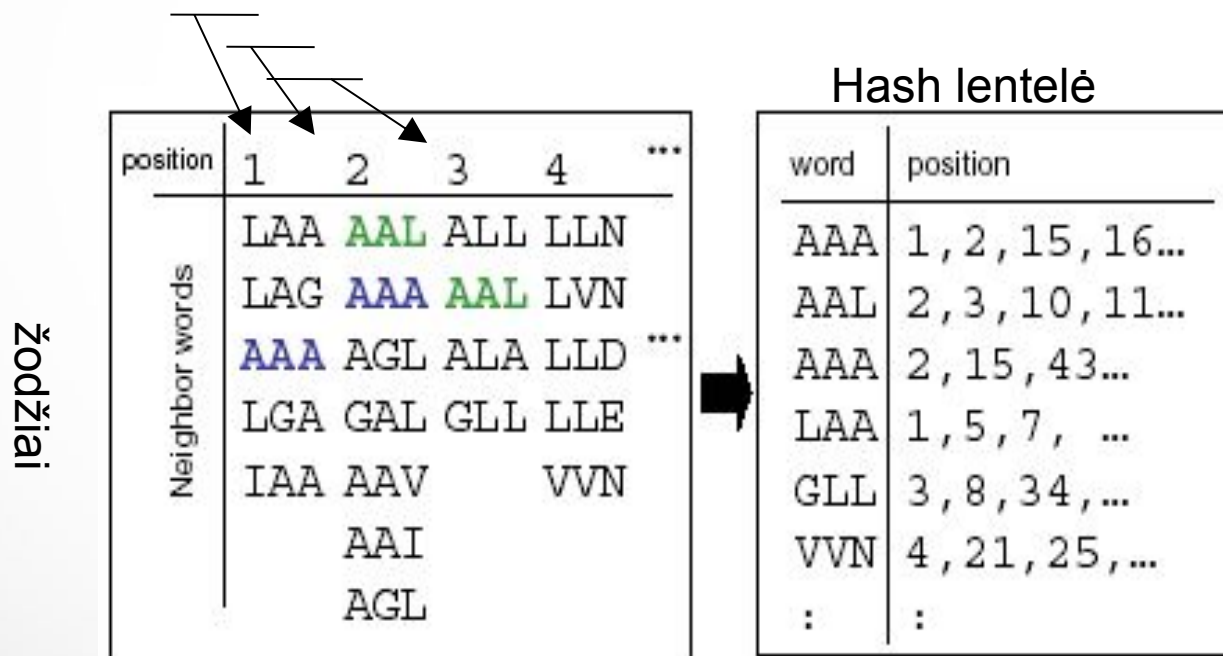
# BLAST algoritmas


- 2 žingsnis – paieška sekų duomenų bazėje. Kiekvienam žodžiui iš sąrašo randami tikslūs radiniai DB



# BLAST algoritmas

- Galima naudotis hash-lentelėmis





Seq\_XYZ: HVTGRSAF\_FS**YYG**YGCYC**GLG**TGKGLPVDATDRCCWA  
Query: QSVFDYI**YYG**CYCGW**GLG**\_GK\_\_PRDA

The diagram shows a sequence alignment between Seq\_XYZ and a Query. Two vertical boxes highlight the regions YYG and GLG in both sequences. Arrows point from a line above to these boxes. Vertical bars indicate gaps in the alignment: two gaps under 'FS' in Seq\_XYZ, one gap under 'I' in Query, and two gaps under 'GK' in Query.

E-val= $10^{-13}$

- 3 žingsnis – optimalaus palyginio paieška.
- Naudojami dviejų žodžių sutapimai, kaip inkarai palyginio konstravimui

# BLAST algoritmas



- 4 žingsnis – palyginio statistinio reikšmingumo įvertinimas. Palyginio plėtimas stabdomas, kai E-reikšmė būna didesnė nei ribinė. Toks rastas segmentas vadinamas didelio įverčio segmentu (High Scoring Segment Pair, HSSP, HSP)

# BLAST algoritmas



- E- reikšmės apibrėžimas:

Tikėtinas HSP, kurių įvertis didesnis nei S, skaičius

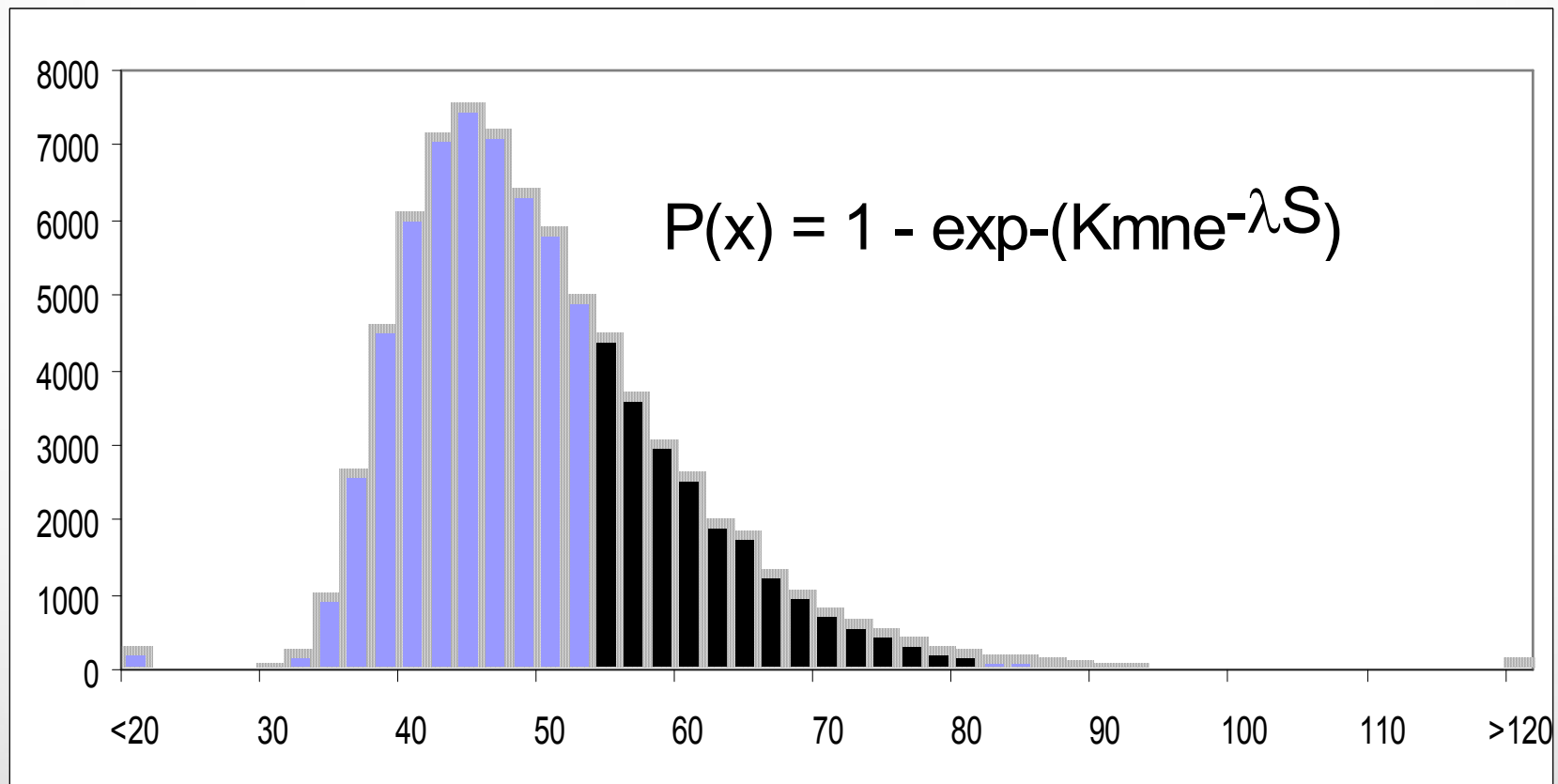
$$E = K * n * m * e^{-\lambda S}$$

$K$ ,  $\lambda$  nuo modelio priklausančios konstantos

$n$ ,  $m$  užklausos ir sekos ilgiai

# BLAST algoritmas

- Sekų įverčiai pasiskirstę pagal ekstremalių verčių dėsnį.





# Algoritmų palyginimas



- Užklausos ilgis – 153
- DB dydis – 5997 sekos

Algoritmas	Trukmė
D.P	16.989 [s]
FASTA	0.618 [s]
BLAST	0.118 [s]

# Algoritmų palyginimas



- Dinaminis programavimas:
  - Jautriausias algoritmas
  - Panaudojama visa informacija
  - Algoritmas lėtas
  - Naudojamos ir bereikšmės sritys

# Algoritmų palyginimas



- FASTA
  - Mažiau jautrus nei DP ir BLAST
  - Naudojama dalinė informacija pagreitinant skaičiavimus
  - Rezultatai nevertinami statistiškai
  - Žymiai greitesnis nei DP

# Algoritmų palyginimas



- BLAST
  - Jautresnis nei FASTA
  - Rezultatai įvertinami statistiškai
  - Greitesnis nei FASTA. Atsižvelgiant į rezultatų patikimumą atmetamas triukšmas ir tokiu būdu sutrumpėja skaičiavimo laikas

# Porinio palyginimo generazizacija



- Dviejų sekų palyginys - dvimatė matrica.
- Analogiškai palyginimą iš trijų sekų galima pavazduoti optimalaus kelio trajektorija trimatėje matricoje.

A T \_ G C G \_

A \_ C G T \_ A

A T C A C \_ A

- Įvertis - kuo mažiau vairuoja stulpeliai tuo geresnis palyginimas.

# Palyginiai= Keliai

- Palyginam tris sekas: ATGC,  
AATC,ATGC

	A	--	T	G	C
--	---	----	---	---	---

	A	A	T	--	C
--	---	---	---	----	---

	--	A	T	G	C
--	----	---	---	---	---



# Palyginimo kelias



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

x koordinatė

	A	A	T	--	C
--	---	---	---	----	---

	--	A	T	G	C
--	----	---	---	---	---

# Palyginimo kelias



- Align the 3 sequences: ATGC, AATC, ATGC

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

x koordinatė

0	1	2	3	3	4
	A	A	T	--	C

y koordinatė

	--	A	T	G	C
--	----	---	---	---	---

•



# Palyginimo kelias



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

x koordinatė

y koordinatė

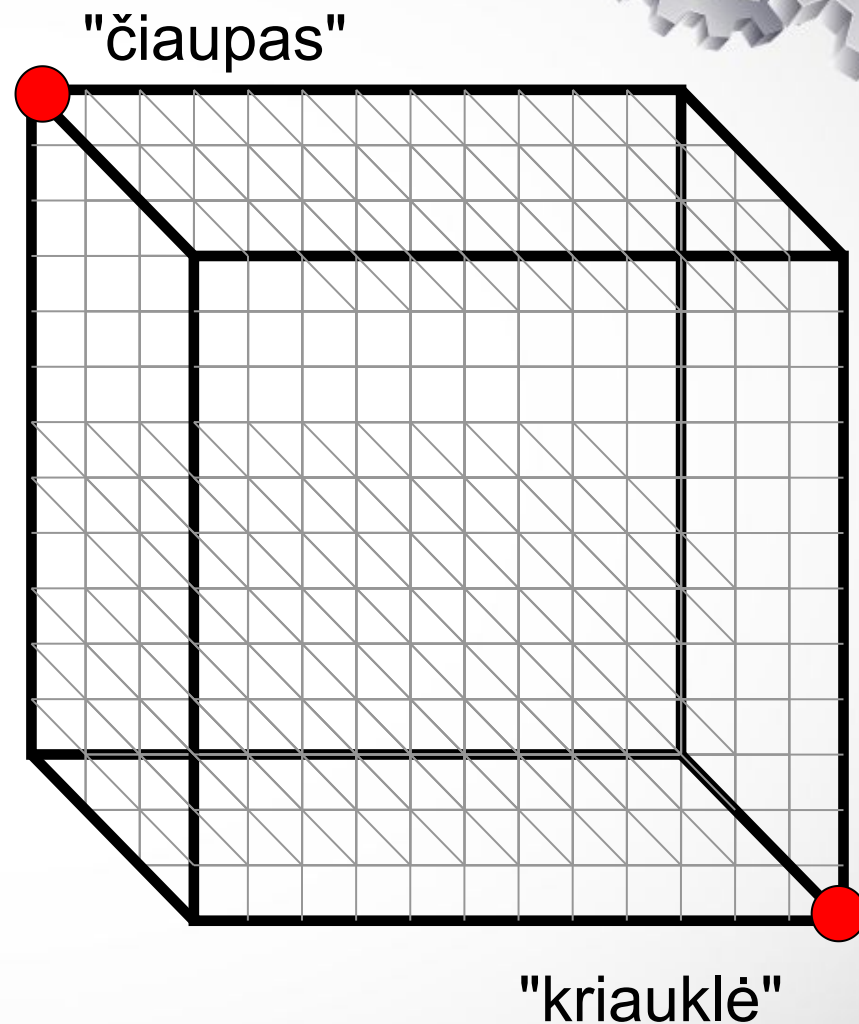
z koordinatė

- Atitinkamas kelias  $(x,y,z)$  erdvėje:

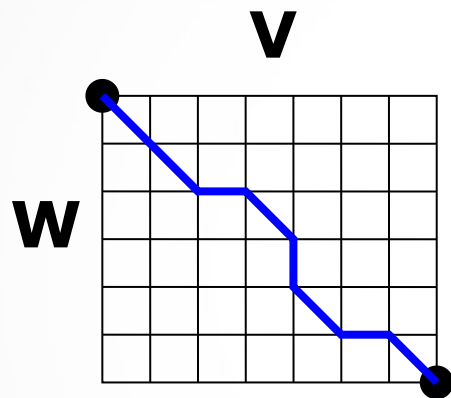
$(0,0,0) \rightarrow (1,1,0) \rightarrow (1,2,1) \rightarrow (2,3,2) \rightarrow (3,3,3) \rightarrow (4,4,4)$

# Trijų sekų lyginimas

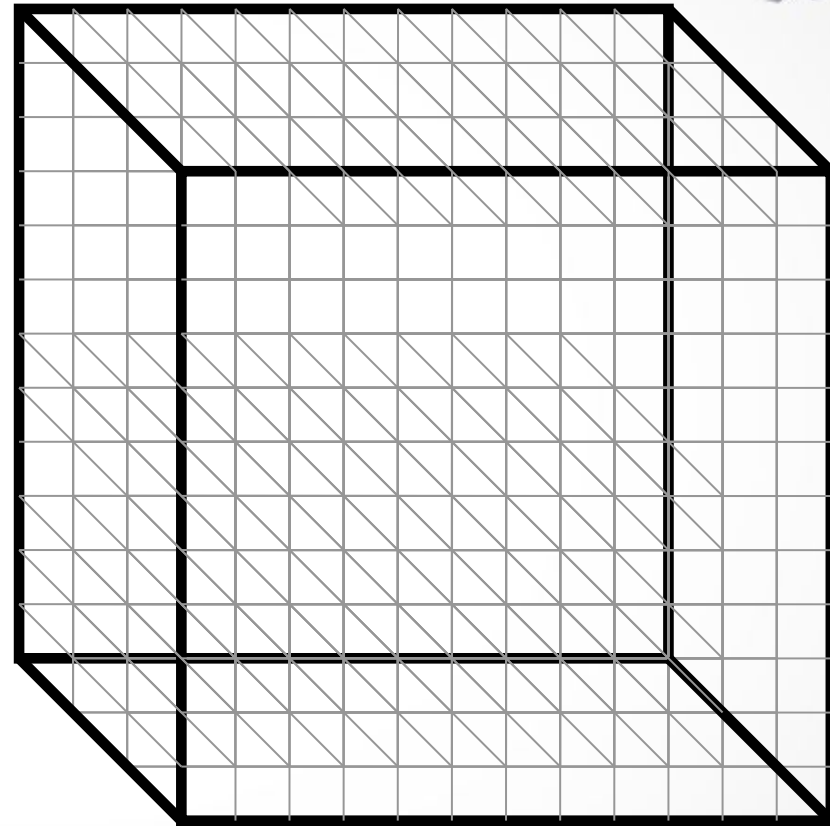
- Ta pati strategija, kaip ir 2-jų sekų atveju
- Naudojamas 3-D "Manheteno kubas", kur kiekviena lyginama seka sutapatinama su koordinačių ašimi.
- Globaliam sulyginimui gauti einama nuo pradžios čiaupo iki kriauklės.



# 2-D vs 3-D Palyginimo tinklelis.

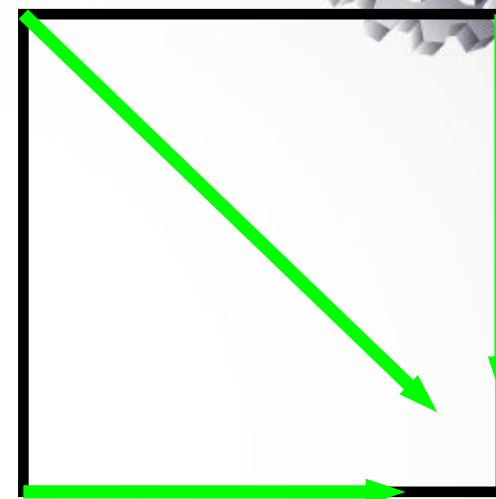
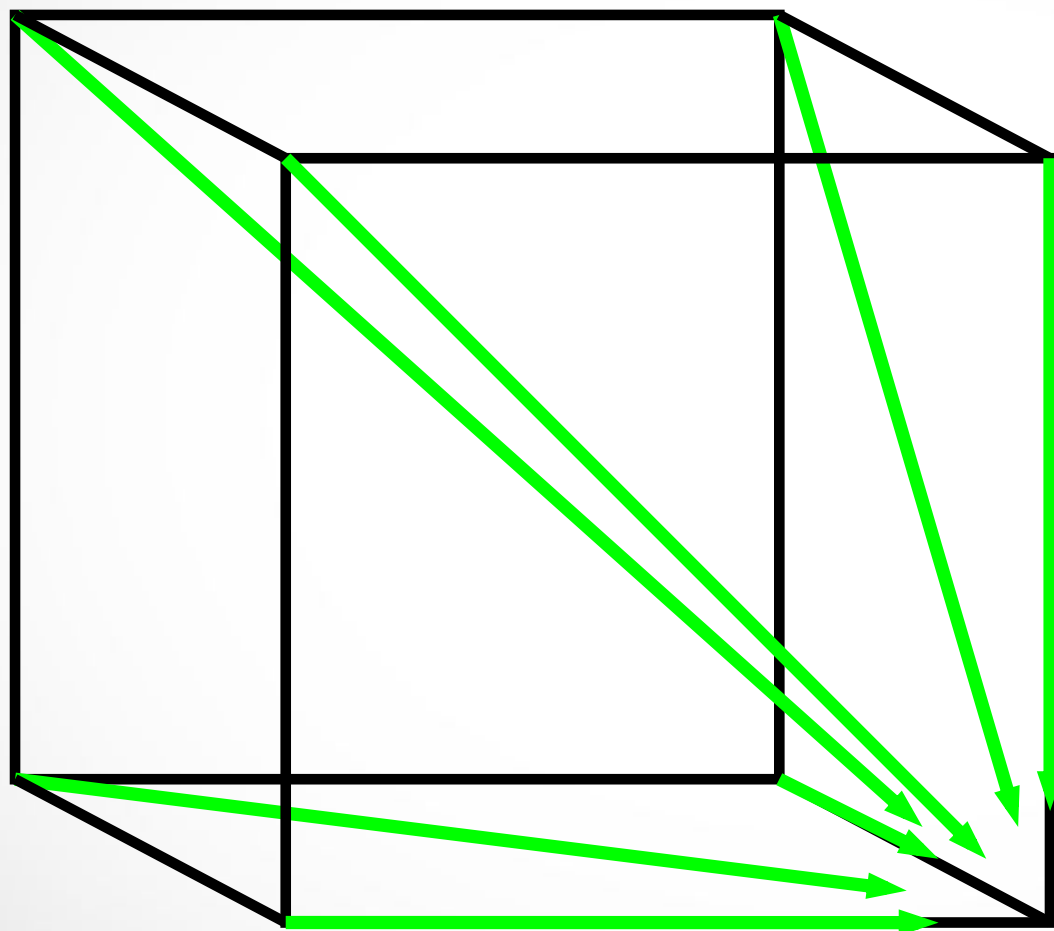


2-D grafas



3-D grafas

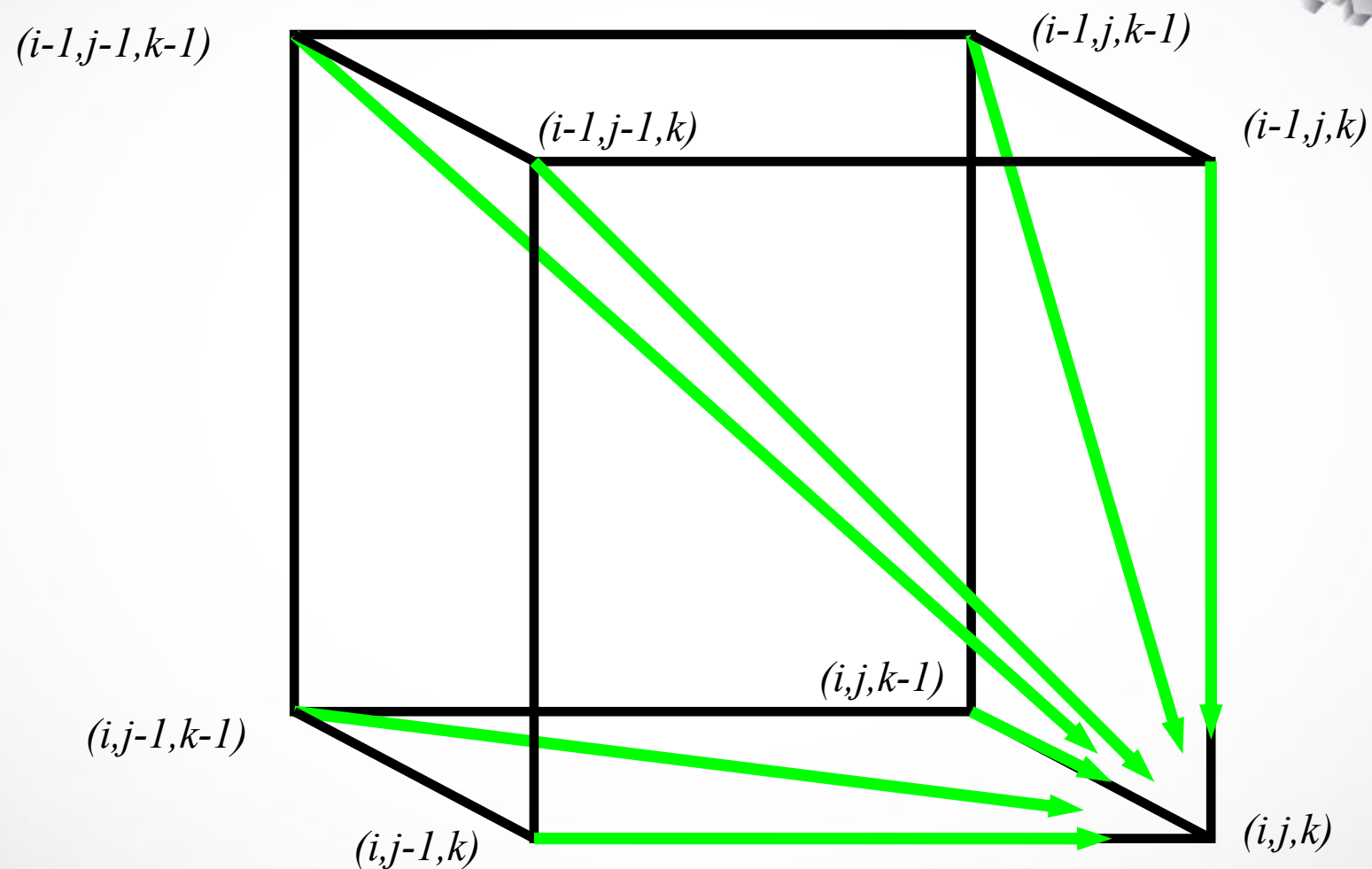
# 3-D vs 2-D palyginimo celė



**2-D**, 3 viršūnės  
vienam lyginimo  
vienetui

**3-D**, 7 viršūnės  
vienam lyginimo  
vienetui

# 3-D sulyginimo celės architektūra



# Daugybinis palyginys: Dinaminis programavimas

- $s_{i,j,k} = \max \left\{ \begin{array}{l} s_{i-1,j-1,k-1} + \delta(v_i, w_j, u_k) \\ s_{i-1,j-1,k} + \delta(v_i, w_j, \_) \\ s_{i-1,j,k-1} + \delta(v_i, \_, u_k) \\ s_{i,j-1,k-1} + \delta(\_, w_j, u_k) \\ s_{i-1,j,k} + \delta(v_i, \_, \_) \\ s_{i,j-1,k} + \delta(\_, w_j, \_) \\ s_{i,j,k-1} + \delta(\_, \_, u_k) \end{array} \right\}$ 
  - kubo diagonalė nėra tarpų
  - plokštumų diagonalės tarpas vienoje sekoje
  - kraštinės diagonalės tarpas dvejose sekose
- $\delta(x, y, z)$  įvertis 3-D įverčių matricoje

# Daugybinis palyginys: vykdymo laikas



- 3-jų  $n$  ilgio sekų palyginimas, globalaus palyginimo laikas  $7n^3$ ;  $O(n^3)$ . (*7-nios diagonalės*)
- $k$  sekoms, atitinka  $k$ -dimensijų paieškos matricą, kurios apskaičiavimo laikas  $(2^k-1)(n^k)$ ;  $O(2^k n^k)$
- Klasikinis dinaminis programavimas lengvai pritaikomas ir išplečiamas daugeliui sekų, bet yra nepraktiškas dėl eksponentiškai didėjančių laiko sąnaudų.

# Daugybinio palyginio profilinė išraiška.



-	A	G	G	C	T	A	T	C	A	C	C	T	G
T	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	T	C	A	C	-	G	G
C	A	G	-	C	T	A	T	C	G	C	-	G	G

A		1				1			.8				
C	.6			1			.4	1		.6	.2		
G			1	.2					.2			.4	1
T	.2				1		.6					.2	
-	.2		.8							.4	.8	.4	



# Daugybinio palyginio profilinė išraiška.



-	A	G	G	C	T	A	T	C	A	C	C	T	G
T	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	C	C	A	-	-	-	G
C	A	G	-	C	T	A	T	C	A	C	-	G	G
C	A	G	-	C	T	A	T	C	G	C	-	G	G

A		1				1			.8				
C	.6			1			.4	1		.6	.2		
G			1	.2					.2			.4	1
T	.2				1		.6					.2	
-	.2		.8							.4	.8	.4	

**Praeityje lygiom seką su seka.**

**Ar galima lyginti seką su profiliu?**

**Ar galima lyginti profilį su profiliu?**

# Lyginant palyginius

- Ar galim sulygtinti du palyginimus?

x	GGGCACTGCAT	
y	GGTTACGTC--	Palyginimas 1
z	GGGAACTGCAG	
w	GGACGTACC--	Palyginimas 2
v	GGACCT-----	



# Lyginant palyginius



- Ar galim sulyginti du palyginimus?
- Galim naudoti atitikamus palyginimus...

**x** GGGCACTGCAT

**y** GGTTACGTC--

**z** GGGAACTGCAG

**w** GGACGTACC--

**v** GGACCT-----

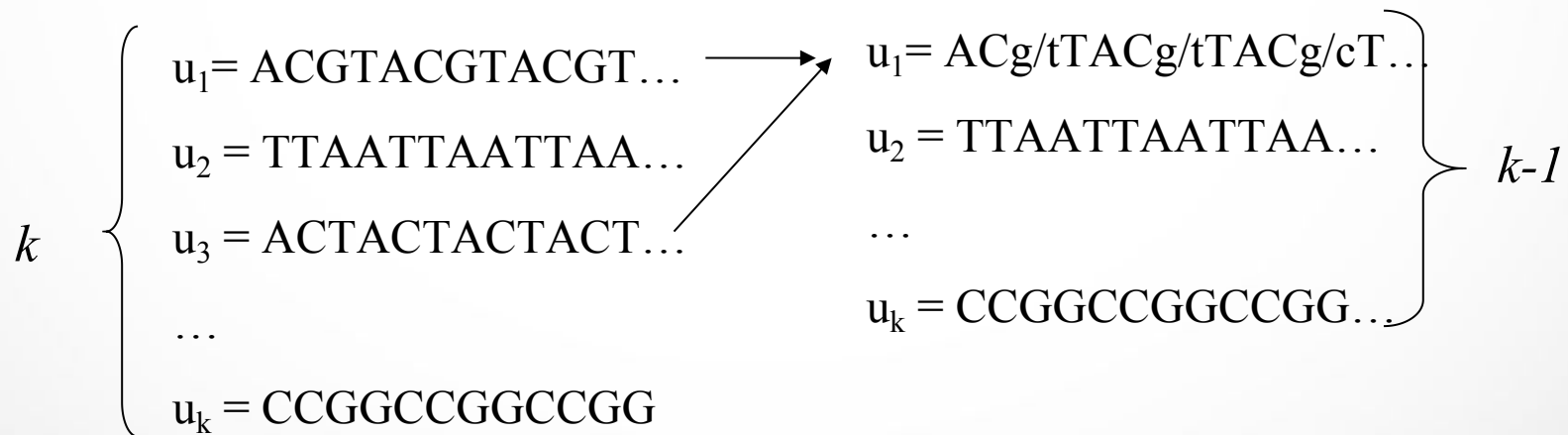
**Apjungti palyginiai**

# Daugybinis palyginis: "godusis" sprendimas

- Pasirink labiausiai panašią sekų porą ir apjunk ją į profilį,  $k$  sekų sulyginimą paverčians į  $k-1$  sekų/profilijų palyginimą.

## Kartojimas

- Euristinis "godusis" metodas:



# "Godusis" sprendimas: Pavyzdys

- Turim keturias sekas:

*s1*    GATTCA

*s2*    GTCTGA

*s3*    GATATT

*s4*    GTCAGC



# "Godusis" sprendimas: Pavyzdys (tesinys)

- Yra  $\binom{6}{4 \atop 2}$  galimi palyginiai

*s2* **GTCTGA**  
*s4* **GTCAGC** (score = 2)

*s1* **GATTCA--**  
*s4* **G-T-CAGC** (score = 0)

*s1* **GAT-TCA**  
*s2* **G-TCTGA** (score = 1)

*s2* **G-TCTGA**  
*s3* **GATAT-T** (score = -1)

*s1* **GAT-TCA**  
*s3* **GATAT-T** (score = 1)

*s3* **GAT-ATT**  
*s4* **G-TCAGC** (score = -1)

# "Godusis" sprendimas: Pavyzdys (tesinys)

$s_2$  ir  $s_4$  yra panašiausi, apjungiam:

$s_2$	GTC <b>TGA</b>	}	$s_{2,4}$ (profilis)	GTC <b>t/aGa/cA</b>
$s_4$	GTC <b>AGC</b>			

naujas 3 sekų palyginiai:

$s_1$	GATTCA
$s_3$	GATATT
$s_{2,4}$	GTC <b>t/aGa/c</b>

# Progresyvus lyginimas



- *Progresyvus lyginimas* yra "godžiojo" algoritmo variantas su kiek inteligentiškesniu pasirinkimu, kurias sekas reikia pirmiausiai apjungti.
- *Progresyvus lyginimas* veikia gerai artimoms sekoms, bet nutolusioms sekoms veikia prastai:
  - Sekos jau sudarytuose profiliuose yra fiksluojamos.
  - Lyginame sekas naudodami profilius.



# ClustalW



- Populiarus ir dar dabar naudojamas algoritmas.
- 'W' reiškia 'weighted' (atskiros palygino dalys turi skirtingus svorius).
- Trijų žingsnių algoritmas
  - 1.) Apskaičiuojam visus galimus porinius palyginius.
  - 2.) Suklasterizuojam sekas pagal panašumą kurdami medį palyginimo "gida".
  - 3.) Kuriam palyginį apjungdami ir lygindami sekas pagal medį.

## Žingsnis 1: Poriniai palyginimai

- Palygina kiekvieną seką su kiekviena. Sukuriama identiškumų matricą.
- Identiškumų matrica = tikslus sutapimas / porinio palyginio ilgis

	$v_1$	$v_2$	$v_3$	$v_4$
$v_1$	–			
$v_2$	.17	–		
$v_3$	.87	.28	–	
$v_4$	.59	.33	.62	–

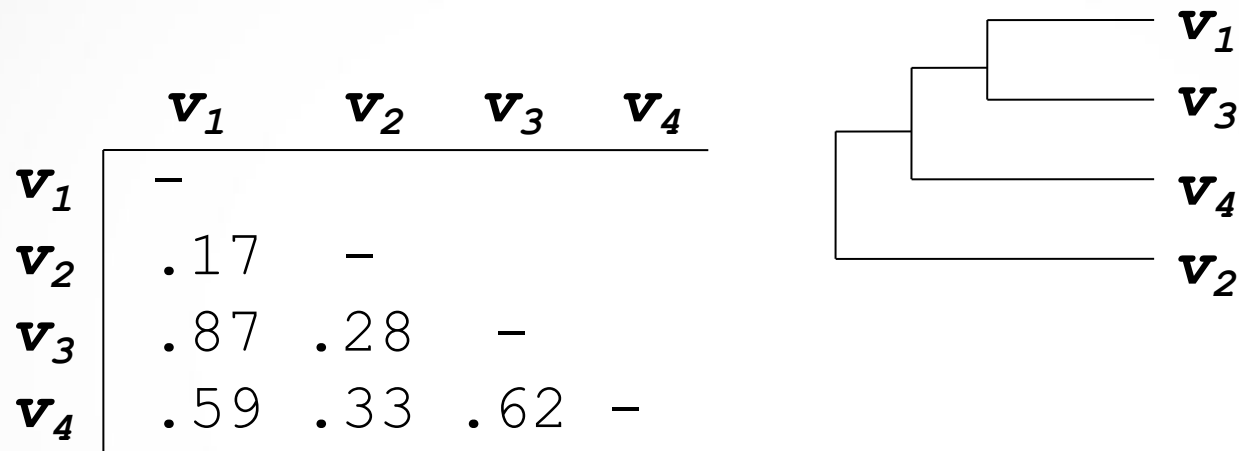
(.17 reiškia kad 17 % identiška)

## Žingsnis 2: Medis - palyginimo "gidas"

- Sukuriamas medis pagal apskaičiuotą identiškumų matricą
- ClustalW naudoja artimiausių kaimynų apjungimo metodiką (sekanti paskaita).
- Naudojamas medis grubiai atititinka evoliucinius ryšius.



## Žingsnis 2: Medis - palyginimo "gidas" (ę)



Skaičiuojam palyginimą:

$v_{1,3}$  = alignment ( $v_1, v_3$ )

$v_{1,3,4}$  = alignment ( $(v_{1,3}), v_4$ )

$v_{1,2,3,4}$  = alignment ( $(v_{1,3,4}), v_2$ )

## Step 3: Progresyvus lyginimas



- Sulyginam dvi panašiausias sekas
- Sekdami medžiu pridedam paeiliui kitas sekas prie palyginio (profilio - sekos lyginimas).

- Insert gaps as necessary

FOS_RAT	PEEMSVTS-LDLTGGLPEATTPESEEAFTLPLLNDPEPK-PSLEPVKNISNMELKAEPFD
FOS_MOUSE	PEEMSVAS-LDLTGGLPEASTPESEEAFTLPLLNDPEPK-PSLEPVKSISNVELKAEPFD
FOS_CHICK	SEELAAATALDLG-----APSPAAAEAFALPLMTEAPPAVPPKEPSG--SGLELKAEPFD
FOSB_MOUSE	PGPGPLAEVRDLPG-----STSAKEDGFGWLLPPPPPPP-----LPFQ
FOSB_HUMAN	PGPGPLAEVRDLPG-----SAPAKEDGFSWLLPPPPPPP-----LPFQ
	. . : ** . :.. *:. * * . *

taškai ir žvaigždutės rodo konservatyviausias sritis

## Daugybinis palyginys: Įvertis

- Sutapimų skaičius (visiškai sutapatintų stulpelių skaičius)
- Entropijos įvertis
- Suminis porinis įvertis(SP-Score)



# Sutapimų skaičiaus įvertis



- Sutapimas skaičiuojamas tik tada, kai visos raidės stulpelyje yra vienodos.

A  
A  
A  
A

AA  
AA  
AT  
TC

- Geras tik visiškai panašioms sekoms.

# Entropijos įvertis



- Apibrėžiam nukleotidų pasikartojimo dažnius stulpeliuose
  - $p_A = 1, p_T=p_G=p_C=0$  (1<sup>st</sup> column)
  - $p_A = 0.75, p_T = 0.25, p_G=p_C=0$  (2<sup>nd</sup> column)
  - $p_A = 0.50, p_T = 0.25, p_C=0.25, p_G=0$  (3<sup>rd</sup> column)
- Compute entropy of each column

$$- \sum_{X=A,T,G,C} p_X \log p_X$$

AAA  
AAA  
AAT  
ATC



# Entropija: Pavyzdys



$$\text{entropy} \begin{pmatrix} A \\ A \\ A \\ A \end{pmatrix} = 0 \quad \text{geriausias atvejis}$$

Worst case

$$\text{entropy} \begin{pmatrix} A \\ T \\ G \\ C \end{pmatrix} = -\sum \frac{1}{4} \log \frac{1}{4} = -4 \left( \frac{1}{4} * -2 \right) = 2$$

# Entropijos įvertis



Daugybino palyginio entropija lygi jo stulpelių entropijų sumai

$$\sum_{\text{per stulpelius}} \sum_{X=A,T,G,C} p_X \log p_X$$

# Palyginio entropija: Pavyzdys



stulpelio entropija:

$$-(p_A \log p_A + p_C \log p_C + p_G \log p_G + p_T \log p_T)$$

A	A	A
A	C	C
A	C	G
A	C	T

- stulpelis 1 =  $-[1 * \log(1) + 0 * \log 0 + 0 * \log 0 + 0 * \log 0]$   
= 0

- stulpelis 2 =  $-[(1/4) * \log(1/4) + (3/4) * \log(3/4) + 0 * \log 0 + 0 * \log 0]$   
=  $-[(1/4) * (-2) + (3/4) * (-1.415)] = +0.811$

- stulpelis 3 =  $-[(1/4) * \log(1/4) + (1/4) * \log(1/4) + (1/4) * \log(1/4) + (1/4) * \log(1/4)]$   
=  $4 * -[(1/4) * (-2)] = +2.0$

- Palyginio entropija =  $0 + 0.811 + 2.0 = +2.811$

# Suminis porinis įvertis



Daugybinį palygininį galima suskaldyti į daugelį porinių

**x:** AC-GCGG-C

**y:** AC-GC-GAG

**z:** GCCGC-GAG

Atitinka:

**x:** ACGCGG-C ;    **x:** AC-GCGG-C ;    **y:** AC-GCGAG

**y:** ACGC-GAC ;    **z:** GCCGC-GAG ;    **z:** GCCGCGAG

# Suminis porinis įvertis(SP-Score)



- Tarkim, kad

$$a_i \text{ ir } a_j$$

yra sekos paimtos iš daugybinio palyginio, kuris sudarytas iš  $k$  sekų.

- Tegul šio porinio palyginio seka bus:

$$s^*(a_i, a_j)$$

- Šių įverčių suma daugybiniam palyginiui ir atitiks SP-Score:

$$s(a_1, \dots, a_k) = \sum_{i,j} s^*(a_i, a_j)$$

# SP-Score skaičiavimas



Daugyninis palygiys iš 4 sekų: 6 poriniai  
payginimai

Sekos  $a_1, a_2, a_3, a_4$ :

$$\begin{aligned} s(a_1 \dots a_4) = \sum s^*(a_i, a_j) = & s^*(a_1, a_2) + s^*(a_1, a_3) \\ & + s^*(a_1, a_4) + s^*(a_2, a_3) \\ & + s^*(a_2, a_4) + s^*(a_3, a_4) \end{aligned}$$

# SP-Score: Pavyzdys

$a_1$  ATG-C-AAT  
A-G-CATAT  
 $a_k$  ATCCCATTT

Einam per visas įmanomas simbolių poras stulpelyje  
ir per visus stulpelius

$$S(a_1 \dots a_k) = \sum_{i,j} s^*(a_i, a_j) \leftarrow \binom{n}{2} \text{ Pairs of Sequences}$$

