

# Metody bioinformatyki (MBI)

Wykład 3 - Bazy sekwencji. Heurystyczne algorytmy badania podobieństwa. Podobieństwo wielu sekwencji.

Robert Nowak

2025Z

# Powtórzenie

- Podobieństwo:
- ▶ globalne (algorytm Needlemana-Wunscha)
  - ▶ lokalne (algorytm Smitha-Watermana)
  - ▶ o liniowej złożoności pamięciowej (algorytm Hirschberga)
  - ▶ z afiniczną funkcją kary

- Macierz podobieństwa:
- ▶ macierze PAM
  - ▶ macierze BLOSUM

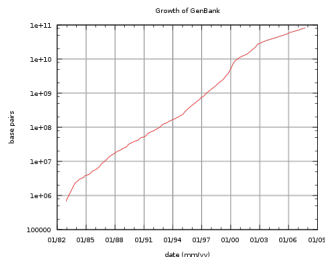
$$e(s_i, t_i) = \log\left(\frac{q_{s_i t_i}}{p_{s_i} p_{t_i}}\right) \text{ gdzie}$$

- $p_a$  estymowane prawdopodobieństwo występowania symbolu  $a$
- $q_{ab}$  estymowane prawd. występowania symbolu  $a$  na jednej nici i symbolu  $b$  na drugiej nici

# *Bazy danych sekwencji biologicznych*

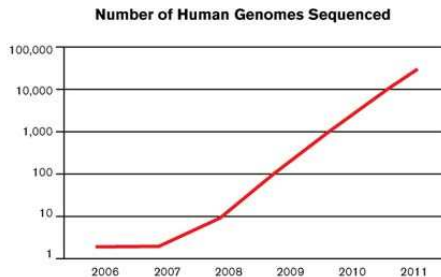
# Bazy danych sekwencji biologicznych

- ▶ GenBank (USA)
- ▶ EMBL (European Molecular Biology Laboratory)
- ▶ DDBJ (DNA Data Bank of Japan)
- ▶ INSD (International Nucleotide Sequence Database) - synchronizuje GenBank, EMBL i DDBJ
- ▶ Swiss-Prot sekwencje białek
- ▶ PSD (Protein Sequence Database) - sekwencje białek
- ▶ PDB (Protein Data Bank) struktura przestrzenna białek
- ▶ ... wiele innych



# Prawo Moore'a dla sekwencji genomów

- ▶ liczba poznawanych genomów podwaja się co 15 miesięcy
- ▶ wielkość sekwencjonowanych genomów podwaja się co 18 miesięcy
- ▶ koszty sekwencjonowania (na bp) zmniejszają się dwukrotnie co 18 miesięcy



# Format danych GenBank - GenBank Flat File(GBFF)

```
LOCUS E01306 229 bp DNA linear PAT 04-NOV-2005
DEFINITION DNA encoding human insulin-like growth factor I(IGF-I).
ACCESSION E01306
VERSION E01306.1 GI:2169565
KEYWORDS JP 1987190088-A/1.
SOURCE synthetic construct
ORGANISM synthetic construct, other sequences, artificial sequences.
REFERENCE 1 (bases 1 to 229)
AUTHORS Raasu,A., Toomasu,M., Berun,N. and Majiasu,U.
JOURNAL Patent: JP 1987190088-A 1 20-AUG-1987;
```

...

## ORIGIN

```
1   gaattctaac ggtcccgaaa ctctgtgcgg tgctgaactg gttgacgctc tgcagtttgt
61  ttgcggtgac cgtgggtttt attttaacaa acccactggg tatgggttctt cttctcgtcg
121 tgctccccag actggtattg ttgacgaatg ctgctttcgt tcttgcgacc tgcgtcgtct
181 ggaaatgtat tgcgctcccc tgaaaccgcg taaatctgct tagaagctt
```

//

# Opis sekwencji DNA i RNA

International Union of Pure and Applied Chemistry (IUPAC)

A	adenine	R	G A (purine)
C	cytosine	Y	T C (pyrimidine)
G	guanine	K	G T (keto)
T	thymine	M	A C (amino)
U	uracil	S	G C
		W	A T
B	G T C	N	A G C T (any)
D	G A T		
H	A C T		
V	G C A		

# Format danych FASTA, FASTQ

FASTA: nagłówek (1 linia), sekwencja (po 70 znaków)

```
>AB000263 |acc=AB000263|descr=Homo sapiens mRNA for ...|len=368  
ACAAGATGCCATTGTCCCCGGCCTCCTGCTGCTGCTCTCCGGGGCCACGGCCACCGCTGCCCTGCC  
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC  
CTCCTGACTTTTCCTCGCTTGTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGG  
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC  
CTGCAGGAACCTTCTTCTGGAAGACCTTCTCCTCCTGCAAATAAAACCTCACCCATGAATGCTCACGCAAG  
TTTAATTACAGACCTGAA
```

FASTQ: nagłówek (1 linia), sekwencja (po 70 znaków), jakość  
każdego symbolu

```
@SEQID description  
GGGTGATGGCCGCTGCCGATGGCGTCAAATCCCACC  
+SEQID description  
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII9IG9IC
```



# Format FASTQ, jakość

Jakość (Phred quality score) obliczana na podstawie prawdopodobieństwa, że symbol jest błędny ( $e$ ):

$$Q = -10 \log_{10} e$$

zapisywana jako znak ASCII o kodzie  $X = Q + 64$ , kod 64 to @

Q	e	dokładność	znak ASCII
10	0.1	90%	<b>J</b>
20	0.01	99%	<b>T</b>
30	0.001	99.9%	<b>^</b>
40	0.0001	99.99%	<b>h</b>
50	0.00001	99.999%	<b>r</b>
60	0.000001	99.9999%	<b> </b>

## Tabela kodonów

UUU	Fenylalanina	UCU	Seryna	UAU	Tyrozyna	UGU	Cysteina
UUC		UCC		UAC		UGC	
UUA	Leucyna	UCA	Seryna	UAA	STOP	UGA	STOP
UUG		UCG		UAG		UGG	Tryptofan
CUU		CCU		CAU	Histydyna	CGU	Arginina
CUC		CCC	Prolina	CAC		CGC	
CUA		CCA		CAA	Kwas glutaminowy	CGC	
CUG		CCG		CAG		CGG	
AUU	Izoleucyna	ACU	Treonina	AAU	Asparagina	AGU	Seryna
AUC		ACC		AAC		AGC	
AUA		ACA		AAA	Lizyna	AGA	Arginina
AUG	Metionina	ACG		AAG		AGG	
GUU	Walina	GCU	Alanina	GAU	Kwas asparaginowy	GGU	Glicyna
GUC		GCC		GAC		GGC	
GUA		GCA		GAA	Kwas glutaminowy	GGA	
GUG		GCG		GAG		GGG	

# Częstości kodonów dla danego organizmu

Każdy organizm preferuje któryś z kodonów do kodowania danego aminokwasu

<http://www.kazusa.or.jp/codon/>

```
CCTGGAGGGTGGCCCCACCGGCCGAGACAGCGAGCATATGCAGGAAGCGGCAGGAATAAGGAAAAGCAGC  
CTCCTGACTTTTCCTCGCTTGGTGGTTTGAGTGGACCTCCCAGGCCAGTGCCGGGCCCCCTCATAGGAGAGG  
AAGCTCGGGAGGTGGCCAGGCGGCAGGAAGGCGCACCCCCCAGCAATCCGCGCGCCGGGACAGAATGCC
```

- ▶ typowe metody kompresji - 2 bity na literę (dla DNA)
- ▶ wykorzystanie tabeli kodonów, zazwyczaj sekwencja koduje białko z wybranego organizmu - 1.6bit / literę
  - ▶ podział sekwencji DNA na ciąg kodonów
  - ▶ zastosowanie typowego algorytmu kompresji (np. Huffmana)

# *Dopasowanie podobnych sekwencji*

# Heurystyczne metody wyznaczania dopasowań

Algorytmy programowania dynamicznego: złożoność  $O(n^2)$

- ▶ niepraktyczne dla bardzo długich sekwencji (np. wyszukiwanie genu w genomie)
- ▶ mało wydajne przy przeszukiwaniu dużych zbiorów sekwencji (baz danych)

Algorytmy dla sekwencji podobnych:

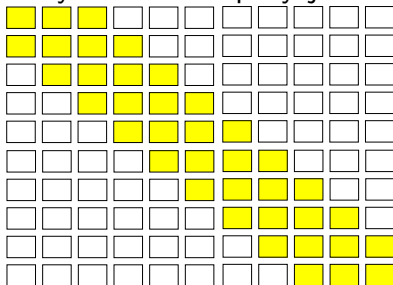
- ▶ pomijają niektóre przypadki
- ▶ złożoność  $O(n)$

Algorytmy heurystyczne:

- ▶ przybliżone badanie podobieństw
- ▶ bardzo wydajne

# Porównywanie podobnych sekwencji

- ▶ bez uwzględniania przerw: algorytm KMP (patrz wykład 1)
- ▶ z uwzględnieniem przerw, założenie – sekwencje nie różnią się maksymalnie na  $k$  pozycjach



- ▶ obliczamy wartości dla pasa  $\langle -k, k \rangle$  wokół przekątnej
- ▶ złożoność czasowa  $O(k*N)$

# Heurystyczne metody wyznaczania dopasowań

Typowa strategii wyszukiwania informacji

1. zawęzić zbiór interesujących sekwencji za pomocą algorytmu heurystycznego
2. sprawdzić te sekwencje dokładnym algorytmem programowania dynamicznego

Algorytmy heurystyczne:

- ▶ FASTA (FAST Alignment)
- ▶ BLAST (Basic Local Alignment Search Tool)

	FASTA	BLAST
czas obliczeń	dłuższy	krótszy
przerwy	uwzględnia	nie uwzględnia
rodzaj sekwencji	DNA, RNA	białka
czułość	bardziej czuły	mniej czuły

# Algorytm FASTA, Pearson i Lipman, 1988

Heurystyka - najlepsze połączenie zawiera identyczne fragmenty

Etapy:

1. inicjacja: wyszukuje fragmenty o długości *ktup* ( $= 6$  dla DNA) identycznych z fragmentami poszukiwanej sekwencji (nazywane 'gorącymi miejscami') wykorzystując tablicę indeksującą do wybrania rekordów w bazie
2. ciągi diagonalne: znajduje 'gorące miejsca' położone blisko siebie
3. łączenie: bada, które fragmenty ciągów diagonalnych mogą być ze sobą połączone
4. algorytm Smitha-Watermana: dla sekwencji zajmujących czołowe miejsca dopasowanie metodą programowania dynamicznego



# FASTA - krok 1, znajdowanie 'gorących miejsc'

Przykład  $ktup = 2$ , alfabet  $\{ A, C, T \}$

tablica indeksująca dla słowa AACACTTTTCAAT

AA	AC	AT	CA	CC	CT	TA	TC	TT
1	2	12	3		5		9	6
11	4		10					7
								8

założmy, że poszukujemy słowa ACTTATCA

AA	AC	AT	CA	CC	CT	TA	TC	TT
	1	5	7		2	4	6	3

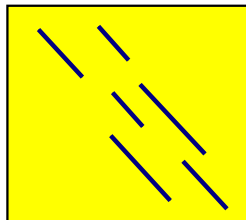
ocena:

AA	AC	AT	CA	CC	CT	TA	TC	TT
	1	7	-4		3		3	3
	3		3					4
								5

Dla tej sekwencji mamy **5 trafień** dla offsetu = 3

# FASTA - krok 2, ciągi diagonalne - ocena

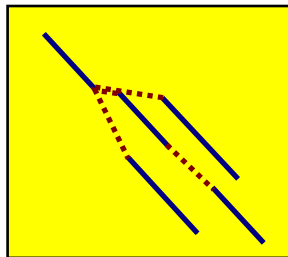
- ▶ 'gorące miejsce': dopasowanie dwóch sekwencji
  - ▶ maksymalny numer zgodnych słów *ktup*
  - ▶ minimalna odległość pomiędzy offsetami dla zgodnych słów
- ▶ łączy blisko położone 'gorące miejsca'
- ▶ stosuje się macierz substytucji do oceny fragmentów (PAM, BLOSUM)
  - ▶ gorące miejsca - wartości dodatnie
  - ▶ przerwy - wartości ujemne
- ▶ znajduje 10 ciągów diagonalnych o najwyższej ocenie



# FASTA - krok 3, łączenie

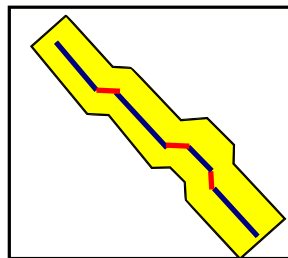
- ▶ buduje graf ciągów diagonalnych z fragmentów ciągów diagonalnych
  - ▶ wierzchołki: fragmenty, krawędzie: przerwy
  - ▶ fragmenty nie pokrywają się
  - ▶ rozważane fragmenty o najwyższych funkcjach oceny

zwracana ścieżka o najwyższej ocenie, ocena to suma ocen wierzchołków i krawędzi (policzonych wcześniej)



# FASTA - krok 4, ocena najlepszego rozwiązania

Algorytm Smitha-Watermana dla pasa  $\langle -k, +k \rangle$  wokół najlepszego rozwiązania



FASTA - zakończenie

- ▶ przegląda wszystkie rekordy w bazie danych dostarczając te, które mają najwyższą funkcję oceny
- ▶ dla wybranych rekordów pełny algorytm dopasowania

# Algorytm BLAST (Basic Local Alignment Search Tool)

Myers, Altschul, Gish, Lipman i Miller, 1990

Heurystyka - najlepsze połączenie zawiera identyczne fragmenty

Etapy:

1. inicjacja: podział analizowanej sekwencji na fragmenty o długości  $W$  ( $= 11$  dla DNA)
2. wyszukiwanie: wyszukiwanie w bazie danych rekordów, które zawierają te słowa o ocenie wyższej niż  $T$  (parametr algorytmu)
3. rozszerzanie
4. ocena: algorytm dokładny (Shmita-Watermana)

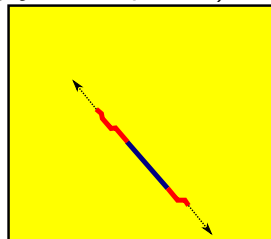
# BLAST - krok 1, 2, inicjacja, wyszukiwanie i ocena

- ▶ inicjacja: podział szukanej sekwencji na fragmenty (słowa)
  - ▶ przykład: założmy, że poszukujemy słowa ACTTATCA, wszystkie słowa o długości  $W=5$  dla sekwencji ACTTATCA: ACTTA, CTTAT, TTATC, TATCA
- ▶ wyszukiwanie: wybieranie z bazy rekordów posiadających dane słowo o ocenie (wartość uliniowienia bez spacji) większej niż  $T$ ,
- ▶ ocena: macierz BLOSUM lub PAM

# BLAST - krok 3, rozszerzanie

rozszerza znalezione dopasowania (bez uwzględniania przerw)

- ▶ rozszerzanie znalezionych obszarów (w lewo lub w prawo)
- ▶ nie uwzględnia przerw
- ▶ aż wartość dopasowania spadnie poniżej C



Parametry algorytmu BLAST:

W - długość słowa

T - wartość dopasowania przy wyszukiwaniu

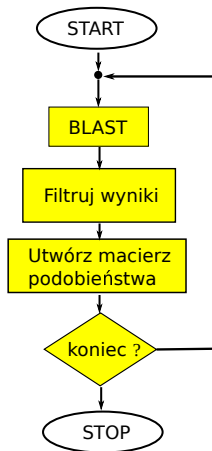
C - wartość dopasowania przy rozszerzaniu  
macierz podobieństwa

# Odmiany programu BLAST

- ▶ BLASTN - baza DNA za pomocą sekwencji DNA
- ▶ BLASTP - baza białek za pomocą sekwencji białek
- ▶ BLASTX - transluje sekwencję DNA we wszystkich 6 ramkach odczytu, uzyskane sekwencje białkowe są przeszukiwane w bazach białek
- ▶ TBLASTN - przeszukuje bazę DNA używając sekwencji białka, rekordy z bazy są analizowane we wszystkich 6 ramkach odczytu
- ▶ TBLASTX - sekwencja DNA translowana na sekwencję białka (6 ramek), baza DNA analizowana we wszystkich 6 ramkach odczytu



# PSI-BLAST (Position-Specific Iterated BLAST)



- ▶ większa czułość i specyficzność BLAST
- ▶ stosowane tam gdzie BLASTP (wyszukiwanie białek w bazie białek)
- ▶ w pierwszym kroku macierz BLOSUM, później tworzy macierz podobieństwa (PSSM, Position-specific scoring matrix)

# *Dopasowanie wielu sekwencji*

# Dopasowanie wielu sekwencji

Jednoczesna analiza wielu sekwencji w porównaniu z analizą poszczególnych par pozwala:

- ▶ łatwiej dostrzec istotne fragmenty
- ▶ łatwiej dostrzec podobieństwa i różnice

Przykład:

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

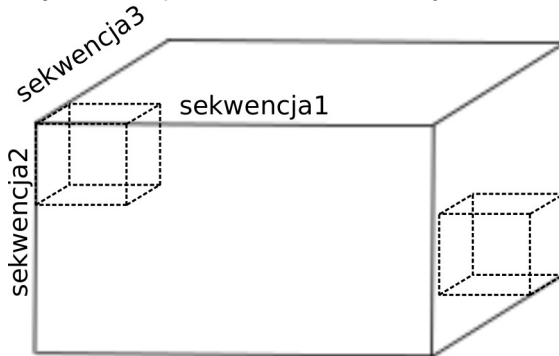
Problemy:

- ▶ algorytmy dokładne – zbyt wolne
- ▶ stosuje się algorytmy heurystyczne

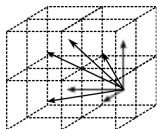
# Algorytm dokładny - programowanie dynamiczne

Dla  $m$  sekwencji długości  $N$  złożoność pamięciowa (i czasowa)  $O(N^m)$

Przykład: dopasowanie 3 sekwencji, macierz 3 wymiarowa

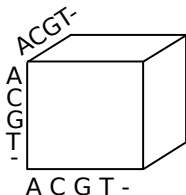


# Programowanie dynamiczne - dopasowanie 3 sekwencji



$$F(i, j, k) = \max \begin{cases} F(i-1, j-1, k-1) + e(s_i, t_j, u_k) \\ F(i-1, j-1, k) + e(s_i, t_j, -) \\ F(i-1, j, k-1) + e(s_i, -, u_k) \\ F(i-1, j, k) + e(s_i, -, -) \\ F(i, j-1, k-1) + e(-, t_j, u_k) \\ F(i, j-1, k) + e(-, t_j, -) \\ F(i, j, k-1) + e(-, -, u_k) \end{cases}$$

Macierz kar i nagród (uwzględnia przerwy)

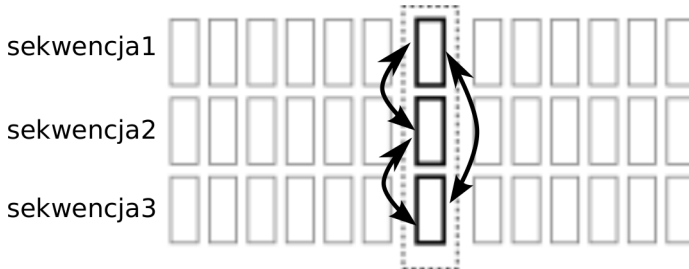


- uwzględnia przerwy
- 125 liczb (dla DNA), ale sporo symetrii

# Miara podobieństwa - suma różnych par

$$e(s_1, \dots, s_k) = \sum_{i=1..j:j=2..k} e(s_i, s_j)$$

gdzie  $s_i \in \{A, C, G, T, -\}$  oraz  $e(-, -) = 0$



Przykład - dla 3 sekwencji:

$$e(s_i, t_j, u_k) = e(s_i, t_j) + e(s_i, u_k) + e(t_j, u_k)$$

## Dopasowanie 3 sekwencji - przykład

	A	B	-
A	2	-1	-2
B	-1	2	-2
-	-2	-2	0

$$e(A,A,A) = e(B,B,B) = 6$$

$$e(A,A,B) = e(A,B,B) = 0$$

$$e(-,-,-) = 0$$

$$e(A,A,-) = e(B,B,-) = -2$$

$$e(A,-,-) = e(B,-,-) = -4$$

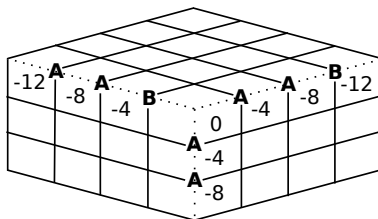
$$e(A,B,-) = -5$$

Sekwencje:

AAB,

AA,

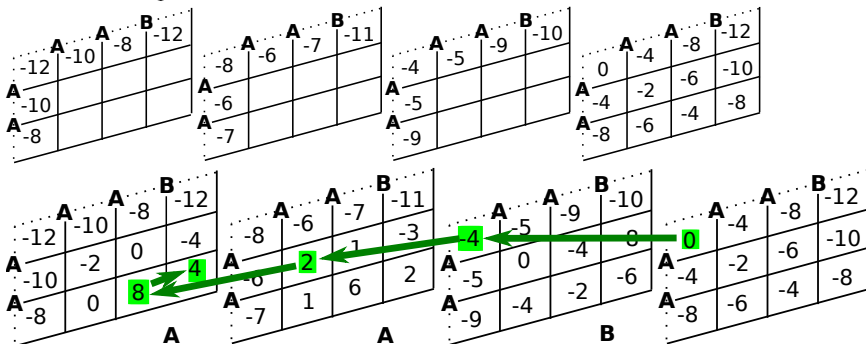
BAA



Rozwiązanie:

-	A	A	B
-	A	A	-
B	A	A	-

-	A	A	B
-	A	A	-
B	A	A	-





## Dopasowanie wielu sekwencji - profile

sekwencja 1	A	T	G	T	-	A	C	C	G	A
sekwencja 2	A	-	-	T	A	G	C	-	-	-
sekwencja 3	T	T	G	-	A	A	C	C	G	A
sekwencja 4	A	G	G	T	A	A	C	C	G	A
sekwencja 5	-	T	G	T	A	-	C	G	G	G

$$p_{-}(i, a) = \frac{\sum_{k=1}^n s_k(i) == a}{\sum_{k=1}^n s_k(i) \neq \{-'\}}$$

Profil:

A	0.75	0	0	0	1	0.75	0	0	0	0.75
C	0	0	0	0	0	0	1	0.75	0	0
G	0	0.25	1	0	0	0.25	0	0.25	1	0.25
T	0.25	0.75	0	1	0	0	0	0	0	0

# Dopasowania dla profili

Algorytmy programowania dynamicznego, ocena

$$e(s_i, t_j) = \sum_{a \in \Sigma} p(i, a) * e(a, t_j)$$

Złożoność:  $O(N * M * |\Sigma|)$

## Przykład

Sekwencje :

**ACG ACT**  
**ACG ACT**  
**ACG ACT**  
**ACG ACT**  
**ACG CCT**

Profil:

	1	2	3
A	0.9	0	0
C	0.1	1	0
G	0	0	0.5
T	0	0	0.5

## Dopasowanie sekwencji do profilu - przykład

Profil:

	1	2	3
A	0.9	0	0
C	0.1	1	0
G	0	0	0.5
T	0	0	0.5

	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

 $d = -5$ 

	G	A	A	T	
1	0	-5	-10	-15	-20
2	-5				
3	-10				
4	-15				

## Dopasowanie sekwencji do profilu - przykład

Profil:

	1	2	3
A	0.9	0	0
C	0.1	1	0
G	0	0	0.5
T	0	0	0.5

	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

 $d = -5$ 

	G	A	A	T	
1	0	-5	-10	-15	-20
2	-5	-1.4			
3	-10				
4	-15				

## Dopasowanie sekwencji do profilu - przykład

Profil:

	1	2	3
A	0.9	0	0
C	0.1	1	0
G	0	0	0.5
T	0	0	0.5

	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

 $d = -5$ 

	G	A	A	T	
1	0	-5	-10	-15	-20
2	-5	-1.4	3.7	-1.3	-18.6
3	-10	-6.4	-1.7	0.7	-1.3
4	-15	-8	-6.7	-3.7	3.2

## Dopasowanie sekwencji do profilu - przykład

Profil:

	1	2	3
A	0.9	0	0
C	0.1	1	0
G	0	0	0.5
T	0	0	0.5

	A	G	C	T
A	10	-1	-3	-4
G	-1	7	-5	-3
C	-3	-5	9	0
T	-4	-3	0	8

 $d = -5$ 

	G	A	A	T	
1	0	-5	-10	-15	-20
2	-5	-1.4	3.7	-1.3	-18.6
3	-10	-6.4	-1.7	0.7	-1.3
4	-15	-8	-6.7	-3.7	3.2

Rozwiązanie:    G    A    A    T  
                      -    1    2    3

# Dopasowanie wielu sekwencji - metoda progresywna

Heurystyczny algorytm dopasowania wielu sekwencji, stopniowe składanie przyrównania wykorzystując profile.

Etapy:

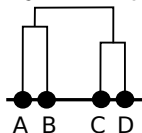
1. utworzenia macierzy odległości (badanie par sekwencji),
2. obliczanie drzewa naprowadzającego,
3. oddzielne przyrównanie najbliższych sekwencji (programowanie dynamiczne),
4. tworzenie sekwencji konsensusowej - profil.

# Dopasowanie wielu sekwencji - metoda progresywna (2)

- ▶ Obliczanie podobieństwa (wszystkie pary), np.

	A	B	C	D
A	-	30	20	20
B		-	10	10
C			-	40
D				-

- ▶ Najbliższa para **{C,D}**, później **{A,B}**



- ▶ obliczenie dopasowania **C** do **D**, utworzenie profilu
- ▶ obliczenie dopasowania **A** do **B**, utworzenie profilu
- ▶ obliczenie dopasowania profilu **CD** do **AB**

**Zastosowanie: program Clustal**



*Dziękuję*