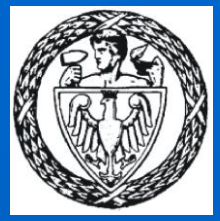
Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska



Problem dopasowywania (uliniawiania) dwóch bądź wielu sekwencji

Plan wykładu

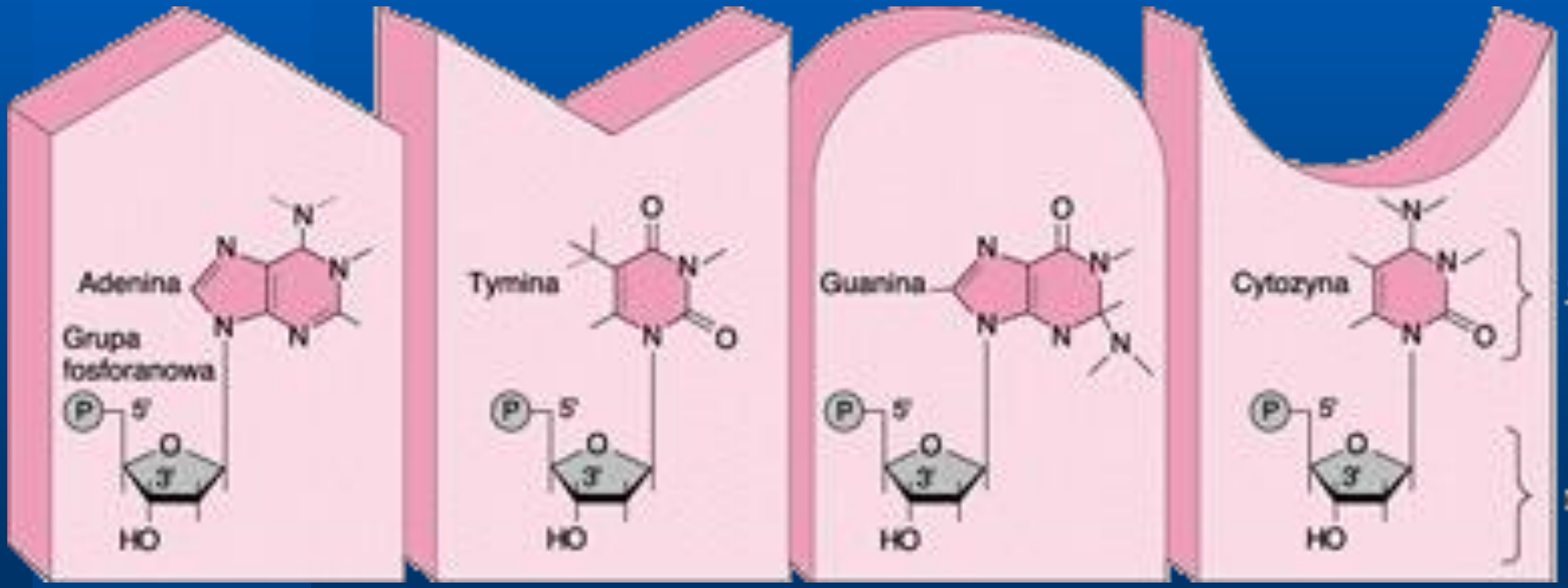
- **Zagadnienie dopasowywania dwóch sekwencji**
- **Algorytmy macierzowe**
  - **Needlemana-Wunscha**
  - **Hirschberga**
- **Problem dopasowywania (uliniowiania) wielu sekwencji**
- **Metoda ewolucyjno-progresywna**

Budowa sekwencji DNA

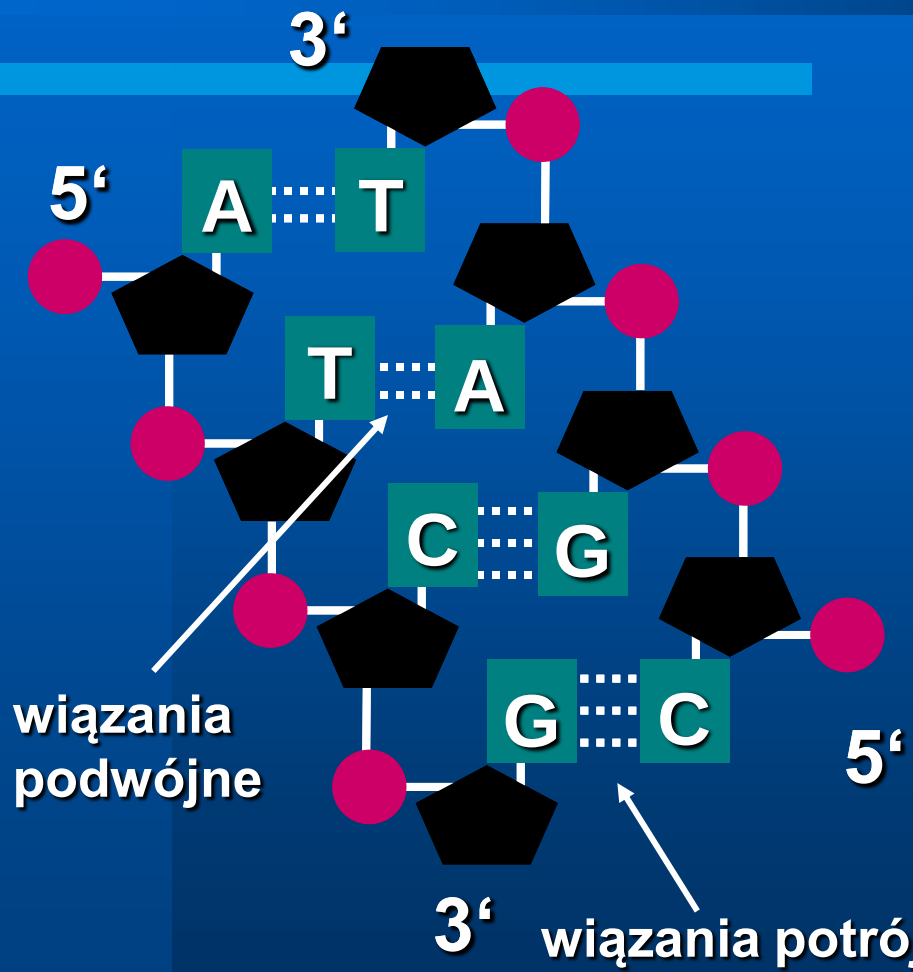
AGTTACGT

Cztery nukleotydy:

Adenina Tymina Guanina Cytozyna



Struktura DNA – podwójna nić

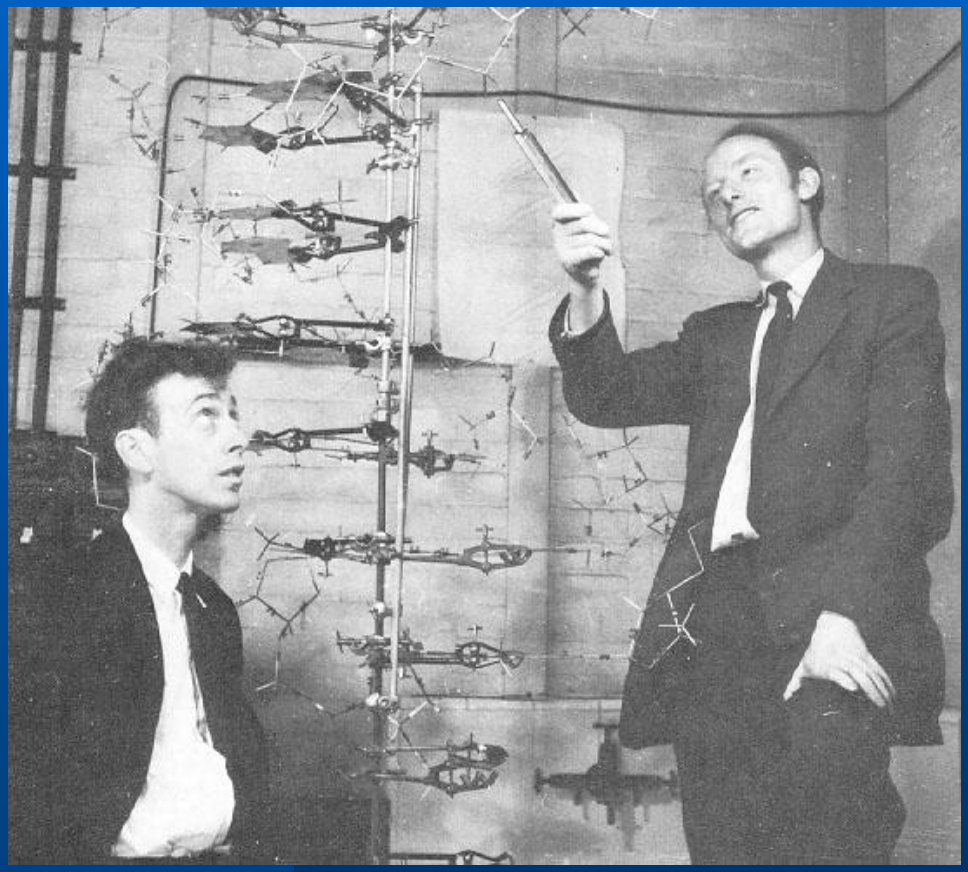


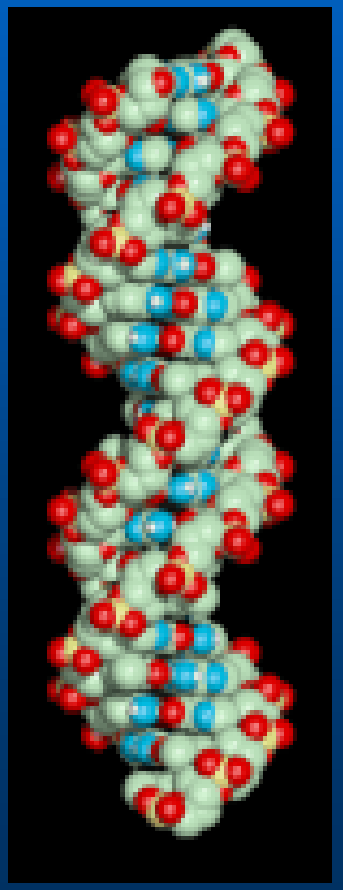
- A łączy się z T (powójnym wiązaniem słabym)
- C łączy się z G (wiązaniem potrójnym)

- Podwójna Helisa Watsona-Cricka
  - 5‘-ATCG-3‘
  - 3‘-TAGC-5‘

Podwójna Helisa

Odkrycie 1953, Nobel 1962 (medycyna)





+ Maurice Wilkins;

Rosalind Franklin (nie dożyła).

Problem dopasowywania dwóch sekwencji DNA lub aminokwasów

- **Problem polega na takim wzajemnym przyporządkowaniu dwóch ciągów, które maksymalizuje funkcję dopasowania.**

- **W funkcji dopasowania nagradzana jest zgodność symboli na odpowiadających sobie pozycjach, natomiast karana jest ich niezgodność oraz wstawianie odstępów.**

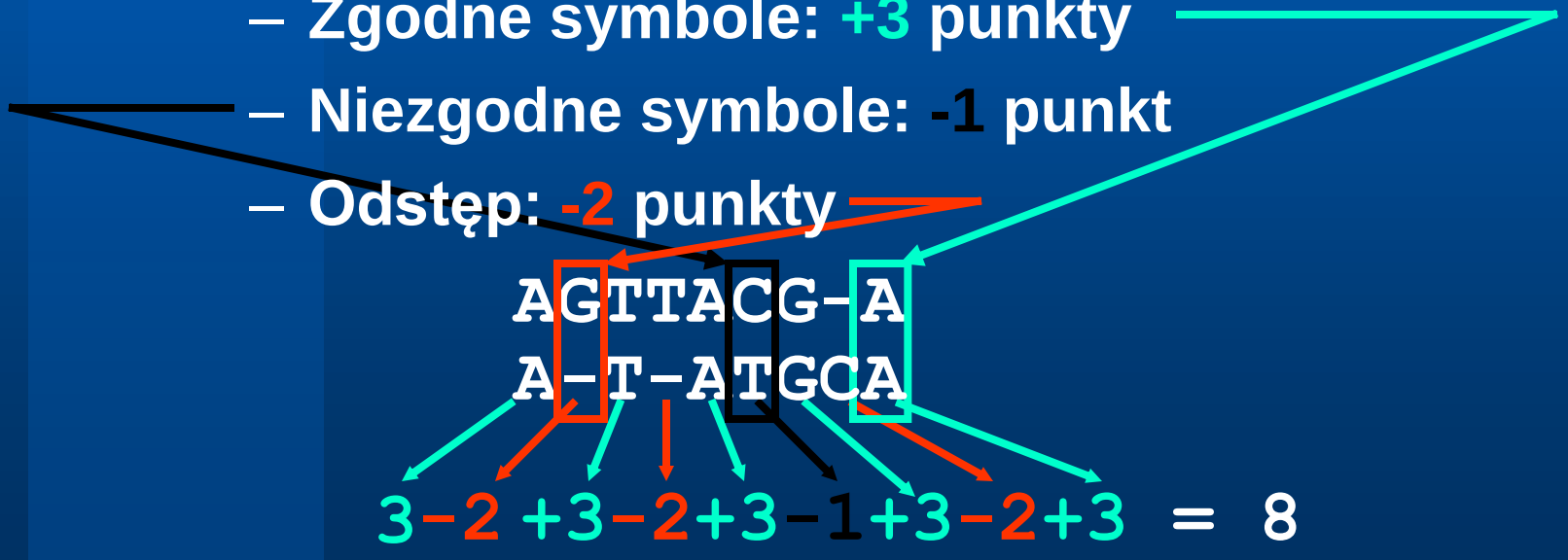
```
AGTTACGTT
ATATACTGCT
```

Warianty problemu dopasowania

- **Jednoczesne dopasowywanie *k > 2* sekwencji - koszt globalny albo lokalny**
- **Znajdowanie maksymalnego wspólnego podciągu dwóch lub większej liczby sekwencji**

Funkcja dopasowania

- **W praktyce stosowanych jest wiele różnych funkcji dopasowania.**
- **W przykładzie modelowym przyjmijmy:**
  - **Zgodne symbole: +3 punkty**
  - **Niezgodne symbole: -1 punkt**
  - **Odstęp: -2 punkty**



Macierz podobieństwa

	A	C	G	T
A	+3	-1	-1	-1
C	-1	+3	-1	-1
G	-1	-1	+3	-1
T	-1	-1	-1	+3

Warianty funkcji dopasowania

- **Afiniczna funkcja kary w przypadku ciągu odstępów:**
  - **f(k)=C + k * p**
  - **C - stały koszt “otwarcia” sekwencji odstępów,**
  - **p - koszt wprowadzenia pojedynczego odstępu**
- **Dla struktur białkowych - tabela kosztów przyporządkowania poszczególnych par symboli (*PAMn* – Point Accepted Mutation lub *BLOSUMn*) – dowolne pary spośród 20 protein**

Macierze PAM (Margaret Dayhoff, 1962)

- **PAM1 - sformułowano na podstawie ręcznego porównywania blisko spokrewnionych sekwencji - różnica w jednym aminokwasie na 100 → p-stwo podstawienia X→Y**

- **zakłada się, że żadna mutacja nie trafiła więcej niż raz w to samo miejsce**

- **Dwie sekwencje od wspólnego przodka**

G → C	G→A	G→A
G → G	G→C	G→A
1 pods/1 różn	2 podst/1różn	2 podst/0 różn

Macierze PAM (Margaret Dayhoff, 1962)

- **PAMn – ekstrapolacja modelu PAM1 po n-krokach**
- **Po pewnym czasie następuje wysycenie – mutacje nadpisują się ➔**

PAMn	1	30	80	110	200	250
% podob.	99	75	50	60	25	20

Macierze PAM (Margaret Dayhoff, 1962)

- **Macierze PAM wyższego rzędu kumulują wyniki porównywania blisko spokrewnionych sekwencji (ale także ewentualne błędy tych porównań!)**

Macierze BLOSUM (BLOcks SUbstitution Matrix) (Henikoff & Henikoff, 1991)

- **notacja BLOSUM, przeciwna do PAM**
- **BLOSUM80 odpowiada 80% identyczności**

PAM vs BLOSUM – którą wybrać?

- **Porównywanie nieznanych – często stosowana jest macierz**

  BLOSUM62

- **Porównywanie odległych (dywergentnych) - „duże PAM, małe BLOSUM” – np.**

  PAM250, BLOSUM30

- **Porównywanie bliskich (podobnych) – „małe PAM, duże BLOSUM” – np.**

  PAM15, BLOSUM80

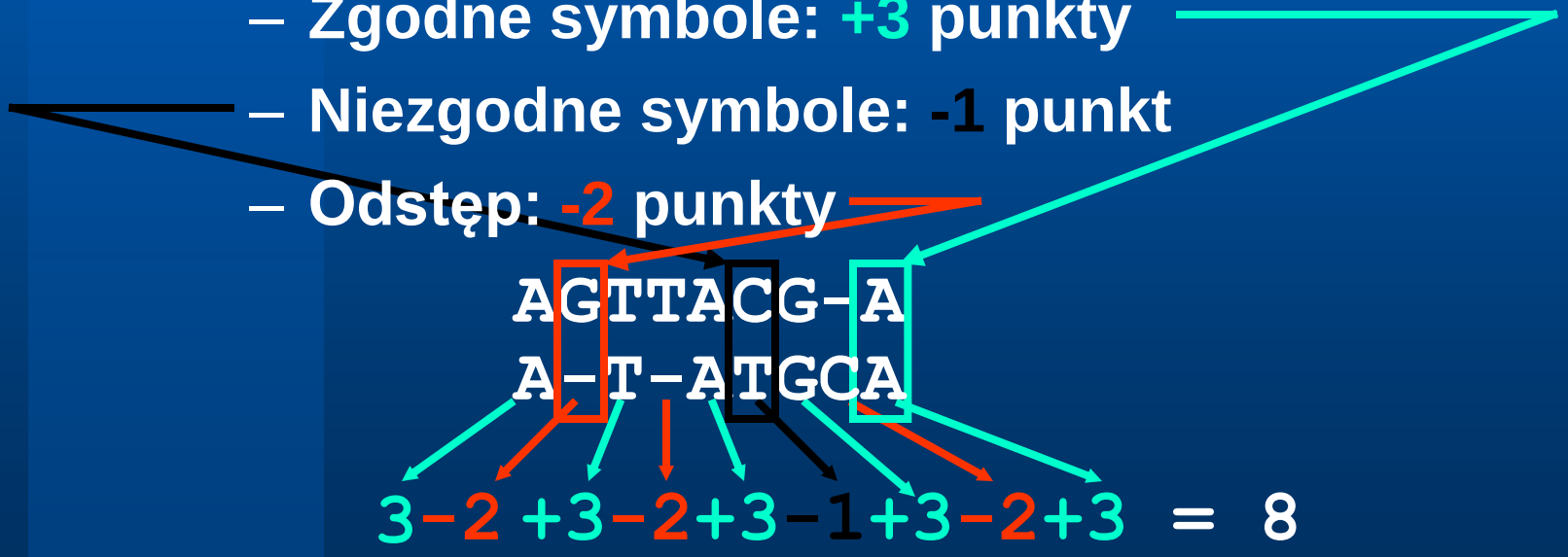
Przyporządkowanie sekwencji aminokwasów

BLOSUM62

```
A  4
R -1  5
N -2  0  6
D -2 -2  1  6
C  0 -3 -3 -3  9
Q -1  1  0  0 -3  5
E -1  0  0  2 -4  2  5
G  0 -2  0 -1 -3 -2 -2  6
H -2  0  1 -1 -3  0  0 -2  8
I -1 -3 -3 -3 -1 -3 -3 -4 -3  4
L -1 -2 -3 -4 -1 -2 -3 -4 -3  2  4
K -1  2  0 -1 -3  1  1 -2 -1 -3 -2  5
M -1 -1 -2 -3 -1  0 -2 -3 -2  1  2 -1  5
F -2 -3 -3 -3 -2 -3 -3 -3 -1  0  0 -3  0  6
P -1 -2 -2 -1 -3 -1 -1 -2 -2 -3 -3 -1 -2 -4  7
S  1 -1  1  0 -1  0  0  0 -1 -2 -2  0 -1 -2 -1  4
T  0 -1  0 -1 -1 -1 -1 -2 -2 -1 -1 -1 -1 -2 -1  1  5
W -3 -3 -4 -4 -2 -2 -3 -2 -2 -3 -2 -3 -1  1 -4 -3 -2 11
Y -2 -2 -2 -3 -2 -1 -2 -3  2 -1 -1 -2 -1  3 -3 -2 -2  2  7
V  0 -3 -3 -3 -1 -2 -2 -3 -3  3  1 -2  1 -1 -2 -2  0 -3 -1  4
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V
```

Funkcja dopasowania

- **W praktyce stosowanych jest wiele różnych funkcji dopasowania.**
- **W przykładzie modelowym przyjmijmy:**
  - **Zgodne symbole: +3 punkty**
  - **Niezgodne symbole: -1 punkt**
  - **Odstęp: -2 punkty**



Rozwiązania optymalne

- **Dla zadanej funkcji dopasowania może istnieć kilka rozwiązań optymalnych. W prezentowanym przykładzie występują trzy rozwiązania:**

```
AGTTACG-T
A-T-ATGCT      3-2+3-2+3-1+3-2+3 = 8

AGT-TACGT
A-TATGC-T      3-2+3-2+3-1+3-1+3 = 8

AGTTACG-T
A--TATGCT      3-2-2+3+3-1+3-2+3 = 8
```

Dwie grupy algorytmów

- **Algorytmy dokładne - optymalne dopasowanie przy zadanej funkcji kosztu.**

- **Algorytmy heurystyczne - rozwiązania suboptymalne**
  - **BLAST (Basic Local Alignment Search Tool - lista lokalnie dopasowanych podciągów)**
  - **MUSCLE**
  - **FASTA**

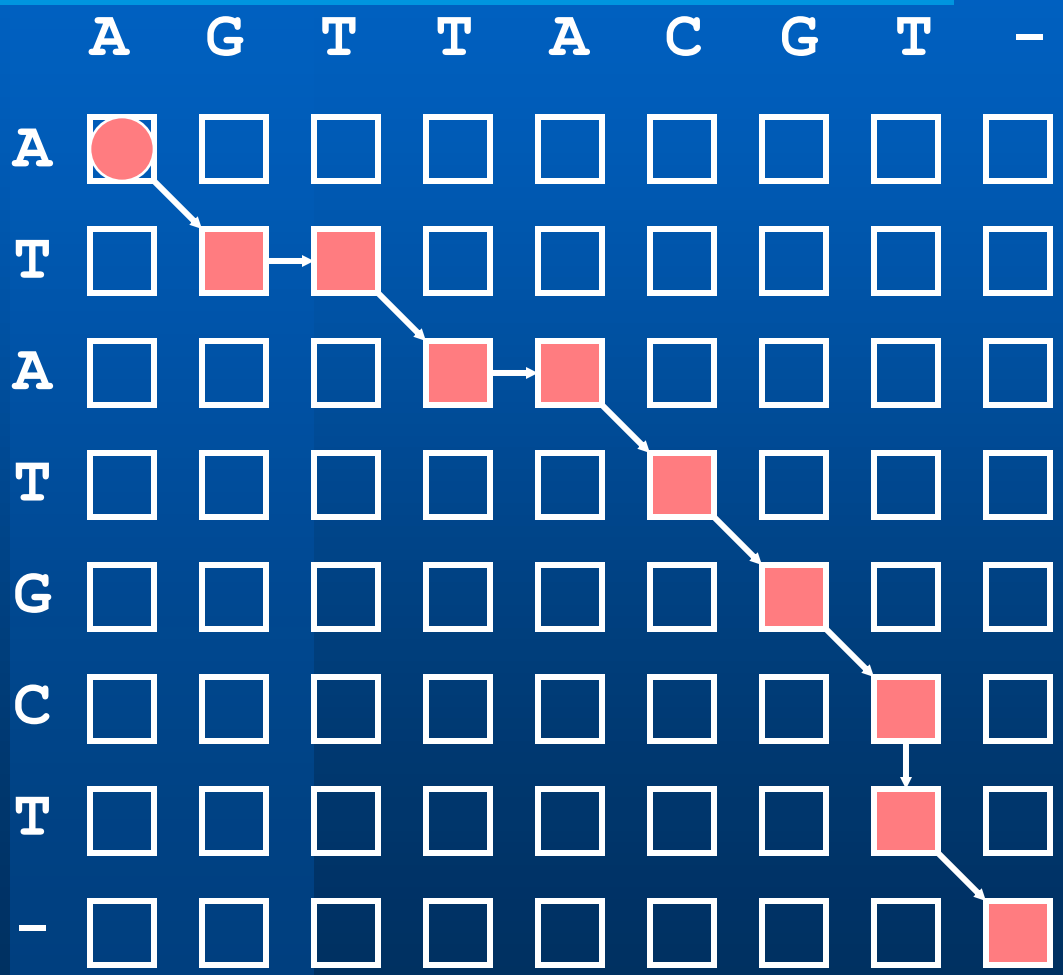
Podstawowe algorytmy dokładne

- **Dwa podstawowe algorytmy dokładne:**
  - **Algorytm Needlemana-Wuncha wykorzystujący pełną reprezentację macierzową**
  - **Algorytm Hirschberga o znacznie ograniczonych wymaganiach pamięciowych**

Reprezentacja problemu

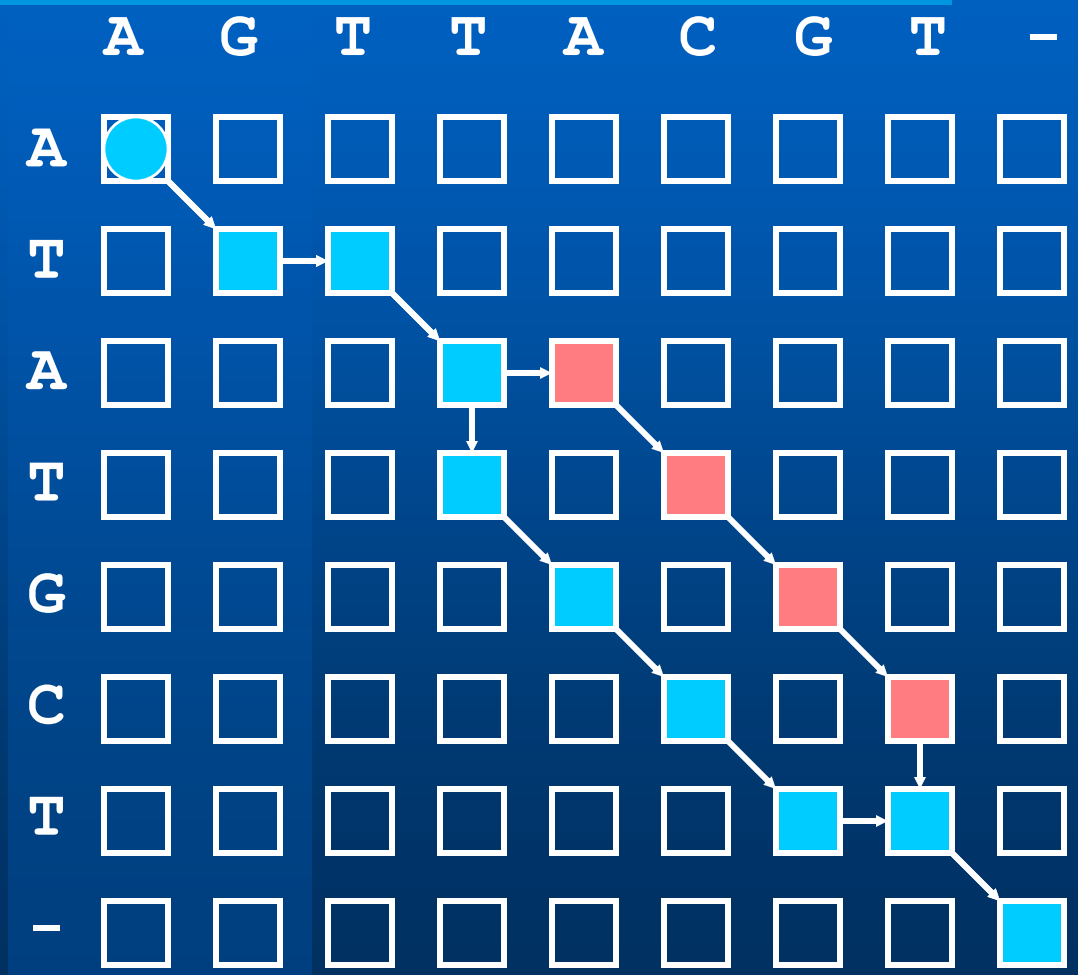
- **Wszystkie potencjalne dopasowania dwóch sekwencji reprezentowane sa w postaci ścieżek w odpowiedniej macierzy.**
- **Macierz zaetykietowana jest kolejnymi symbolami występującymi w rozważanych sekwencjach.**
- **Każda scieżka od lewego-górnego do prawego-dolnego elementu macierzy reprezentuje syntaktycznie poprawne (dopuszczalne) rozwiązanie.**

Rozwiązania jako ścieżki



A	G	T	T	A	C	G	–	T
A	–	T	–	A	T	G	C	T

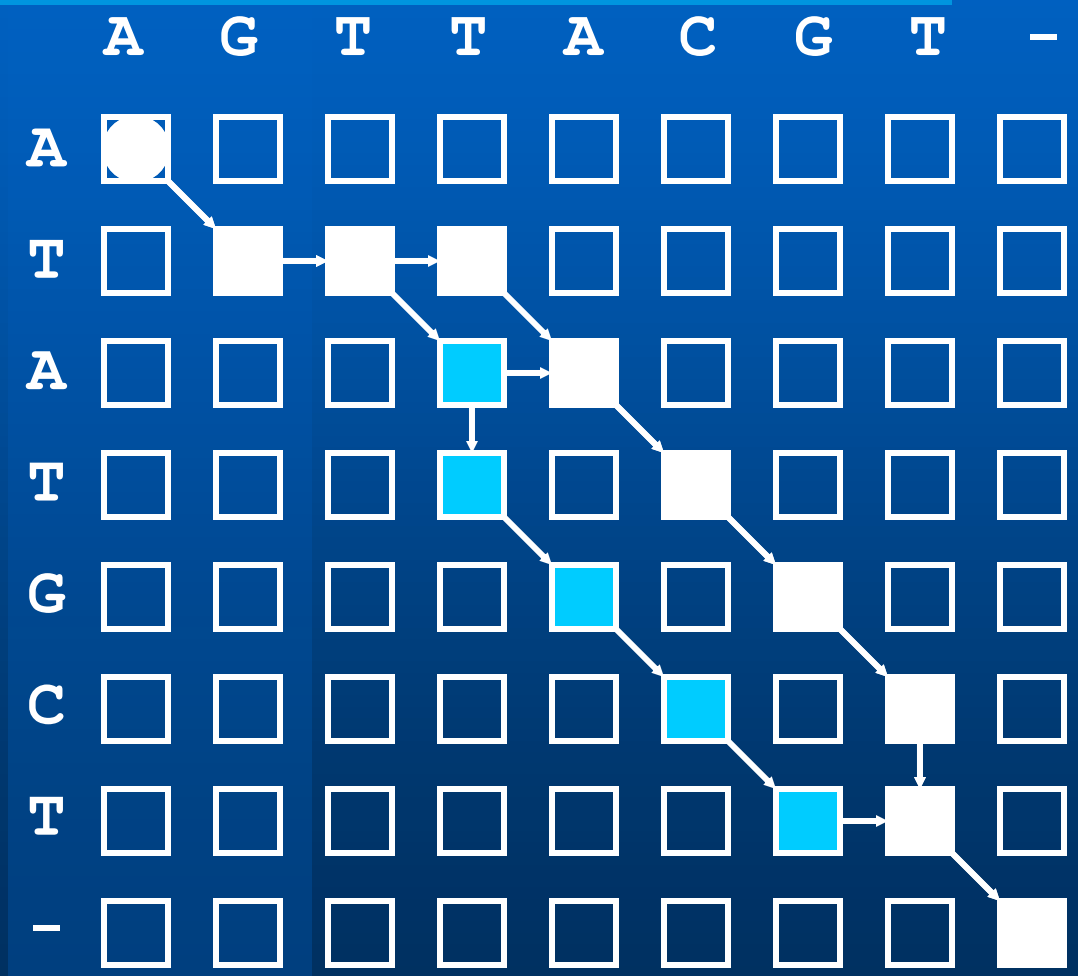
Pozostałe dwa rozwiązania



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

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

Pozostałe dwa rozwiązania



A	G	T	T	A	C	G	–	T
A	–	T	–	A	T	G	C	T

A	G	T	–	T	A	C	G	T
A	–	T	A	T	G	C	–	T

A	G	T	T	A	C	G	–	T
A	–	–	T	A	T	G	C	T

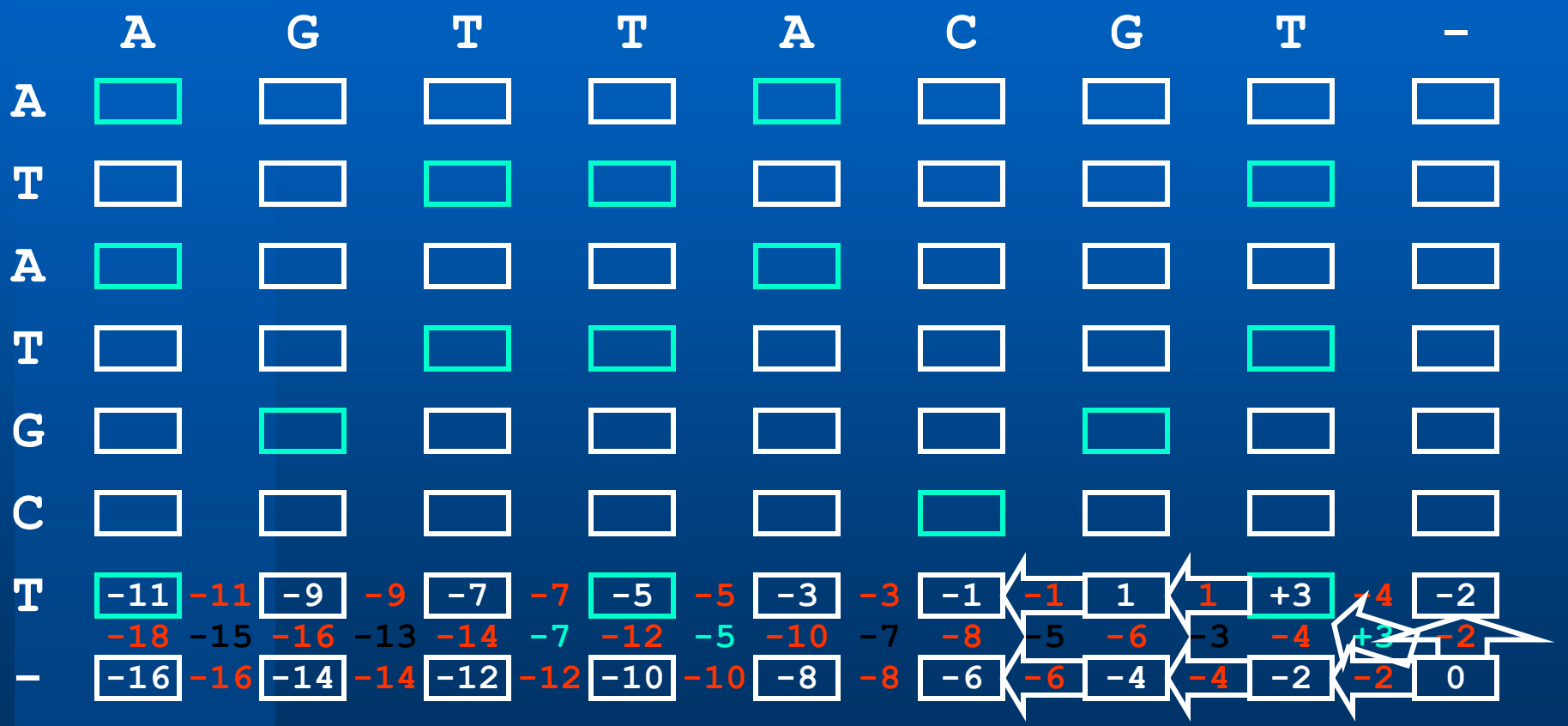
Schemat algorytmu macierzowego

- **Algorytmy rozwiązujące problem dopasowania składają się z dwóch faz:**
  - **oblicz_koszt**
  - **znajdź_ścieżkę**

Faza oblicz_koszt

- **W fazie oblicz_koszt znajdowane są koszty wszystkich ścieżek od prawego-dolnego do lewego-górnego elementu macierzy.**
- **Wartości propagowane są w macierzy z prawej do lewej i z dołu do góry.**
- **Po skompletowaniu całej macierzy koszt dopasowania optymalnego odczytywany jest w lewym-górnym elemencie.**

Faza oblicz_koszt



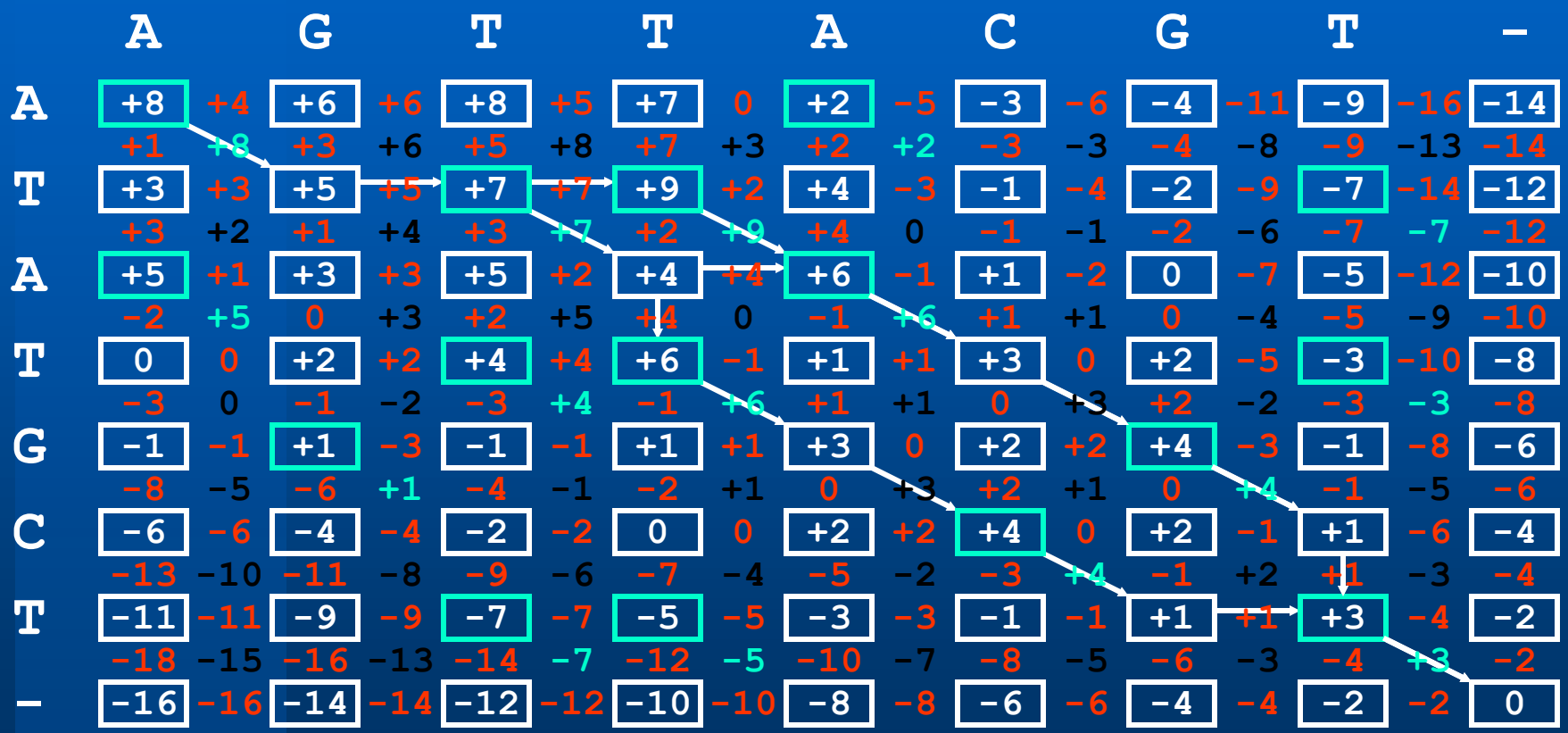
Faza oblicz_koszt

	A		G		T		T		A		C		G		T		–
A	+8	4	+6	+6	+8	+5	+7	0	+2	-5	-3	-6	-4	-11	-9	-16	-14
	+1	+8	+3	+6	+5	+8	+7	+3	+2	+2	-3	-3	-4	-8	-9	-13	-14
T	+3	+3	+5	+5	+7	+7	+9	+2	+4	-3	-1	-4	-2	-9	-7	-14	-12
	+3	+2	+1	+4	+3	+7	+2	+9	+4	0	-1	-1	-2	-6	-7	-7	-12
A	+5	+1	+3	+3	+5	+2	+4	+4	+6	-1	+1	-2	0	-7	-5	-12	-10
	-2	+5	0	3	+2	+5	+4	0	-1	+6	+1	+1	0	-4	-5	-9	-10
T	0	0	+2	+2	+4	+4	+6	-1	+1	+1	+3	0	+2	-5	-3	-10	-8
	-3	0	-1	-2	-3	+4	-1	+6	+1	+1	0	+3	+2	-2	-3	-3	-8
G	-1	-1	+1	-3	-1	-1	+1	+1	+3	0	+2	+2	+4	-3	-1	-8	-6
	-8	-5	-6	+1	-4	-1	-2	+1	0	+3	+2	+1	0	+4	-1	-5	-6
C	-6	-6	-4	-4	-2	-2	0	0	+2	+2	+4	0	+2	-1	+1	-6	-4
	-13	-10	-11	-8	-9	-6	-7	-4	-5	-2	-3	+4	-1	+2	+1	-3	-4
T	-11	-11	-9	-9	-7	-7	-5	-5	-3	-3	-1	-1	+1	+1	+3	-4	-2
	-18	-15	-16	-13	-14	-7	-12	-5	-10	-7	-8	-5	-6	-3	-4	+3	-2
–	-16	-16	-14	-14	-12	-12	-10	-10	-8	-8	-6	-6	-4	-4	-2	-2	0

Faza znajdź_ścieżkę

- W fazie znajdź_ścieżkę przeszukiwanie zaczyna się od lewego-górnego elementu.
- W każdym z elementów wybierany jest kierunek kontynuacji: w dół, w prawo, bądź po skosie w oparciu o maksymalny wkład punktowy do tego elementu (z rozważanych kierunków).
- Jeżeli wkład punktowy z dwóch (bądź trzech) kierunków jest jednakowy, to oba (trzy) kierunki generują optymalne ścieżki.

Faza znajdź_ścieżkę



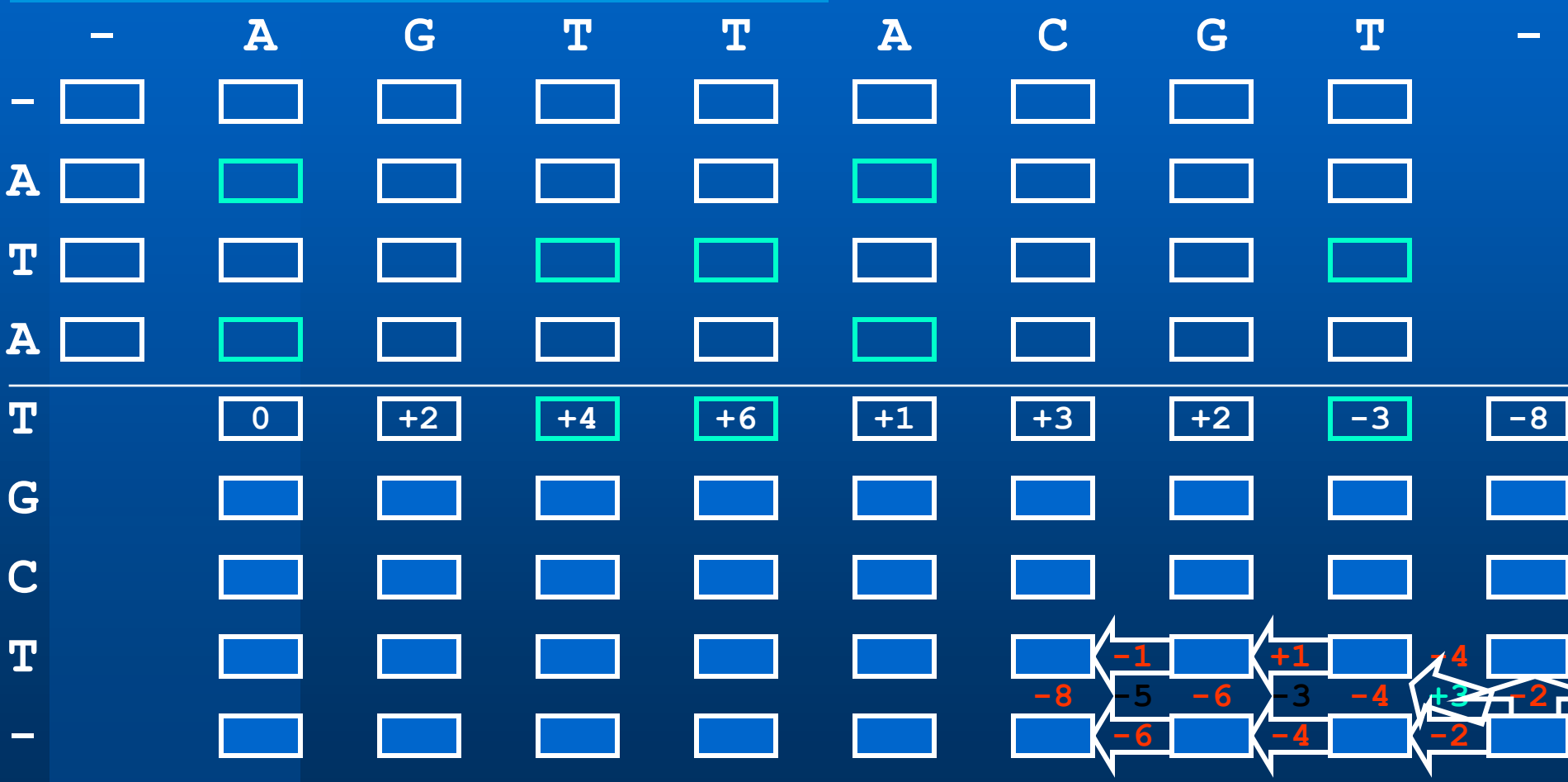
Koszt algorytmu

- **Algorytm w obu fazach (oblicz_koszt, znajdź_ścieżkę) wymaga jednoczesnego przechowywania w pamięci całej macierzy.**
- **Dla sekwencji o długościach *n* i *m* przechowywanych jest *n* x *m* elementów.**
- **Faza oblicz_koszt wymaga wykonania *n* x *m* operacji.**
- **Faza znajdź_ścieżkę wymaga wykonania około *m* + *n* operacji.**

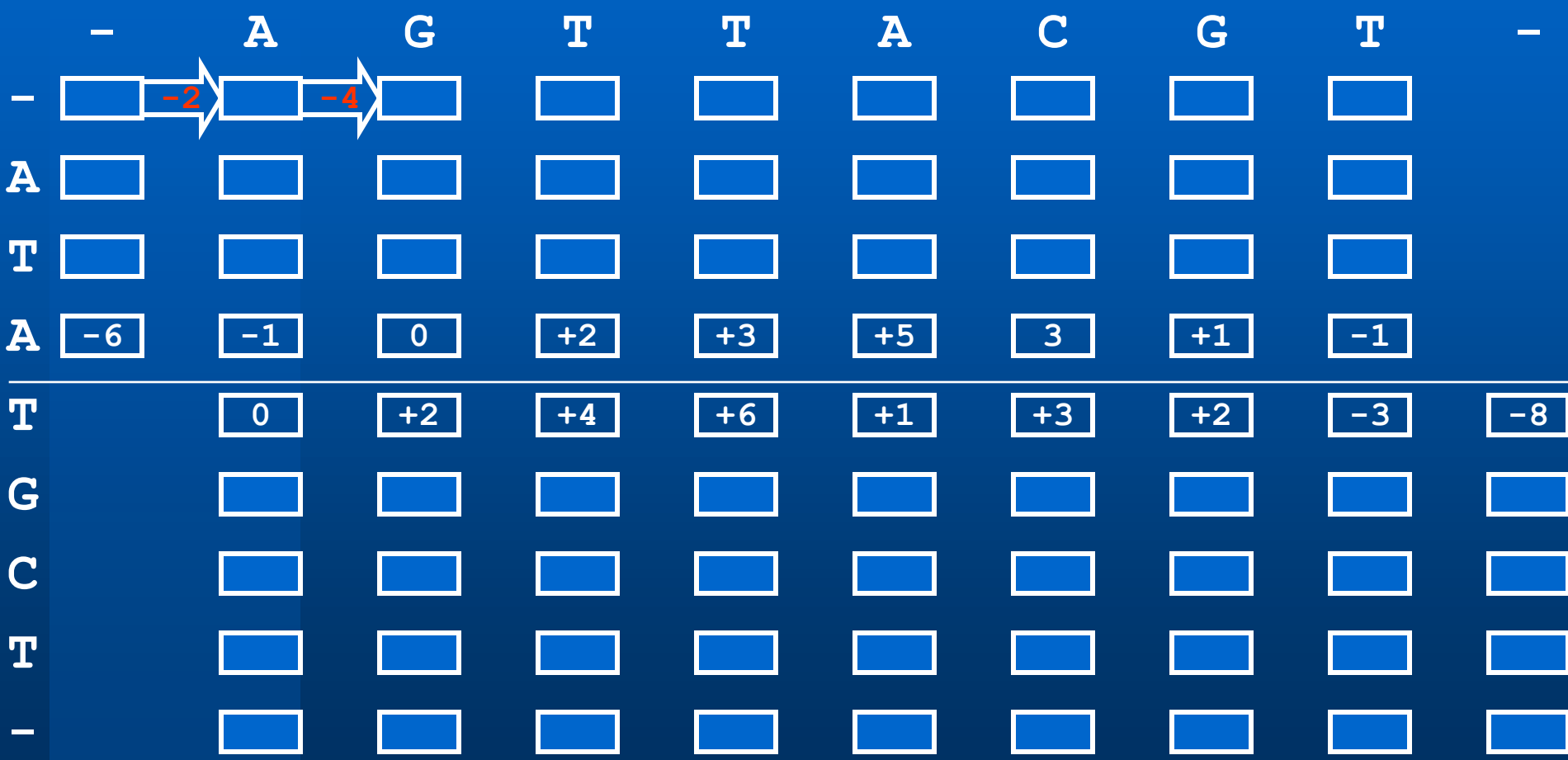
Algorytm Hirschberga

- **Algorytm Hirschberga znajduje optymalne rozwiązania wykorzystując mniej pamięci.**
- **Idea polega na podzieleniu jednej z sekwencji na dwie (możliwie równe) części i przeprowadzeniu fazy oblicz_koszt w każdej części oddzielnie, przy czym na jednej z nich w kierunku odwrotnym.**
- **W pamięci przechowywany jest jedynie bieżący wiersz obliczeń z każdej podmacierzy (w sumie *2n* elementów).**

Faza oblicz_koszt



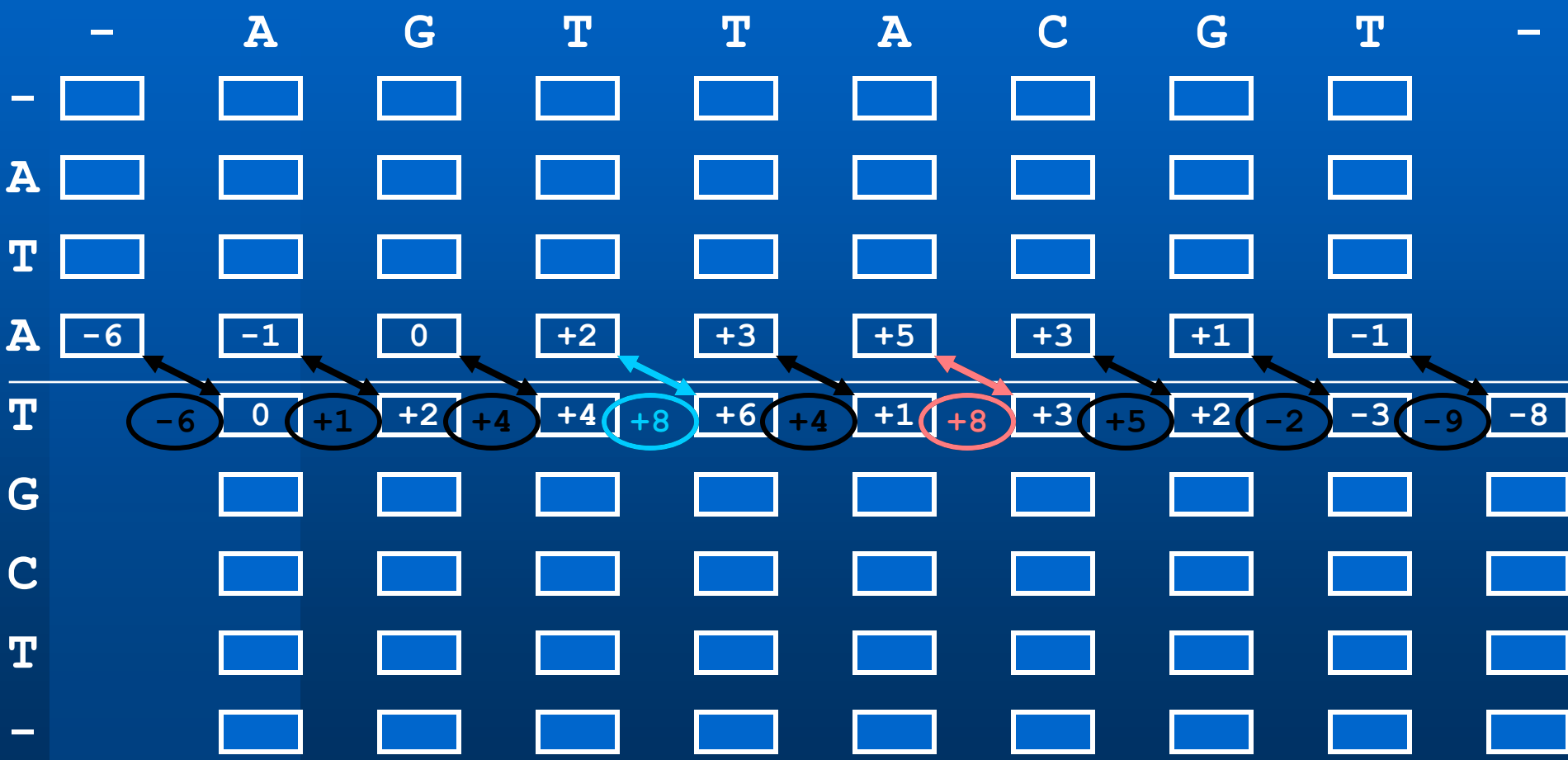
Faza oblicz_koszt



Algorytm Hirschberga

- **Po zakończeniu fazy oblicz_koszt, wiersze wynikowe w każdej z dwóch podmacierzy wykorzystywane są do znalezienia optymalnego punktu łączenia obu częściowych dopasowań.**

Faza oblicz_koszt



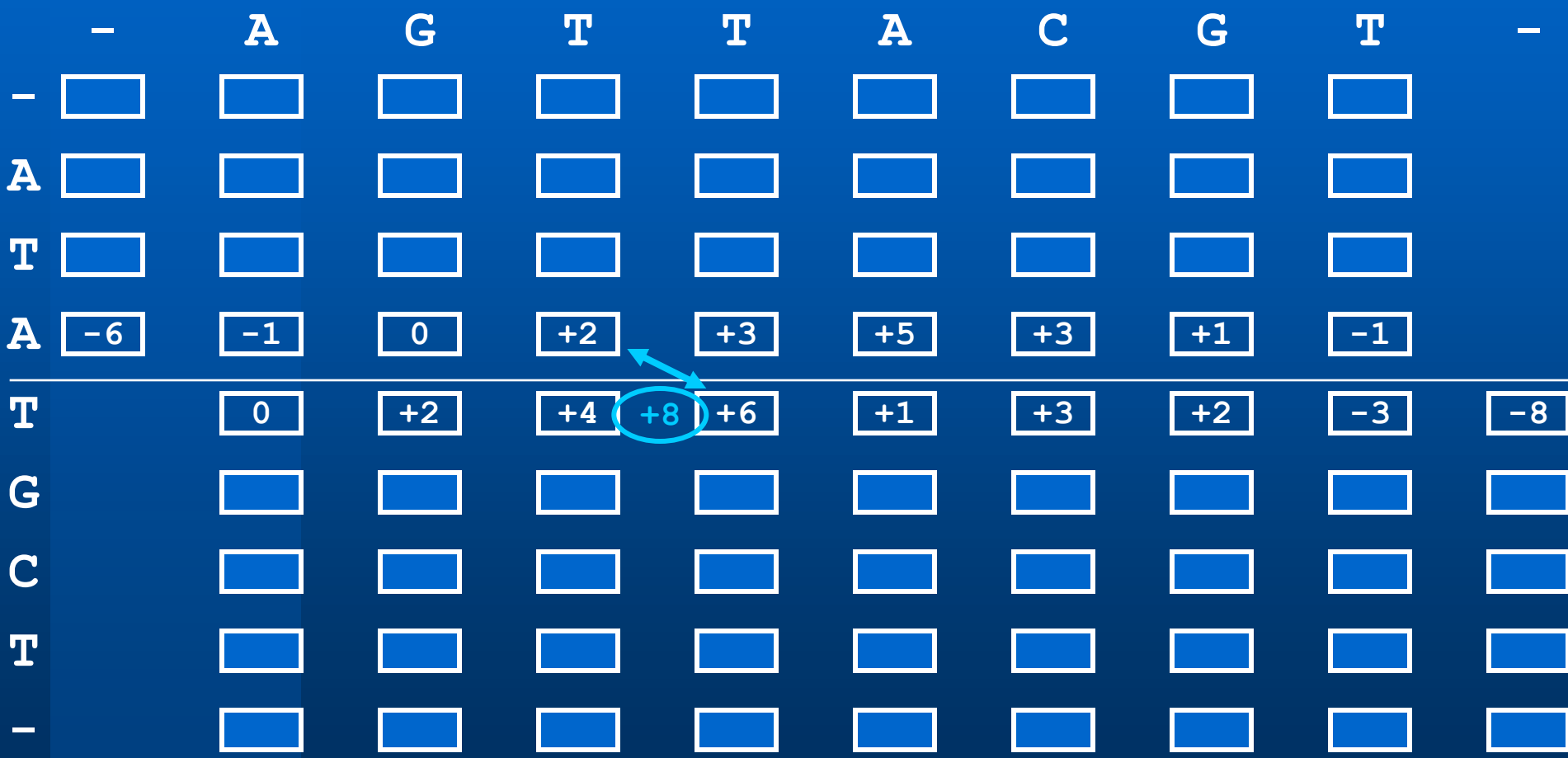
Podział na podproblemy

- Znaleziono dwa punkty optymalnego podziału: niebieski oraz różowy.
- Podział różowy odpowiada dwóm rozwiązaniom optymalnym: różowemu oraz *białemu*.

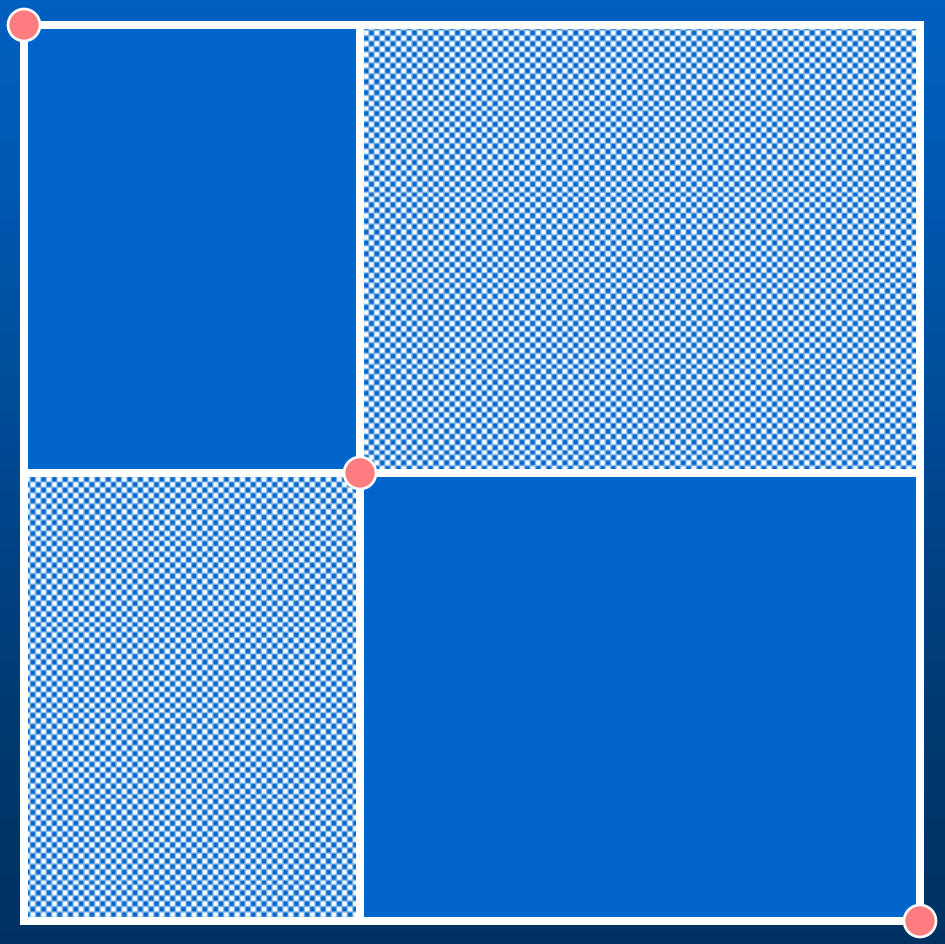
```
AGT    TAGCT        AGTTA   CGT
ATT    ATGCT        ATA-A   TGCT

                    AGTTA   CGT
                    ATATA   TGCT
```

Faza oblicz_koszt



Eliminacja połowy macierzy

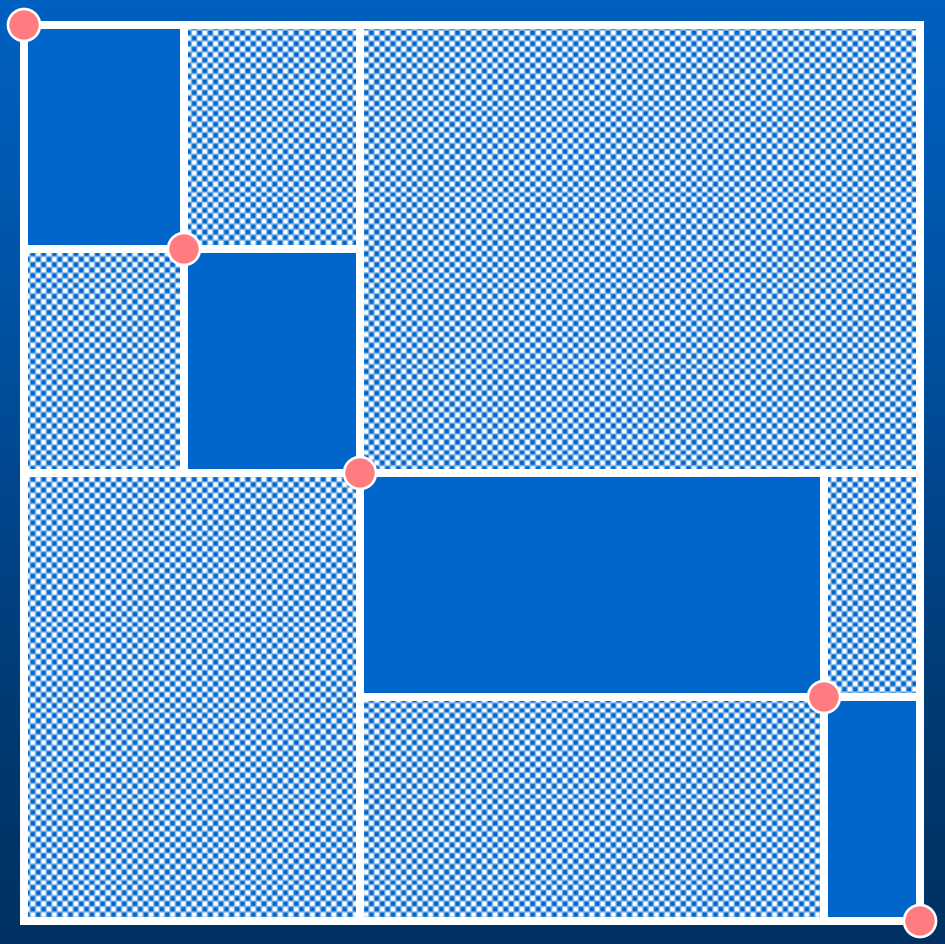


Algorytm Hirschberga

- **Następnia algorytm wywoływany jest ponownie dla każdej pary powstałej z pierwotnych “półsekwencji”.**

- **Rekurencyjne wywołanie algorytmu kończy się w przypadku, gdy długość porównywanych sekwencji jest równa 1.**

Schemat rekursji Hirschberga



Koszt algorytmu

- **Algorytm Hirschberga wymaga jedynie pamięci liniowej - *2* x *n*.**

- **Kosztem ok. dwukrotne zwiększenie wymagań czasowych (do *2* x *m* x *n*) w związku z koniecznością powtórnego obliczania części elementów macierzy.**

Podsumowanie: złożoność algorytmów

	N-W	H
pamięć	n^2	**2n**
czas oblicz_koszt	n^2	**$2n^2$**
czas znajdź_ścieżkę	**2n**	

Tyle o algorytmach dokładnych