Algorytmy w bioinformatyce sekwencji

Algorytmy badania podobieństw 2 sekwencji

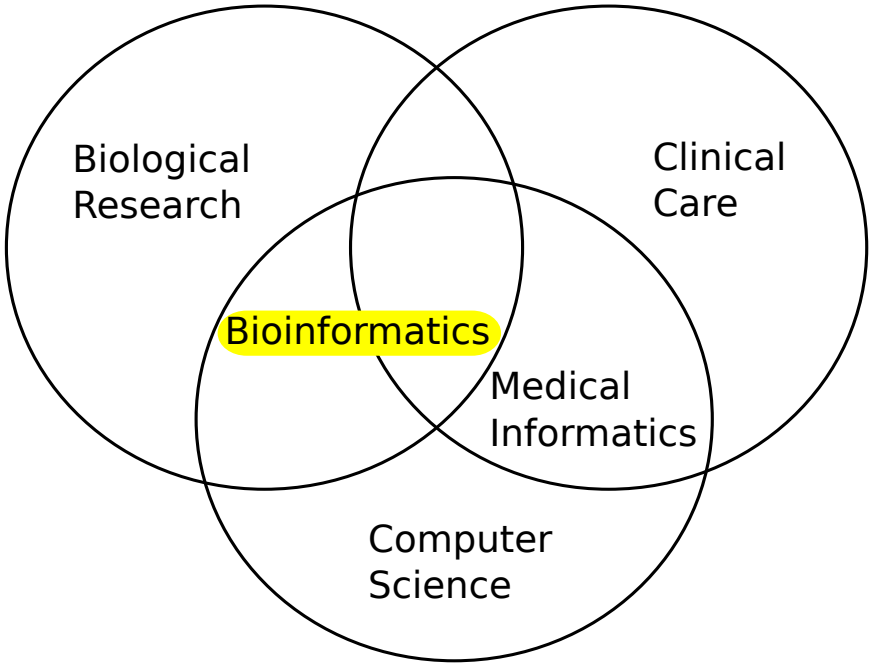
Robert Nowak

2024

Miejsce bioinformatyki wśród innych dziedzin wiedzy

Dziedzina na styku:

- matematyki
- informatyki
- statystyki
- biologii molekularnej
- medycyny

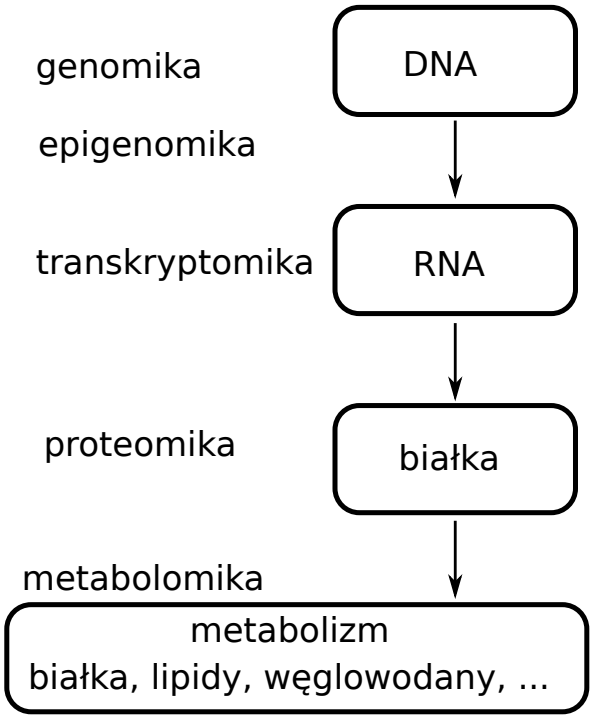


Główny cel: **zrozumieć procesy biologiczne**

Bioinformatyka - ważne daty

data	wydarzenie
1970	pierwszy raz użyto słowa 'bioinformatyka'
1970	algorytm uliniowienia 2 sekwencji
1971	PDB, Protein Data Bank
1978	algorytm PAM znajdowania macierzy podobieństwa
1982	GenBank
1990	algorytm BLAST
1997	algorytm PSI-BLAST
2003	sekwencja genomu człowieka
2010	pierwszy genom syntetyczny
2012	system CRISPR-Cas9, edycja DNA
2020	AlphaFold2 poprawnie przewiduje str. przestrzenną białka

Bioinformatyka - rodzaj analizowanych danych



- genomika - analiza genomu:
  - genomika strukturalna,
  - genomika funkcjonalna,
  - genomika porównawcza;
- epigenomika – analiza modyfikacji chromatyny;
- transkryptomika – analiza transkryptomów;
- proteomika – analiza białek;
- metabolomika – analiza przemian chemicznych zachodzących w organizmach;
- **metagenomika** – analiza materiału izolowanego z nisz ekologicznych.

Bioinformatyka - rodzaj narzędzi

- zarządzanie danymi (data management), np. bazy danych, ontologie;
- obliczenia (data computation), np. algorytmy do pozyskiwania danych, asemblacja;
- odkrywanie wiedzy z danych (data-mining, biological discoveres);
- modelowanie i symulacje (modeling/simulation).

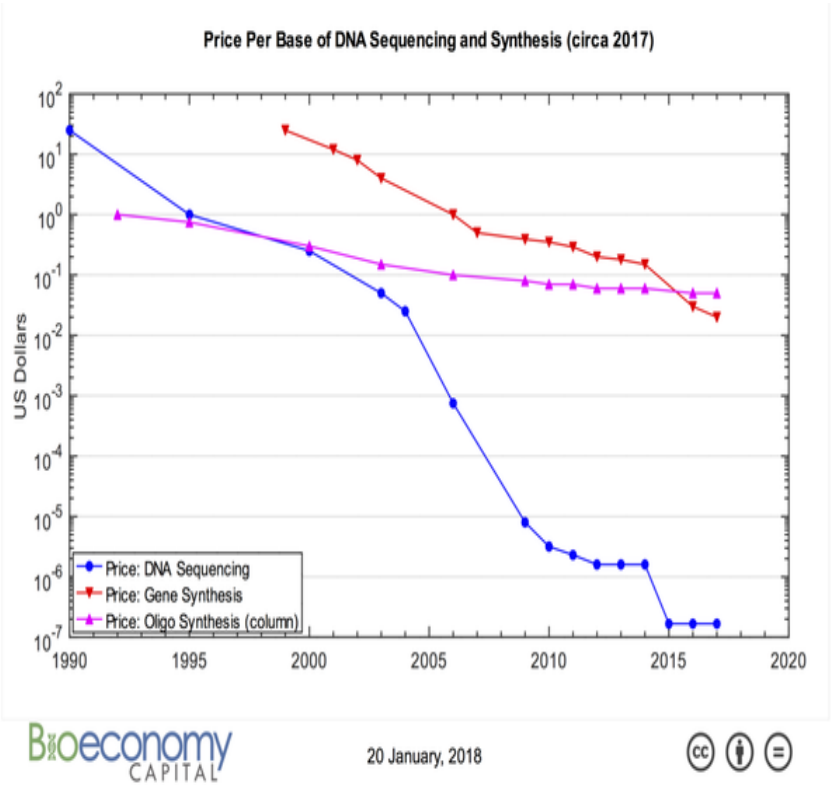
Dlaczego warto coś wiedzieć o bioinformatyce?

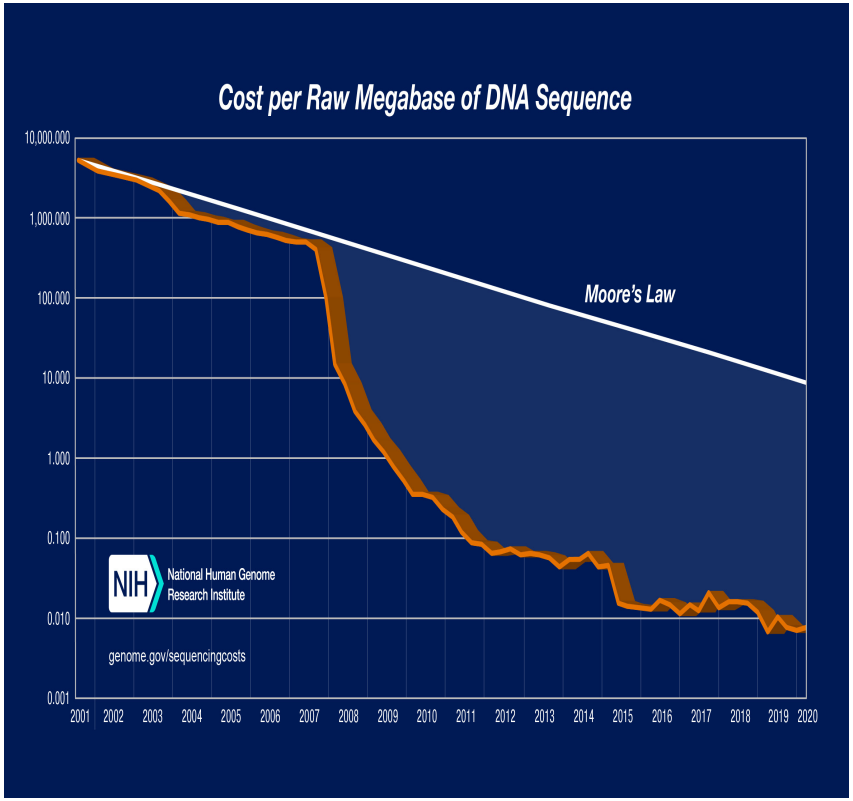
- wielkie nakłady na rozwój tej dziedziny
- potencjalnie duży obszar zastosowań (medycyna, farmacja, sądownictwo, ubezpieczenia)
- potrzeba wykorzystywania wyrafinowanych algorytmów

Specyfika aplikacji bioinformatycznych

- duże ilości danych
- złożone algorytmy
- wydajność jest niezwykle istotna
- szerokie wykorzystanie super-komputerów oraz obliczeń rozproszonych (np. grid-computiong)

Dlaczego bioinformatyka - koszty sekwencjonowania





- 10^5 Genomics England, 1+ Million Genomes in EU (by 2023), $\sim 10^6$ All of Us in America, $\sim 10^5$ Genome Asia
- jeden genom to plik 15GB
- dane genetyczne za kilka lat będą **drugim największym** zbiorem danych

Cel i zakres przedmiotu

Przedstawienie algorytmów stosowanych w bioinformatyce sekwencji

Zakładana znajomość:

- podstawowej wiedzy o DNA, RNA i białkach (transkrypcja, translacja, itp.)
- znajomość podstawowych algorytmów i struktur danych
- umiejętność programowania strukturalnego

Podstawowe dane o przedmiocie

Godziny wykładów: środy $10^{30} - 12^{00}$, online MS Teams

Strona przedmiotu:
`https://staff.elka.pw.edu.pl/~rnowak2/dyd/abs`

Prowadzący:

dr hab. inż. Robert Nowak, prof. uczelni,
`robert.nowak@pw.edu.pl`
dr inż. Wiktor Kuśmirek,
`wiktor.kusmirek@pw.edu.pl`

Tematyka wykładów

- bioinformatyka jako dziedzina informatyki;
- programowanie dynamiczne, uliniowienie dwu sekwencji;
- tworzenie macierzy podobieństwa;
- bazy danych sekwencji biologicznych, algorytm BLAST;
- badanie podobieństw wielu sekwencji;
- tworzenie drzew filogenetycznych;
- wyszukiwanie motywów w sekwencjach;
- assemblery DNA, algorytmy oparte o graf de Brujna;
- ukryte modele Markowa dla sekwencji;
- markery genetyczne, badanie pokrewieństw;
- badanie rozkładu haplotypów, algorytm EM;
- grupowanie i redukcja wymiarów, algorytm PCA;
- biologia syntetyczna, obliczanie struktur drugorzędowych.

Literatura

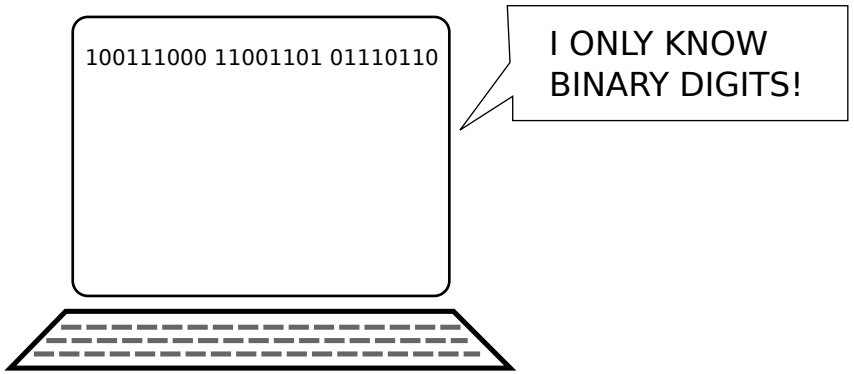
- Jin Xiong, Podstawy bioinformatyki, PWN, 2011
- P.Higgs, T.Attwood, Bioinformatyka i ewolucja molekularna, PWN, 2008
- V. Makinen, D. Belazzougui, F. Cunial, A. Tomescu, Genome-Scale Algorithm design, Cambridge 2015
- Wing-Kin Sung, Algorithms for next-generation sequencing, CRC Press 2017
- R.Durbin, S.Eddy, A.Krogh, G.Mithison, Biological sequence analysis. Cambridge 2007

Zaliczenie przedmiotu

- egzamin: 0 – 50pkt
- ćwiczenia: 0 – 50pkt

91 – 100 pkt.	ocena 5
81 – 90 pkt.	ocena $4\frac{1}{2}$
71 – 80 pkt.	ocena 4
61 – 70 pkt.	ocena $3\frac{1}{2}$
51 – 60 pkt.	ocena 3
0 – 50 pkt.	ocena 2

Reprezentacja cyfrowa



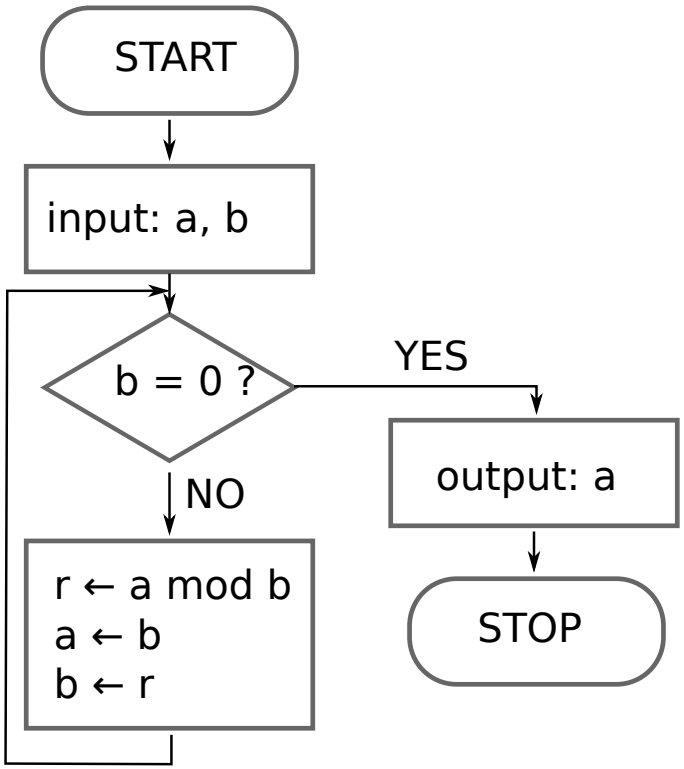
Komputer może przetwarzać tylko informację dostępną w formie cyfrowej

Poprawna reprezentacja informacji jest bardzo istotna.

1974×2024 to typowe zadanie dla ucznia szkoły podstawowej, ale: *MCMLXXIV* $\times$ *MMXXIV*	Reprezentacja liczb w kodzie U2: $\begin{array}{cccccc} & 0 & 1 & 0 & 0 & (+4) \\ + & 1 & 1 & 0 & 1 & (-3) \\ \hline & 0 & 0 & 0 & 1 & (+1) \end{array}$

Algorytm

Musimy dostarczyć algorytm, aby komputer mógł rozwiązać problem.




```
Euclid(a, b)                    // argumenty
begin
    while b > 0 do
        r ← a % b               // reszta
        a ← b
        b ← r
    end
    return b                    // result
end
```

Tworzenia algorytmów i programów

- Komputer potrzebuje programów (instrukcji)
- Tylko człowiek może stworzyć program, **silna Sztuczna Inteligencja (strong AI) nie istnieje!**

Programowanie:

- bardzo kosztowne, wieloletnia praca zespołowa,
- większość projektów wytwarzania kończy się porażką!

Wiele wysiłku wkłada się w wielokrotne wykorzystanie (reusability).

Reprezentacja cyfrowa biopolimerów

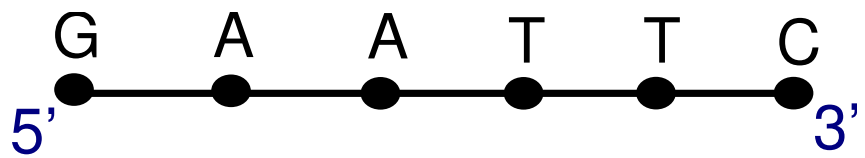
- struktura pierwszorzędowa – sekwencja symboli,
- struktura drugorzędowa – uwzględnienie oddziaływania nukleotydów (DNA, RNA) lub aminokwasów (białka),
- struktura trzeciorzędowa – położenie atomów w przestrzeni 3D.

Σ - alfabet, $\Sigma_{DNA} = \{A, C, G, T\}$

s - sekwencja nukleotydów $s = s_1 s_2 ... s_n$, gdzie

n - długość sekwencji s, oznaczana $|s|$,

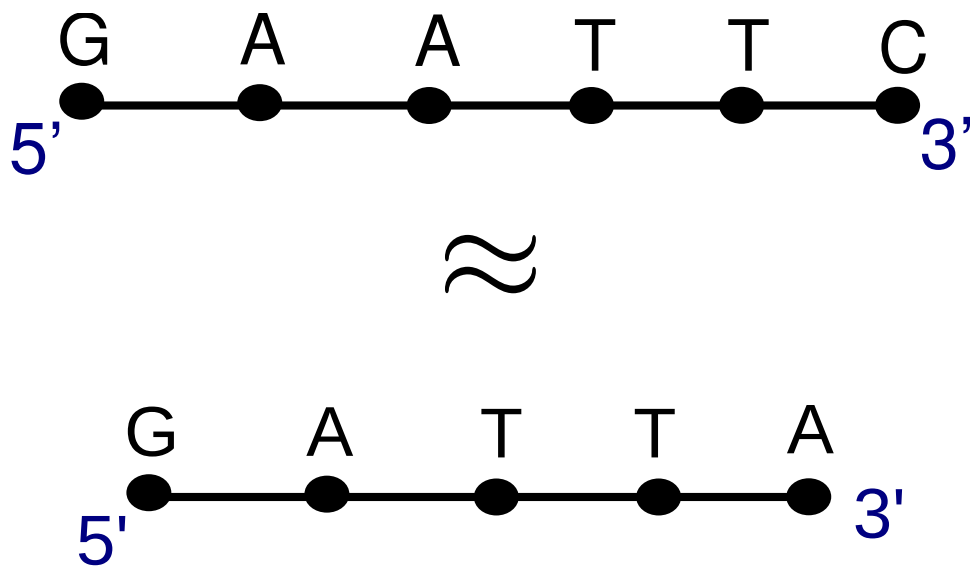
s_i - symbol (o indeksie i) sekwencji s, $s_i \in \Sigma$



Konwencja: ciągi piszemy od końca 5'.

Badanie podobieństw dwu sekwencji

Podobieństwo sekwencji

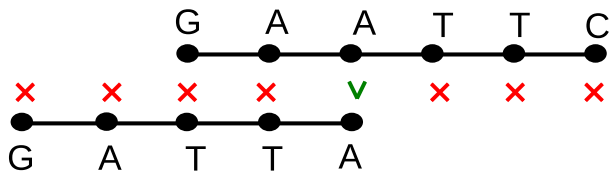
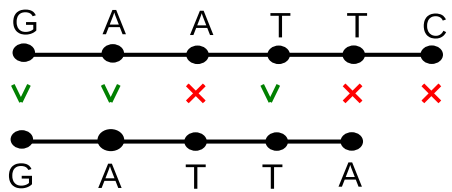
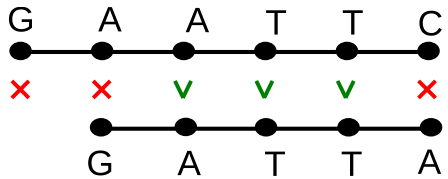


Operacje zmieniające sekwencje:

- mutacje
- wstawienia (pojedyncze, łańcuchy)
- usunięcia (pojedyncze, łańcuchy)

Algorytm 'naiwny' - nie uwzględnia przerw

- nie uwzględnia wstawiania i usuwania nukleotydów
- nagroda za zgodność = 1
- kara za niezgodność = -1

offset = -2		score = -6
offset = 0		score = 0
offset = 1		score = 0

Macierz podobieństwa (jako miara zgodności)

	A	G	C	T
A	1	-1	-1	-1
G	-1	1	-1	-1
C	-1	-1	1	-1
T	-1	-1	-1	1

d = -1 (za brak)

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

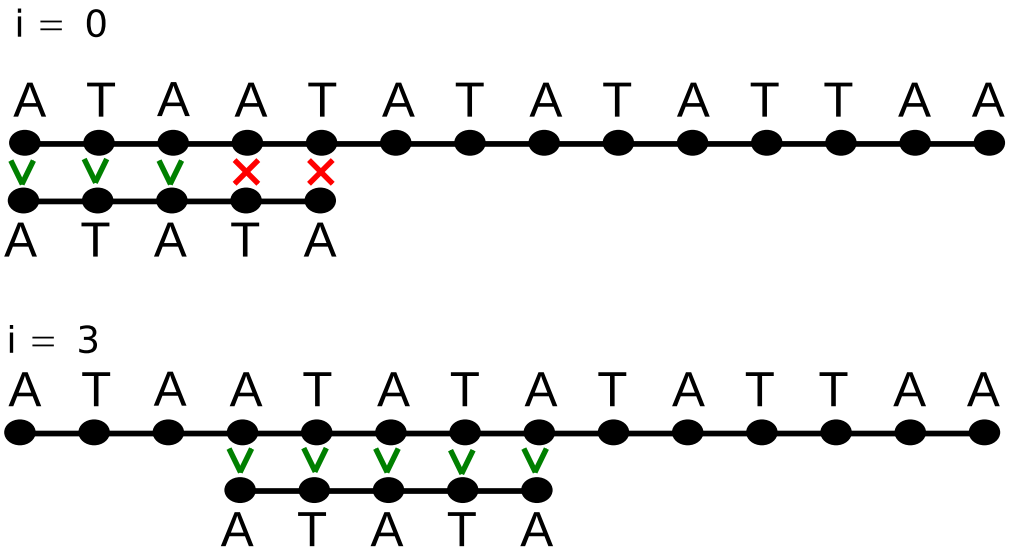
d = -5 (za brak)

Przykładowe miary zgodności:

offset -2	`--GAATTC` `GATTA---`	$S_1 = -6$	$S_2 = -22$
offset 0	`GAATTC` `GATTA-`	$S_1 = 0$	$S_2 = 12$
offset 1	`GAATTC` `-GATTA`	$S_1 = 0$	$S_2 = 17$

Problem z algorytmem naiwnym - złożoność obliczeniowa

```
//T[N] - łańcuch
//S[M] - wzorzec
//M < N
for(int i=0; i<N-M; ++i) {
  for(int j=0; j<M; ++j)
    if(T[i+j] != S[j])
       break;
  if(j==M) //znaleziono;
}
```



Złożoność czasowa: **O(N*M)**

Prosty algorytm ma kwadratową złożoność czasową

Tablica prefikso-sufiksowa

Prefikso-sufiks: najdłuższy właściwy prefiks tekstu, będący jednocześnie jego sufiksem

Przykład: **ATATA**

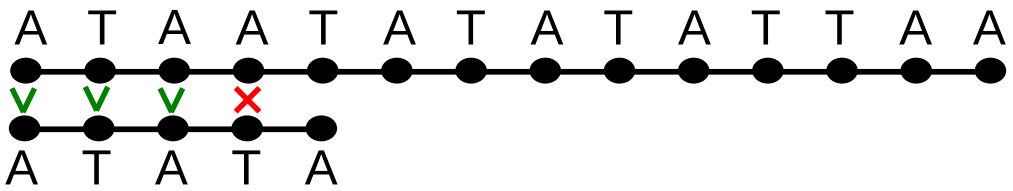
i	napis	prefikso-sufiks	P[i]
0	-	-	0
1	A	-	0
2	AT	-	0
3	ATA	A	1
4	ATAT	AT	2
5	ATATA	ATA	3

Przykład: **ATAAT**

i	napis	p-s	P[i]
0	-	-	0
1	A	-	0
2	AT	-	0
3	ATA	A	1
4	ATAA	A	1
5	ATAAT	AT	2

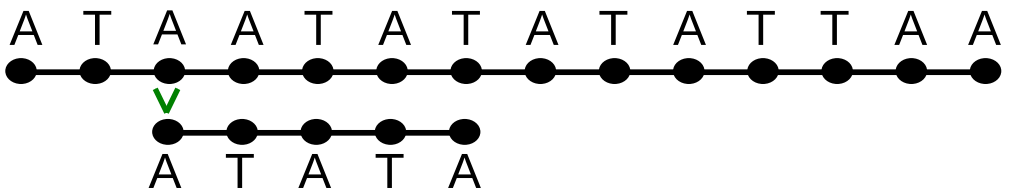
Algorytm Knutha-Morrisa-Prata, złożoność O(N+2M)

i = 0



- jedno porównanie, symbolu ze wzorcem gdy wynik jest pozytywny
- gdy wynik jest negatywny wykorzystuje informację z tablicy prefiksów - sufiksów

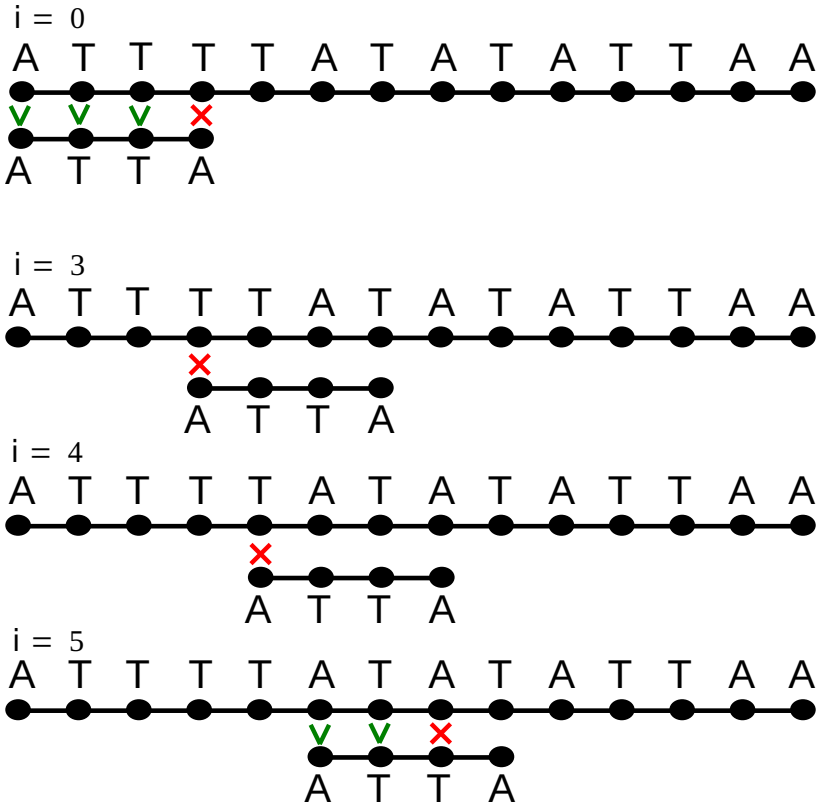
i = 2




```
//algorytm MP
for(int i=0,int j=0; i<N-M;)
{
  for(;j<M; ++j)
    if(T[i+j] != S[j])
      break;
  if(j==M) //znaleziono
    cout << i;

  if(j > 0) {
    i += j - P[j];
    j = P[j];
  } else {
    i += 1;
  }
}
```

Algorytm Knutha-Morrisa-Prata, przykład 2



i	napis	p-s	P[i]
1	A	-	0
2	AT	-	0
3	ATT	-	0
4	ATTA	A	1

```
for(int i=0,int j=0; i<N-M;)
{ for(;j<M; ++j)
    if(T[i+j] != S[j])
      break;
  if(j==M) //znaleziono
    cout << i;
  if(j > 0) {
    i += j - P[j];
    j = P[j];
  } else { ++i; }
}
```

Uwzględnienie przerw w rozwiązaniu 'naiwnym'

Uwzględnienie przerw w rozwiązaniu przeglądającym wszystkie rozwiązania - problem o złożoności wykładniczej, bo liczba możliwych połączeń łańcuchów s i t (o długości n)

$$C(s,t) = \begin{pmatrix} |s|+|t| \\ |s| \end{pmatrix} \simeq \frac{(2n)!}{(n!)^2} \simeq \frac{2^{2n}}{\sqrt{n}} \text{ gdy } |s| \simeq |t| \simeq n$$

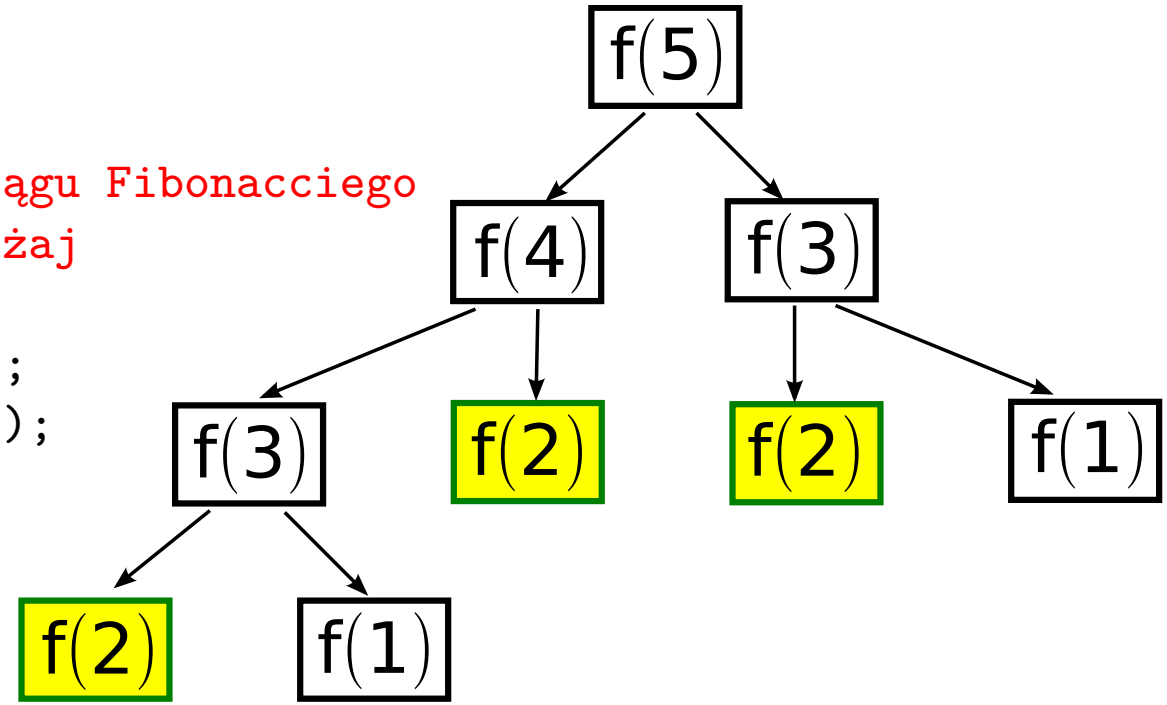
ale: miara dopasowania jest addytywna, więc można konstruować optymalne rozwiązanie na podstawie wyników wcześniejszych obliczeń.

czyli
rekurencyjnie zdefiniować problem
a następnie go rozwiązać

- metodą dziel i zwyciężaj
- metodą programowania dynamicznego

Metoda dziel i zwyciężaj : ciąg Fibonacciego

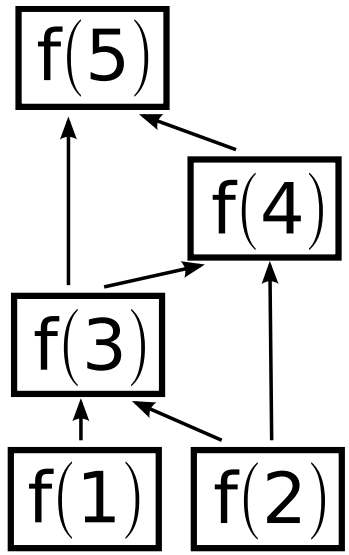
```
//obliczanie wyrazów ciągu Fibonacciego
//metodą dziel i zwyciężaj
int f(int n) {
    if(n <= 2) return 1;
    return f(n-1)+f(n-2);
}
```



Wielokrotnie wykonuje te same obliczenia

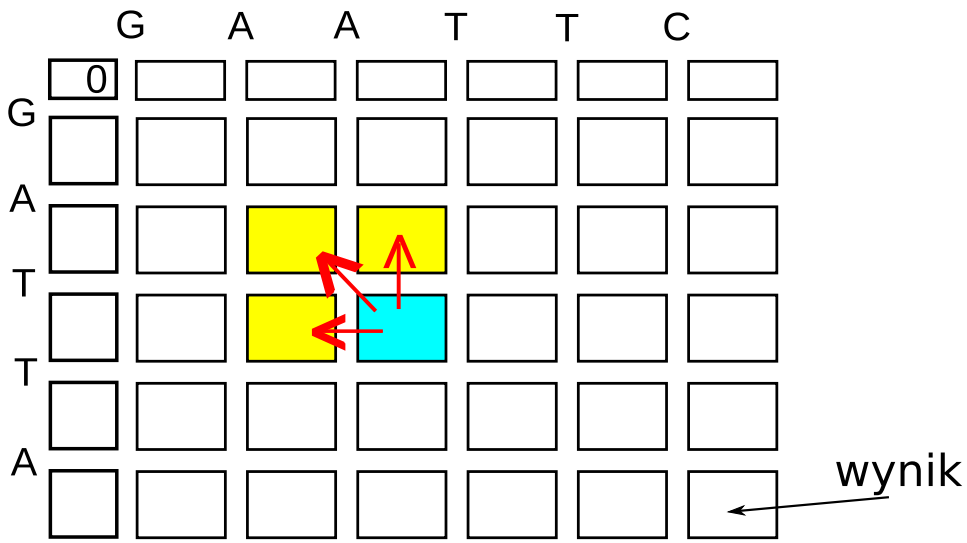
Metoda programowania dynamicznego : ciąg Fibonacciego

```
//obliczanie wyrazów ciągu Fibonacciego
// metodą programowania dynamicznego
int f(int n) {
    boost::scoped_array<int> tab = new int[n];
    tab[0] = 1;
    tab[1] = 1;
    for(int i=2; i<n;++i)
       tab[i] = tab[i-1] + tab[i-2];
    return tab[n-1];
};
```



oblicza optymalne rozwiązanie metodą wstępującą (bottom-up)

Rekurencyjna definicja problemu



Algorytm Needlemana-Wunscha

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + e(s_i, t_j) \text{ //połączenie} \\ F(i-1,j) + d \text{ //przerwa na łańcuchu s} \\ F(i,j-1) + d \text{ //przerwa na łańcuchu t} \end{cases}$$

Algorytm Needlemana-Wunscha

- znajduje $maxM(s,t)$
- złożoność $O(n^2)$

Założenia:

- znana macierz podobieństwa E
- kara za „przerwę" $\gamma(g) = |g| * d$
- wykorzystuje obliczenia dla krótszych łańcuchów

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + e(s_i,t_j) \text{ //połączenie} \\ F(i-1,j) + d \text{ //przerwa na łańcuchu s} \\ F(i,j-1) + d \text{ //przerwa na łańcuchu t} \end{cases}$$

Algorytm Needlemena-Wunscha (przykład)

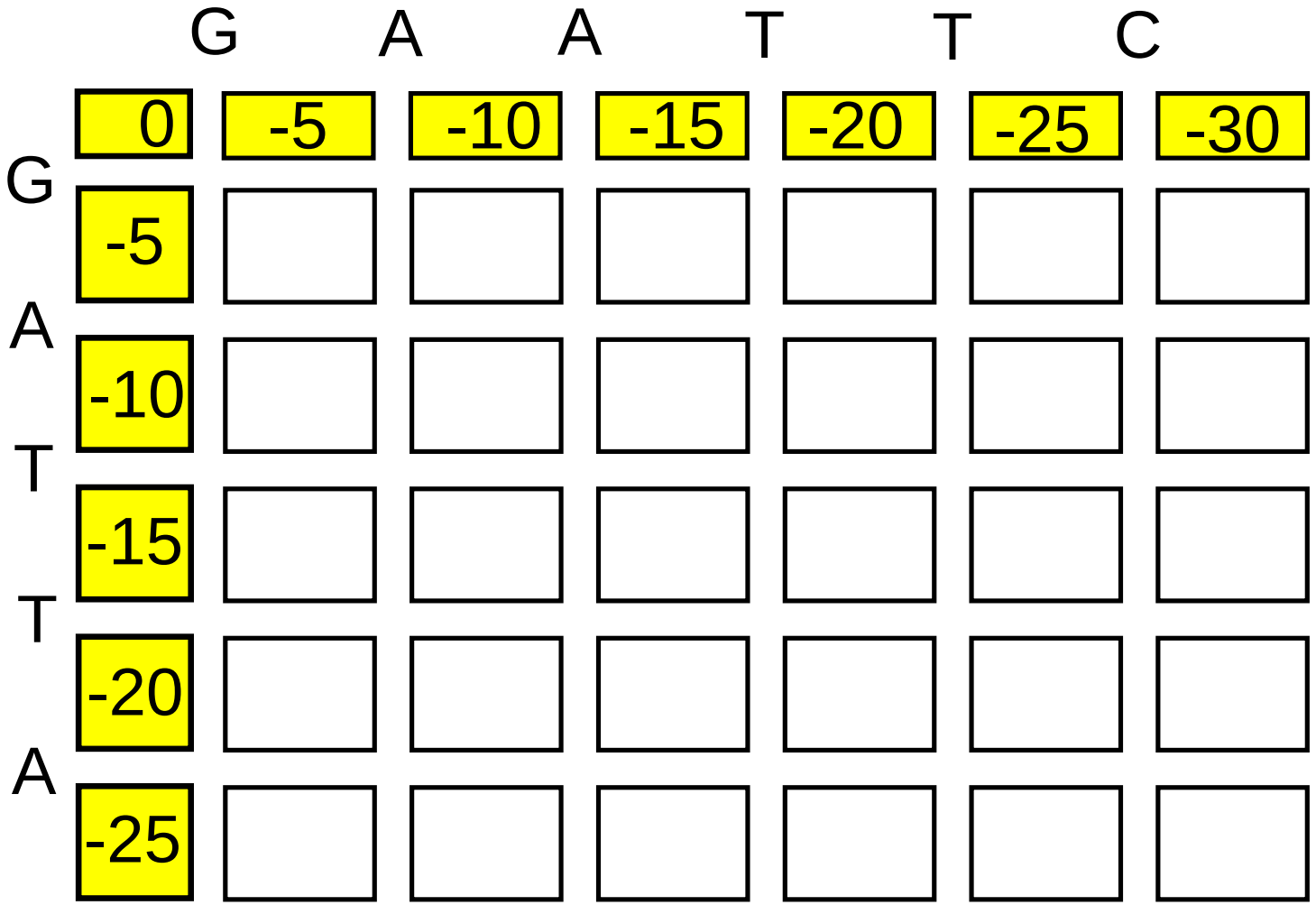
Przykładowa macierz kar i nagród

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

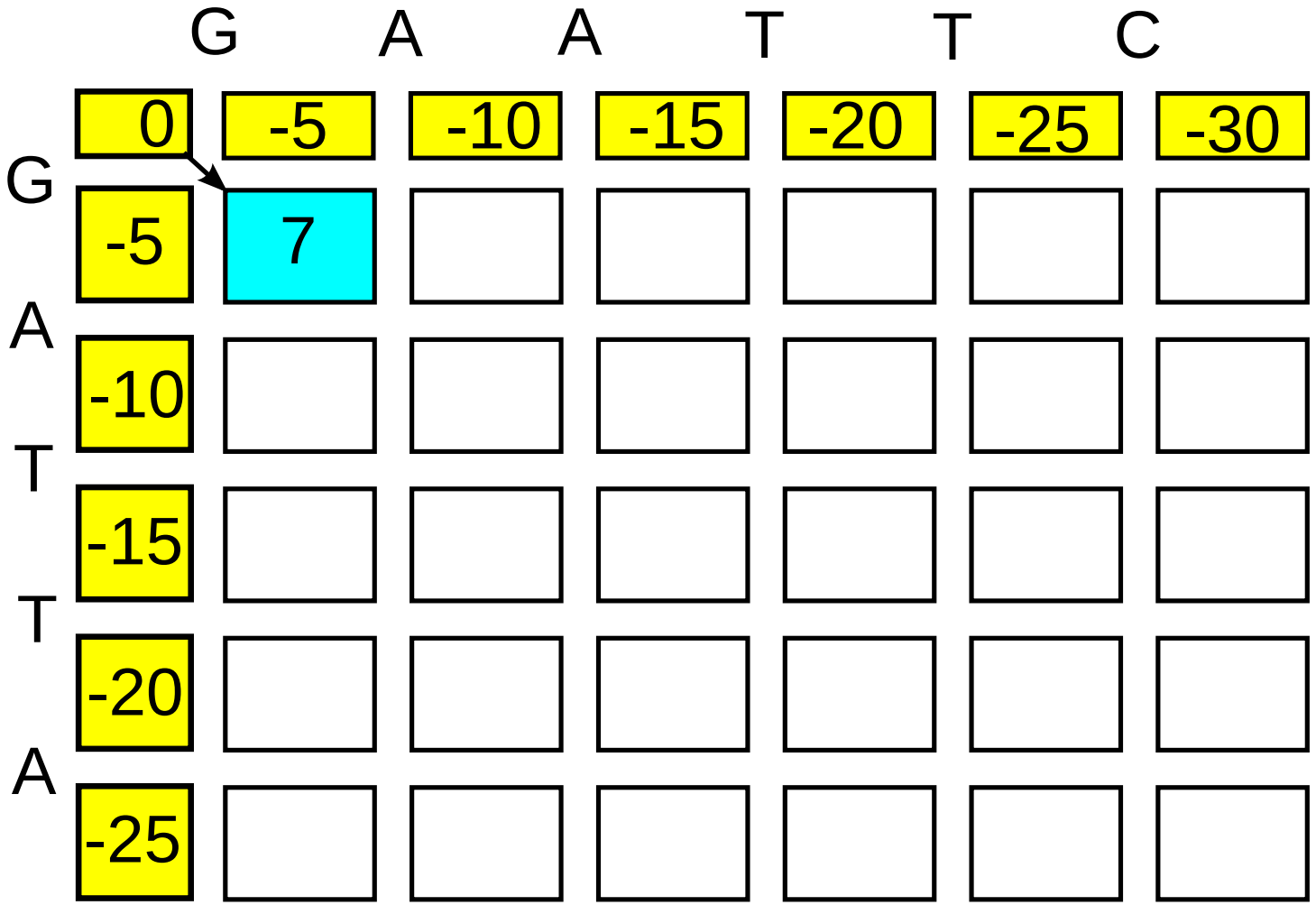
Kara za przerwę:
$d = -5$

Kara za wiszący nukleotyd:
$d = -5$

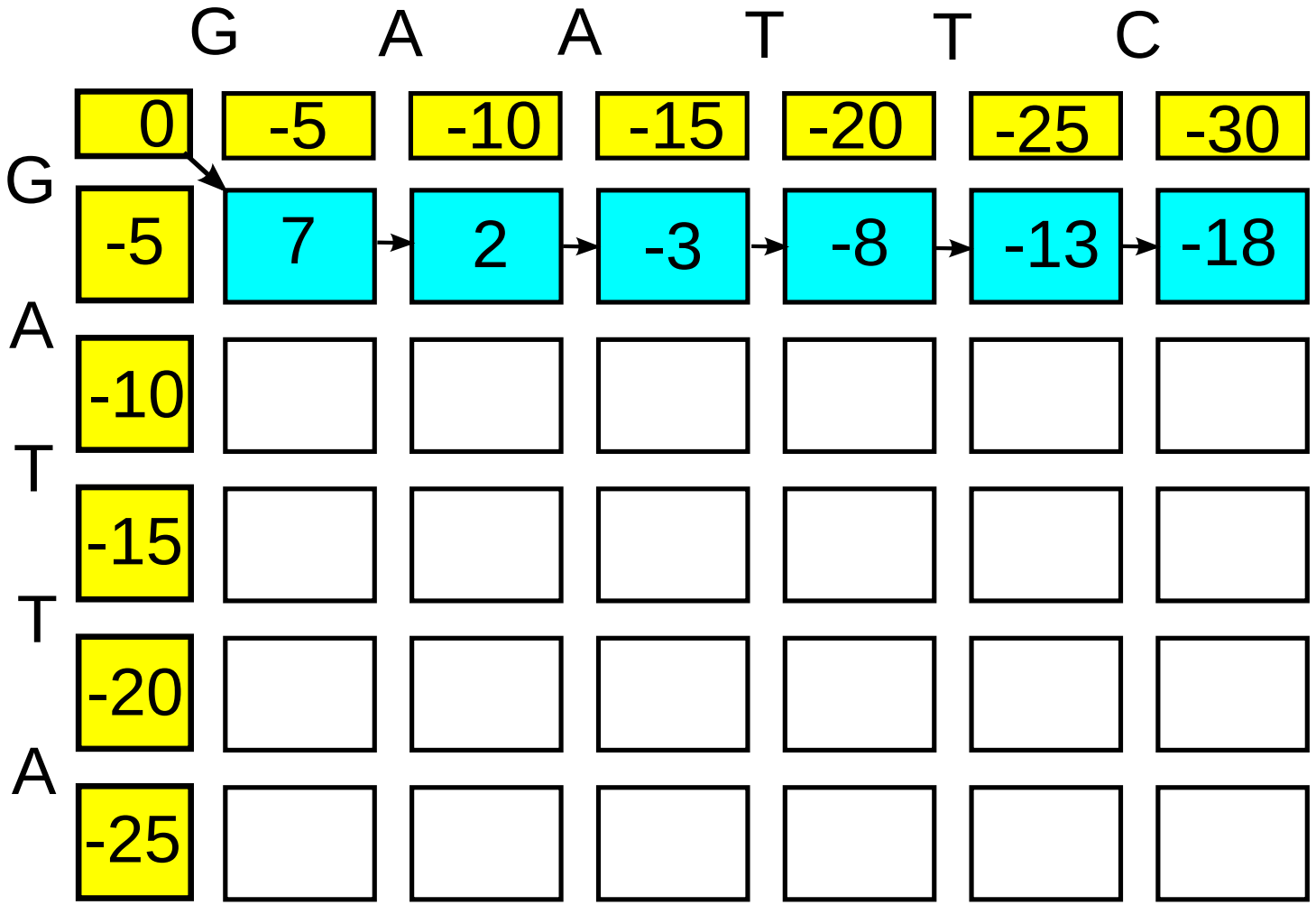
Algorytm Needlemana-Wunscha (przykład)



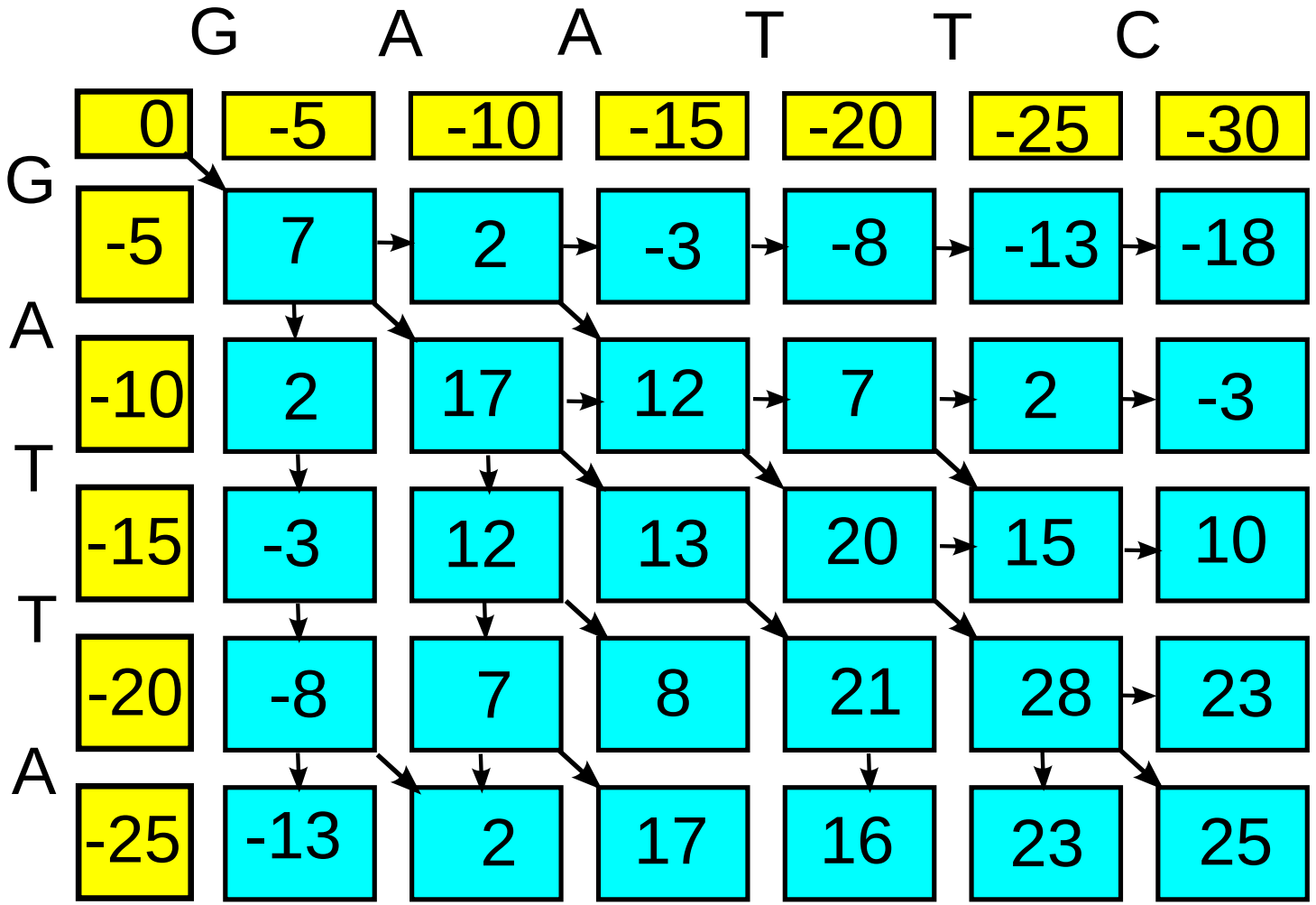
Algorytm Needlemana-Wunscha (przykład)



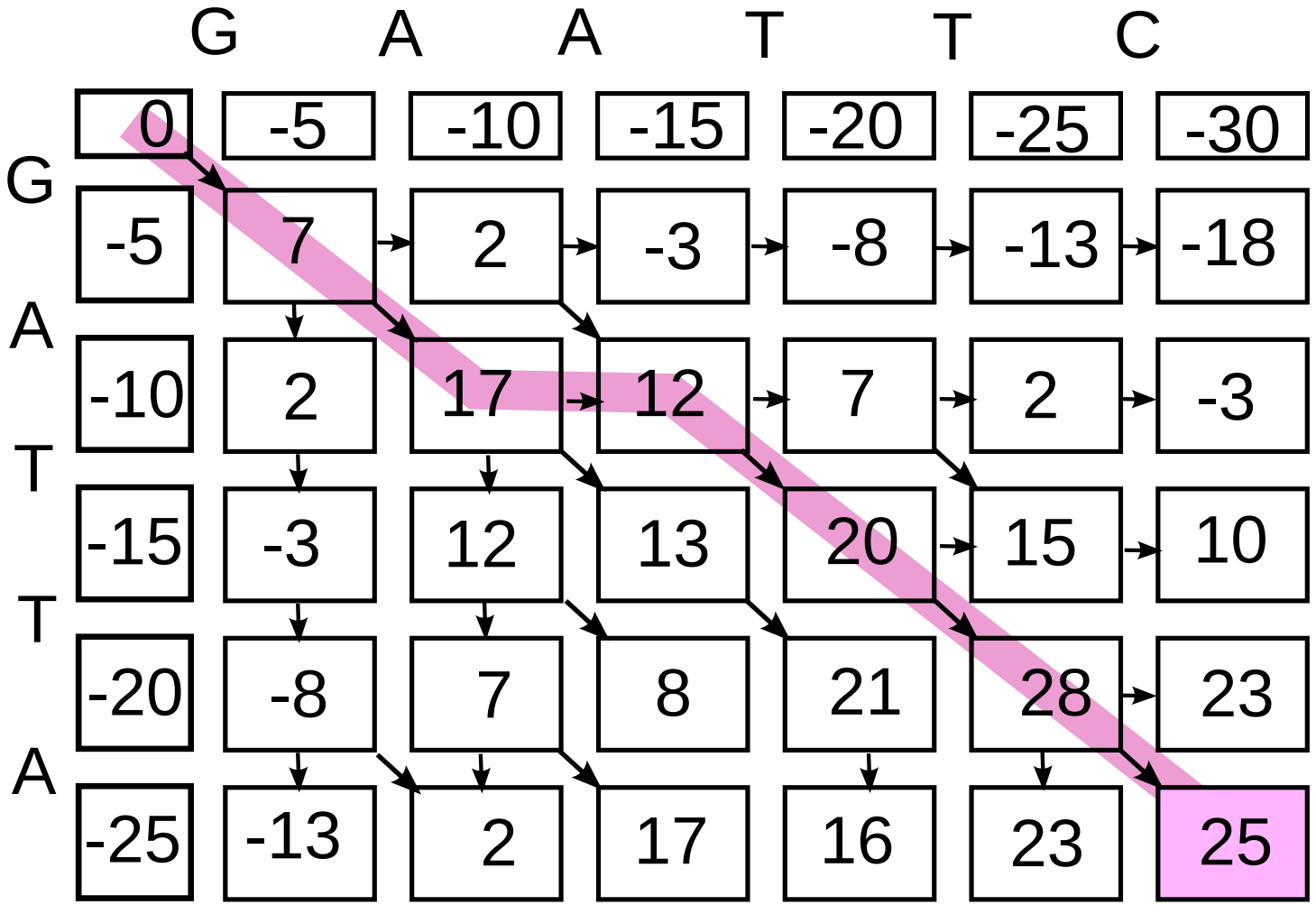
Algorytm Needlemana-Wunscha (przykład)



Algorytm Needlemana-Wunscha (przykład)



Algorytm Needlemana-Wunscha (przykład)



Algorytm Needlemana-Wunscha (przykład)

2 rozwiązania:

```
G A A T T C     G A A T T C
| |   | |   ,   |   | | |   .
G A - T T A     G - A T T A
```

Parametry:

- złożoność czasowa: O(m*n)
- złożoność pamięciowa: O(m*n)

program demonstracyjny pokazujący kolejne kroki obliczeń `http://staff.elka.pw.edu.pl/~rnowak2/dyd/abs2023/demo_align2`

Dziękuję