

<https://hackmd.io/@bsobocki/subsequences>

## Największy Wspólny Podciąg

Artykuł, z którego korzystałem: [wikipedia](#) :)

Mamy dwa ciągi:

$$A = a_1, a_2, \dots, a_n$$

$$B = b_1, b_2, \dots, b_m$$

Interesuje nas ciąg  $C$  taki, że:

- $\forall c \in C \quad c \in A \cap B$
- elementy w  $C$  mają zachowany porządek ciągów  $A$  i  $B$
- $|C| - maksymalny$

Prościej rzecz ujmując chcemy znaleźć jak największy *podciąg* ciągów  $A$  i  $B$ , czyli ciąg  $C$  składający się z elementów należących do  $A$  i do  $B$  z zachowaniem ich porządku.

Podciąg wygląda tak, że bierzemy ciąg  $A$  w porządku jakim się znajduje, usuwamy elementy, które nam nie pasują i powstaje ciąg  $C$ . Tak samo z ciągu  $B$  usuwamy odpowiednie elementy i otrzymamy ciąg  $C$ . Wtedy  $C$  jest wspólnym podciągiem  $A$  i  $B$ .

### PRZYKŁAD PROBLEMU

$$\begin{aligned} A &= b \ b \ a \ b \ a \ a \ b \ b \ a \\ B &= a \ b \ b \ a \ c \ d \ a \ g \end{aligned}$$

Możemy zauważyć, że największymi wspólnymi podciągami będą:

$$\begin{aligned} C1 &= b \ b \ a \ a \\ C2 &= a \ b \ b \ a \end{aligned}$$

### ROZWIĄZANIE

Powyższy przykład rozwiążemy za pomocą **programowania dynamicznego**.

#### Co chcemy zrobić?

Tworzymy tablicę  $T$  rozmiaru  $m \times n$ , która będzie reprezentowała nasz problem.

Chcemy wypełniać ją tak, że w danej komórce  $T[i][j]$  będziemy mieć informacje na temat tego, jak długi jest NWP ciągów  $A[1..i]$  i  $B[1..j]$ , czyli ciągi składające się tylko z tych rozpatrzonych już elementów i aktualnych elementów ( $A[i]$  i  $B[j]$ ).

#### Reguły postępowania

Zastanówmy się, jakie reguły postępowania możemy wyznaczyć.

Rozpatrujemy  $T[i][j]$ , czyli porównujemy  $A[i]$  i  $B[j]$ .

- Jeśli  $A[i] = B[j]$  to do naszego podciągu wynikowego możemy dodać ten element (zachowaliśmy porządek ciągów – idąc od początku po kolej – więc jak natkniemy się na elementy o wartości równej w obu podciągach to wiemy, że w  $NWP$  podciągów  $A[1..i]$  i  $B[1..j]$  ten element występuje).

Wiemy jaką długość ma  $NWP$  ciągów  $A$  i  $B$  bez  $A[i]$  i  $B[j]$ , to znaczy wiemy ile wynosi  $NWP$  ciągów  $A[1..i-1]$  i  $B[1..j-1]$ .

Jest to wartość  $T[i-1][j-1]$ , a więc do tej wartości dodajemy 1.

- Jeśli  $A[i] \neq B[j]$  to oznacza, że nie mamy czego dodawać do  $NWP$ , a zatem jego długość jest równa  $NWP$  dla ciągów  $A[1..i-1]$  i  $B[1..j]$ , lub równa długości  $NWP$  dla  $A[1..i]$  i  $B[1..j-1]$ , ponieważ będzie taka sama jak po dodaniu tylko  $B[j]$  lub tylko  $A[i]$ , a więc bierzemy musimy wziąć wartość maksymalną z tych dwóch długości.

Powstaje więc wzór na wypełnienie tablicy  $T$ :

$$T = \begin{cases} T[i-1][j-1] + 1 & \text{jeśli } A[i] = B[j] \\ \max(T[i-1][j], T[i][j-1]) & \text{jeśli } A[i] \neq B[j] \end{cases}$$

### Wypełnianie tabeli

Zacznijmy od zbudowania tabeli  $T$ :

	$\emptyset$	a	b	b	a	c	d	a	g
$\emptyset$									
b									
b									
a									
b									
a									
a									
b									
b									
a									

Oczywiście zbiór pusty nie posiada wspólnych elementów z żadnym innym podzbiorem, stąd wyzerujemy odpowiednie komórki tablicy:

	$\emptyset$	a	b	b	a	c	d	a	g
$\emptyset$	0	0	0	0	0	0	0	0	0
b	0								
b	0								
a	0								
b	0								
a	0								
a	0								
b	0								
b	0								
a	0								

Posługując się ustalonymi przez nas regułami wypełniamy tabelę:

	$\emptyset$	a	b	b	a	c	d	a	g
$\emptyset$	0	0	0	0	0	0	0	0	0
b	0	0	1	1	1	1	1	1	1
b	0	0	1	2	2	2	2	2	2
a	0	1	1	2	3	3	3	3	3

b	0	1	2	2	3	3	3	3	3
a	0	1	2	2	3	3	3	4	4
a	0	1	2	2	3	3	3	4	4
b	0	1	2	3	3	3	3	4	4
b	0	1	2	3	3	3	3	4	4
a	0	1	2	3	4	4	4	4	4

Wiemy dokładnie, że gdy ciągi  $A$  i  $B$  mają odpowiednio długości  $n$  i  $m$  to  $T[n][m]$  jest równe *długości NWP* tych ciągów, co jest odpowiedzią na pytanie:

Jaką długość ma Najdłuższy Wspólny Podciąg ciągów A i B?

## IMPLEMENTACJA ROZWIĄZANIA

```
int nwp_size(string A, string B){
    int n = A.size();
    int m = B.size();

    // zauważmy fakt, że dodając zbiory puste do tablicy
    // musimy dodać 1 do obu jej wymiarów
    int T[n+1][m+1];

    //zerowanie pierwszej kolumny
    for (int i=0; i<=n; i++)
        T[i][0] = 0;

    //zerowanie первого wiersza
    for (int j=0; j<=m; j++)
        T[0][j] = 0;

    for (int i=1; i<=n; i++)
        for (int j=1; j<=m; j++){
            //nasza tablica powiększona jest o zbiór pusty
            //więc musimy brać indeksy o 1 mniejsze dla A i B
            if (A[i-1] == B[j-1])
                T[i][j] = T[i-1][j-1] + 1;
            else
                T[i][j] = max(T[i-1][j], T[i][j-1]);
        }

    return T[n][m];
}
```

Jednak chcieliśmy znaleźć ten podciąg (być może więcej takich podciągów), a nie jego długość!

- Jak możemy znaleźć NWP?
- Odtwarzając串 na podstawie danych!

Możemy zauważyc, że idąc od  $T[n][m]$  w góre i w lewo dopóki komórki na lewo lub do góry mają tę samą wartość dojedziemy do skraju figury tworzonej przez wartości odpowiadające znalezionej długości NWP (w tym przypadku 4), to znaczy do momentu, gdy komórka na lewo i komórka na prawo mają już mniejszą wartość.

	Ø	a	b	b	a	c	d	a	g
Ø	0	0	0	0	0	0	0	0	0
b	0	0	1	1	1	1	1	1	1
b	0	0	1	2	2	2	2	2	2
a	0	1	1	2	3	3	3	3	3

	Ø	a	b	b	a	c	d	a	g
Ø	0	0	0	0	0	0	0	0	0
b	0	0	1	1	1	1	1	1	1
b	0	0	1	2	2	2	2	2	2
a	0	1	1	2	3	3	3	3	3

b	0	1	2	2	3	3	3	3	3
a	0	1	2	2	3	3	3	4	4
a	0	1	2	2	3	3	3	4	4
b	0	1	2	3	3	3	3	4	4
b	0	1	2	3	3	3	3	4	4
a	0	1	2	3	4	4	4	4	4

b	0	1	2	2	3	3	3	3	3
a	0	1	2	2	3	3	3	4	4
a	0	1	2	2	3	3	3	4	4
b	0	1	2	3	3	3	3	4	4
b	0	1	2	3	3	3	3	4	4
a	0	1	2	3	4	4	4	4	4

Tak samo postępujemy z kolejnymi wartościami:

	ø	a	b	b	a	c	d	a	g
ø	0	0	0	0	0	0	0	0	0
b	0	0	1	1	1	1	1	1	1
b	0	0	1	2	2	2	2	2	2
a	0	1	1	2	3	3	3	3	3
b	0	1	2	2	3	3	3	3	3
a	0	1	2	2	3	3	3	4	4
a	0	1	2	2	3	3	3	4	4
b	0	1	2	3	3	3	4	4	4
b	0	1	2	3	3	3	4	4	4
a	0	1	2	3	4	4	4	4	4

	ø	a	b	b	a	c	d	a	g
ø	0	0	0	0	0	0	0	0	0
b	0	0	1	1	1	1	1	1	1
b	0	0	1	2	2	2	2	2	2
a	0	1	1	2	3	3	3	3	3
b	0	1	2	2	3	3	3	3	3
a	0	1	2	2	3	3	3	4	4
a	0	1	2	2	3	3	3	4	4
b	0	1	2	3	3	3	4	4	4
b	0	1	2	3	3	3	4	4	4
a	0	1	2	3	4	4	4	4	4

Przy każdym dojściu do komórki, której lewy i górny sąsiad mają inną wartość, niech to będzie  $T[k][l]$ , zapamiętujemy element  $A[k]$  bądź  $B[l]$ .

Wynikiem jest nasz podciąg, lecz w kolejności odwrotnej (ponieważ szliśmy od końca do początku!).

W tym wypadku lewe zaznaczenie odpowiednich wartości z  $T$  daje nam:

$$C = bbaa,$$

natomiast prawe:

$$C = abba$$

## Największy Podciąg Ciągu Będący Palindromem

*palindrom* – słowo  $w$ , które od tyłu ma dokładnie taką samą sekwencję znaków, co odpowiadające mu słowo  $w'$  będące jego odrwóceniem.

przykłady:

*kajak, zakaz, 1001, 10101, 2390932, aabnsnbaa*

Aby znaleźć długość największego wspólnego podciągu będącego palindromem dla danego ciągu posłużymy się metodą **programowania dynamicznego**. Utworzymy tabelę, którą będziemy wypełniać odpowiednimi wartościami.

Do wyszukania danego podciągu użyjemy dwóch iteratorów.

Iterator  $i$  oraz iterator  $j$ .

Sprawdzamy długość najdłuższego podciągu będącego palindromem, który zawiera się w podciagu utworzonego z elementów od  $i$  do  $j$ .

Dla przykładu będziemy rozpatrywać ciąg

A A F G J A J A K

iterujemy po elementach ciągu

A A F G J A J A K  
^ ^  
i j

Wszystkie iteracje  $j$  będą odpowienimi kolumnami naszej tabeli, natomiast iteracje  $i$  będą kolejnymi wierszami tabeli.

Wiemy że w sytuacji, gdy  $i$  i  $j$  wskazują na ten sam element ciągu, to długość sprawdzanego podciągu jest równa 1, stąd długość podciągu od  $i$  do  $j$  będącego palindromem jest równa 1.

Stąd przekątną naszej tabeli wypełniamy 1.

Następnie zauważmy, że gdy przesuniemy któryś z iteratorów na kolejną pozycję, to mamy dwie możliwości:

- $$\bullet \quad {}^*i = {}^{\circ}i$$

długość naszego ciągu zwiększa się o 2 względem krótszego ciągu składającego się z elementów od  $i + 1$  do  $j - 1$  (rozszerzyliśmy ten ciąg w lewo i w prawo o elementy równej wartości, czyli palindrom się zwiększa)

- $$\bullet \text{ } {}^*i \neq {}^*j$$

nie dodajemy żadnego elementu do podciągu będącego palindromem, a więc dalej mamy taką samą długość jak po zweżeniu od lewej lub od prawej strony (max z tych dwóch długości)

Stad mając tabelę  $T[0..n][0..n]$  mamy warunki

aafqajajak

$$T[i][j] = \begin{cases} T[i+1][j-1] + 2 & \text{ gdy } i = j \\ \max(T[i+1][j], T[i][j-1]) & \text{ gdy } *i \neq *j \end{cases}$$

## Co nas interesuje

Interesuje nas wypełnienie górnego prawego trójkąta tabeli.

### UWAGA

Gdy chcemy wypełnić  $T[0][1]$ , a elementy 0 i 1 mają taką samą wartość. Wtedy potrzebujemy wartości  $T[1][0]$ , która nie ma wartości zainicjalizowanej. Wystarczy, że zainicjalizujemy ją jako 0

LibreOffice Calc robi to automatycznie.

Zauważmy, że dla danego  $i$  i dla danego  $j$  rozpatrujemy podciąg złożony z elementów od  $i$ -tego do  $j$ -ego, więc w komórce  $T[0][n - 1]$  bedziemy mieć długość najdłuższego podciagu będącego palindromem.

W tym przypadku takie podciągi to:

a a f a a

a a g a a

a a j a a

a j a j a

Film, który pomógł mi w opanowaniu tego zagadnienia:

<https://www.youtube.com/watch?v=TLaGwTnd3HY>

## DWUMIAN NEWTONA

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

$$\binom{n}{k} = \begin{cases} 1 & \text{dla } k=0 \text{ lub } k=n \\ \binom{n-1}{k-1} + \binom{n-1}{k} & \text{dla } 0 < k < n \end{cases}$$

gdzie  $\binom{i}{0} = \binom{i}{i} = 1$ ,  $0 \leq k \leq n$ ,  $0 \leq i$

$$T[i][j] = \begin{cases} B[i-1][j-1] + B[i-1][j], & 0 \leq j < i \end{cases}$$

| 1 ,  $j=0$  lub  $j=i$

## OPTIMALNA KOLEJNOŚĆ MNOŻENIA MACIERZY

Przyjmujemy, że mnożenie macierzy rozmiaru  $p \times q$  przez macierz rozmiaru  $q \times r$  wymaga  $p \cdot q \cdot r$  operacji. Chcemy obliczyć iloczyn  $M = M_1 \cdot M_2 \cdot M_3 \cdot M_4$  o wymiarach odpowiednio  $[10 \times 20]$ ,  $[20 \times 50]$ ,  $[50 \times 1]$ ,  $[1 \times 100]$ . Obliczmy  $M$  wg schematu  $M = M_1 \cdot (M_2 \cdot (M_3 \cdot M_4))$

$$M_{34} = M_3 \cdot M_4 = 50 \cdot 1 \cdot 100 = 5000 - \text{koszt mnożenia, wymiar powstałej macierzy } M_{34} [50 \times 100]$$

$$M_{234} = M_2 \cdot M_{34} = 20 \cdot 50 \cdot 100 = 100000 - \text{koszt mnożenia, wymiar macierzy } M_{234} \text{ to } [20 \times 100]$$

$$M_1 \cdot M_{234} = 10 \cdot 20 \cdot 100 = 20000 - \text{koszt mnożenia}$$

$$5000 + 100000 + 20000 = 125000 - \text{całkowity koszt mnożenia}$$

Policzmy teraz  $M$  dla schematu

$$(M_1 \cdot (M_2 \cdot M_3)) \cdot M$$

$$M_{23} = M_2 \cdot M_3 = 20 \cdot 50 \cdot 1 = 1000 - \text{koszt mnożenia, wymiar } M_{23} \text{ to } [20 \times 1]$$

$$M_{123} = M_1 \cdot M_{23} = 10 \cdot 20 \cdot 1 = 200 - \text{koszt operacji, wymiar } [10 \times 1]$$

$$M_{123} \cdot M_4 = 10 \cdot 1 \cdot 100 = 1000 - \text{koszt mnożenia, wymiar } [10 \times 100]$$

$$1000 + 200 + 1000 = 2200 - \text{całkowity koszt mnożenia}$$

Gdybyśmy chcieli znaleźć najmniejszy koszt mnożenia macierzy to moglibyśmy sprawdzić wszystkie możliwe schematy, ale to byłby czas wykładniczy. Za pomocą programowania dynamicznego możemy ograniczyć czas do  $O(n^3)$ .

Niech  $m_{ij}$  będzie minimalnym kosztem obliczenia  $M_i \cdot M_{i+1} \cdot \dots \cdot M_j$  dla  $1 \leq i \leq j \leq n$ .

Wówczas

$$m_{ij} = \begin{cases} 0 & \text{jeżeli } i = j \\ \min_{i \leq k < j} (m_{ik} + m_{k+1,j} + r_{i-1} \cdot r_k \cdot r_j) & \text{jeżeli } j > i \end{cases}$$

gdzie

$m_{ik}$  - minimalny koszt mnożenia macierzy  $M_i \cdot M_{i+1} \cdot \dots \cdot M_k$  o rozmiarze  $r_{i-1} \times r_k$

$m_{k+1,j}$  - minimalny koszt mnożenia macierzy  $M_{k+1} \times M_{k+2} \times \dots \times M_j$  o rozmiarze  $r_k \times r_j$   
 $r_{i-1} \cdot r_k \cdot r_j$  - koszt mnożenia macierzy  $M_{ik} \times M_{k+1,j}$  o rozmiarze  $r_{i-1} \times r_j$   
Dla  $i < j$  wartość  $k = i, i+1, i+2, \dots, j-1$   
Rozmiary macierzy umieszczamy w tablicy  $R : [r_0, r_1, \dots, r_{n-1}, r_n]$ . Wartości  $m_{ij}$  zapamiętujemy w tablicy dwuwymiarowej.

Ciąg dalszy przykładu

$n = 4$ ,  $M = M_1 \cdot M_2 \cdot M_3 \cdot M_4$  o wymiarach odpowiednio  $[10 \times 20]$ ,  $[20 \times 50]$ ,  $[50 \times 1]$ ,  $[1 \times 100]$   
 $R[10, 20, 50, 1, 100]$

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ & m_{22} & m_{23} & m_{24} \\ & & m_{33} & m_{34} \\ & & & m_{44} \end{bmatrix}$$

Licząc z definicji wiemy, że

$$m_{11} = m_{22} = m_{33} = m_{44} = 0$$

$$m_{12} = \min_{1 \leq k < 2} (m_{1k} + m_{k+1,2} + r_0 \cdot r_k \cdot r_2) = \min \underbrace{(m_{11} + m_{22} + r_0 \cdot r_1 \cdot r_2)}_{k=1} = 0 + 0 + 10 \cdot 20 \cdot 50 = 10000$$

$$m_{23} = \min_{2 \leq k < 3} (m_{2k} + m_{k+1,3} + r_1 \cdot r_k \cdot r_3) = \min \underbrace{(m_{22} + m_{33} + r_1 \cdot r_2 \cdot r_3)}_{k=2} = 0 + 0 + 20 \cdot 50 \cdot 1 = 1000$$

$$m_{34} = \min_{3 \leq k < 4} (m_{3k} + m_{k+1,4} + r_2 \cdot r_k \cdot r_4) = \min \underbrace{(m_{33} + m_{44} + r_2 \cdot r_3 \cdot r_4)}_{k=3} = 0 + 0 + 50 \cdot 1 \cdot 100 = 5000$$

$$m_{13} = \min_{1 \leq k < 3} (m_{1k} + m_{k+1,3} + r_0 \cdot r_k \cdot r_3) = \min \underbrace{(m_{11} + m_{23} + r_0 \cdot r_1 \cdot r_3)}_{k=1}, \underbrace{(m_{11} + m_{33} + r_0 \cdot r_2 \cdot r_3)}_{k=2} = \min(0 + 1000 + 10 \cdot 20 \cdot 1, 50000 + 0 + 10 \cdot 50 \cdot 1) = 1200$$

$$m_{24} = \min_{2 \leq k < 4} (m_{2k} + m_{k+1,4} + r_1 \cdot r_k \cdot r_4) = \min \underbrace{(m_{22} + m_{34} + r_1 \cdot r_2 \cdot r_4)}_{k=2}, \underbrace{(m_{23} + m_{44} + r_1 \cdot r_3 \cdot r_4)}_{k=3} = \min(0 + 5000 + 20 \cdot 50 \cdot 100, 1000 + 0 + 20 \cdot 1 \cdot 100) = 3000$$

$$m_{14} = \min_{1 \leq k < 4} (m_{1k} + m_{k+1,4} + r_0 \cdot r_k \cdot r_4) = \min \underbrace{(m_{11} + m_{24} + r_0 \cdot r_1 \cdot r_4)}_{k=1}, \underbrace{(m_{12} + m_{34} + r_0 \cdot r_2 \cdot r_4)}_{k=2}, \underbrace{(m_{13} + m_{44} + r_0 \cdot r_3 \cdot r_4)}_{k=3} = \min(0 + 3000 + 10 \cdot 20 \cdot 100, 50000 + 5000 + 10 \cdot 50 \cdot 100, 1200 + 0 + 10 \cdot 1 \cdot 100) = 2200$$

$$M = \begin{bmatrix} 0 & 10000 & 1200 & 2200 \\ & 0 & 1000 & 3000 \\ & & 0 & 5000 \\ & & & 0 \end{bmatrix}$$

w prawym górnym roku wyraz  $m_{14} = 2200$  określa nam minimalną ilość operacji potrzebną do policzenia iloczynu.

~~~~~

Jesli  $i=j$  to ciąg skończy się tylko z

jednej macierzy  $A_1 \dots = A_i$ . Wtedy

$m[i,i] = 0$  dla  $i = 1, 2, \dots, m$

Aby obliczyć  $m[i,j]$  dla  $i < j$  trzeba wykorzystać optymalne rozwiązań.

Obliczenie iloczynu macierzy  $A_1 \dots k$  i

$A_{k+1} \dots j$  kosztuje  $p_{i-1} p_k p_j$  skalarowych mnożeń, to otrzymujemy wzór:

$$m[i,j] = m[i,k] + m[k+1,j] + p_{i-1} p_k p_j$$

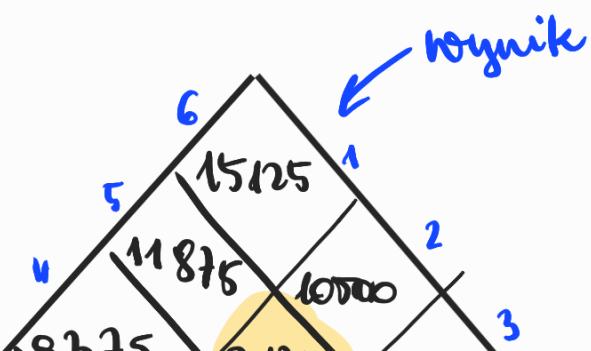
Nie wiemy, które  $k$  jest dobre. Jest  $j-i$  możliwych wartości  $k$  ( $k = i, i+1 \dots j-1$ ).

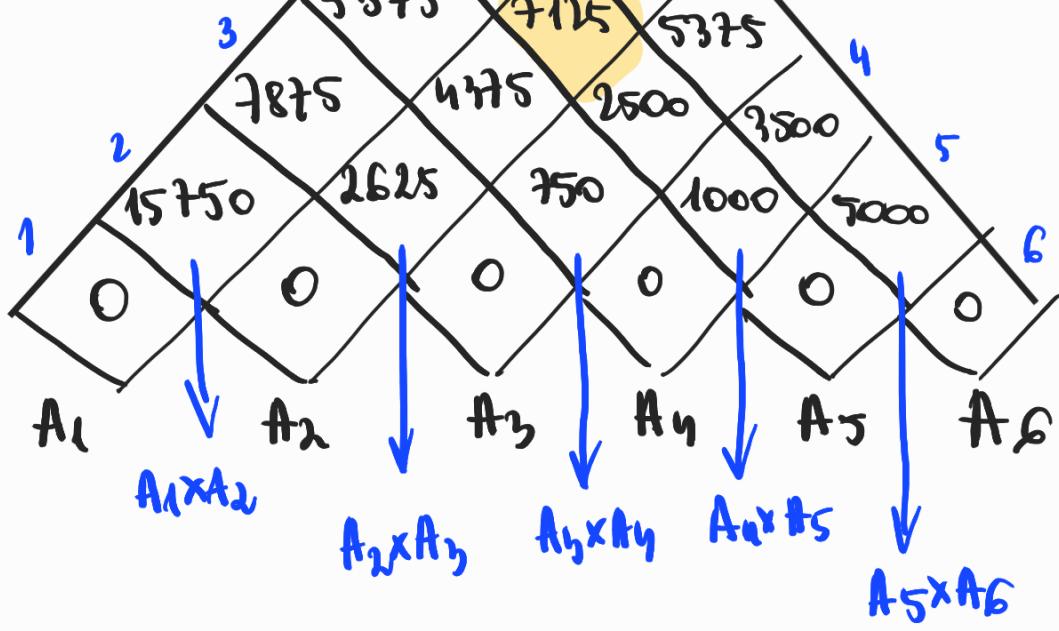
Musimy wybrać najlepszą:

$$m[i,j] = \begin{cases} 0, & i=j \\ \min_{i \leq k < j} \{ m[i,k] + m[k+1,j] + p_{i-1} p_k p_j \}, & i < j \end{cases}$$

PRZMKTAD:

| Macierz  | $A_1$          | $A_2$          | $A_3$         | $A_4$         | $A_5$          | $A_6$          |
|----------|----------------|----------------|---------------|---------------|----------------|----------------|
| Rozmiary | $30 \times 35$ | $35 \times 15$ | $15 \times 5$ | $5 \times 10$ | $10 \times 20$ | $20 \times 25$ |





$$m[2,5] = \min \left\{ \begin{array}{l} m[2,2] + m[3,5] + p_1 p_2 p_5 = \\ = 0 + 2500 + 35 \cdot 15 \cdot 20 = 13000 \\ \\ m[2,3] + m[4,5] + p_1 p_3 p_5 = \\ = 2625 + 1000 + 35 \cdot 5 \cdot 20 = 7125 \\ \\ m[2,4] + m[5,5] + p_1 p_4 p_5 = \\ = 4375 + 0 + 35 \cdot 10 \cdot 20 = 11375 \end{array} \right.$$

= 7125

## PROBLEM PLECAKOWY

DYSKRETNY PROBLEM PLECAKOWY

(0-1 knapsack problem) ↑ NIE ZDAŁĘ  
ALG. LAGHT

$n$  przedmiotów

$i$ -ty przedmiot jest wart  $v_i$  i waży  $w_i$  kg

Do plecaka nie możemy spakować więcej niż  $k$  kilogramów.

każdy przedmiot może być dobrany albo zostawiony.

PODSTAWIE PIERWSZEJ  
MŁODZIAŻECKA

## CIĄGTY PROBLEM PLECAKOWY

NIE! Nie musimy podejmować decyzji pakować / nie pakować. Możemy pakować wtórne ciągły.

$O(nlg n)$

### 2.3.1 Wersja z powtórzeniami

PROBLEM:

Dane: ciąg  $w_1, \dots, w_n \in \mathcal{N}$

ciąg  $v_1, \dots, v_n \in \mathcal{R}$

liczba  $W \in \mathcal{N}$

Wynik: wielozbiór zbiór  $\{i_1, \dots, i_k\}$  taki, że  $\sum_{j=1}^k w_{i_j} \leq W$  oraz  $\sum_{j=1}^k v_{i_j}$  jest maksymalna

Zakładamy, że waga każdego przedmiotu nie przekracza  $W$ .

Podproblemy: mniejszy plecak.

$K(w) =$  maksymalna wartość plecaka osiągalna dla plecaka o pojemności  $w$ .

Fakt 1

$$K(w) = \begin{cases} 0 & \text{jeśli } w = 0, \\ \max_{i: w_i < w} \{K(w - w_i) + v_i\} & \text{jeśli } w > 0 \end{cases}$$

□

Czas działania:  $O(nW)$ .

### 2.3.2 Wersja bez powtórzeń

PROBLEM:

Dane: ciąg  $w_1, \dots, w_n \in \mathcal{N}$

ciąg  $v_1, \dots, v_n \in \mathcal{R}$

liczba  $W \in \mathcal{N}$

Wynik: zbiór  $\{i_1, \dots, i_k\}$  taki, że  $\sum_{j=1}^k w_{i_j} \leq W$  oraz  $\sum_{j=1}^k v_{i_j}$  jest maksymalna.

Podproblemy: mniejszy plecak pakowany podzbiorem przedmiotów.

$K(w, j) =$  maksymalna wartość plecaka osiągalna dla plecaka o pojemności  $w$  oraz przedmiotów  $\{1, \dots, j\}$ .

Fakt 2

$$K(w, j) = \begin{cases} 0 & \text{jeśli } w = 0 \text{ lub } j = 0 \\ \max\{K(w - w_j, j - 1) + v_j, K(w, j - 1)\} & \text{wpp} \end{cases}$$

□

Czas działania:  $O(nW)$ .

↓  
ony

↓  
ony  
i p

Najlepszy  
i ok go  
wzorcu  
jedna po  
nieważnej

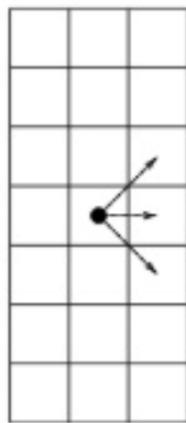
# PRIEJŚCIE PO TABLICY NAJBARDZIEJ OPTYMALNAZ TRASĄ

## PROBLEM:

Dane: Tablica  $\{a_{i,j}\}$  liczb nieujemnych ( $i = 1, \dots, n; j = 1, \dots, m$ )

Wynik: Ciąg indeksów  $i_1, \dots, i_m$  taki, że  $\forall_{j=1, \dots, m-1} |i_j - i_{j+1}| \leq 1$ , minimalizujący sumę  $\sum_{j=1}^m a_{i_j, j}$

INTERPRETACJA: Ciąg  $i_1, \dots, i_m$  wyznacza trasę wiodącą od pierwszej do ostatniej kolumny tablicy  $a$ . Startujemy z dowolnego pola pierwszej kolumny i kończymy w dowolnym polu ostatniej kolumny. W każdym ruchu przesuwamy się o jedno poziomo albo w prawo na wprost albo w prawo na ukos (jak pokazano na rysunku 1). Chcemy znaleźć trasę o minimalnej długości rozumianej jako suma liczb z pól znajdujących się na trasie.



Rysunek 1: Możliwe kierunki ruchu w tablicy  $a$ .

Jak łatwo sprawdzić liczbę wszystkich prawidłowych tras jest wykładnicza, więc rozwiązanie siłowe nie wchodzi w rachubę.

Rozważmy najpierw nieco prostsze zadanie, polegające na znalezieniu długości optymalnej trasy. Potem pokażemy w jaki sposób zorganizować obliczenia, by wyznaczenie samej trasy było proste.

Niech  $d_{i,j}$  oznacza minimalną długość trasy wiodącej od dowolnego pola pierwszej kolumny do pola  $(i,j)$ .

Niech  $a_{i,k}$  oznacza minimalną długość trasy wiodącej od dowolnego pola pierwszej kolumny do pola  $a_{i,k}$ , a  $P(i, k)$  - problem wyznaczenia  $d_{i,k}$ . Rozwiążanie  $P(i, k)$  (dla  $k > 1$ ) można łatwo otrzymać z rozwiązań trzech prostszych podproblemów, a mianowicie  $P(i-1, k-1)$ ,  $P(i, k-1)$  i  $P(i+1, k-1)$  (w przypadku  $P(1, k)$  i  $P(n, k)$  - dwóch podproblemów). Problem spełnia więc wymagane kryterium optymalności.

Jeśli za rozmiar  $P(i, k)$  przyjmiemy wartość  $k$ , to problem rozmiaru  $k$  redukujemy do trzech podproblemów rozmiaru  $k-1$ . To zbyt skromna redukcja, by stosować metodę dziel i zwyciężaj. Z drugiej strony przestrzeń wszystkich podproblemów jest stosunkowo niewielka - składa się z  $nm$  elementów (zawiera wszystkie  $P(i, j)$  dla  $i = 1, \dots, n$ ,  $j = 1, \dots, m$ ), możemy więc zastosować programowanie dynamiczne.

```

for  $j = 1$  to  $m$  do  $d_{0,j} \leftarrow d_{n+1,j} \leftarrow \infty$ 
for  $i = 1$  to  $n$  do  $d_{i,1} \leftarrow a_{i,1}$ 
for  $j = 2$  to  $m$  do
    for  $i = 1$  to  $n$  do  $d_{i,j} \leftarrow a_{i,j} + \min\{d_{i-1,j-1}, d_{i,j-1}, d_{i+1,j-1}\}$ 
return  $\min\{d_{i,m} \mid i = 1, \dots, n\}$ 

```

Pozostaje wyjaśnić, w jaki sposób można odtworzyć optymalną trasę. Niech  $i_0$  będzie wartością  $i$ , dla której osiągane jest  $\min\{d_{i,m} \mid i = 1, \dots, n\}$ , a więc  $a_{i_0,m}$  jest ostatnim polem optymalnej trasy. Aby wyznaczyć przedostatnie pole wystarczy sprawdzić, która z trzech wartości  $d_{j,m-1}$  (dla  $j \in \{i_0-1, i_0, i_0+1\}$ ) jest minimalna. Postępując dalej rekurencyjnie wyznaczymy całą trasę.

```

procedure trasa( $i, j$ )
{
    if  $j = 1$  then return  $i$ 
    if  $d_{i-1,j-1} < d_{i,j-1}$  then  $k \leftarrow i - 1$  else  $k \leftarrow i$ 
    if  $d_{i+1,j-1} < d_{k,j-1}$  then  $k \leftarrow i + 1$ 
    return concat(trasa( $k, j - 1$ ),  $i$ )
}
.....
write(trasa( $i_0, m$ ))

```

## CYK

- dynamiczny algorytm sprawdzający, czy słowo należy do języka kontekstowego. Język kontekstowy musi być przedstawiony w postaci normalnej Chomskiego.  $O(n^3 \cdot |G|)$

$\downarrow$   
m-dł. słowa

$\downarrow$   
normalny gramatyki

Tworzymy tablicę  $T[i, j, x]$ , dla  $1 \leq i \leq j \leq n$ , zaś  $x$  przebiega wszystkie nieterminale (czy też równoważnie ich numery), wszystkie jej wartości ustawiając na 0

Dla każdego znaku  $a$  na pozycji  $i$ , i dla każdego  $X$  takiego, że w gramatyce jest produkcja  $X \rightarrow a$ , ustawiamy w tablicy  $T[i, 1, X] := 1$

Dla każdej długości  $i$  od 2 do  $n$ :

Dla każdego początku  $j$  od 1 do  $n - i + 1$ :

Dla każdego podziału  $k$  od 1 do  $i - 1$ :

Jeśli w tablicy są ustawione  $T[j, k, X]$  i  $T[j + k, i - k, Y]$ , a w gramatyce mamy produkcję  $Z \rightarrow XY$ , ustawiamy  $T[j, i, Z] := 1$

Słowo należy do języka, jeśli  $T[1, n, S] = 1$ , gdzie  $S$  to symbol startowy gramatyki

Dana jest gramatyka bezkontekstowa w postaci normalnej Chomsky'ego:

- [1]  $S \rightarrow AC$
- [2]  $C \rightarrow SB$
- [3]  $S \rightarrow AB$
- [4]  $A \rightarrow a$
- [5]  $B \rightarrow b$

Formalnie:

$$G = \{a^n b^n \mid n \geq 1\}$$

Pytanie:  $aabb \in G$ ?

Inicjalizacja tabeli:

|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | a | a | b | b | b |
| 2 |   |   |   |   |   |
| 3 |   |   |   |   |   |
| 4 |   |   |   |   |   |
| 5 |   |   |   |   |   |

Wyrazy długości 1:

pola  $[1, 1] = [1, 2] = \{A\}$ , z racji istnienia reguły [4]

pola  $[1, 3] = [1, 4] = [1, 5] = \{B\}$ , z racji istnienia reguły [5]

|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | a | a | b | b | b |
| 2 |   |   |   |   |   |
| 3 |   |   |   |   |   |
| 4 |   |   |   |   |   |
| 5 |   |   |   |   |   |

Wyrazy długości 2:

pole  $[2, 1] = \emptyset$ , ponieważ nie istnieje żadne reguła, która miałaby po prawej stronie ciąg symboli nieterminálnych  $AA$

|   | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 |   |   |   |   |   |
| 2 |   |   |   |   |   |
| 3 |   |   |   |   |   |
| 4 |   |   |   |   |   |
| 5 |   |   |   |   |   |

|   | 1   | 2   | 3   | 4   | 5   |
|---|-----|-----|-----|-----|-----|
|   | a   | a   | b   | b   | b   |
| 1 | {A} | {A} | {B} | {B} | {B} |
| 2 | -   |     |     |     |     |
| 3 |     |     |     |     |     |
| 4 |     |     |     |     |     |
| 5 |     |     |     |     |     |

pole [2, 2] = {S}, z racji produkcji [3]

|   | 1   | 2   | 3   | 4   | 5   |
|---|-----|-----|-----|-----|-----|
|   | a   | a   | b   | b   | b   |
| 1 | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} |     |     |     |
| 3 |     |     |     |     |     |
| 4 |     |     |     |     |     |
| 5 |     |     |     |     |     |

pole [2, 3] =  $\emptyset$ , ponieważ nie istnieje żadna reguła, która miałaby po prawej stronie ciąg symboli nieterminalnych BB

|   | 1   | 2   | 3   | 4   | 5   |
|---|-----|-----|-----|-----|-----|
|   | a   | a   | b   | b   | b   |
| 1 | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} | -   |     |     |
| 3 |     |     |     |     |     |
| 4 |     |     |     |     |     |
| 5 |     |     |     |     |     |

pole [2, 4] =  $\emptyset$ , ponieważ nie istnieje żadna reguła, która miałaby po prawej stronie ciąg symboli nieterminalnych BB

|   | 1   | 2   | 3   | 4   | 5   |
|---|-----|-----|-----|-----|-----|
|   | a   | a   | b   | b   | b   |
| 1 | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} | -   | -   |     |
| 3 |     |     |     |     |     |
| 4 |     |     |     |     |     |
| 5 |     |     |     |     |     |

Wyrazy długości 3:

pole [3, 1] =  $\emptyset$ , ponieważ nie istnieje żadna reguła, która miałaby po prawej stronie ciąg symboli nieterminalnych A lub tylko B

|   | 1   | 2   | 3   | 4   | 5   |   | 1   | 2   | 3   | 4   | 5   |
|---|-----|-----|-----|-----|-----|---|-----|-----|-----|-----|-----|
|   | a   | a   | b   | b   | b   |   | a   | a   | b   | b   | b   |
| 1 | {A} | {A} | {B} | {B} | {B} | 1 | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} | -   | -   |     | 2 | {S} | -   | -   | -   |     |
| 3 | -   |     |     |     |     | 3 | -   |     |     |     |     |
| 4 |     |     |     |     |     | 4 |     |     |     |     |     |

pole  $[3, 2] = \{C\}$ , z racji reguły [2]

| 1 | 2   | 3   | 4   | 5   | 1   | 2 | 3   | 4   | 5   |     |     |
|---|-----|-----|-----|-----|-----|---|-----|-----|-----|-----|-----|
| a | a   | b   | b   | b   | a   | a | b   | b   | b   |     |     |
| 1 | {A} | {A} | {B} | {B} | {B} | 1 | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} | -   | -   | 2   | - | {S} | -   | -   | -   |     |
| 3 | -   | -   |     |     | 3   | - | {C} |     |     |     |     |
| 4 |     |     |     |     | 4   |   |     |     |     |     |     |
| 5 |     |     |     |     | 5   |   |     |     |     |     |     |

pole  $[3, 3] = \emptyset$ , ponieważ nie istnieje żadna reguła, która miałaby po prawej stronie symbol  $B$

| 1 | 2   | 3   | 4   | 5   | 1   | 2 | 3   | 4   | 5   |     |     |
|---|-----|-----|-----|-----|-----|---|-----|-----|-----|-----|-----|
| a | a   | b   | b   | b   | a   | a | b   | b   | b   |     |     |
| 1 | {A} | {A} | {B} | {B} | {B} | 1 | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} | -   | -   | 2   | - | {S} | -   | -   | -   |     |
| 3 | -   | {C} | -   |     | 3   | - | {C} | -   |     |     |     |
| 4 |     |     |     |     | 4   |   |     |     |     |     |     |
| 5 |     |     |     |     | 5   |   |     |     |     |     |     |

Wyrazy długości 4:

pole  $[4, 1] = \{S\}$ , z racji reguły [1]

| 1 | 2   | 3   | 4   | 5   | 1   | 2 | 3   | 4   | 5   | 1   | 2   | 3   | 4   | 5   |     |     |     |
|---|-----|-----|-----|-----|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| a | a   | b   | b   | b   | a   | a | b   | b   | b   | a   | a   | b   | b   | b   |     |     |     |
| 1 | {A} | {A} | {B} | {B} | {B} | 1 | {A} | {A} | {B} | {B} | {B} | 1   | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} | -   | -   | 2   | - | {S} | -   | -   | 2   | -   | {S} | -   | -   | -   | -   | -   |
| 3 | -   | {C} | -   |     | 3   | - | {C} | -   |     | 3   | -   | {C} | -   |     |     |     |     |
| 4 | {S} |     |     |     | 4   | - |     |     |     | 4   | -   |     |     |     |     |     |     |
| 5 |     |     |     |     | 5   |   |     |     |     | 5   |     |     |     |     |     |     |     |

pole  $[4, 2] = \emptyset$ , ponieważ nie istnieje żadna reguła, która miałaby po prawej stronie symbol  $A$ ,  $S$  lub ciąg symboli nieterminalnych  $CB$ .

pole  $[4, 2] = \emptyset$ , ponieważ nie istnieje żadna reguła, która miałaby po prawej stronie symbol  $A$ ,  $S$  lub ciąg symboli nieterminalnych  $CB$ .

| 1 | 2   | 3   | 4   | 5   | 1   | 2 | 3   | 4   | 5   | 1   | 2   | 3   | 4   | 5   |     |     |     |
|---|-----|-----|-----|-----|-----|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| a | a   | b   | b   | b   | a   | a | b   | b   | b   | a   | a   | b   | b   | b   |     |     |     |
| 1 | {A} | {A} | {B} | {B} | {B} | 1 | {A} | {A} | {B} | {B} | {B} | 1   | {A} | {A} | {B} | {B} | {B} |
| 2 | -   | {S} | -   | -   | 2   | - | {S} | -   | -   | 2   | -   | {S} | -   | -   | -   | -   | -   |
| 3 | -   | {C} | -   |     | 3   | - | {C} | -   |     | 3   | -   | {C} | -   |     |     |     |     |
| 4 |     |     |     |     | 4   |   |     |     |     | 4   |     |     |     |     |     |     |     |
| 5 |     |     |     |     | 5   |   |     |     |     | 5   |     |     |     |     |     |     |     |

|   |     |   |
|---|-----|---|
| 4 | {S} | - |
| 5 |     |   |

|   |     |   |
|---|-----|---|
| 4 | {S} | - |
| 5 |     |   |

|   |     |   |
|---|-----|---|
| 4 | {S} | - |
| 5 |     |   |

Wyrazy długości 5:

pole [5, 1] = {C}, z racji reguły [2]

|       |     |     |     |     |       |     |     |     |     |       |     |     |     |     |       |     |     |     |     |
|-------|-----|-----|-----|-----|-------|-----|-----|-----|-----|-------|-----|-----|-----|-----|-------|-----|-----|-----|-----|
| 1     | 2   | 3   | 4   | 5   | 1     | 2   | 3   | 4   | 5   | 1     | 2   | 3   | 4   | 5   | 1     | 2   | 3   | 4   | 5   |
| a     | a   | b   | b   | b   | a     | a   | b   | b   | b   | a     | a   | b   | b   | b   | a     | a   | b   | b   | b   |
| 1 {A} | {A} | {B} | {B} | {B} | 1 {A} | {A} | {B} | {B} | {B} | 1 {A} | {A} | {B} | {B} | {B} | 1 {A} | {A} | {B} | {B} | {B} |
| 2 -   | {S} | -   | -   | -   | 2 -   | {S} | -   | -   | -   | 2 -   | {S} | -   | -   | -   | 2 -   | {S} | -   | -   | -   |
| 3 -   | {C} | -   | -   | -   | 3 -   | {C} | -   | -   | -   | 3 -   | {C} | -   | -   | -   | 3 -   | {C} | -   | -   | -   |
| 4 {S} | -   | -   | -   | -   | 4 {S} | -   | -   | -   | -   | 4 {S} | -   | -   | -   | -   | 4 {S} | -   | -   | -   | -   |
| 5 -   | -   | -   | -   | -   | 5 -   | -   | -   | -   | -   | 5 -   | -   | -   | -   | -   | 5 {C} | -   | -   | -   | -   |

Ponieważ symbol startowy  $S$  nie jest podzbiorem zbioru w polu [5, 1], czyli  $\{C\}$ , wyraz  $aabb$  nie jest elementem gramatyki  $G$ .

|           |       |       |     |     |     |
|-----------|-------|-------|-----|-----|-----|
| 5 letters | C,S,A | -     | B   | a   | b   |
| 4 letters | -     | A,S,C | -   | -   | -   |
| 3 letters | -     | B     | B   | -   | -   |
| 2 letters | A,S   | B     | S,C | A,S | -   |
| 1 letter  | B     | A,C   | A,C | B   | A,C |
|           | b     | a     | a   | b   | a   |

$$S \rightarrow AB \mid BC$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow CC \mid b$$

$$C \rightarrow AB \mid a$$

1. ba, aa, ab, ba
2. baa (b U aa  $\mid$  ba U a), aab (a U ab  $\mid$  aa U b), aba (a U ba  $\mid$  ab U a)
3. baab (baa U b  $\mid$  b U aab  $\mid$  ba U ab), aaba (aab U a  $\mid$  a U aba  $\mid$  aa U ba)
4. baaba (baa U ba  $\mid$  ba U aba  $\mid$  b U aaba)



