Programowanie dynamiczne

Problem (LCS)

Dane: $X = x_1, x_2, x_3, \dots, x_n$

$Y = y_1, y_2, \dots, y_m$

Wynik: Jakikolwiek $Z \in LCS(X,Y)$

Podciąg:

abaaba → aaaa

→ ε

$X = x_1, \dots, x_n$

$Z = z_1, \dots, z_k$

Z jest podciągiem X jeśli: $\exists_{i_1,\dots,i_k}\ i_1 < i_2 < \dots < i_k$

$\forall_j\ z_j = x_{i_j}$

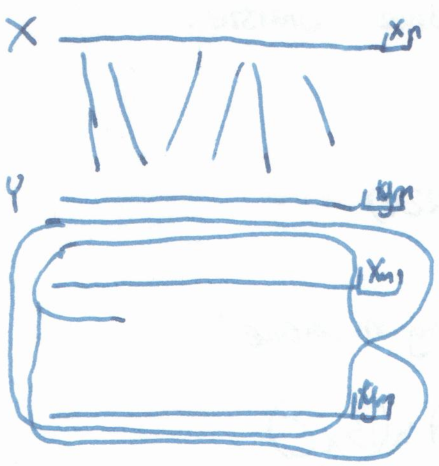
Ozn.

$LCS(X,Y) = \{Z :$ Z jest podciągiem X-a jak i Y-a i nie istnieje Z' o tej własności, który jest dłuższy od Z$\}$

Przykład: $X =$ ababab, $Y =$ bbbaaa

podciągi: ε, a, b, bbb, aaa, ba, itd. $LCS(X,Y) = \{$aaa, bbb, bba, baa$\}$

Ozn: $X = x_1, x_2, x_3, \dots x_n$, to $X_i = x_1, x_2, \dots, x_i$ ← i-literowy prefix



Spostrzeżenie:

- $x_n = y_m$ → to każdy ciąg z $LCS(X,Y)$ kończy się literą $x_n (= y_m)$

~~$x_n = y_m$ → każdy podciąg $LCS(X_{n-1}, Y_{m-1})$~~

czyli szukając $LCS(X,Y)$ możemy poszukać $LCS(X_{n-1}, Y_{m-1})$, a następnie dopisać do niego $x_n (= y_m)$

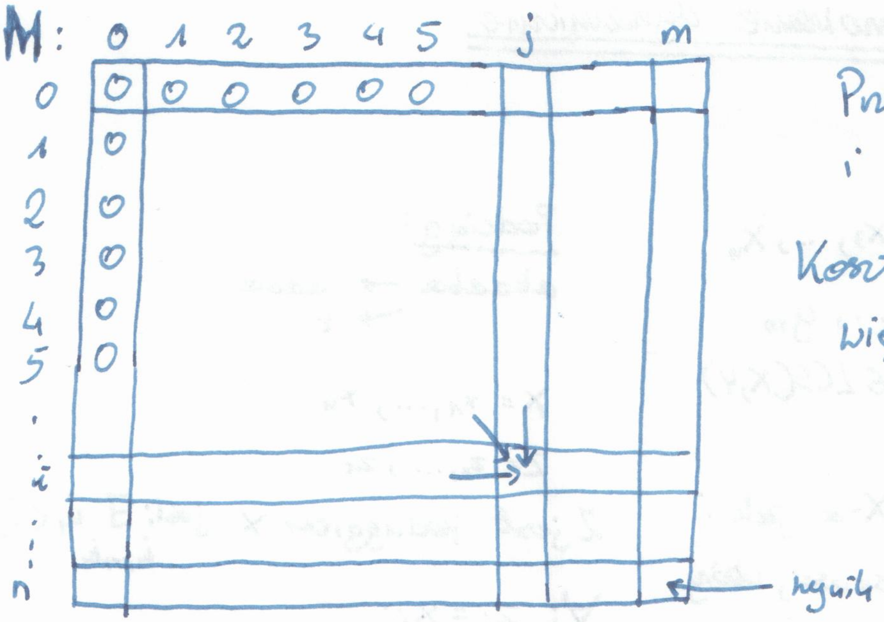
- $y_m \neq x_n$ → szukając $LCS(X,Y)$ możemy poszukać $LCS(X_{n-1}, Y)$ oraz $LCS(X, Y_{m-1})$ i wybrać dłuższy z nich

\# podproblemów $= O(nm)$

Ozn.

M_{ij} = "długość ciągów z $LCS(X_i, Y_j)$", $M_{i0} = 0\ \forall_{0 \le i \le n}$, $M_{0j} = 0\ \forall_{0 \le j \le m}$

$$M_{ij} = \begin{cases} 1 + M_{i-1,j-1} & x_i = y_j \\ \max(M_{i-1,j}, M_{i,j-1}) & x_i \neq y_j \end{cases}$$



Przechodzimy wierszami i siema

Koszt: $n \cdot m$ elementów $O(1)$, więc $O(n \cdot m)$

Jak z tablicy M odtworzyć jakiś ciąg z LCS?

Jeśli na i-tej i j-tej pozycji jest ta sama litera, to idziemy po skosie. Wpp. idziemy w kierunku ↑ lub ←, którego wartość w M jest taka sama. Zatem nas $O(n+m)$

Trochę lodi nas pamięć: gdy interesuje nas tylko wartość LCSa to easy: po prostu pamiętamy dwa wiersze.

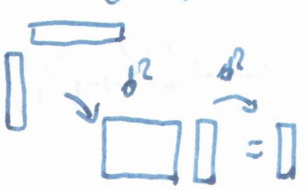
Problem (OPTYMALNA KOLEJNOŚĆ MNOŻENIA MACIERZY)

$\underset{(d_0\times d_1)}{M_1} \times \underset{(d_1\times d_2)}{M_2} \times \dots \times \underset{(d_{n-1}\times d_n)}{M_n}$, gdy mamy tylko trzy macierze:

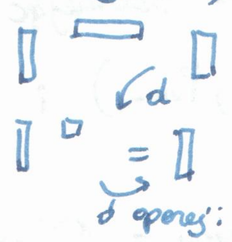
macierz

$(A\times B)\times C$ lub $A\times(B\times C)$

ilustracja problemu: $(\overset{d\times1}{A} \times \overset{1\times d}{B}) \times \overset{d\times1}{C}$, ale $\overset{d\times1}{A} \times (\overset{1\times d}{B} \times \overset{d\times1}{C})$



$\Theta(d^2)$ operacji



$\Theta(d)$ operacji

Założenie: iloczyn macierzy o rozmiarach $a\times b$ i $b\times c$ wymaga $\Theta(abc)$ operacji

Dane: $d_0, d_1, \ldots, d_n \in \mathbb{N}$ (interp: $d_{i-1} \times d_i$ to wymiary macierzy M_i)

Wynik: Kolejność wykonywania obliczeń w $M_1 \times \ldots M_n$ o minimalnym koszcie

Naiwnie:

$$M_1 \times M_2 \times \ldots \times M_i \times M_{i+1} \times \ldots \times M_n$$
$$(\qquad\qquad) \times (\qquad\qquad)$$

↑ to jest ostatnie

$S(n)$ = # sposobów przemnożenia ciągu n macierzy

$$S(n) = \begin{cases} S(n) = 2, & n = 2 \\ S(n) = \sum S(n-i)S(i), & n > 2 \end{cases} \leftarrow \text{liczba Catalana}$$

No ogólnie dramat

DP:

Podproblemy: OPT. KOLEJNOŚĆ obliczenia $M_i \times M_{i+1} \times \ldots \times M_j$

Podproblemów: $\Theta(n^2)$

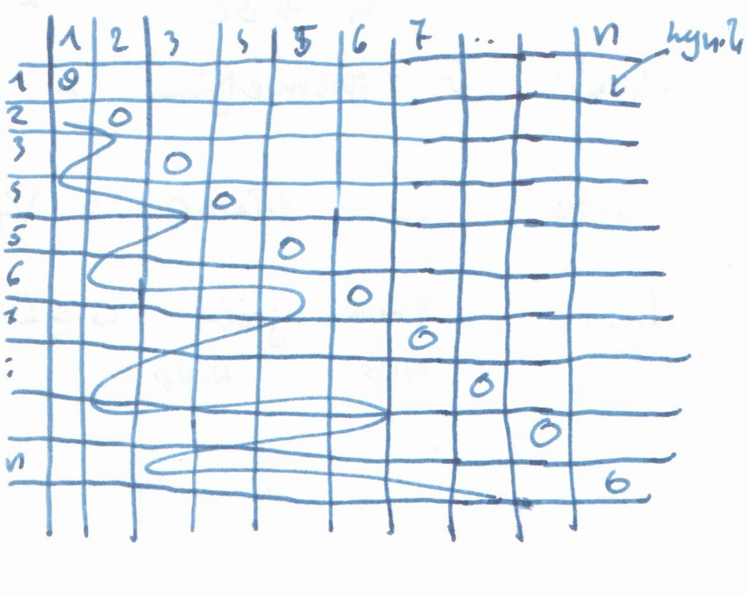
Ozn. m_{ij} = opt. koszt obliczenia $M_i \times \ldots \times M_j$

$$m_{ij} = \begin{cases} 0, & i = j \\ \min \{ m_{ik} + m_{k+1,j} + d_{i-1} d_k d_j, & i < j \end{cases}$$

$$\overbrace{M_i \times \ldots \times M_k}^{\text{wym: } d_{i-1} \times d_k} \times \overbrace{M_{k+1} \times \ldots \times M_j}^{\text{wym: } d_k \times d_j}$$

↑

$m_{ik} + d_{i-1} d_k d_j \, m_{k+1,j}$



Będziemy chcieli chodzić po przekątnych

Koszt obliczenia elementu z s-tej przekątnej.

$m_{i,i+s}$ – min po s wartościach; każda wartość obliczana w $\Theta(1)$

Koszt alg: $\sum_{s=0}^{n} \underbrace{(n-s)}_{\#el.} \cdot s = \Theta(n^3)$

Pytanie jak odtworzyć? Oprócz pamiętania wartości m_{ij} należy pamiętać liczbę k, która mówi że k-te mnożenie jest ostatnie i teraz schodzimy do m_{ik} i $m_{k+1,j}$

Problem: (Przynależność słowa do języka CFL)

REMINDER: Gramatyka bezkontekstowa

$$(\underbrace{V_T}_{\text{alf terminaly}}, \underbrace{V_N}_{\text{alf nieterm}}, \underbrace{P}_{\text{produkcje}}, S) \leftarrow \text{sym. początkowy } S_0 \in V_N$$

Produkcje: pary $A \to \alpha$ $\quad A \in V_N$

$\alpha \in (V_T \cup V_N)^*$

Gramatyka jest w postaci Chomsky'ego jeśli produkcje są postaci:

$A \to a \qquad A, B, C \in V_N$

$A \to BC \qquad a \in V_T$

Niech G - gramatyka w normalnej postaci Chomsky'ego

Dane: ω - SŁOWO z V_T^*

Wynik: TAK jeśli $\omega \in L(G)$

NIE w.p.p.

Naiwnie:

$x \in (V_N \cup V_T)^*$ $\quad F(x) = \{ y : y \in (V_N \cup V_T)^* \; x \Rightarrow y \}$

↑ w jednym kroku

$F_0 = \{S\}$

$F_{i+1} = \bigcup_{x \in F_i} F(x)$

Wystarczy policzyć F_i do $i \leq 2n+1$

na końcu

return $w \in F_{2n-1}$

DP:

$$(S, w) \swarrow \searrow$$

$(A, a_1, \dots, a_i)$ & $(B, a_{i+1} \dots a_n)$

Podproblemy (X, w)

$\swarrow \quad \searrow$

$X \in V_N$ $\quad$ podsłowo w

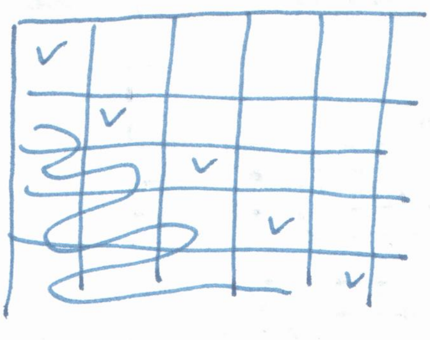
podproblemów $O(n^2)$

$$a_1 \dots \underbrace{a_i \dots a_j}_{m_{ij}} \dots a_n$$

Niech m_{ij} = "zbiór symb. nieterminalnych, z których da się wyprowadzić $a_i \dots a_j$"

$m_{ii} = \{ X : X \to a_i$ jest produkcją w $G\}$

$m_{ij} = \{ X : \exists_{i \leq k < j} \underbrace{\exists_{A,B: X \to AB} A \in m_{ik} \;\&\; B \in m_{k+1,j}}_{m_{ik} \otimes m_{k+1,j}} \}$ dla $i < j$



← przekątne tablicy

Koszt: $O(n^3)$ operacji $\otimes$

Jaki jest koszt pojedynczej operacji $\otimes$?

Koszt: $O(1)$

Wyjaśnienie op. $\otimes$:

$Z_1, Z_2 \subseteq V_N$, $\quad Z = Z_1 \otimes Z_2$, $\quad Z = \{ X \in V_N : \exists_{A \in Z_1, B \in Z_2} (X \to AB) \in P \}$

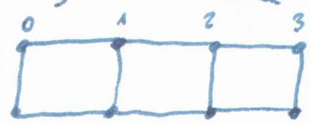
↑ produkcje w G

zatem inaczej $m_{ij} = \bigcup_{k=i}^{j-1} m_{ik} \otimes m_{k+1,j}$

Jak pamiętać mij? Wektory charakterystyczne, listy elementów

Problem (MST DRABIN)

D_3 - drabinka



Dane: n - rozmiar drabiny

$a_1\ b_1$
$\vdots$
$a_d\ b_d$

} krawędzie kolorowe

$k \in \mathbb{N}$

Wynik: # drzew rozpinających, które zawierają dokładnie k krawędzi kolorowych

1. Zapomnijmy o kolorowych krawędziach. Chcemy policzyć #MST drabiny D_n.

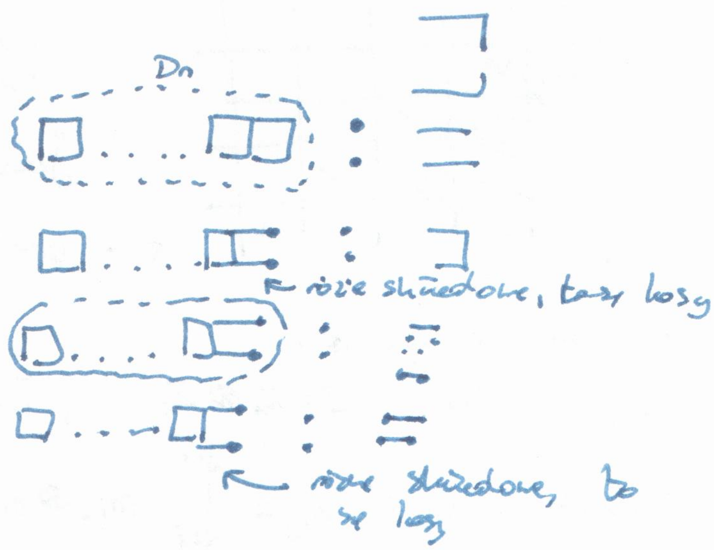
Ozn. S_n - # drzew spinających D_n, N_i - # lasów, o których może jest to

drzewo spin D_{n-1} przecięte z D_n może być:

- drzewem spinającym D_n
- lasem ~~[illegible]~~ spinającym D_n złożonym z dwóch drzew, takich że wierzchołki n i n' należą do różnych drzew

$$S_{n+1} = 3 \cdot S_N + N_N$$

$$N_{N+1} = 2 S_N + N_N$$



Jak to teraz zmodyfikować do krawędzi kolorowych?

Teraz musimy obliczać $S_i(r)$ – # drzew spinających z r kolorowymi krawędziami oraz $N_i(r)$ # lasów spinających z r kolorowymi krawędziami

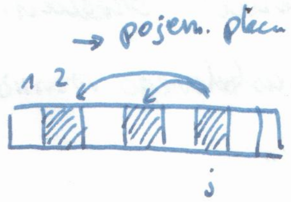
Problem (PLECAKOWY Z POWTÓRZENIAMI)

Dane: $w_1, w_2, \dots, w_n \in \mathbb{N}$ $\quad W \in \mathbb{N}$

$v_1, v_2, \dots, v_n \in \mathbb{R}$

interp: mamy n przedmiotów, w_i to ich wagi, a v_i to ich wartości, a W to pojemność plecaka

Wynik: wielozbiór $\{i_1, i_2, \dots, i_k\}$, t. że $\sum_{j=1}^{k} w_{i_j} \leq W$ oraz $\sum_{j=1}^{k} v_{i_j}$ - maksym



t_i - max wartość przedmiotów, które zmieszczą się w plecaku o poj. i

$$t_i = \begin{cases} 0, & \text{jeśli } i=0 \\ \max\limits_{w_l < i} \{ t_{i-w_l} + v_l \} & \text{jeśli } i>0 \end{cases}$$

koszt: $O(W \cdot n)$

wszystkich przedmiotów

pojemności

Mówimy o koszcie wielomianowym, gdy koszt zależy od rozmiaru danych. Tutaj alg. nie zależy tylko od n, bo gdy pozostawimy n bez zmian, a zmienimy W to zmieni się koszt. Taki algorytm nazywamy pseudowielomianowy.