Mnożenie macierzy

czwartek, 6 czerwca 2024 11:44

$(A \cdot B) \cdot C$

$A \cdot (B \cdot C)$

⇒ może to być inna praca, chcemy znaleźć to tańsze

A: $d \times 1$, B: $1 \times d$, C: $d \times 1$

$A \cdot B \Rightarrow d \times d$ → $O(d^2)$

$(A \cdot B) \cdot C \Rightarrow d \times 1$, $O(d^2)$

$B \cdot C \Rightarrow 1 \times 1$ → $O(d)$

$(B \cdot C) \cdot A \to O(d)$

czyli to się bardzo opłaca kontrolować

$M_1, M_2 \dots M_n$ → jest tyle najmniej rozwiązań, ile jest różnych nawiasowań → DUŻO → liczba Catalana → ta metoda do zapomnienia

wykładniczo dużo

$d_0 \times d_1 \quad d_1 \times d_2 \quad \dots \quad d_{n-1} \times d_n$

Dane: $d_0 \dots d_n$

$\underbrace{M_1 \dots M_{k-1}}_{d_0 \times d_{k-1}} \times \underbrace{M_k \dots M_n}_{d_k \times d_n}$ ← spekulacja

$\times$ → ostatnie mnożenie ⇒ czyli to będzie $d_0 \cdot d_{k-1} \cdot d_n$

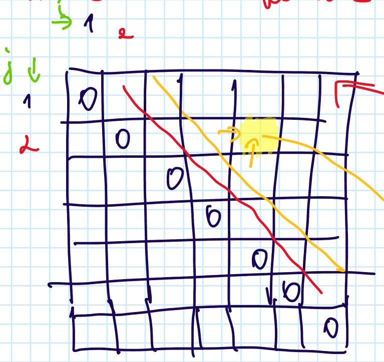
założenie: Obliczenie macierzy $A \cdot B$ gdzie A ma wymiar $a \times b$ a B wymiar $b \times c$ ma koszt: $\Theta(a \cdot b \cdot c)$

Podproblem: Obliczyć optymalny koszt obliczenia $M_i \times \ldots M_j$ ← oznaczenie m_{ij} dla $i \leq j$

liczba podproblemów: $O(n^2)$

czyli m_{1n} to opt. koszt obliczenia $M_1 \times M_{1+1} \ldots \times M_n$

M_i $\forall_i$ $m_{ii} = 0$ koszt policzenia jednej macierzy



tu jest wynik

czyli żeby policzyć to pole musimy rekurencyjnie policzyć → ↑ te pola i dodatkowo policzyć koszt ich mnożenia

$$m_{ij} = \min_{i \leq k < j} \left(m_{ik} + m_{k+1\,j} + \underbrace{d_{i-1} \cdot d_k \cdot d_j}_{\text{macierz która powstanie z tego mnożenia}} \right)$$

Policzenie $M_i \times \ldots \times M_k$

$M_{k+1} \times \ldots \times M_j$

Bo wiemy jeszcze mamy coś od $M_0 \ldots M_{i-1}$

Koszt obliczenia macierzy m: $\sum_{s=0}^{n-1} \Theta(s) \cdot (n-s) = O(n^3)$

s – przekątne

$\Theta(s)$ – koszt wyliczenia elementu

$(n-s)$ – liczba elementów na s-tej przekątnej

procedure $dyn - matmult(d[0..n])$;
 int $m[1..n, 1..n]$, $p[1..n, 1..n]$
 for $i \leftarrow 1$ **to** n **do** $m_{ii} \leftarrow 0$;
 for $s \leftarrow 1$ **to** $n-1$ **do**
  for $i \leftarrow 1$ **to** $n-s$ **do**
   $j \leftarrow i + s$
   $m_{ij} \leftarrow \min_{i \leq k < j}(m_{i,k} + m_{k+1,j} + d_{i-1} d_k d_j)$
   $p_{ij} \leftarrow$ "to k, przy którym osiągane było minimum dla m_{ij}"

procedure $dyn - matmult(d[0..n])$;
  int $m[1..n, 1..n]$, $p[1..n, 1..n]$
  for $i \leftarrow 1$ **to** n **do** $m_{ii} \leftarrow 0$;
  for $s \leftarrow 1$ **to** $n-1$ **do**
    for $i \leftarrow 1$ **to** $n-s$ **do**
      $j \leftarrow i + s$
      $m_{ij} \leftarrow \min_{i \le k < j}(m_{i,k} + m_{k+1,j} + d_{i-1} d_k d_j)$
      $p_{ij} \leftarrow$ "to k, przy którym osiągane było minimum dla m_{ij}"
  return $p[1..n, 1..n]$

Problem plecakowy

środa, 14 sierpnia 2024 22:50

n przedmiotów ?

wagi przedmiotów → $w_1, \dots, w_n \in \mathbb{N}$

wartości przedmiotów → $v_1, \dots, v_n \in \mathbb{R}$

liczba $W \in \mathbb{N}$ pojemność plecaka

Wynik: przedmioty w plecaku które mają największą wartość

cyli ~~wielozbiór~~ zbiór $\{i_1, \dots, i_k\}$ t. że $\sum_{j=1}^{k} w_{i_j} \le W$ oraz $\sum_{j=1}^{k} v_{i_j}$ maksymalne

wersje z powtórzeniami

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

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

Czas działania: $O(nW)$

Przykład

w_i	3	4	7
v_i	2	3	6

$K(w)$ dla $w = 0, 1, 2, \dots, W$

0	1	2	3	4	5	6	7	8	9	10
0	0	0	2	3	3	4	6			

...... itd

zaglądamy do poprzednich pojemności by wybrać to co się najbardziej opłaca

$2 + K[0]$

dwie opcje:
$3 + K[0]$ bo ma wagę 4
lub $2 + K[1]$ bo ma wagę 3

to wybieramy

bo dwa razy bierzemy przedmiot o wadze 3

rozmiar danych problemu

$w_1 \quad w_n$
$v_1 \quad v_n$
W

$\}$ # liczb to $2n+1 = \bar{n}$

dla takich problemów rozmiarem danych jest liczba liczb

Chcemy by złożoność to była funkcja rozmiaru

cyli $O(n \cdot W) = f(\bar{n})$

W ma się większ do n'a

Zatem ten problem ma złożoność wielomianową względem wielkości liczb

Nasz problem będzie rozwiązać tym więcej im większa jest wartość W

zatem nasz algorytm jest pseudowielomianowy - bo rozwiązanie zależy od wartości liczb

W przypadku bez powtórzeń:

n przedmiotów ?
wagi przedmiotów → $w_1, \dots, w_n \in \mathbb{N}$
wartości przedmiotów → $v_1, \dots, v_n \in \mathbb{R}$

liczba $W \in \mathbb{N}$ pojemność plecaka

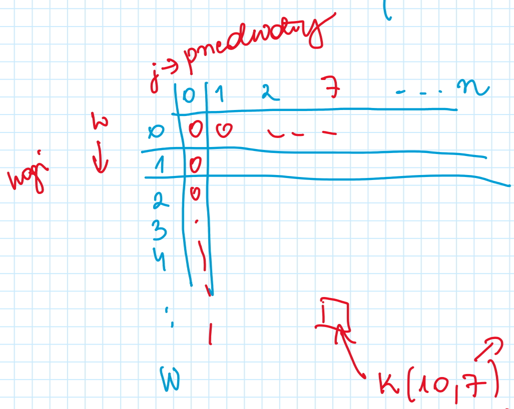
Wynik: przedmioty w plecaku które mają największą wartość

cyli ~~wielozbiór~~ zbiór $\{i_1, \dots, i_k\}$ t.że $\sum_{j=1}^{k} w_{i_j} \le W$ oraz $\sum_{j=1}^{k} v_{i_j}$ maksymalne

Podproblem: Mniejszy plecak pakowany podzbiorem przedmiotów

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

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



→ tablica ma rozmiar tyle, ile jest podproblemów cyli $O(W \cdot n)$

$K(10,7)$ → przedmioty od 1 do 7

Cyli rozważany przedmiot 7 może on uczestniczyć w tym zapakowaniu lub może okazać się będzie można optymalnie zapakować plecak bez niego

Przynależność do języka bezkontekstowego

środa, 14 sierpnia 2024 23:38

Σ → alfabet

PRZYKŁAD

$w \in \{0,1\}^*$: w jest rozwinięciem binarnym liczby pierwszej

Gramatyka = $\langle V_N, V_T, P, S \rangle$

dwie listy, x mała

V_N – nieterminale, V_T – terminale, P – produkcje, $S \in V_N$ – symbol startu

$A \to \alpha$, α – napis nad V_N lub V_T

$A \in V_N$

$A \to abA$

$S \to AB$

Interesuje nas jakie słowa możemy utworzyć nad tym alfabetem z liter terminalnych

$V_T = \{ (,), [,] \}$

$S \to ()$
$S \to []$
$S \to SS$
$S \to A$
$S \to B$

$B \to [A]$
$B \to (B)$
$A \to []$
$B \to ()$

Chcemy wyprowadzić (()) []

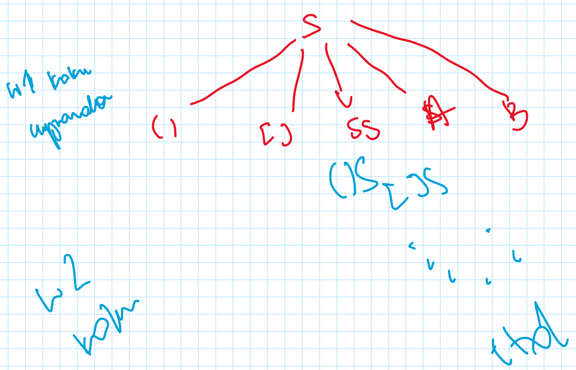
$S \to SS \to BS \to (B)S \to (B)[\,] \to (())[\,]$

Będziemy mieli zadaną gramatykę i słowo i chcemy utworzyć algorytm który będzie sprawdzał czy to słowo możemy utworzyć w tej gramatyce

Będziemy używać postaci Chomskyego

$A \to XY$ $A \to a$

Możemy zrobić drzewo wyprowadzeń



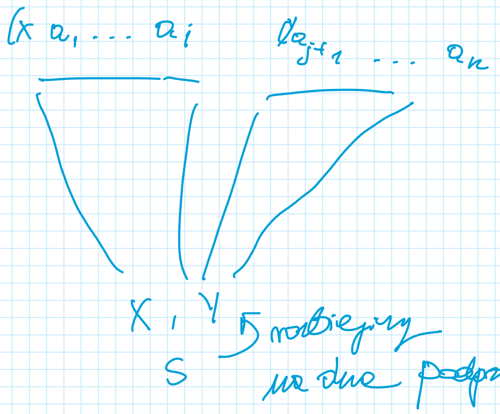
Algorytm naiwny: sprawdzamy, co się da wyprowadzić w poszczególnych

Kodeki drzewa wyprowadzeń

Fakt: słowa w gramatyce Chomskiego n literowe słowa da się ograniczyć do $2n-1$ poziomów drzew wyprowadzeń

↓

na ostatnim poziomie będą wszystkie możliwe wyprowadzenia



rozbiegający na dwa podproblemy

podproblemów $O(|V_n| \cdot n^2)$

$|V_n|$ → liczba nieterminali

n^2 → spójnych podciągów

$a_1 \ldots a_n$

$a_1 a_2, a_3 \ldots a_n$

⋮

itd