Dziel i zwyciężaj

piątek, 31 maja 2024 11:43

Master theorem

$$T(n) = \begin{cases} b & \text{dla } n=1 \\ aT\left(\frac{n}{c}\right) + bn & \text{dla } n>1 \end{cases}$$

rozwiązaniem równania jest

$$T(n) = \begin{cases} \Theta(n) & a<c \\ \Theta(n \log n) & a=c \\ \Theta(n^{\log_c a}) & a>c \end{cases}$$

Mnożenie bardzo dużych liczb - algorytm Karatsuby

$A = \underbrace{a_{n-1}\ a_{n-2}\ \dots}_{A_1}\ \underbrace{a_2\ a_1\ a_0}_{A_0}$

$B = \underbrace{b_{n-1}\ b_{n-2}\ \dots}_{B_1}\ \underbrace{b_2\ b_1\ b_0}_{B_0}$

Klasycznie: $\Theta(n^2)$

$$A \cdot B = (A_1 \cdot 2^{\frac{n}{2}} + A_0)\cdot(B_1 \cdot 2^{\frac{n}{2}} + B_0) = \underbrace{A_1 B_1}_{C_0} \cdot 2^n + \underbrace{(A_1 \cdot B_0 + A_0 B_1)}_{C_1} 2^{\frac{n}{2}} + \underbrace{A_0 B_0}_{C_2}$$

mało zmyślny sposób, doprowadzi nas też do złożoności $\Theta(n^2)$

Algorytm Karatsuby będzie używał 3 zamiast 4 mnożeń:

$Z \leftarrow A_0 B_0 - C_0$

$X \leftarrow A_1 B_1 - C_2$

$$Y = (A_0 + A_1)(B_0 + B_1) - C_0 - C_2 = A_0B_0 + A_0B_1 + A_1B_0 + A_1B_1 - C_0 - C_2$$

$= C_1$

jedno mnożenie — czyli korzystamy z wcześniej policzonych mnożeń

bo problem jest przy procedurze multiply(a_0+a_1, b_0+b_1)

może się okazać, że a_0+a_1 będzie liczbą $\frac{n}{2}+1$ bitową, tak samo b_0+b_1

Czyli algorytm Karatsuby ma złożoność:

$$T(n) = 2T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}+1\right) + \Theta(n)$$

a_0+a_1: $[a'|a'']$ $= \frac{n}{2}+1 = a' \cdot 2^{\frac{n}{2}} + a''$

b_0+b_1: $[b'|b'']$ $= \frac{n}{2}+1 = b' \cdot 2^{\frac{n}{2}} + b''$

najbardziej znaczący bit

$$\Downarrow$$

$$a' \cdot b' \cdot 2^n + (a'' \cdot b' + a' \cdot b'') \cdot 2^{\frac{n}{2}} + a'' \cdot b''$$

- $a' \cdot b' \cdot 2^n$ — 0 lub 1 przesunięte o 2^n
- $(a'' \cdot b' + a' \cdot b'')$ — dodanie dwóch liczb $\frac{n}{2}$ bitowych w czasie liniowym + przesunięcie
- $a'' \cdot b''$ — jedyne mnożenie $\frac{n}{2}$ bitowych liczb

```
multiply(a, b)
    n ← max(|a|, |b|)    (* |x| oznacza długość liczby x *)
    if n jest małe then  pomnóż a i b klasycznym algorytmem
                         return obliczony iloczyn
    p ← ⌊n/2⌋
    a1 ← ⌊a/2^p⌋; a0 ← a mod 2^p
    b1 ← ⌊b/2^p⌋; b0 ← b mod 2^p
    z ← multiply(a0, b0)
    y ← multiply(a1 + a0, b1 + b0)
    x ← multiply(a1, b1);
    return 2^(2p) x + 2^p (y − x − z) + z
```

zadanie z egzaminu:

STUDENT DEBIL

???

3 rekurencje

$z \in$ mult(a_0, b_0)

$y \in$ mult$(a_1+a_0, b_1+b_0) = a_1b_1 + a_1b_0 + a_0b_1 + a_0b_0$

$x \in$ mult$(a_1 - a_0, b_1 - b_0)$

$\quad = a_1b_1 - a_0b_1 - a_1b_0 + a_0b_0$

Potrzebujemy: $a_1 b_1 \cdot 2^{2s} + (a_0 b_1 + a_1 b_0) 2^s + a_0 b_0$

$$2^{2p}\left(\frac{x+y}{2} - z\right) + 2^p\left(\frac{y-x}{2}\right) + z$$

Jedyna wątpliwość: $a_0 + a_1$ mogą być postaci $n/2+1$ bit

Zatem zapiszemy je jako a'' i a'

pojedynczy bit najbardziej znaczący

$a'' \cdot 2^{\frac{n}{2}}$ i a'

$$(a_1+a_0)(b_1+b_0) = (a'' \cdot 2^{\frac{n}{2}} + a')(b'' \cdot 2^{\frac{n}{2}} + b') =$$

$$= \underbrace{a''b''2^n + a''b' \cdot 2^{\frac{n}{2}} + a'b'' \cdot 2^{\frac{n}{2}}}_{\text{zwykłe mnożenia przez stałe } (a'', b'')} + a'b'$$

$a'b'$ — tu rekurencyjne mnożenie jedno, czyli tak jak miało być

przez stałe(...) ... jedno, cyfr ... może być

czyli zmiany mamy to złożoność jedynie o pewną stałą

wzór ogólny Karatsuba dla k podziałów:

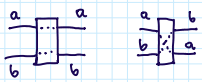
$$\log_k(2k-1) \Rightarrow O\left(n^{\log_k 2k-1}\right)$$

* Karatsuba dla liczb nie 2^k tylko z jednym bitem

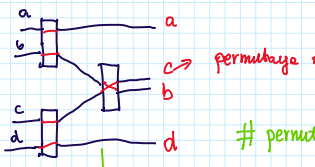
Sieci przełączników

niedziela, 18 sierpnia 2024 11:41

Sieć składa się z:
- przełączników
- drutów



Połączenia tych przełączników utworzą nam pewną "magiczną" dzięki której otrzymamy permutację wejściowych elementów:



permutacja a, b, c, d

permutacji n elementowych = n!

to nie będzie sieć ojaka nam chodzi, bo będzie $2^3 = 8$ różnych permutacji a nie $4! = 24$

Zadanie

Dane: n

Mamy skonstruować sieć $Perm_n$ realizującą WSZYSTKIE permutacje z S_n.

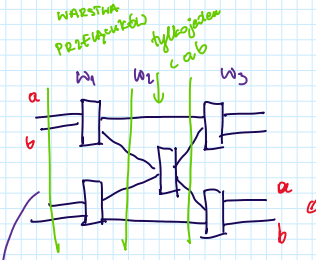
Przykład

$n = 4 \Rightarrow$ 24 permutacje

Czyli liczba przełączników k $2^k \geq 24$ musi być +.ie

$\Downarrow$

$k \geq 5$



tak się nie da

Ale zauważmy, że ta sieć nie jest dobra, bo a, b nie mogą obie przejść via dobry przełącznik

głębokość tej sieci jest równa 3

FAKT:
liczba wejść i wyjść drutów
w sieci przełączników jest taka sama

Kryterium głębokości sieci:
długość maksymalnej drogi od portu wejściowego do wyjściowego (czyli liczba przełączników przez jakie przejdziemy)

Mamy dane n stanów

przełączników: k mamy, że musi zachodzić $2^k \geq n!$ czyli $k \geq n \log n$ ze wzoru Stirlinga

czyli wszystkich przełączników musi być $n \log n$

a minimalna liczba warstw to $\log n$.

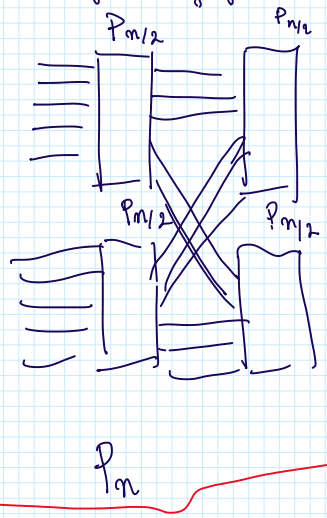
czyli na każdej warstwie $\frac{n}{2}$ przełączników

minimalna głębokość: d:

$2^d \geq n$

$d \geq \log n$

Próba konstrukcji komutującej z dwóch i wyczerpującej zał: $n = 2^k$



Niech d_n = głębokość P_n

$$d_n = 2 \cdot d_{n/2} = 2 \cdot (2 \cdot d_{n/4}) = 4 \cdot d_{n/4} = 8 \cdot d_{n/8} = \frac{n}{2} \cdot \overset{\uparrow}{\underset{\parallel}{d}}_2 = \frac{n}{2}$$

czyli wychodzi słabo

tu chcemy głębokość 1

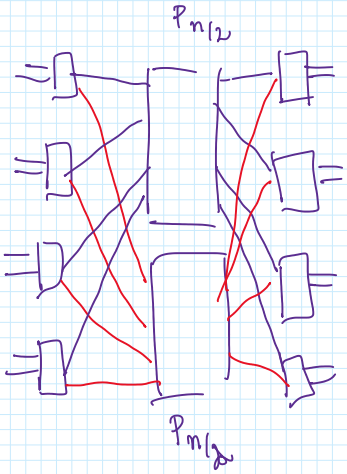
$d_n = d_{n/2} + C$

po dojściu do $\log n$

w_1 - przełączniki, których wejścia są dwukami wejściowymi

w_i - przełączniki, których wejścia są dwukami wejściowymi lub wyjściowymi z poprzednich warstw w_j $(j < i)$

Sieć BW



głębokość : $2 \log n - 1$

$$G(2^k) = \begin{cases} 1 & k=1 \\ G(2^{k-1}) + 2 & k>1 \end{cases}$$

liczba przełączników $\Theta(n \log n)$

$$P(2^k) = \begin{cases} 1 & k=1 \\ 2P(2^{k-1}) + 2^k & k>1 \end{cases}$$

Dlaczego to dwójka?