DZIEL I ZWYCIĘŻAJ

Przykłady: Mergesort, Quicksort

Pytanie: Jak scalać mergesorta w miejscu?

X - dane

0° Jeśli X małe, to rozw. ad hoc

1° Podziel X na $X_1, X_2, \dots X_n$

2° $\forall i$ $y_i \leftarrow$ rozwiązanie problemu na X_i

3° Utwórz rozwiązanie dla X z $y_1 \dots y_n$

Tw.

Niech $a, b, c \in \mathbb{N}$

Rozwiązaniem równania

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

dla n będących potęgą liczby c jest

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

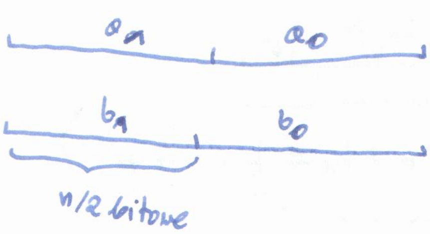
Problem (mnożenie liczb)

Dane: $a, b \in \mathbb{N}$ (n-bitowe)

Wynik: c (t. że $c = a \cdot b$)

Ilustracja:

Zał. $n = 2^k$



$a = a_1 \cdot 2^{n/2} + a_0$

$b = b_1 \cdot 2^{n/2} + b_0$

Wówczas:

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

Mult (a, b)

if n-mała to adhoc (a,b)

$x_0 \leftarrow \text{Mult}(a_0, b_0)$

$x_1 \leftarrow \text{Mult}(a_1, b_0)$

$x_2 \leftarrow \text{Mult}(a_0, b_1)$

$x_3 \leftarrow \text{Mult}(a_1, b_1)$

return $x_3 2^n + (x_1 + x_0) 2^{n/2} + x_0 \rightsquigarrow$ czas $O(n)$

$$T(n) = \underbrace{4\,T(n/2)} + O(n)$$

ale to jest $O(n^2)$, więc słabo

Więc chcemy zaoszczędzić na liczbie mnożeń, z 4 chcemy zmniejszyć do trzech.

$x_0 \leftarrow a_0 b_0$

$x_1 \leftarrow (a_0 + a_1)(b_0 + b_1)$

$x_2 \leftarrow a_1 \cdot b_1$

3 wywołanie rek.

return $x_2 2^n + (x_1 - x_2 - x_0) 2^{n/2} + x_0$

Stąd $T(n) = 3T(n/2) + O(n) = O(n^{\log_2 3})$

Zauważmy, że $(a_0 + a_1)$ i $(b_0 + b_1)$ ~~a_1~~ może być ~~$n/2$~~ $+1$ bitowe

$a_0 + a_1$: a' | a''

$b_0 + b_1$: b' | b''

$a_0 + a_1 = a' \cdot 2 + a''$

$b_0 + b_1 = b' \cdot 2 + b''$

1 bitowy

$n/2$ bitowe

Uogólnienie metody: podzielić na 3 części

$$a = \; | \; a_2 \; | \; a_1 \; | \; a_0 \; | \qquad a = a_2 \cdot 2^{\frac{2}{3}n} + a_1 \cdot 2^{\frac{1}{3}n} + a_0$$

$$b = \; | \; b_2 \; | \; b_1 \; | \; b_0 \; | \qquad b = b_2 2^{\frac{2}{3}n} + b_1 2^{\frac{1}{3}n} + b_0$$

$$a \cdot b = \underbrace{(a_2 b_2)}_{c_4} \cdot 2^{\frac{4}{3}n} + \underbrace{(a_2 b_1 + a_1 b_2)}_{c_3} 2^{\frac{3}{3}n} + \underbrace{(a_2 b_0 + a_1 b_1 + a_0 b_2)}_{c_2} 2^{\frac{2}{3}n} +$$

$$+ \underbrace{(a_1 b_0 + a_0 b_1)}_{c_1} 2^{\frac{1}{3}n} + \underbrace{(a_0 b_0)}_{c_0}$$

$T(n) = 5 \cdot T\left(\frac{n}{3}\right) + \Theta(n)$, rozw. tego równania $T(n) = \Theta\left(n^{\log_3 5}\right)$

Pytanie teraz jak liczyć $c_0, \dots, c_4$, bo widać, że to jest szybsze od podziału na dwa?

$$w_0 = a_0 b_0 = c_0$$
$$w_1 = a_2 b_2 = c_4$$
$$w_2 = (a_0 + a_1 + a_2)(b_0 + b_1 + b_2) = c_0 + c_1 + c_2 + c_3 + c_4$$
$$w_3 = (a_0 - a_1 + a_2)(b_0 - b_1 + b_2) = c_0 - c_1 + c_2 - c_3 + c_4$$
$$w_4 = (a_0 + 2a_1 + 4a_2)(b_0 + 2b_1 + 4b_2) = c_0 + 16c_4 + 2c_1 + 4c_2 + 8c_3$$

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 2 & 4 & 8 & 16 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

tu np. może być 1, 3, 9, 27, 81

Mult(a, b)

Niech $a_0, a_1, a_2, b_0, b_1, b_2$ to części a b

$w_1 \leftarrow$ Mult(a_0, b_0)

$\vdots$

$w_4 \leftarrow$ Mult$(a_0 + 2a_1 + 4a_2, \; b_0 + 2b_1 + 4b_2)$

$c_j \leftarrow \sum_{i=0}^{n} a_{ji} w_i$ dla $j = 0, 1, \dots 4$

return $c_4 2^{4/3 n} + \dots + c_0$

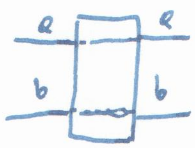
Przy podziale na k części:

- a_i, b_j są $\frac{n}{k}$ bitowe
- # niewiadomych $= 2k-1$

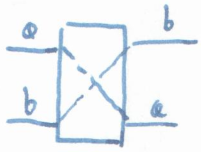
$$T(n) = (2k-1)\,T\left(\frac{n}{k}\right) + \Theta(n) = $$
$$= \Theta\left(n^{\log_k 2k-1}\right)$$

Sieci przełączników

st.0



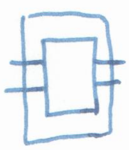
st.1



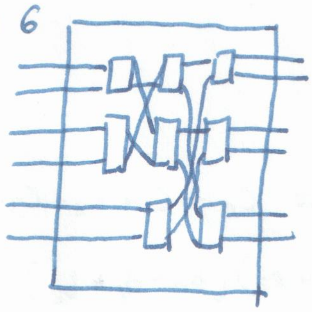
<u>Cel:</u> $\forall n$ skonstruować sieć przełączników o n wejściach takich, która umożliwia otrzymanie dowolnej permutacji inf wejściowych, ale taką, żeby była jak najpłytsza

Dla prostoty $n = 2^k$

$k=1$



$n: 6$



← to jest przykład, że się nie da. Chcemy 6! wyników, ale różnych układów sieci jest 2^8, co więcej ~~te~~ różne układy przełączników mogą generować te same permutacje

$2^8 = 256 < 6! = 720$

<u>Parametry sieci:</u>

- rozmiar $S(n)$ – # przełączników
- głębokość $D(n)$ – # warstw

<u>Ograniczenie na głębokość:</u>

$$D(n) \geq \log n$$

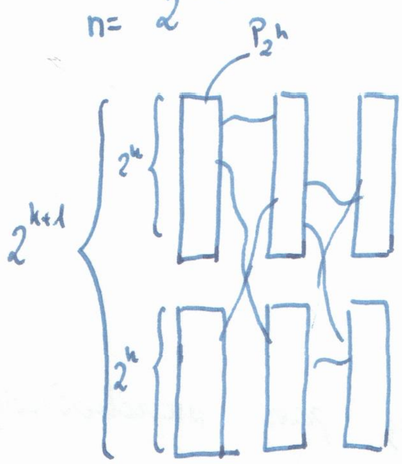
<u>Ograniczenie na rozmiar:</u>

$$2^{S(n)} \geq n!$$

$$S(n) \geq \log(n!)$$

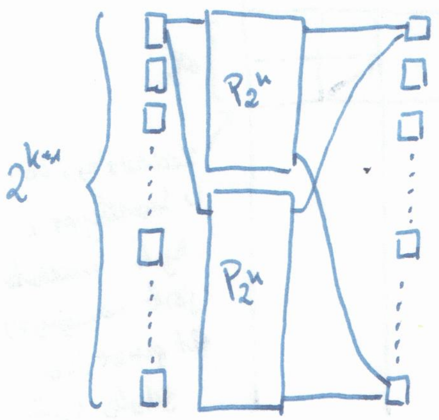
Niech P_n to sieć permutacyjna (taka, która umożliwia otrzymanie wszystkich permutacji) o n wejściach.

$n = 2^{k+1}$



$$D(n) = \begin{cases} 3\,D(\frac{n}{2}) \\ 1 \end{cases} \qquad D(n) = \Theta(n^{\log_2 3})$$

Sieć Beneša - Waksmana



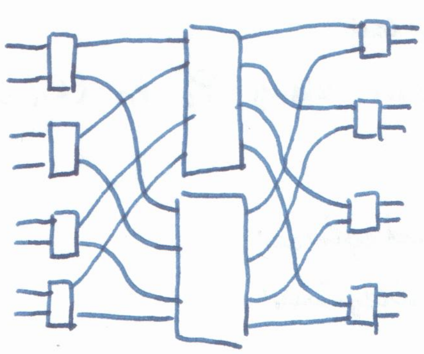
Fakt: Sieć B-W ma głębokość

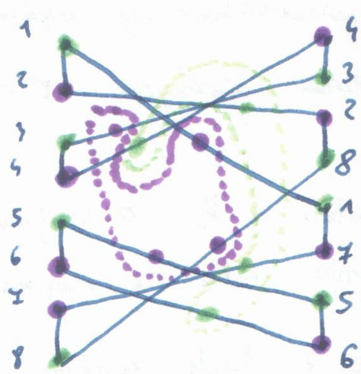
$$D(n) = \begin{cases} 1 & \text{dla } n=2 \\ D(\frac{n}{2})+2 & \text{dla } n>2 \end{cases} = \Theta(\log n)$$

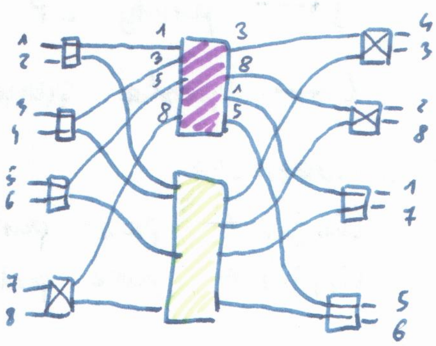
A jeśli chodzi o rozmiar

$$S(n) = D(n) \cdot \frac{n}{2} = \Theta(n \log(n))$$

Zostaje do pokazania, że ta sieć jest dobra







Fakt: Każdy wierzchołek w tym grafie ma st $= 2$

$\forall \pi$ Graf G_π jest sumą rozłącznych cykli (o długości parzystej)

Dana jest $\pi : 1 \ldots n \to 1 \ldots n$

Najbliższa para punktów

Dane: $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, $\dots$ $p_n = (x_n, y_n)$ ← punkty $\mathbb{R}^2$

Zadanie: Znaleźć $1 \leq i_1 \neq i_2 \leq n$

$$d(p_{i_1}, p_{i_2}) = \min_{i \neq j} d(p_i, p_j)$$

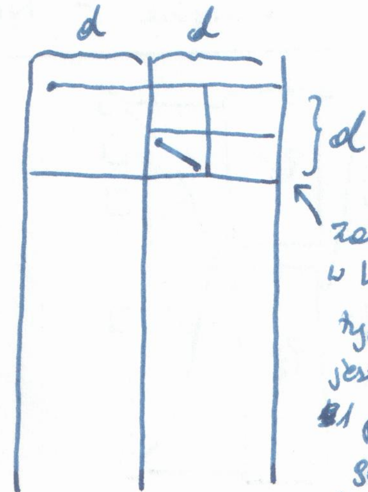
odległość Euklidesowa

Alg naiwny: obliczyć odległości między każdą parą wierzchołków, zatem $O(n^2)$

Teraz lepiej

$P = \{p_1, p_2, \dots, p_n\}$

$d \leftarrow \min(d_L, d_R)$



zauważ, że w każdym z tych kwadratów jest conajwyżej 1 punkt, bo gdyby były dwa to wynik P_R nie byłby by d, gdzie $\frac{d\sqrt{2}}{2}$

Alg

1° $X \leftarrow$ punkty z P posortowane wg współrzędnej x-owej

$Y \leftarrow$ punkty z P posortowane wg współrzędnej y-owej

$l \leftarrow$ prosta pionowa dzieląca P na równoliczne zbiory P_L, P_R (± 1 element)

2° REKURENCJA

$(i_1, j_1) \leftarrow$ para punktów z P_L o najmniejszej odległości

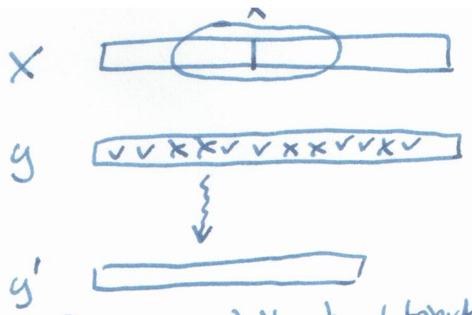
$(i_2, j_2) \leftarrow$ para punktów z P_R o najmniejszej odległości

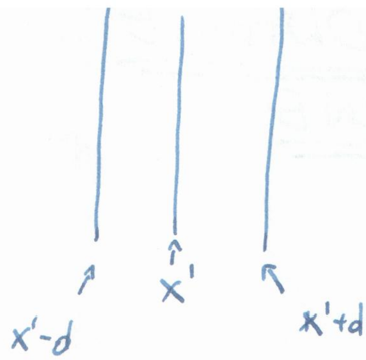
3° Niech $(i', j') \leftarrow$ lepsza z tych dwóch poprzednich

$d =$ odl między i' a j'

Sprawdź, czy istnieje para (t, s) $t \in P_L$, $s \in P_R$ t. że $d(t, s) < d$

$T(n) = 2 \cdot T\left(\frac{n}{2}\right)$ $\Theta(n \log n)$





← bierzemy tylko te, których wsp x jest ok. Teoretycznie można się ograniczyć nawet tylko do jednej połówki

Teraz możemy posłużyć się po y'. Zatem pkt 3° robimy w $\Theta(n)$. Punkt 1° w $\Theta(n \log n)$, a pkt 2° w $2 \cdot T(\frac{n}{2})$, zatem

$$T(n) = 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n \log n) = \Theta(n \log^2 n)$$

Zauważmy, że nie trzeba w każdym wywołaniu sortować wg x-sów. W wywołaniu rekurencyjnym podzbiór x-sów w poł. mamy od razu i są posortowane. Wtedy możemy podzielić y w czasie liniowym

← podzbiór x determinuje, w której połówce jest y.

Wówczas możemy podzielić pkt 1° na dwa. Pierwsze sortowanie jest w $\Theta(n \log n)$, a potem już liniowo, więc $T(n) = 2T(\frac{n}{2}) + \Theta(n) = \Theta(n \log n)$

PROGRAMOWANIE DYNAMICZNE

Dane : $n, k \in \mathbb{N}$

Problem: $\binom{n}{k}$

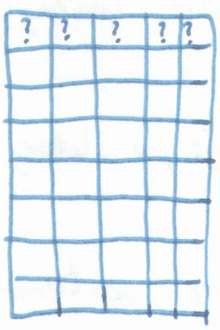
$$\binom{n}{k} = \begin{cases} 1 & \text{jeśli } k=0 \lor n=k \\ \binom{n-1}{k} + \binom{n-1}{k-1} & \text{w p.p.} \end{cases}$$

Naiwnie: rekurencja ← te same podproblemy liczone się wielokrotnie

Podproblemy $\binom{n'}{k'}$ $n' \leq n$, $k' \leq k$. Zatem liczba podproblemów $\leq n \cdot k$. Remedium: spamiętywanie

tab $\xrightarrow{n}$

n ↓

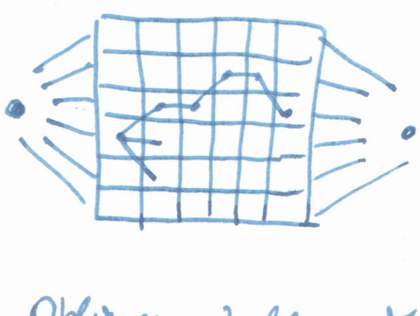


NpoK (n, k)
- if $(n=k \lor k=0)$ then $tab_{nk} \leftarrow 1$, return 1
- if $tab_{n-1,k} = ?$ then $tab_{n-1,k} \leftarrow$ NpoK$(n-1, k)$
- if $tab_{n-1,k-1} = ?$ then $tab_{n-1,k-1} \leftarrow$ NpoK$(n-1, k-1)$
- $tab_{n,k} \leftarrow tab_{n-1,k} + tab_{n-1,k-1}$
- return $tab_{n,k}$

Dane: tablica$(n, m;\ t_{ij} \in \mathbb{N})$

Znaleźć najlżejszą (tj. o najmniejszej wadze) ścieżkę przechodzącą

Alg naiwny : # ścieżek za dużo



Podproblem : $\forall i,j$ znaleźć najkrótszy koszt dojścia do pola (i,j)

podproblemów $n \cdot m$.

Obliczamy tablicę M_{ij} = najmniejszy koszt ścieżki kończącej się w polu (i,j)

$M_{i,1} = t_{i1}$ $\forall i=1\ldots n$, $M_{ij} = t_{ij} + \min\{M_{i-1,j-1}; M_{i,j-1}; M_{i+1,j-1}\}$ $j>1$, zatem $\Theta(n \cdot m)$