Dziel i Zwyciężaj

Master Theorem

Każdy rekurencyjny algorytm możemy oszacować z góry:

Wersja pełna (od Abdul Bari)

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \qquad \text{gdzie } f(n) = O(n^k \log^p n)$$

1. $\log_b a > k \rightarrow O(n^{\log_b a})$
2. $\log_b a = k$
   a) $p > -1 \rightarrow O(n^k \log^{p+1} n)$
   b) $p = -1 \rightarrow O(n^k \log\log n)$
   c) $p < -1 \rightarrow O(n^k)$
3. $\log_b a < k$
   a) $p \geqslant 0 \rightarrow O(n^k \log^p n)$
   b) $p < 0 \rightarrow O(n^k)$

Wersja uproszczona

Rozwiązaniem równania rekurencyjnego dla

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

gdy n jest potęgą liczby c jest:

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

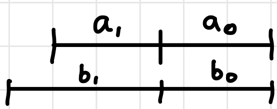
Karatsuba

Jest to algorytm szybkiego mnożenia bardzo dużych liczb.

1. Oblicz minimalną długość dwóch liczb:

$$dl = \min(\text{len}(a), \text{len}(b))$$

$$s = \left\lfloor \frac{dl}{2} \right\rfloor$$



2. Podziel liczby na część lewą i prawą:

$$a = a_1 \cdot B^s + a_0$$

$$b = b_1 \cdot B^s + b_0$$

Gdzie B jest systemem liczbowym.

3. Oblicz podstawiając pod znak mnożenia wywołanie rekurencyjne karatsuba.

$$p_1 = a_0 \cdot b_0$$

$$p_2 = (a_1 + a_0)(b_1 + b_0)$$

$$p_3 = a_1 \cdot b_1$$

Skąd się to bierze?

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

$$a_0 b_1 + a_1 b_0 + a_0 b_0 - a_0 b_0 + a_1 b_1 - a_1 b_1 =$$

$$= (a_1 + a_0)(b_1 + b_0) - a_1 b_1 - a_0 b_0$$

4. Oblicz wynik:

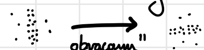
$$p_3 B^{2s} + (p_2 - p_3 - p_1) B^s + p_1$$

Złożoność: $O(n^{\log_2 3})$

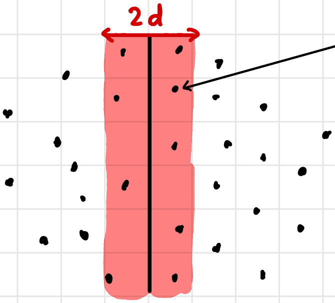
Para najbliżej położonych punktów

1. Tworzymy dwie tablice
   1. punkty posortowane rosnąco po x ←
   2. punkty posortowane rosnąco po y

W przypadku gdy punkty są bardziej rozłożone na osi y niż x zamieniamy tablice

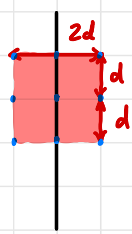


2. Dzielimy zbiór punktów po x na pół (prosta pionowa)
3. Rekurencyjnie wywołujemy tą funkcję na lewej i prawej stronie prostej pionowej.
4. Otrzymujemy dwa wyniki minimalne z lewej i prawej. Bierzemy mniejszy z tych wyników, oznaczmy go jako (p_1, p_2).
5. Sprawdzamy na wąskim pasku leżącym na tej prostej czy nie ma innego punktu leżącego bliżej.



Dla każdego punktu w tym wąskim pasku sprawdzamy możliwości tylko d w górę i w dół. Zatem $O(n)$

↳ Skoro d jest najmniejszą odległością, to może być max 8 takich innych punktów niż nasz rozpatrywany. Zatem $O(1)$ operacji na każdy punkt



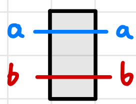
Cały algorytm ma złożoność czasową $O(n \log n)$ ← $2T(\frac{n}{2}) + O(n)$

Sieci Benesa Waksmana

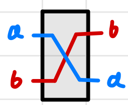
Dla $n=4$ otrzymujemy 24 permutacje $(4\cdot 3\cdot 2\cdot 1)$

Problem permutacji sygnałów przełączników.

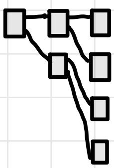
Budowa przełącznika:



lub

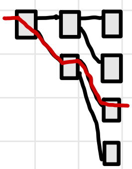


Żeby permutować 1 sygnał potrzebujemy $\log n$ przełączników:



Na przykład:

albo



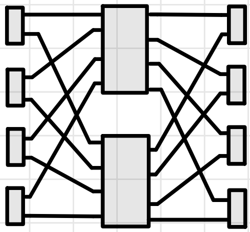
dowolnie!

Zatem aby permutować n sygnałów potrzebujemy $n \log n$ przełączników. Jest to minimalna wartość, jako, że permutowanie jest formą sortowania danych.

Rekurencyjna konstrukcja sieci:

Rekurencyjna sieć dla $\frac{n}{2}$

$n=8$



Głębokość sieci $2\log n - 1$

usunięcie duplikatu na samym końcu rekurencji

dla tego przykładu jest to $2\log 8 - 1 = 5$

Koniec rekurencji

głębokość: $2\log 4 - 1 = 4 - 1 = 3$

$n=4$