$L_i \leftarrow$ uporządkowany ciąg wektorów opisujących ~~wierzch.~~ wierzchoł. z S_i

if $L_1 \neq L_2$ return "nieizomorficzne"

$\forall u \in S_i$ kod$(u) = 1 +$ rank (key(v) wśród z L_i)

Na początku L_i dołącz wszystkie liście z poziomu j

<u>Pytanie</u>: Jak szybko to działa?

Porządkowanie ciągów wektorów: liniowo od sumy długości wektorów

Długość L_i w czasie k-tej iteracji $=$ # wierzchołków na niższym poziomie

Stąd $\sum_{\text{po iteracjach}} |L_i| \leqslant O(\#$ wierzchołków w $T_i)$

<u>Quick sort</u>

```
Quicksort (A[1...n], p, r)      ← sortujemy od elem p-tego do r-tego
    if (r-p) - małe then insertsort (A[p...r])
    else
        choose_pivot (A, p, r)
        q = partition (A, p, r)
        Quicksort (A, p, q)
        Quicksort (A, q+1, r)

Partition (A[1...n], p, r)
    x ← A[p]
    i ← p-1
    j ← r+1
    while i<j
        repeat j ← j-1 until A[j] ≤ x
        repeat i ← i+1 until A[i] ≥ x
        if (i<j) swap (A[i], A[j])
        else ...
```

$$T(n) = \begin{cases} \Theta(1) & \text{dla } n < \dots \text{ (małe)} \\ T(k) + T(n-k) + \Theta(n) & \text{o.p.p.} \end{cases}$$

↓ w przypadku wyboru mediany jako pivota

$\Theta(n \log n)$

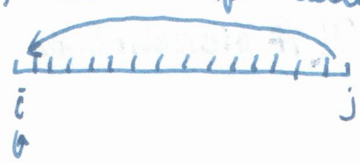
Smuteczek

Wybór pivota determinuje złożoność. Pesymistycznie złożoność – $\Omega(n^2)$, optymistycznie $O(n \log n)$

Zajmijmy się wyborem pivota. Ogólnie te metody muszą być 1° deterministyczne albo 2° losowe (deterministyczne, np. pierwszy element tablicy). Chcemy, żeby ten wybór był szybki.

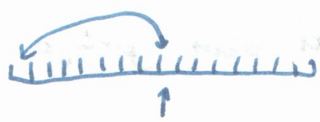
Ad 1°

Możemy mieć pecha np. tablica może już być posortowana



Więc jesteśmy w ... gdy tablica prawie posortowana.

Ad 2°



choose_pivot(A, p, r)
  $i \leftarrow$ random(p, r)
  swap (A[p], A[r])

$T(n)$ – oczekiwana liczba porównań

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

Ponieważ $T(1) = \Theta(1)$, a $T(n-1)$ jest $O(n^2)$, więc $\frac{1}{n}\left(T(1) + T(n-1)\right) = O(n)$

$$T(n) = \sum_{k=1}^{n-1} \frac{1}{n}\left[T(k) + T(n-k)\right] + \Theta(n)$$

$$T(n) = \frac{1}{n} \sum_{k=1}^{n-1} (T(k) + T(n-k)) + \Theta(n)$$

$$T(n) = \frac{2}{n} \sum_{k=1}^{n} T(k) + \Theta(n)$$

Mamy silne przeczucie, że $T(n) = O(n \log n)$. Sprawdźmy, że tak jest, czyli że istnieją stałe a, b: $T(n) \leq an\log n + b$

Niech b będzie takie, że $T(1) \leq b$.

Dla $n > 1$ mamy

$$T(n) \leq \frac{2}{n} \sum_{k=1}^{n-1} (ak\log k + b) + \Theta(n) \leq \frac{2a}{n} \sum_{k=1}^{n-1} k\log k + \frac{2b(n-1)}{n} +$$

$$+ \Theta(n),$$

FAKT

$$\sum_{k=1}^{n-1} k\log k \leq \frac{1}{2} n^2 \log n - \frac{1}{8} n^2$$

$$\leq \frac{2a}{n}\left(\frac{1}{2}n^2\log n - \frac{1}{8}n^2\right) + \frac{2b(n-1)}{n} + \Theta(n) \leq$$

$$\leq an\log n - \frac{a}{4}n + 2b + \Theta(n) = an\log n + b + \underbrace{(\Theta(n) + b - \tfrac{a}{4}n)}_{\leq 0}$$

dla odpowiedniego a

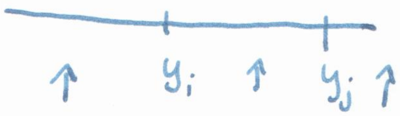
Inna metoda (inny dowód)

Chcemy oszacować liczbę porównań w QS

$$X_{ij} = \begin{cases} 1, & \text{jeśli } y_i \text{ był porównywany z } y_j \\ 0, & \text{w p.p.} \end{cases} \longrightarrow y_i, y_j \text{ elem tablicy}$$

$$X = \sum_{1 \leq i < j \leq n} X_{ij} \text{ — \# porównań w QS}$$

Chcemy oszacować $E[X] = E\left[\sum_{1\leq i<j\leq n} X_{ij}\right] = \sum_{1\leq i<j\leq n} E[X_{ij}]$



Prawdopodobieństwo tego, że y_i będą porównane z y_j wynosi $\frac{2}{j-i+1}$.

Jeśli trafimy z pivotem na lewo od y_i lub na prawo y_j, to nic się nie dzieje, bo odwlekamy decyzję o tym, czy będą porównane czy nie. Jeśli pivot jest wewnątrz przedziału y_i do y_j to elementy trafią do różnych tablic i nie będą porównywane.

$$E[X_{ij}] = \frac{2}{j-i+1}$$

$$E[X] = \sum_{1 \leq i < j \leq n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \approx O(n\log n)$$

Modyfikacje:

- pivot - mechanika trzech losowych elem. i wybieramy środkowy z tych trzech
- jeśli w ciągu jest dużo powtórzeń to trój podział (mniejsze, równe i większe od pivota)
- stała pamięć

SELEKCJA

Problem:

Dane: $T[1 \ldots n]$ - ciąg, $k \in \mathbb{N}$ $(1 \leq k \leq n)$

Wynik: k-ty co do wielkości element