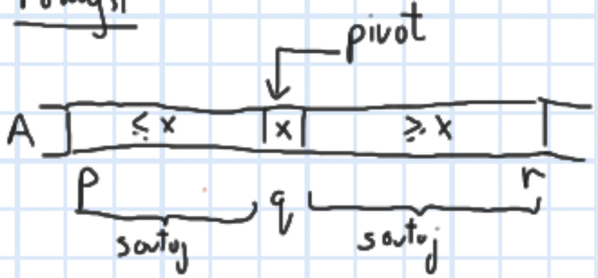
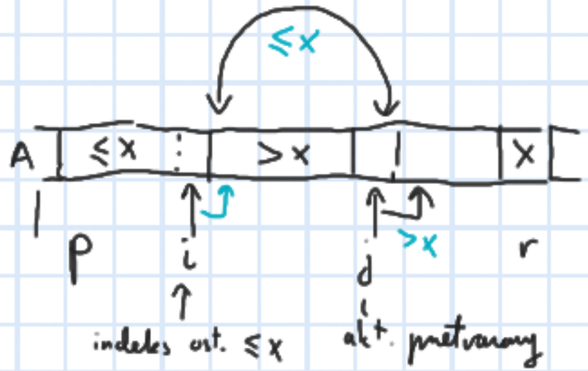
Algorytmy i Struktury Danych

Wykład 3

Algorytm Quick Sort

Pomysł






```
def quicksort(A, p, r):
    if p < r:
        q = partition(A, p, r)
        quicksort(A, p, q-1)
        quicksort(A, q+1, r)
```

```
def partition(A, p, r):
    x = A[r]
    i = p-1
    for j in range(p, r):
        if A[j] ≤ x:
            i += 1
            swap(A[i], A[j])
    swap(A[i+1], A[r])
    return i+1
```

Algorytm Lomuto

Algorytm Hoare'a

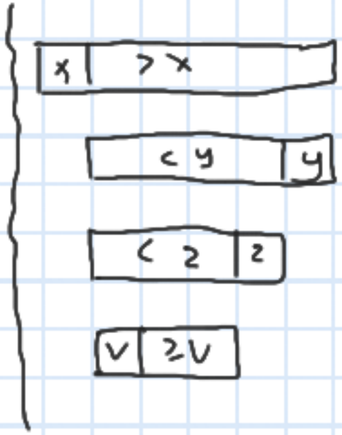
Złożoność obliczeniowa

Idealne podziały

$$T(n) = \begin{cases} c, & n \le 1 \\ 2T\left(\frac{n}{2}\right) + cn, & n > 1 \end{cases} \quad T(n) = \Theta(n \log n)$$

Pesymistyczne podziały

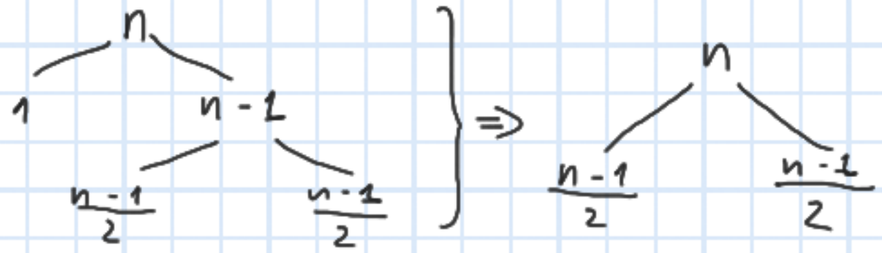
$$T(n) = \begin{cases} c, & n \le 1 \\ T(n-1) + cn, & n > 1 \end{cases}$$



$$T(n) = T(n-1) + cn$$
$$= T(n-2) + c(n-1) + cn$$
$$\vdots$$
$$= c + 2c + 3c + \ldots + c(n-1) + cn$$
$$= c\left(\frac{n(n+1)}{2}\right) = \Theta(n^2)$$

Mieszane podziały

Na co drugim poziomie rekurencji podziały pechowe, a na co drugim idealne



Quicksort bez rekurencji ogonowej

```
def quicksort(A, p, r):
    while p < r:
        q = partition(A, p, r)
        quicksort(A, p, q-1)
        p = q+1
```

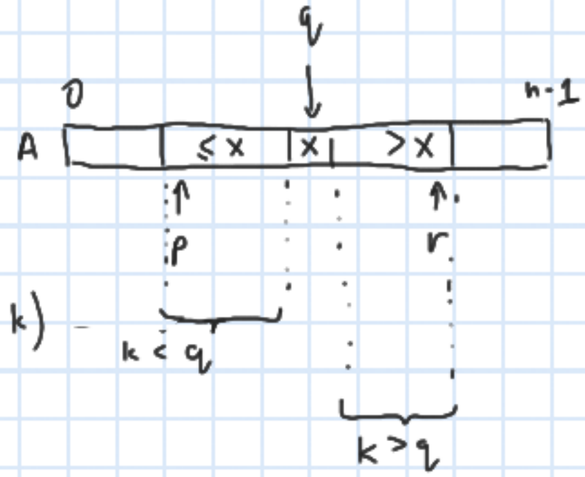
Statystyki pozycyjne

k-ta statystyka pozycyjna — element na pozycji k po posortowaniu

min/max — oczywisty algorytm $\Theta(n)$

mediana — ??

```
def select(A, p, r, k):
    if p == r: return A[p]
    q = partition(A, p, r)
    if q == k: return A[q]
    elif k < q: return select(A, p, q-1, k)
    else: return select(A, q+1, r, k)
```



$$T(n) = \begin{cases} c, & n \le 1 \\ T\left(\frac{n}{2}\right) + cn, & n > 1 \end{cases} \Rightarrow$$

$$\begin{aligned} T(n) &= cn + T\left(\frac{n}{2}\right) \\ &= cn + \frac{cn}{2} + T\left(\frac{n}{4}\right) \\ &\vdots \\ &= c\left(n + \frac{n}{2} + \frac{n}{4} + \ldots + 1\right) \\ &= 2cn = \Theta(n) \end{aligned}$$

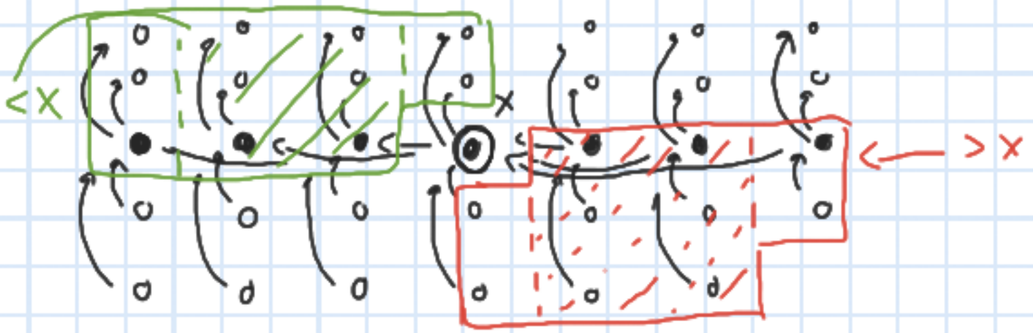
Magiczne piątki – statystyki pozycyjne w czasie liniowym

Algorytm

① podziel tablicę na $\lceil \frac{n}{5} \rceil$ grup po 5 elementów, w każdej wyznaczamy medianę

② rekurencyjnie wyznacz x jako medianę median

③ kontynuuj jak w zwykłym "select", ale używając x do wykonania partition (jako pivot)



Ile jest elementów większych od x?

$$3 \cdot \left(\left\lceil \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rceil - 2 \right) \geq \frac{3n}{10} - 6$$

Złożoność czasowa

$$T(n) = \begin{cases} \Theta(1) & , n \leq \text{pewna stała} \\ T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + \Theta(n) & , n \geq \bullet \end{cases}$$

$T(\lceil \frac{n}{5} \rceil)$: med. median

$T(\frac{7n}{10} + 6)$: zajmuje jak w select

Twierdzimy, że $T(n) \leq cn$ dla pewnej stałej c

Dowód indukcyjny:

$$T(n) \leq c \lceil \tfrac{n}{5} \rceil + \frac{7nc}{10} + 6c + an$$

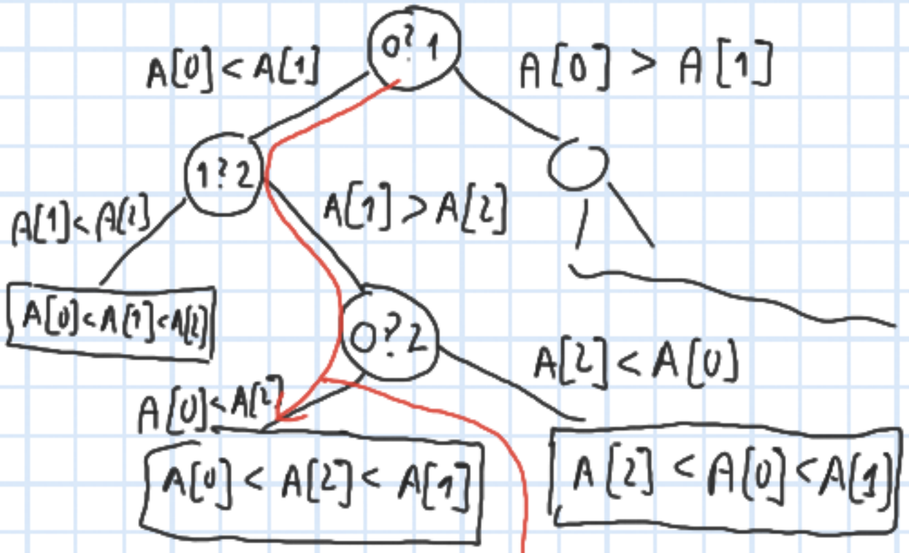
(an: jakaś stała)

$$\leq \frac{2cn}{10} + \frac{7cn}{10} + 6c + an + c$$

$$= cn + \left(-\frac{1}{10} cn + 7c + an \right)$$

wybieramy c tak duże, że ta wartość jest ujemna dla odp. dużych n

Dolne ograniczenie na złożoność sortowania



najdłuższa ścieżka w tutaj drzewie
to pesymistyczna liczba porównań do wykonania

$$\frac{n}{2}(\log n - 1) = \frac{n}{2}\log\frac{n}{2} = \log\left(\frac{n}{2}\right)^{\frac{n}{2}} \leq \log n! \leq \log n^n = n\log n$$

$$\downarrow$$

$$\Theta(n\log n)$$

Jeśli A ma n elementów to

drzewo musi mieć $\geq n!$ liści

Drzewo o wysokości h ma najwyżej 2^h liści
(binarne)

$$n! \leq 2^h$$

$$h \geq \log n!$$