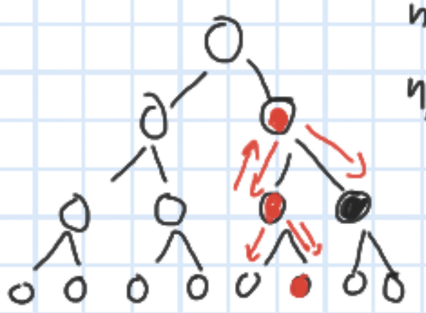
ASD - Wykład 3



$n/16$, 4

$n/8$, 3

$n/4$, 2

$\frac{n}{2}$, 1

$$\frac{n}{2} + \frac{n}{4}\cdot 2 + \frac{n}{8}\cdot 3 + \ldots = \sum_{i=1}^{\lfloor \log n \rfloor} \left(\frac{n}{2^i} \cdot i \right)$$

$$= n \sum_{i=1}^{\lfloor \log n \rfloor} \frac{i}{2^i}$$

$$\leq n \sum_{i=0}^{\infty} \frac{i}{2^i} \leq 2n$$

dla $x = \frac{1}{2}$

$$f(x) = 1 + x + x^2 + x^3 + \ldots = \frac{1}{1-x}$$

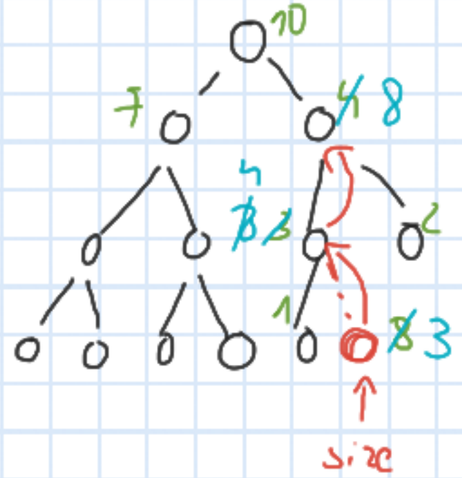
$$f'(x) = 1 + 2x + 3x^2 + 4x^3 + \ldots = \frac{1}{(1-x)^2}$$

$$x f'(x) = x + 2x^2 + 3x^3 + 4x^4 + \ldots = \frac{x}{(1-x)^2}$$

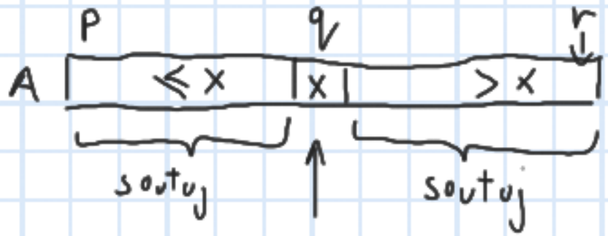
Kolejka priorytetowa

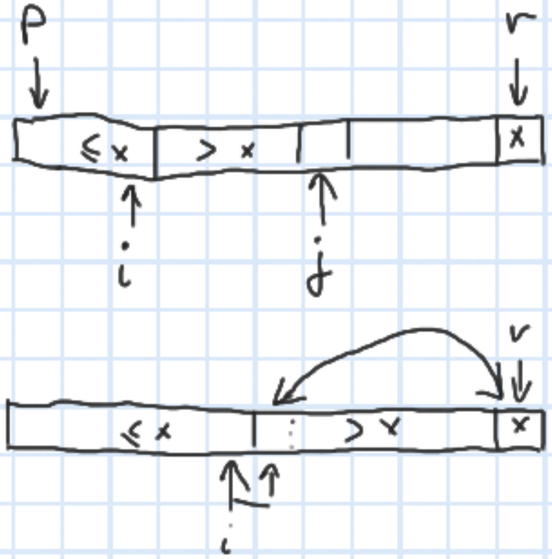
"Coś, do niego można wstawiać elementy w dowolnej kolejności i wyciągać w kolejności zgodnej z priorytetem"

```
class PQ:
  def __init__(self, n):
    self.T = [None] * n
    self.size = 0
```



Quick Sort






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

def partition(A, p, r):
    x = A[r]          //<- zamiana A[r] z losowym elementem z A[p],...,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
```

Złożoność czasowa algorytmu Quick Sort

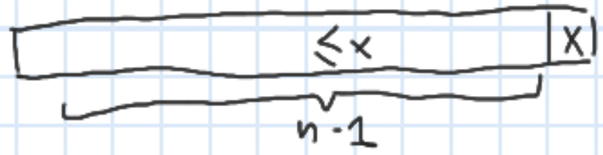
Idealne podziały

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

$T(n) = \Theta(n \log n)$

Pechowe podziały

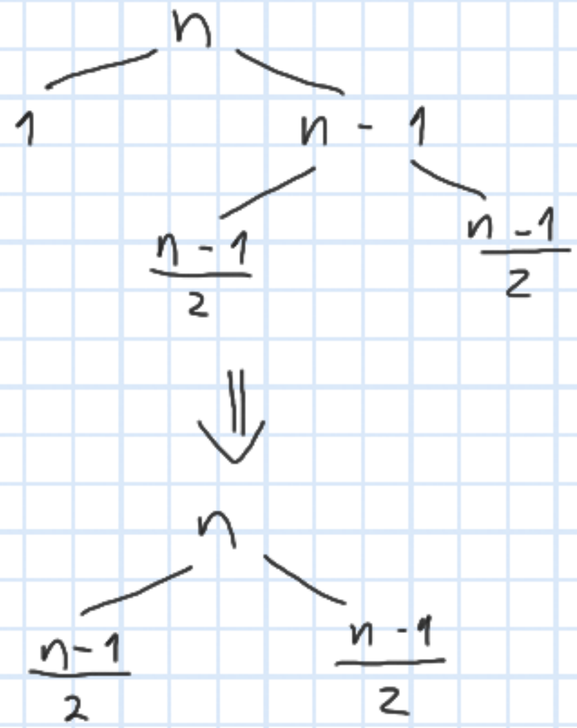
$$T(n) = \begin{cases} c, & n \leq 1 \\ T(n-1) + cn, & n > 1 \end{cases}$$



$T(n) = T(n-1) + cn = T(n-2) + c(n-1) + cn$

$= c(1 + 2 + 3 + \ldots + n) = \Theta(n^2)$

Mieszanka podziałów – "co drugi pechowy, pozostałe idealne"



Statystyki pozycyjne

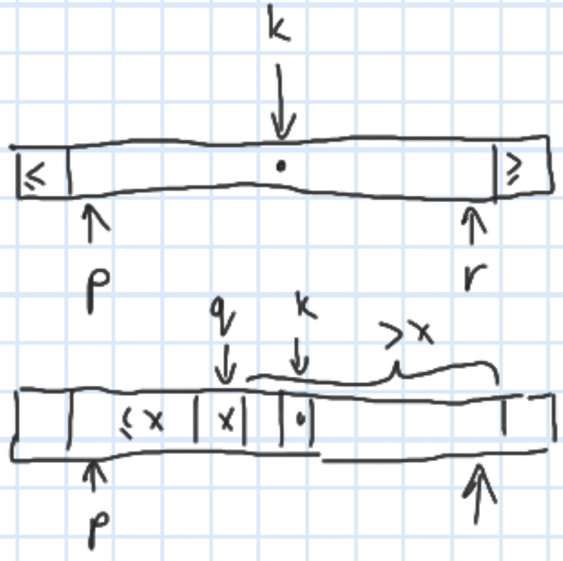
min/max – oczywisty algorytm $\Theta(n)$

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

W ogólności chcemy obliczyć element, który po posortowaniu byłby pod indeksem k

$$T(n) = cn + \frac{cn}{2} + \frac{cn}{4} + \ldots = \Theta(n)$$

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



Statystyki pozycyjne w pesymistycznym czasie liniowym — Magiczne Piątki

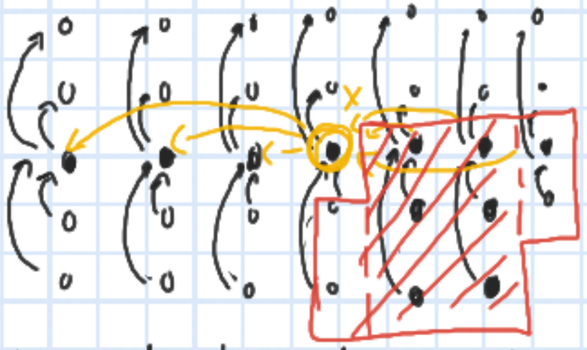
Algorytm

① Podziel wejściową tablicę na $\lceil \frac{n}{5} \rceil$ grup po 5 elementów

U każdej grupie wyznaczamy medianę

② Rekurencyjnie wyznaczamy x jako medianę median

③ Kontynuujemy tak jak w select, traktując x jak pivot w partition



Ile jest elementów większych od x?

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

Złożoność czasowa

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

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

Dowód indukcyjny

$$T(n) \leq c\lceil \frac{n}{5} \rceil + \frac{7nc}{10} + 6c + an$$
$$\leq \frac{2cn}{10} + \frac{7cn}{10} + 7c + an$$
$$= cn + \left(-\frac{1}{10}cn + 7c + an\right)$$

możemy wybrać tak duże c, że ta wartość jest zawsze ujemna