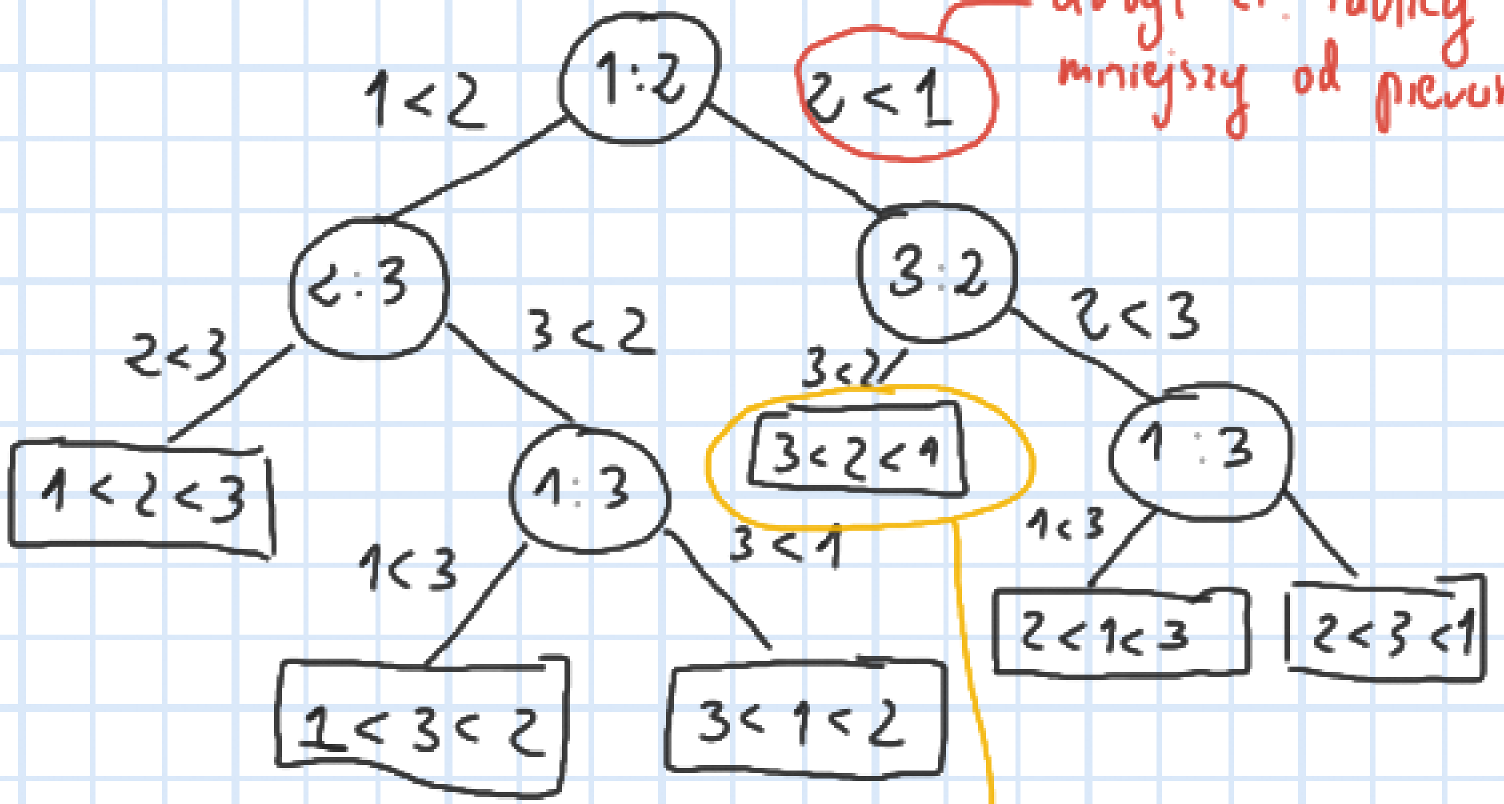
Algorytmy i Struktury Danych

Wykład 3

Dolne ograniczenie na szybkość sortowania



drugi el. tablicy mniejszy od pierwszego

każda permutacja musi się pojawić jako liść

wysokość tego drzewa to liczba porównań, którą algorytm wykonuje w najgorszym przypadku

$$1 \cdot 2 \cdot \ldots \cdot \frac{n}{2} \cdot \underbrace{\ldots \cdot n}_{\frac{n}{2} \cdot \frac{n}{2} \cdot \ldots \cdot \frac{n}{2}}$$

$$\log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right) \le \log(n!) \le \log(n^n)$$

$$\frac{n}{2}(\log n - 1) \qquad n \log n$$

jest $\Theta(n \log n)$

Tablica n elementowa $\Rightarrow$ $n!$ permutacji

Drzewo binarne wysokości h ma najwyżej 2^h liści

Jeśli nasze drzewo jest wysokości h, to $n! \le 2^h \Rightarrow h \ge \log(n!)$ $\Rightarrow$ h jest rzędu $n \log n$

Sortowanie przez zliczanie

Chcemy posortować pewną tablicę rozmiaru n, zawierającą liczby (naturalne) od 0 do k-1

```
def countsort(A, k):
    C = [0] * k
    B = [0] * len(A)
    for i in range(len(A)):
        C[A[i]] += 1
    for i in range(1, k):
        C[i] += C[i-1]
    for i in range(len(A)-1, -1, -1):
        C[A[i]] -= 1
        B[C[A[i]]] = A[i]
    for i in range(len(A)):
        A[i] = B[i]
```

```
for x in A:
    C[x] += 1
```

$O(n+k)$

$k = 5$

A:

1	3	2'	4	0	3	2''

C:

0	1	2	3	4
1	1	2	2	1

C:

1	2	4	6	7

// C[i] = ile liczb jest ≤ liczbie i

B:

0	1	2	3	4	5	6
0	1	2'	2''	3	3	4

Zadanie obowiązkowe

Proszę zaproponować jak najszybszy algorytm sortujący n elementową tablicę zawierającą liczby ze zbioru $\{0, 1, 2, \ldots, n^2-1\}$

Sortowanie kubełkowe

Sortujemy tablicę n liczb pochodzących
z rozkładu jednostajnego nad $[0,1)$

$O(n)$ gdy spełnione założenia

0.42, 0.93, 0.07, 0.21, 0.91, 0.13, 0.37

tworzymy n kubełków // list jednokierunkowych

[0, 0.15)	[0.15, 0.30)	[0.30, 0.45)	[0.45, 0.6)	[0.6, 0.75)	[0.75, 0.9)	[0.9, 1)
0.13	0.21	0.42				0.91
0.07		0.37				
0.13						
0.07	0.21	0.37				0.91
0.13		0.42				
0.13						

$x \in [0,1)$

x ląduje w kubełku

$\lfloor x \cdot n \rfloor$

$\downarrow$

k

0.07, 0.13, 0.13, 0.21, 0.37, 0.42, 0.91

Radix Sort – sortowanie pozycyjne

kra		kra		kil		art
art		ati		kit		ati
kot	⇒	kil	⇒	kot	⇒	kil
kit		art		kra		kit
ati		kot		art		kot
kil		kit		ati		kra

Zadanie obowiązkowe

Tablica T jest długości n, ale zawiera tylko $\lceil \log n \rceil$ różnych wartości. Proszę zaproponować jaknajszybszy algorytm sortujący taką tablicę

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

$$T(n) = cn + c\frac{n}{2} + c\frac{n}{4} + c\frac{n}{8} + \ldots + c\frac{n}{2^{\log n}}$$

$$= cn\left(1 + \frac{1}{2} + \frac{1}{4} + \ldots\right) \leq 2cn$$

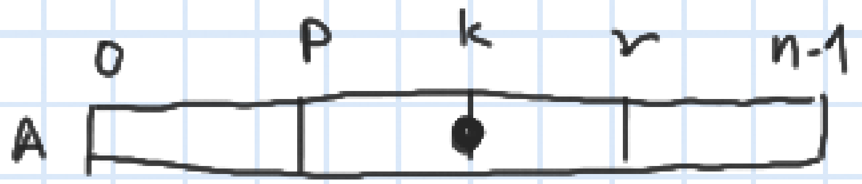
Statystyki pozycyjne

Zadanie: Wyznaczyć element, który po posortowaniu tablicy znalazłby się na pozycji k-ej

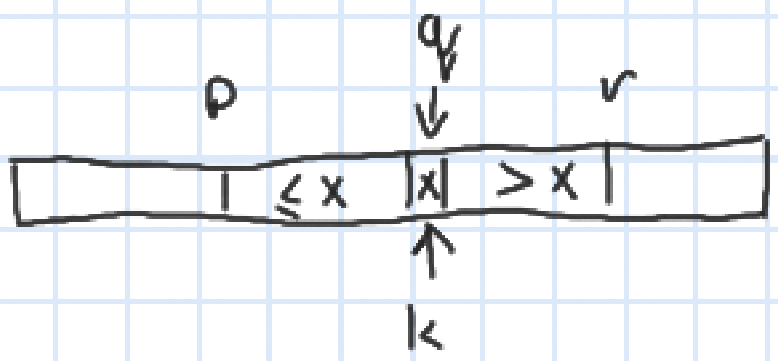
Przykłady elementarne

$k = 0 \rightarrow$ minimum

$k = n-1 \rightarrow$ maksimum



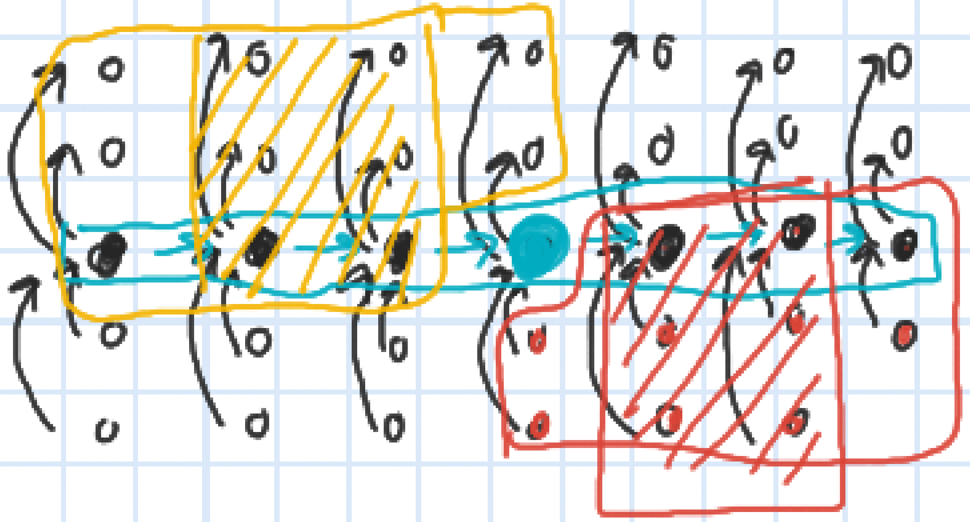

```
def select(A, p, r, k):
    if p == r:
        return A[p]
    q = partition(A, p, r)
    if q == k:
        return A[q]
    elif k < q:
        return select(A, p, q-1, k)
    else:
        return select(A, q+1, r, k)
```



Algorytm: Magiczne Piątki

A - wejściowa n-elementowa tablica

1. Podziel wejściową tablicę na $\lceil \frac{n}{5} \rceil$ grup po 5 elementów
2. W każdej grupie wyznacz medianę
3. Rekurencyjnie wyznacz x jako medianę median
4. Kontynuujemy jak w algorytmie select, traktując x jak piwot przy podziale



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

Złożoność czasowa

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

$$\frac{2}{10}n + \frac{7}{10}n = \frac{9}{10}n$$

Twierdzimy, że $T(n) \leq cn$

Dowód przez indukcję:

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

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

$$= cn + \left(-\frac{1}{10}cn + \underline{6c + an + c}\right) \leq cn$$

$$T(n) = T\left(\left\lceil \frac{n}{3}\right\rceil\right) + T\left(\frac{2}{3}n\right)$$

$$\frac{n}{3} + \frac{2}{3}n$$

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