Úloha: Najít k-tý nejmenší z $x_1 — x_n$

① vybereme pivota $\in x_1 — x_n$

② rozdělíme vstup na L, S, P ($<p$, $=p$, $>p$)

③ pokud $k \le |L|$: rekurze na L, k
jinak pokud $k \le |L|+|S|$: vrátíme p
jinak: rekurze na P, $k-|L|-|S|$

Volba pivota
- maximum $\Theta(n^2)$
- medián $\Theta(n)$] nevíme jak
- pseudomedián $\Theta(n)$] nevíme jak
- náhodně $\mathbb{E} = \Theta(n)$

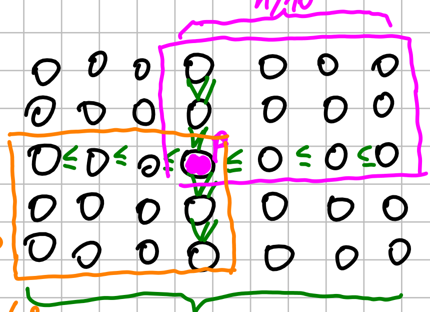
Select $(x_1 — x_n; k)$:

1. Rozdělíme $x_1 — x_n$ na pětice $P_1 — P_t$ ($t = \lceil n/5 \rceil$)
   doplníme podle potřeby $+\infty$
2. Najdeme mediány pětic: $\forall i$ $m_i \leftarrow$ medián(P_i)
3. $p \leftarrow$ Select$(m_1 — m_t, \lceil t/2 \rceil)$

↳ pivot v předchozím alg.

Věta: Select má čas. složitost $\Theta(n)$.

Dk:



vždy zahodím aspoň $\frac{3}{10}n$ prvků
→ zbývá nejvýš $\frac{7}{10}n$ prvků.

$$T(n) = T(n/5) + T\left(\tfrac{7}{10}n\right) + \cancel{\Theta(n)}\, n$$

$$T(1) \in \Theta(1)$$

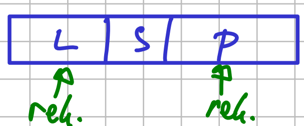
Uhodneme $T(n) = cn$

$$cn = \underbrace{\tfrac{1}{5}cn + \tfrac{7}{10}cn}_{\frac{9}{10}cn} + n$$

$$\tfrac{1}{10}cn = n$$

$$\boxed{c = 10}$$

QuickSort $(x_1 — x_n)$



0. Pokud $n \le 1$: vrátíme vstup
1. Zvolíme pivota p.
2. Rozdělíme $x_1 — x_n$ podle p na L, S, P.
3. $L \leftarrow$ QuickSort(L)
   $P \leftarrow$ QuickSort(P)
4. Vrátíme $L \| S \| P$.

Složitost — pokud p bude medián $\rightarrow T(n) = 2T(n/2) + \Theta(n)$
$\Downarrow$
$T(n) = \Theta(n \log n)$

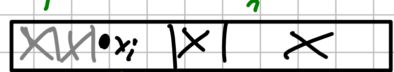
— pokud p bude min/max $\rightarrow T(n) = \underbrace{T(n-1)} + \underbrace{T(0)}_{\Theta(1)} + \Theta(n)$
$\Downarrow$
$T(n) = \Theta(n^2)$

Věta: QuickSort s náhodnou volbou pivota má časovou složitost se střední hodnotou $\Theta(n \log n)$.

čas $= \Theta(\#\text{porovnání})$

Dk #1: porovnání účtujeme prvku, který nebyl pivotem
Kolik porovnání jsme naúčtovali prvku $x_i \rightarrow P_i$
sledujeme velikost pp., ve kterém je x_i
dělíme na fáze, končí volbou p = pseudomedián

$\#\text{porovnání} = \sum_i P_i$
$\Downarrow$
$\mathbb{E}[\#\text{porovnání}] = \sum_i \mathbb{E}[P_i]$



① fáze zmenší n na max. 3/4 n → # fází $\in O(\log n)$
② $\mathbb{E}[\text{kroků ve fázi}] \le 2$

Dk #2: y_1 — y_n setříděné pořadí prvků (výstup)

$$C_{ij} := \begin{cases} 0 \\ 1 & \text{pokud jsme porovnali } y_i \text{ s } y_j \end{cases}$$
(stane se max. 1×)

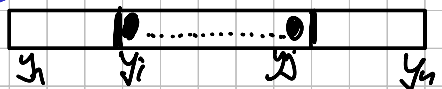
$1 \le i < j \le n$

Celkový #porovnání $C = \sum_{i,j} C_{ij}$

linearita $\mathbb{E}$

$$\mathbb{E}[C] = \sum_{i,j} \mathbb{E}[C_{ij}]$$

$$\mathbb{E}[C_{ij}] = \Pr[y_i, y_j \text{ bylo porovnáno}]$$



1) právě je y_i nebo y_j pivot
2) žádný z $y_{i+1}, \dots, y_{j-1}$ ještě pivotem nebyl

} y_i nebo y_j se z prvků y_i — y_j stalo pivotem jako první

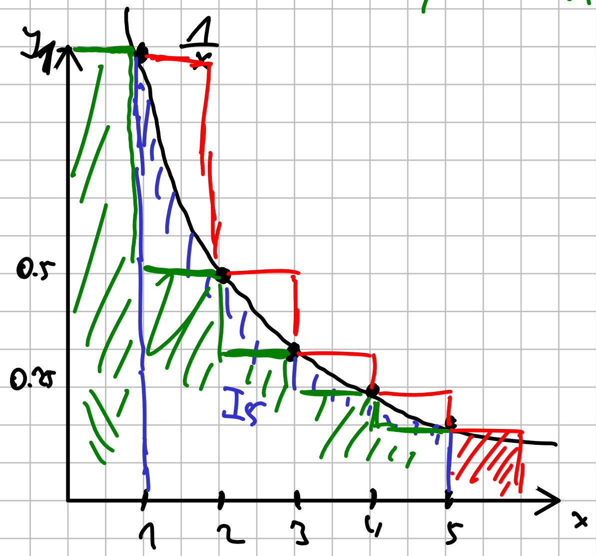
$$= \frac{2}{j-i+1}$$

$$\mathbb{E}[C] = \sum_{1 \le i < j \le n} \frac{2}{j-i+1} = \sum_{1 \le d \le n} \frac{2}{d} \cdot \underbrace{(n-d+1)}_{\le n} \le 2n \sum_{1 \le d \le n} \frac{1}{d} \in \Theta(n \log n)$$

H_n n-té harmonické číslo

Lemma: $\ln n \le H_n \le \ln n + 1$, kde $H_n = \sum_{1 \le i \le n} \frac{1}{i}$.

Dk:



$$I_n := \int_1^n \frac{1}{x}\,dx = [\ln x]_1^n = \ln n$$

(zelená) $= H_n \rightarrow H_n \le I_n + 1 = \ln n + 1$

(červená) $= H_n \rightarrow H_n \ge I_n + \frac{1}{n} \ge I_n = \ln n$

Dynamické Programování

Fibonacciho čísla

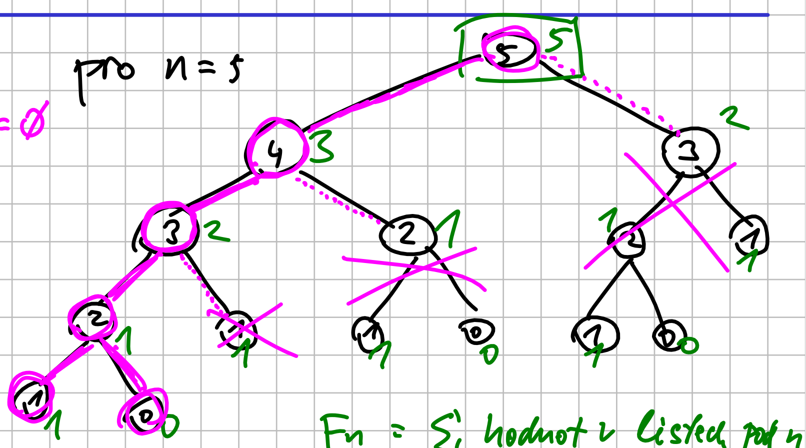
paměť P: $\forall i$ $P[i] = \emptyset$

F(n): pokud $P[n] \ne \emptyset$: vrátíme $P[n]$

1. Pokud $n \le 1$: vrátíme n.
2. Jinak vrátíme $F(n-1) + F(n-2)$. a zapíšeme do $P[n]$

$\forall$ $F(i)$ pro $0 \le i \le n$ počítáme jen 1× a trvá to $O(1)$
→ celkem $O(n)$.

pro $n = 5$



$F_n = \sum$ hodnot v listech pod n
$\le$ # listů pod n

už víme: $F_n \ge 2^{n/2}$ → exponenciální algo.

↳ jiná možnost: vyplňovat cyklem $P[0]$ — $P[n]$ → už známe

$O(n)$ ↳ víme: $P[i]$ potřebuje znát $P[i-1]$ a $P[i-2]$

Postup: ① začneme rekurzivním alg. ... exponenciálně pomalý

② počítáme mockrát totéž

③ paměť na známé výsledky ⟨ kešování / memoizace

↳ poly složitost

④ rekurzi nahradíme vyplňováním keše cyklem ve správném pořadí } také poly, ale jednodušší

Základní princip DP

Nejdelší rostoucí podposloupnost (NRP) v x_1 — x_n (BÚNO $x_0 \leftarrow -\infty$, $x_{n+1} \leftarrow +\infty$)

NRP(i): ← délka NRP vybrané z x_i — x_{n+1} začínající x_i → voláme NRP(0)

(0. Pokud $i = n+1$: vrátíme 1.)
1. $d \leftarrow 1$
2. Pro $j = i+1$ — $n+1$:
3. Pokud $x_i < x_j$:
4. $d \leftarrow \max(NRP(j)+1, d)$
5. Vrátíme d.

① exp. složitost: pokud vstup rostoucí, vyzkoušíme všech 2^n podposloupností!

② $i \in \{0$ — $n+1\}$ → volání se opakují

③ přidáme keš → fce zavolána $O(n)$-krát, pokaždé počítá v čase $O(n)$ } $O(n^2)$

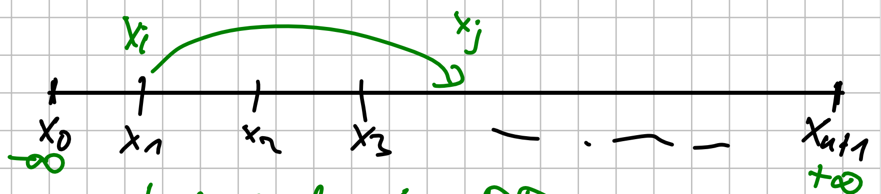
④ vyplňujeme tabulku cyklem ↳ $O(n^2)$

NRP(n):
1. $P[n+1] \leftarrow 1$
2. Pro $i = n, \ldots, 0$ pozpátku:
3. $d \leftarrow 1$
4. Pro $j = i+1$ — $n+1$:
5. Pokud $x_i < x_j$:
6. $d \leftarrow \max(P[j]+1, d)$ ⊛ lze v $O(\log n)$
7. $P[i] \leftarrow d$
8. Vrátíme $P[0]$

Mimochodem: jde to v $O(n \log n)$, např. vhodnou DS pro ⊛

$(x_i, x_j) \in E \equiv i < j \;\&\; x_i < x_j$

Alternativně: sestrojíme graf



cesty v grafu ⟺ RP

↓

graf je DAG ← hledáme nejdelší cestu z x_0 do x_{n+1}

↓

má topolog. pořadí x_0 — x_{n+1}

↳ cestu najít indukcí podle topol. poř. v $O(n+m) = O(n^2)$