Nalezení k-tého nejmenšího prvku posloupnosti $x_1 \dots x_n$

1) Vyberu pivota $\in x_1 \dots x_n$
2) Rozdělím vstup na L, S, P
   $<p,\ =p,\ >p$
3) Pokud $k \le |L|$: rekurze na L, k
   Pokud $k \le |L| + |S|$: vrátíme pivota
   Jinak rekurze na $P,\ k - |L| - |S|$

Rychlost závisí na faktoru zmenšení

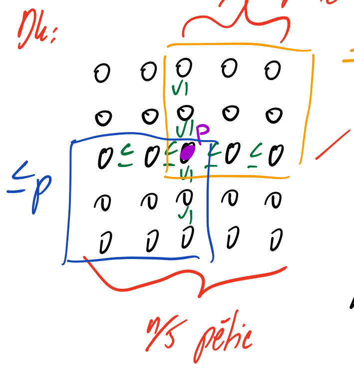
Volba pivota:
- maximum $\Rightarrow \Theta(n^2)$
- medián $\Rightarrow \Theta(n)$ — zatím neumíme jak
- pseudomedián $\Rightarrow \Theta(n)$ — celkově
- náhodně $\Rightarrow \mathbb{E} = O(n)$
  ↳ přestože v nejhorším $O(n^2)$

Select $(x_1 \dots x_n ; k)$:

1) Rozdělíme $x_1 \dots x_n$ na pětice $P_1 \dots P_t$ $\left(t = \lceil \frac{n}{5} \rceil\right)$
2) Najdeme mediány pětic: $\forall i \ \ m_i \leftarrow$ medián (P_i)
3) Najdeme medián mediánů pětic:
   : Select $(m_1 \dots m_t, \lceil t/2 \rceil)$
   ↳ pivot z předchozího algoritmu

Věta: Select má časovou složitost $\Theta(n)$

Dk:



→ Celkem $\frac{3}{10}n$ prvků

alespoň jednu takovou část musím zahodit.

↳ Tedy určitě alespoň $\frac{3}{10}$ prvků zahodíme v každém kolu
$\Rightarrow$ zbyde $\frac{7}{10}$ prvků nejvýše

$$T(n) = T(n/5) + T\left(\tfrac{7}{10}n\right) + \cancel{\Theta(n)}\ n$$
$$T(1) = \Theta(1)$$

Uhodneme $T(n) = cn$

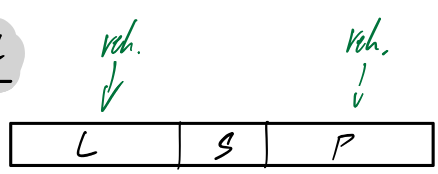
$cn = \frac{1}{5}cn + \frac{7}{10}cn + n$

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

$\boxed{c = 10}$ $\Rightarrow T(n) = cn$

↳ Tedy v nejhorším případě má lineární složitost

QuickSort



0. Pokud $n \le 1$: vrátíme vstup
1. Zvolíme pivota p.
2. Rozdělíme $x_1 - x_n$ podle L, S, P
3. $L \leftarrow \text{quicksort}(L)$
   $R \leftarrow \text{quicksort}(R)$ → slepení
4. Vrátíme $L \,\|\, S \,\|\, P$.

Složitost:

- pokud p bude medián.

$$T(n) = 2T(n/2) + \Theta(n)$$
$$\downarrow$$
$$T(n) = \Theta(n \log n)$$

→ už umíme najít pivota v čase $O(n)$

- pokud p min/max:

$$T(n) = T(n-1) + \underbrace{T(0)}_{\Theta(n)} + \Theta(n)$$
$$\Downarrow$$
$$T(n) = \Theta(n^2)$$

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

Dk #1:

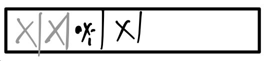
Porovnám, kolikrát se každý prvek zúčastní porovnání.

– porovnání účtujeme prvku, který nebyl pivotem (ten bude porovnáván furt...)

Kolik porovnání jsme naúčtovali x_i ? → P_i

$$\text{čas} = \Theta(\#\text{porovnání})$$
$$\#\text{porovnání} = \sum_i P_i$$
$$E[\#\text{porovnání}] = \sum_i E[P_i]$$

Sledujeme velikost podproblému, ve kterém je x_i.



– pokud použiju pseudomedián jako pivot, tak v každém kroku vyhodím alespoň $\frac{1}{4}$ prvků

Dělíme na fáze: končí volbou $p :=$ pseudomedián

① Fáze zmenší n na $\frac{3}{4}n$: $\#\text{fází} \in O(\log n)$

② $E[\text{kroků ve fázi}] \le 2$

Takže jedna fáze trvá $O(\log n)$, celkem může být až n porovnání, tedy celkem $O(n \log n)$

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

C_{ij} : $1 \le i < j \le n$

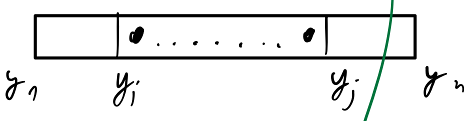
$C_{ij} \begin{cases} 0 \\ 1 & \text{pokud jsme porovnali } y_i \text{ s } y_j \end{cases}$ (stane se nejvýš jednou)

Při porovnání je jeden ze dvou pivot a ten nepostupuje do rekurze

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

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

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



→ y_i, y_j se účastní porovnání, pokud jeden z nich je pivot a pokud jsou ve stejném intervalu.

Všechno dohromady

$$\mathbb{E}[C] = \sum_{1\le i<j\le n} \frac{2}{j-i+1} = \frac{2}{j-i+1}$$

délka intervalu

proto

tohle

- tedy žádný prvek mezi y_i, y_j ještě nemohl být pivotem, protože y_i a y_j by se potkaly ... při finálním dělení.
- tedy y_i nebo y_j se z prvků $y_i - y_j$ stalo pivotem jako první.

$$= \sum_{1\le d\le n} \frac{2}{d}\cdot\underbrace{(n-d+1)}_{\le n} \le 2n \sum_{1\le d\le n}\frac{1}{d} \in \Theta(n\log n)$$

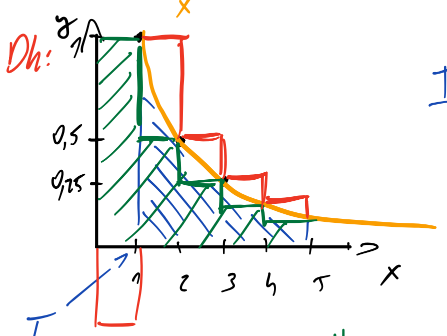
harmonická suma - roste logaritmicky rychle

→ harmonická řada

Lemma: $\ln n \le H_n \le \ln n+1$

$\frac{1}{x}$

Dk:



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

I_5

tohle se nepatří do oblasti, co integrujeme

green $= H_n$ → $H_n \le I_n + 1 = \ln n + 1$

red $= H_n$ → $H_n \ge I_n + \frac{1}{n} \ge I_n = \ln n$

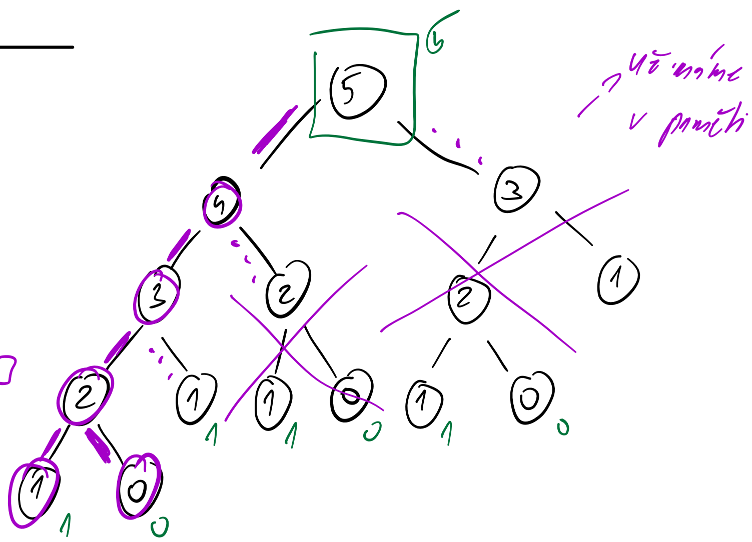
- takže jsme to zavřeli mezi dva předpisy

Dynamické programování

Výpočet Fib. čísel:

paměť $P[i] = 0$

F(n):
0) Pokud $P[n] \ne 0$: vrátíme $P[n]$
1) Pokud $n \le 1$: vrátíme n
2) Jinak vrátíme $F(n-1) + F(n-2)$



F(n) je součet listů v podstromu

$F_n \ge 2^{n/2}$ → exponenciální alg!

- Počítáme spoustu věcí znovu, takže si to cachujeme.
- každé $F(i)$ pro $0 \le i \le n$ počítáme právě jednou a trvá to $O(1)$, tedy celkem $O(n)$

→ Jiná možnost: Vyplňovat $P[0]$ — $P[n]$? už známe

↳ víme $P[i]$ potřebuje znát $P[i-1]$ a $P[i-2]$

Postup:

- Začneme rekurzivním alg., který je zpravidla velmi pomalý
- počítáme mockrát to stejný
- pamět na známé výsledky (cachování)
  - ↳ poly složitost
- rekurzi vyměníme za vyplnění cache v polynomiálním čase

Hledáme nejdelší vybranou podposloupnost v $x_1 \dots x_n$ (Buno $x_0 = -\infty$, $x_{n+1} = +\infty$)

NRP(i): $\to$ délka NRP vybrané z $x_i \dots x_{n+1}$ začínající x_i

1. $d \leftarrow 1$ (zatím nejlepší délka)
2. Pro $j = i+1 \dots n+1$
3.   Pokud $x_i < x_j$:
4. $d \leftarrow \max(NRP(j)+1, d)$
5. Vrátíme d.

① Opět exponenciální posloupnost

② $i \in \{0 \dots n+1\}$
$\to$ volání se opakují

③ Přidání cache
$\to$ fce zavolání $O(n)$-krát
pokaždé počítá v čase $O(n)$

④ Vyplňujeme tabulku cyklem

1. $P[n+1] \leftarrow 1$
2. Pro $i = n \dots 0$ pozpátku:
3. $d \leftarrow 1$
4.   Pro $j = i+1 \dots n+1$:
5.   Pokud $x_i < x_j$
6. $d \leftarrow \max(P[j]+1, d)$
7. $P[i] \leftarrow d$
8. Vrátíme $P[0]$

- nyní běží v $O(n^2)$