Cv 16.11.'10

(chudil)

$n \in O(n^2)$

def: $n_0 \geq 0$

$c > 0$

$\forall n > n_0 : g(n) \leq c \cdot fn$

$n \leq c \cdot n^2 \,/\, n \quad n_0 > 0$

$1 \leq c \cdot n$

$\frac{1}{c} \leq n$

Libovolně si můžu zvolit c a n_0 tak, aby def. platila.

např. $c = 1$ pak $n_0 = 1 \Rightarrow \frac{1}{c} \leq n$ platí

(PŘ) ~~důkaz, že n roste rychleji~~

$n \in o(n^2)$

. dokázat, že „n^2" roste rychleji než n

$(n+1) - n \leq (n+1)^2 - n^2$

$1 \leq n^2 + 2n + 1 - n^2$

$1 \leq 2n + 1$

$0 \leq 2n$

pokud $n > 0$, pak ~~roste~~ n^2 roste rychleji

Invariant

= pravidlo, které musí platit předtím, než udělám novou operaci

Algoritmy - třídění

Co mě zajímá?, Typy vlastností implementace:

STABILITA

SLOŽITOST - v nejlepším, nejhorším, amortiz. případě

PAMĚŤOVÁ = kolik paměti potřebujeme navíc
SLOŽITOST
(METODY SLOŽ)

DATOVÁ = kárislost na vstupních datech
CITLIVOST

DETERMINISMUS - je daný determin. postup třídění/řazení

IN-PLACE - pro řazení jim stačí vstupní pole
OUT-PLACE - potřebuje další pole

	STAB	SLOŽ	M. SLOŽ	DAT. CITL	DETER	IN-PLACE
INSERT SORT	A	$\Theta(n^2)$ (lineární u setříděného pole)	$O(1)$	A	A	A
MERGE SORT	A	$\Theta(n \log n)$	$\Theta(n)$	N	A	N (pro slévání potřebuji jiné jiného pole)
QUICK SORT (VAR. 1)	A	$O(n^2)$ = nejhorší $\Theta(n)$ průměr. složit. = $n \log n$ $\Theta(n \log n)$	$\Theta(n)$	A	A v případě že je ~~špatně~~ ~~vstup~~ pro "randomizovaná" pole N	N
QS (VAR. 2)	N	$O(n^2)$	$O(n)$ v případě rekurzivního algoritmu	A	dtto	A
SELECT. SORT	N	$\Theta(n^2)$	$O(1)$	N	A	A
	stabilní např. nad spojitým seznamem					

Quick Sort - VAR 1

2_1	5	8	2_2	1	12	2_3	13

PIVOT = 2 (náhodný výběr, vzali jsme první číslo)

1	2_1	2_2	2_3	5	8	12	13

c_1 c_2

1) vlevo pivota
2) vlevo čísla < 2
3) číslo = 2
4) číslo > 2
5) seřadit části před c_1 a za c_2 = může použít stejný algoritmus

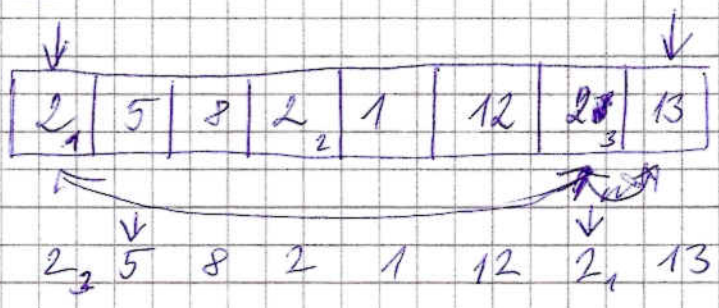
c_1 - v našem případě OK

c_2

5	8	12	13

c_1 c_2

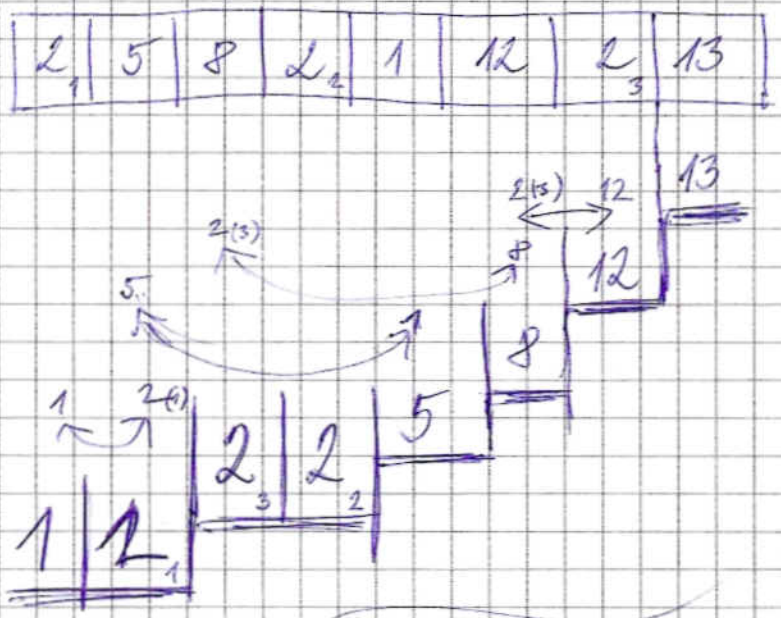
Quick Sort - VAR. 2



1) pivot = 2
2) ~~pohledání~~ hledá z obou konců v L 2< číslo
P 2 > číslo

kde není shoda, čísla se prohodí a ukazatel se posune směrem do středu

Selection Sort - VAR 1



hledám max a řadím od konce a vždy pouze hranici prohledávání dolevá

stačí setřídit n-1 čísel

~~SS je dobré~~

SS funguje stabilně nad spojovým seznamem, kde přesun je otázkou změny reference

---

Dú - je na odevzdání

Ex. 5.21

Ex. 5.28 a)

Sekvenční ~~f~~ výpočty - algoritmy

Najděte „k-té" největší číslo.

K dispozici máte POUZE „k" paměťových míst.

např.

1	2	8	3	16	5	4	2	1	8

$k = 5$

~~mám~~ pole smím prohledat POUZE jednou ⇒ prohledávám a postupně probarvuji v 5-ti prvkovém poli

1 2 8

1 2 3 8 16

1 2 3 5 8

1 2 3 4 5

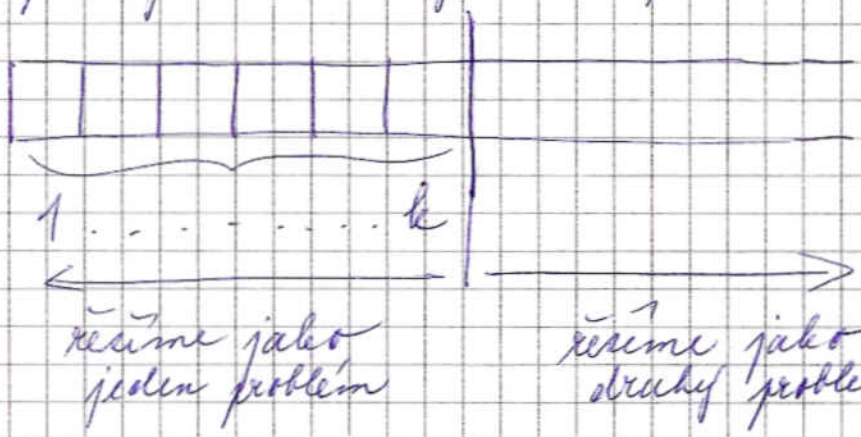
⋮ atd.

1	1	2	2	3

k-té největší číslo

PŘ 5.27

máme k nejmenších prvků na indexech 1–k
použijeme k vyřešení QS



Porovnat 3 čísla na 2 porovnání

A ≠ B ≠ C

přes A – B A – C
pak porovnat výsledky