

# KVADRATICKÉ ŘAZENÍ A QUICK-SORT

---

Karel Horák, Petr Ryšavý

13. dubna 2016

Katedra počítačů, FEL, ČVUT

Jak seřadí následující osoby podle věku stabilní řadící algoritmus?

1. (*Franta*, 32)
2. (*Pepa*, 17)
3. (*Petr*, 25)
4. (*Jana*, 23)
5. (*David*, 23)
6. (*Martina*, 25)
7. (*Lojza*, 7)
8. (*Hanka*, 23)

V dokumentaci zjistěte, zda je standardní řazení v Javě `Arrays::sort` (nebo vašem oblíbeném jazyce) stabilní.

V materiálech máte metodu `Sorting::insertSort`. Řazení je naimplementováno tak, že není stabilní. Opravte kód tak, aby řazení bylo stabilní.

V materiálech máte metodu `Sorting::selectionSort`. Řazení je naimplementováno tak, že není stabilní. Opravte kód tak, aby řazení bylo stabilní.

V určitém problému je velikost zpracovávaného pole s daty rovna  $2n^3 \log n$ , kde  $n$  charakterizuje velikost problému. Pole se řadí pomocí

1. *selection sortu.*
2. *insert sortu.*

Jaká je asymptotická složitost jednotlivých algoritmů nad uvedeným polem?

Pole  $n$  různých prvků je uspořádáno od druhého prvku sestupně, první prvek má nejmenší hodnotu ze všech prvků v poli. Vyberte níže všechny možnosti, které alespoň přibližně charakterizují asymptotickou složitost

Selection sortu,

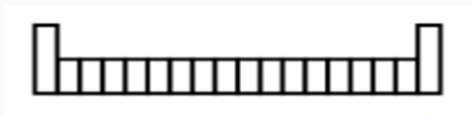
Insert sortu

pracujícího nad tímto konkrétním polem.

1.  $\mathcal{O}(n)$
2.  $\Omega(n)$
3.  $\Theta(n)$
4.  $\mathcal{O}(n^2)$
5.  $\Omega(n^2)$
6.  $\Theta(n^2)$

Jedenáct prvků řadíme pomocí *insert sortu*. Jaký je minimální a maximální možný počet porovnání prvků během tohoto řazení?

*Insert sort* řadí do neklesajícího pořadí pole o  $n$  prvcích, kde jsou stejné všechny hodnoty kromě první a poslední, které jsou větší a navzájem stejné.



1. Kolik porovnání dvou prvků se provede během tohoto řazení?
2. Kolik operací swap nad řazeným polem se provede během tohoto řazení?



V materiálech máte metodu `Sorting::abcSort`. Bez toho aniž byste kód používali, určete, zda řadí pole vzestupně nebo sestupně. Poté se přesvědčte, že tomu tak je a upravte kód tak, aby řadil pole opačným směrem. Všimněte si, že `abcSort` je stabilní. Vaše verze, která řadí opačným směrem by měla být také stabilní.

Pole se řadí pomocí *Quick-sortu*. Určete, jak bude pole rozděleno na „malé“ a „velké“ hodnoty po jednom průchodu, pokud jako pivotní hodnotu použijeme

1. 6

2. 8

6, 10, 8, 5, 7, 2, 3, 9, 1, 4

Předpokládejme, že vždy, když *Quick-sort* rozdělí daný úsek pole na „malé“ a „velké“ hodnoty, bude jeden z těchto úseků třikrát delší než ten druhý.

Určete asymptotickou složitost *Quick-sortu* v tomto případě.