KVADRATICKÉ ŘAZENÍ A QUICK-SORT

Karel Horák, Petr Ryšavý

13. dubna 2016

Katedra počítačů, FEL, ČVUT

Příklad 1

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.

Řešení 1

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

Řazení v Javě je stabilní. [`https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html#sort-java.lang.Object:A-`]

Příklad 2

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

Příklad 3

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

Příklad 4

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?

Řešení 4

Složitost je

$$\Theta(n^6 \log^2 n)$$

v případě selection sortu a

$$\mathcal{O}(n^6 \log^2 n)$$

v případě insert sortu.

Příklad 5

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. $\Theta(n^2)$

Řešení 5

Pro oba algoritmy řazení jsou správné odpovědi

$\Omega(n)$

$\mathcal{O}(n^2)$

$\Omega(n^2)$

$\Theta(n^2)$

Příklad 6

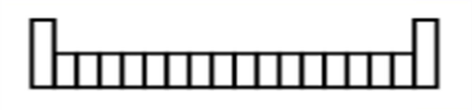
Jedenáct prvků řadíme pomocí *insert sortu*. Jaký je minimální a maximální možný počet porovnání prvků během tohoto řazení?

Řešení 6

Minimálně potřebujeme 10 porovnání a maximálně 55 porovnání.

Příklad 7

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 pole se provede během tohoto řazení?

Řešení 7

1. Potřebujeme $0 + 1 + \underbrace{2 + 2 + \cdots + 2}_{(n-3)\times} + 1 = 2(n-2)$ porovnání.
2. Potřebujeme $0 + \underbrace{1 + 1 + \cdots + 1}_{(n-2)\times} + 0 = n - 2$ operací `swap`.

Příklad 8

V materiálech máte metodu `Sorting::abcSort`. Bez toho aniž byste kód pouštěli, 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.

Příklad 9

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`

Řešení 9

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

Příklad 10

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.

Řešení 10

V prvních $\log_4 n$ patrech vykonáme n práce. Díky tomu je asymptotická složitost $\Omega(n \log n)$.

Pater je $\log_{\frac{4}{3}} n$ a v každém vykonáme nejvýše $\mathcal{O}(n)$ práce. Celkově je tedy třeba $\mathcal{O}(n \log n)$ práce.

Dohromady je tedy asymptotická složitost $\Theta(n \log n)$.