Typografie a publikování

Rychlé řazení

Dmitrii Ivanushkin

12. května 2022

Definice

Řadicí nebo třídicí **algoritmus** je algoritmus zajišťující uspořádání dané sady (pole, seznamu, souboru) datových záznamů do požadovaného pořadí.

Příklady univerzálních druhů algoritmů

- **Bublinkové řazení**

Definice

Řadicí nebo třídicí **algoritmus** je algoritmus zajišťující uspořádání dané sady (pole, seznamu, souboru) datových záznamů do požadovaného pořadí.

Příklady univerzálních druhů algoritmů

- **Bublinkové řazení**
- **Řazení haldou**

Definice

Řadicí nebo třídicí **algoritmus** je algoritmus zajišťující uspořádání dané sady (pole, seznamu, souboru) datových záznamů do požadovaného pořadí.

Příklady univerzálních druhů algoritmů

- **Bublinkové řazení**
- **Řazení haldou**
- **Řazení vkládáním**

Definice

Řadicí nebo třídicí **algoritmus** je algoritmus zajišťující uspořádání dané sady (pole, seznamu, souboru) datových záznamů do požadovaného pořadí.

Příklady univerzálních druhů algoritmů

- **Bublinkové řazení**
- **Řazení haldou**
- **Řazení vkládáním**
- **Řazení slučováním**

Definice

Řadicí nebo třídicí **algoritmus** je algoritmus zajišťující uspořádání dané sady (pole, seznamu, souboru) datových záznamů do požadovaného pořadí.

Příklady univerzálních druhů algoritmů

- **Bublinkové řazení**
- **Řazení haldou**
- **Řazení vkládáním**
- **Řazení slučováním**
- **Rychlé řazení**

Proč se naučit rychlé řazení?

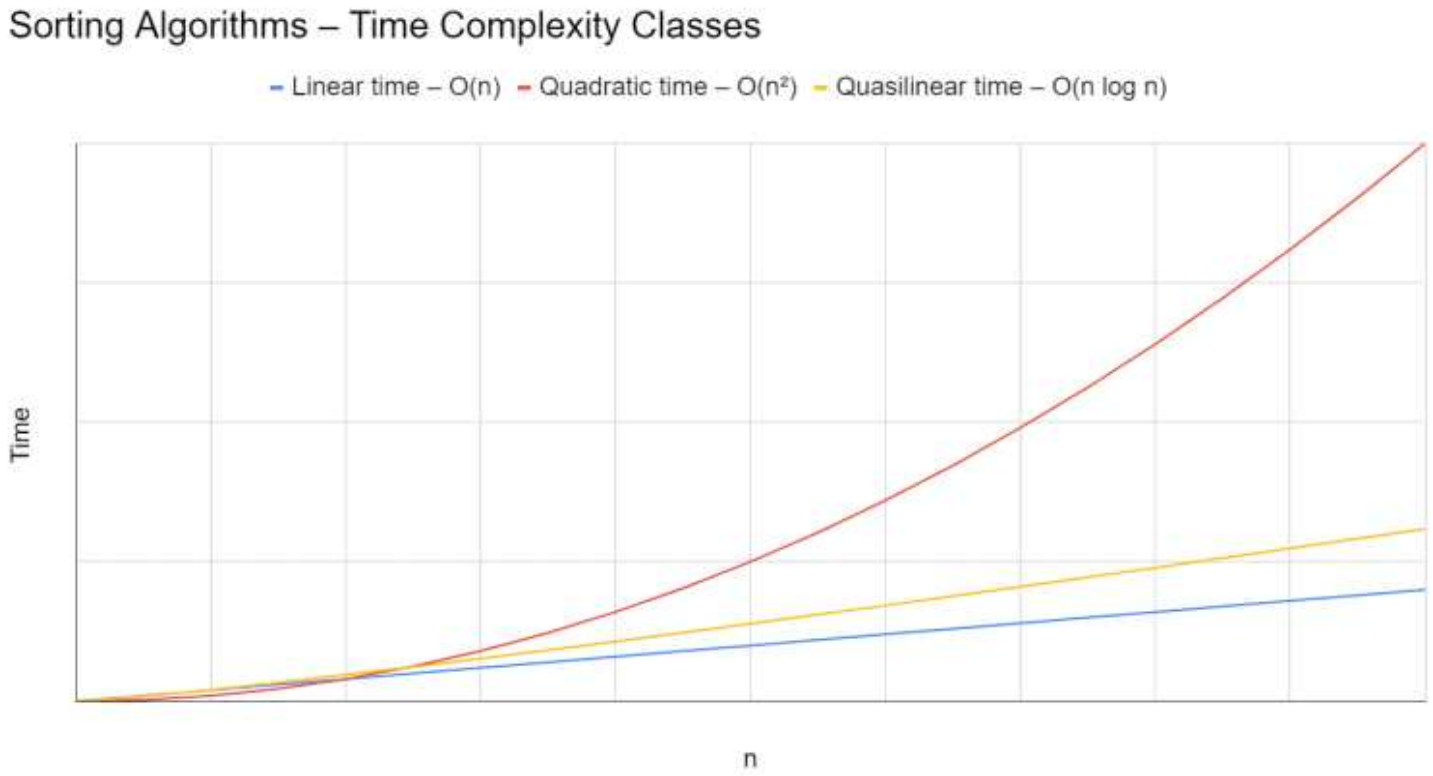
- **Dosahuje mnohem lepší časové složitosti O(n log n).**

Proč se naučit rychlé řazení?

- **Dosahuje mnohem lepší časové složitosti O(n log n).**
- **Je vhodný i pro velké datové sady s miliardami prvků.**

Proč se naučit rychlé řazení?

- **Dosahuje mnohem lepší časové složitosti O(n log n).**
- **Je vhodný i pro velké datové sady s miliardami prvků.**



Porovnání efektivity O(n log n) časové složitosti s jinými

Popis algoritma

Základní myšlenkou algoritmu rychlého řazení je rozdělení řazené posloupnosti čísel na dvě přibližně stejné části (rychlé řazení patří mezi algoritmy typu rozděl a panuj). V jedné části jsou čísla větší a ve druhé menší, než nějaká zvolená hodnota (nazývaná pivot – anglicky „střed otáčení“). Pokud je tato hodnota zvolena dobře, jsou obě části přibližně stejně velké. Pokud budou obě části samostatně seřazeny, je seřazené i celé pole. Obě části se pak rekurzivně řadí stejným postupem, což ale neznamená, že implementace musí taky použít rekurzi.

Základní kroky algoritmu:

1. **volba pivotu**

Základní kroky algoritmu:

1. **volba pivotu**
2. **přesun prvků menších než pivot před něj**

Základní kroky algoritmu:

1. **volba pivotu**
2. **přesun prvků menších než pivot před něj**
3. **přesun prvků větších nebo rovných pivotu za něj**

Základní kroky algoritmu:

1. volba pivotu
2. přesun prvků menších než pivot před něj
3. přesun prvků větších nebo rovných pivotu za něj
4. spuštění algoritmu na část pole obsahující menší prvky

Základní kroky algoritmu:

1. volba pivotu
2. přesun prvků menších než pivot před něj
3. přesun prvků větších nebo rovných pivotu za něj
4. spuštění algoritmu na část pole obsahující menší prvky
5. spuštění algoritmu na část pole obsahující větší prvky (prvky stejné jako pivot je možné přeskočit)

C

```c
void rychlerazeni(int array[], int levy_zacatek, int pravy_zacatek)
{
int pivot = array[(levy_zacatek + pravy_zacatek) / 2];
int levy_index, pravy_index, pom;
levy_index = levy_zacatek;
pravy_index = pravy_zacatek;
do {
    while (array[levy_index] < pivot && levy_index < pravy_zacatek)
        levy_index++;
    while (array[pravy_index] > pivot && pravy_index > levy_zacatek)
        pravy_index--;

    if (levy_index <= pravy_index) {
        pom = array[levy_index];
        array[levy_index++] = array[pravy_index];
        array[pravy_index--] = pom;
    }
} while (levy_index < pravy_index);

if (pravy_index > levy_zacatek) rychlerazeni(array, levy_zacatek, pravy_index);
if (levy_index < pravy_zacatek) rychlerazeni(array, levy_index, pravy_zacatek);
}
```

Použité zdroje


J. Rychlík. Programovací techniky. 2., upravené vyd. České Budějovice: KOPP, 1995. 188 s. ISBN 80-85828-05-7. Kapitola 6 – Řazení.


S. Woltmann. Happy coders. [online], [vid. 2022-05-05]. www.happycoders.eu/algorithms/sorting-algorithms/


Wikipedia. [online], [vid. 2022-05-05]. https://en.wikipedia.org/wiki/Quicksort


V. Hordejčuk. Quick sort (rychlé řazení). [online], [vid. 2022-05-05]. http://voho.eu/wiki/quick-sort/