INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Číslo projektu	CZ.1 .07/1.1 .36/02.0066
Autor	Ladislav Kašpárek
Předmět	Programování a vývoj aplikací
Téma	Řadící algoritmus Quick sort
Metodický pokyn	výkladový text s ukázkami

Řadící algoritmus Quick-sort

Problém

Vstup: Máte zadáno pole čísel, nebo čehokoliv jiného, co se dá a co má smysl řadit.
Úkol: Máte najít algoritmus, který seřadí podle zadaného pravidla prvky v poli.

Nyní podrovněji

Máme zadáno pole - tzn. homogenní datovou strukturu, u které máme přístup k libovolné její položce v konstantním čase na základě indexu.
Máme najít algoritmus, který seřadí prvky podle zadaného pravidla, (u čísel obvykle velikost), ale obecně by nám měla postačit jakákoliv funkce, která pro dva prvky určí jejich pořadí.

Co nás zajímá?

U řadících algoritmů nás zajímá, podobně jako u jiných, jejich složitost. Čili jak roste výpočetní čas (doba trvání algoritmu), když zvyšuji požadavky (rozsah vstupních dat).
Složitost řazení se zapisuje jako funkce, která nám říká, kolik musíme udělat elementárních operací nad zadanými daty, abychom mohli uživateli s jistotou říci, že nyní je celé pole seřazeno.
U řazení to bývá trochu problém, protože mohu měřit dvě elementární operace:
Jednak porovnání dvou prvků (abych zjisitl zda jsou ve správném pořadí) - to musím provést vždy.
A jednak přesun prvků - to se někdy provádět nemusí (protože třeba už jsou ve správném pořadí).
Minule jsme si ukázali kvadratické (pomalé) řadící algoritmy a dnes bych rád představil zástupce těch rychlejších řadících algoritmů.

Quick sort

Základní myšlenka:

Publikoval ho sir Charles Anthony Richard Hoare v roce 1962.
Základní myšlenkou je rozdělení řazené posloupnosti čísel na dvě přibližně stejné části.

V jedné části jsou čísla větší a ve druhé menší, než nějaká zvolená hodnota nazývaná pivot.
Pokud je tato hodnota zvolena dobře, jsou obě části přibližně stejně velké.
Pokud budou obě části samostatně seřazeny, je seřazené i celé pole.
Obě menší části se pak rekurzivně řadí stejným postupem.

Kód:

```
public static void quicksort(int l, int r, int[] Pole, Random rand) {
    int i = l;
    int j = r;
    int pivot = Pole[(rand.nextInt(r - l)) + l];
    do {
        while (Pole[i] < pivot) {
            i++;
        }
        while (Pole[j] > pivot) {
            j--;
        }
        if (i <= j) {
            int pom = Pole[i];
            Pole[i] = Pole[j];
            Pole[j] = pom;
            i++;
            j--;
        }
    } while (i < j);
    if ((j - l) > 0) {
       quicksort(l, j, Pole, rand);
    }
    if ((r - i) > 0) {
       quicksort(i, r, Pole, rand);
    }
}
```

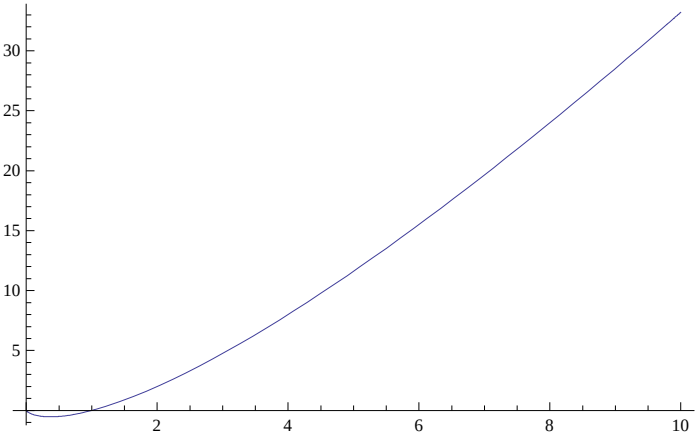
Složitost :

Nejprve tu máme počet půlení pole.
$\log_2(N)$

Dále tu máme procházení pole a přehazování prvků.
N

In[10]:=
```
Plot[x Log2[x], {x, 0, 10}]
```

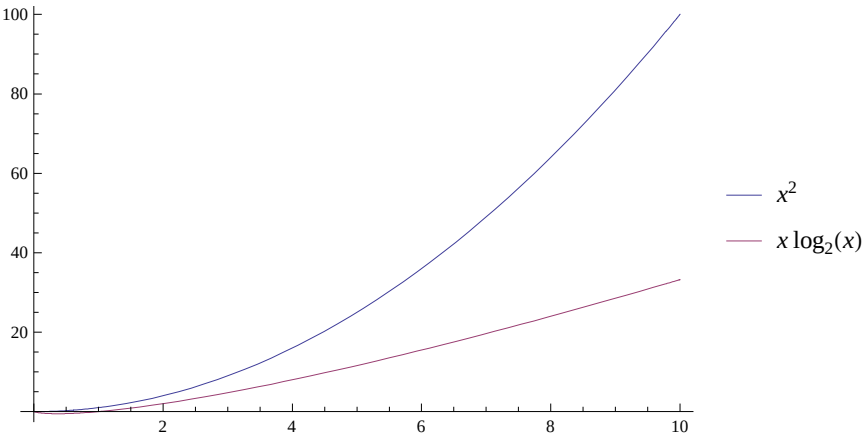
Out[10]=



Pokud to porovnáme s kvadratickým algoritmem:

In[9]:=
```
Plot[{x^2, x Log2[x]}, {x, 0, 10}, PlotLegends → "Expressions"]
```

Out[9]=



Zhodnocení :

V průměrném případě je tedy Quick sort asymptoticky méně složitý jak kvadratické řadící algoritmy, ale pozor pokud je pivot volen špatně, můžeme se dostat do situace kdy nám pivot rozdělí posloupnost na jeden prvek a zbytek pole a pokud se toto opakuje N krát dostáváme opět kvadratickou složitost.

Jiná varianta :

Publikoval jí Nico Lomuto v roce 1984.
Zatímco, v Hoareho variantě je snaha trefitt pivota na střed a vytvořit dvě stejné půlky.
Zde je základní myšlenka následjící: postupně umisťujeme prvky menší jak pivot do levé půlky pole, která se tímto zvětšuje.

Kód:

```
public static void QuickSortLomuto(int l, int r, int[] pole) {
    int pivot = pole[l];
    int pmin = l;
    for (int i = l + 1; i <= r; i++) {
        if (pole[i] < pivot) {
            pmin++;
            int tmp = pole[pmin];
            pole[pmin] = pole[i];
            pole[i] = tmp;
        }
    }
    int tmp = pole[l];
    pole[l] = pole[pmin];
    pole[pmin] = tmp;
    if (l < (pmin - 1)) QuickSortLomuto(l, pmin - 1, pole);
    if (r > (pmin + 1)) QuickSortLomuto(pmin + 1, r, pole);
}
```

Zdroje

http://reference.wolfram.com/mathematica/ref/Plot.html