UNICORN COLLEGE s.r.o. Praha

DÚ č.2 z předmětu základy návrhu a optimalizace algoritmů

Hrabičkové třídění

Pavel MAJER

2016

Obsah

Seznam grafů

Kontext

Existuje mnoho třídících algoritmů, které využívají mnoho různých principů pro seřazení dat. Cílem tohoto cvičení seznámit se s jedním konkrétním třídícím algoritmem (Hrabičkové třídění) a porovnat jeho chování s dalšími algoritmy, o kterých víme, jakou mají teoretickou složitost.

Hrabičkové třídění

Hrabičkové třídění si bere jako inspiraci jednotlivé průchody Insertion sortu, ve kterých se vezme další nový prvek z pole a zatřídí se do již setříděné části posloupnosti. Hrabičkové tříděné pracuje ve fázích, kde se porovnávají prvky s větší vzdáleností od sebe (dá se přirovnat k pomyslným hrábím). Tato vzdálenost (označovaná jako „k") se postupně zkracuje až k číslu 1, kde se pak algoritmus chová stejně jako Insertion sort.

Předpokládá se, že tímto způsobem by se měla čísla přesouvat na větší vzdálenosti s použitím mnohem menšího počtu iterací, než u běžného Insertion sortu.

Způsob měření

Pro účely měření jsem vytvořil v ruby program hrabisort.rb a hrabibench.rb podle zadání.

- Hrabisort obsahuje metodu, která nastavuje velikosti všech rozestupů hrabiček (k) podle parametru K. Poté provede samotné třídění pro každou velikost k.
- Hrabibench vytváří různé sady dat, a tato data se postupně setřídí s pomocí 5 třídících algoritmů (Hrabičky 1..3, Insertion sort, Quicksort), a změří se čas, potřebný k setřídění. Pro vyšší vypovídající hodnotu se testy na malých polích provádí vícekrát za sebou.

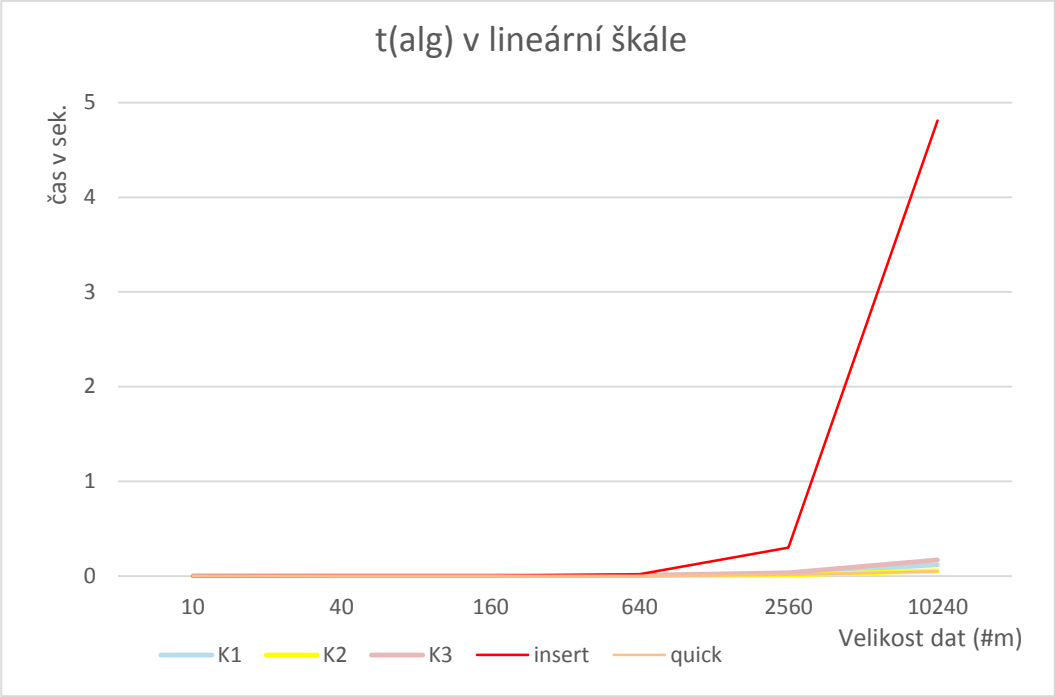
Výsledky měření

Výsledky měření jsou zaznamenané ve výstupních souborech times.txt a ratios.txt.

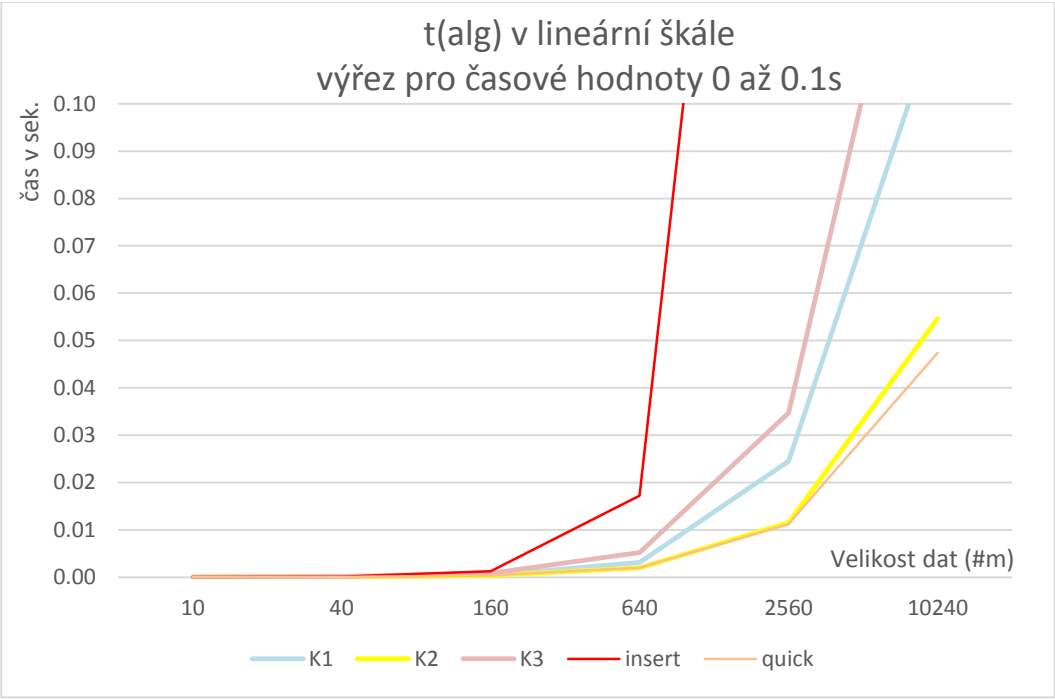
- Times.txt ukazuje pro každý algoritmus čas, který je potřebný k setřídění polí s „m" prvky.
- Ratios.txt ukazuje pro každý algoritmus a velikost pole, kolik času je potřeba navíc při přidání dalšího prvku. Ukazuje, jak velký vliv na čas třídění má velikost tříděných polí. Čas je navíc pokrácen o log (m), který reprezentuje běžný minimální čas navíc, který je potřeba na setřídění i u těch nejlepších třídících mechanismů.

Times.txt [t(alg)]

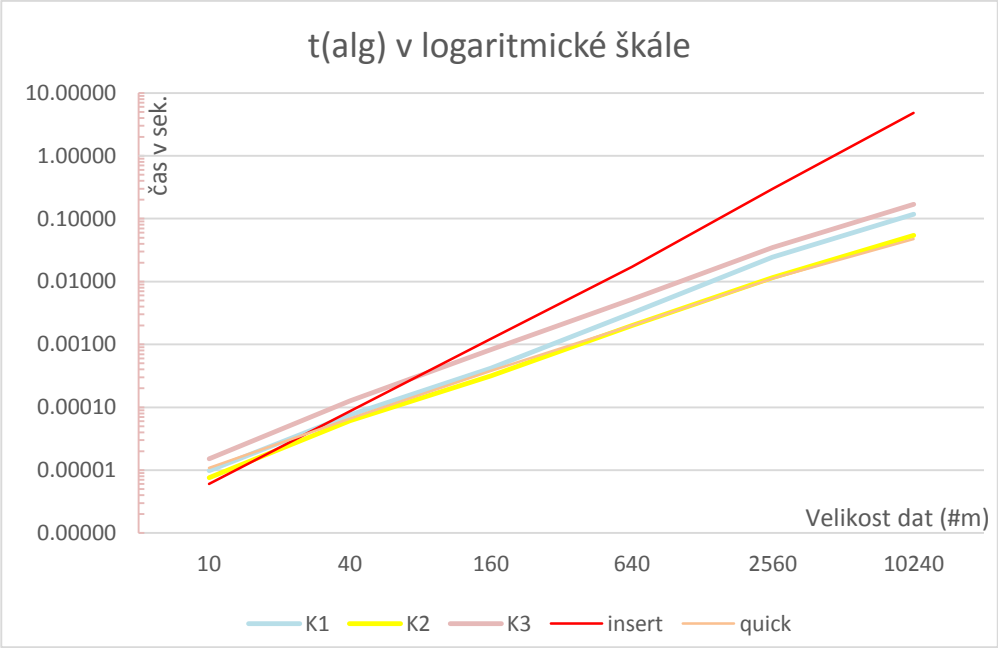
# m	K1	K2	K3	Insert	Quick
10	9.86E-06	7.56E-06	1.52E-05	6.00E-06	1.09E-05
40	7.53E-05	6.15E-05	1.27E-04	8.68E-05	6.57E-05
160	4.18E-04	3.16E-04	8.27E-04	1.22E-03	3.77E-04
640	3.13E-03	1.99E-03	5.22E-03	1.72E-02	2.00E-03
2560	2.45E-02	1.15E-02	3.47E-02	2.99E-01	1.12E-02
10240	1.18E-01	5.46E-02	1.69E-01	4.81E+00	4.74E-02



Graf 1 – časová náročnost algoritmů t (alg) v lineární škále



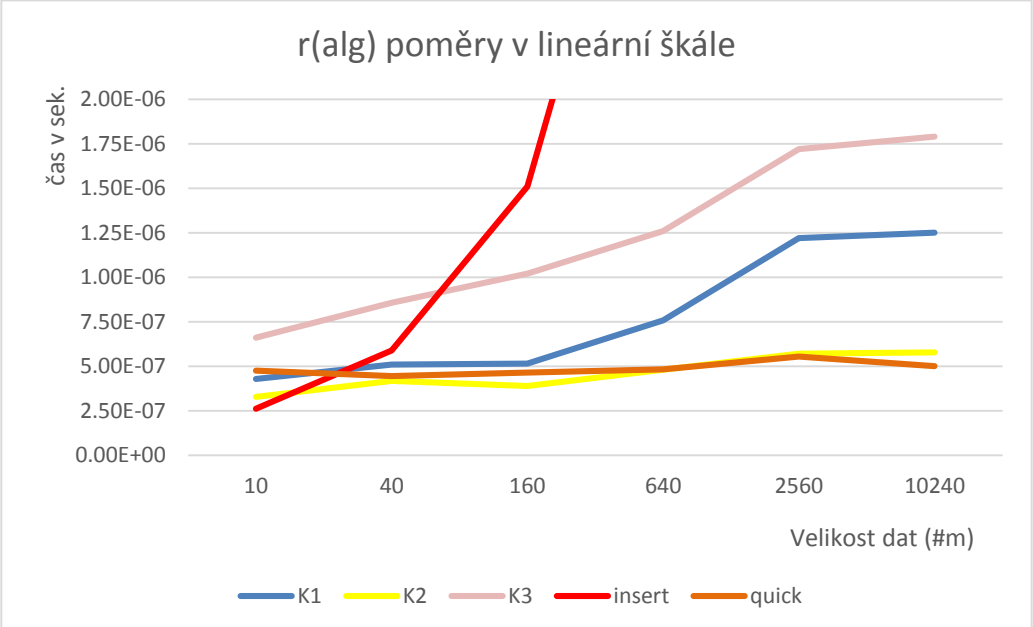
Graf 2 - časová náročnost algoritmů t (alg) v lineární škále s omezením hodnot od 0 do 0,1



Graf 3 - časová náročnost algoritmů t (alg) v logaritmické škále

Ratios.txt [t(alg)/mlogm)]

# m	K1	K2	K3	Insert	Quick
10	4.28E-07	3.28E-07	6.61E-07	2.61E-07	4.75E-07
40	5.10E-07	4.17E-07	8.57E-07	5.88E-07	4.45E-07
160	5.15E-07	3.89E-07	1.02E-06	1.51E-06	4.64E-07
640	7.57E-07	4.80E-07	1.26E-06	4.16E-06	4.82E-07
2560	1.22E-06	5.70E-07	1.72E-06	1.49E-05	5.55E-07
10240	1.25E-06	5.77E-07	1.79E-06	5.09E-05	5.01E-07



Graf 4 – poměrná časová náročnost na třídící algoritmy při přidání dalšího prvku [t(alg)/mlogm)]

Zhodnocení Hrabičkového třídění

Algoritmus Hrabičkového třídění se při optimálním nastavení mezery mezi pomyslnými hroty (k) je o mnoho rychlejší než Insertion sort. Může blížit i rychlosti Quicksortu. O Quicksortu víme, že má složitost $O(n.\log_2 n)$, tudíž i rychlost tohoto algoritmu se této hodnotě může blížit.

V testovaných případech se jevila varianta K2 Hrabičkového třídění ($\frac{3^a-1}{2}$, do k < n/3) jako nejoptimálnější ze všech zkoumaných. V Grafu č.4 jde žlutá křivka (K2) téměř souběžně s hnědou (Quicksort) na všech velikostech testovaných prvků. Zvýšení množství tříděných prvků na 10tis. nemělo na poměrnou délku třídění K2 téměř žádný vliv (v porovnání s Insertion sortem, nebo i s K1 a K3)

Zhruba do 640 řazených prvků byl Hrabičkový sort K2 dokonce rychlejší než Quicksort. Jde pravděpodobně o určitou podobnost s Insertion sortem, který je ze všech testovaných algoritmů nejrychlejší, pokud se třídí velmi malé množství dat (např. s 10 prvky). Případně to může být spojeno s určitou neefektivitou Quicksortu při třídění menšího množství dat.

Insertion Sort ve variantě K1 a K3 jsou o něco pomalejší než K2. Například pole s 10tis prvky s použitím těchto variant setřídění trvalo cca o 0.05 sekundy déle. V porovnání s Insertion sortem, který trval cca o 4sekuny déle, jde téměř o zanedbatelné zpomalení. O Insertion sortu víme, že má složitost O (n^2), a z těchto testů plyne, že se Hrabičkový sort v žádné ze svých variant této složitosti neblíží.

Vhodná volba optimální mezery K je velice důležitá, a to hlavně pro třídění více prvků. Při řazení 10tis prvků byl rozdíl mezi variantami K1-K3 pouhé 0.05s, ale při řazení 100 násobně většího množství prvků začíná být rozdíl mezi Quicksortem s K1,K2,K3 více zřetelný. V tomto dodatečném měření Quicksort trval 6s, K2 14s, K1 23s a varianta K3 trvala dokonce 33s.

Závěr

V testovaných případech se jevila varianta K2 Hrabičkového třídění jako nejoptimálnější pro naši nasimulovanou situaci. Při řazení do 10tis prvků se varianta K2 chová podobně jako Quickosort. Větší rozdíly nastanou při dalším navýšení velikosti řazené množiny na stonásobek. Zde se pak ukazuje Quicksort za nejvýhodnější, a Hrabisort K2 je až na druhém místě.

Dá předpokládat, že na volbu vhodného třídícího algoritmu může mít i vliv způsob, jak jsou data zamíchána. Při našem testování jsme používali zcela náhodná data. Je pravděpodobné, že by se výsledky mohly lišit, pokud by data byla setříděna z 95% ze začátku, a třídilo by se pouze nově přidaných 5% a tak podobně.

Toto téma je pro mě velmi zajímavé a přínosné i pro mé zaměstnání. Plánuji ho budoucna zkoumat hlouběji, například s modelováním určitých reálných situací a následným zkoumáním toho, jak se jednotlivé případy chovají.