DSA – Zadanie 1

Andrej Stuchlý

Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

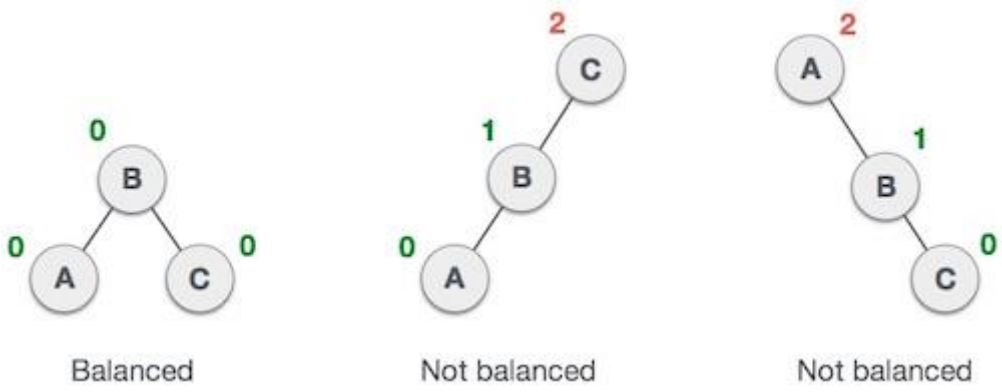
2023

Zadanie

Našou úlohou v tomto zadaní bolo porovnať viacero implementácií dátových štruktúr z hľadiska efektivity operácií **insert**, **search** a **delete** v rozličných situáciách. Ja som si na porovnávanie vybral tieto 4 dátové štruktúry: AVL Tree, Red-Black Tree, Chaining Hashtable, Open Addressing Hashtable.

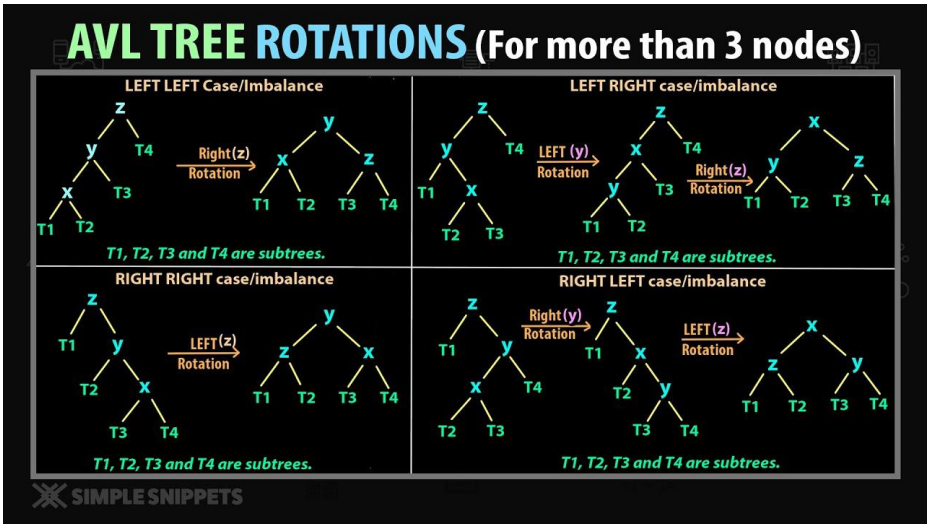
1. Adelson-Velskii and Landis (AVL) tree

AVL strom funguje na princípe, že sa pre každý uzol rozdiel výšky dvoch podstromov detských uzlov líšia najviac o jednotku, preto je dôležité pri každom vkladaní do stromu a mazaní zo stromu kontrolovať vyváženosť stromu a ak príde k tomu že výšky dvoch podstromov sa líšia o viac ako jednotku je potrebné vykonať rotácie.



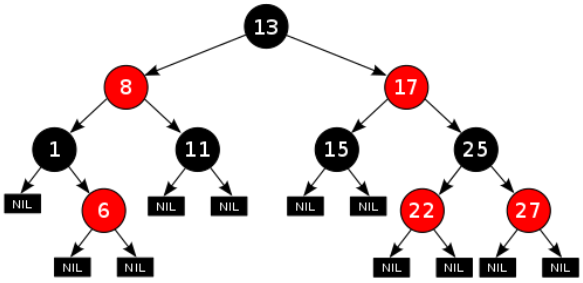
Rotácie

Rotácie sa využívajú v prípade, že strom nie je vybalansovaný a prevažuje na jednu stranu.



Funkcie insert, search aj delete majú zložitosť O(log n) v priemernom aj najhoršom prípade.

2. Red-Black (RB) Tree



Červeno čierny strom pracuje na nasledujúcom princípe:

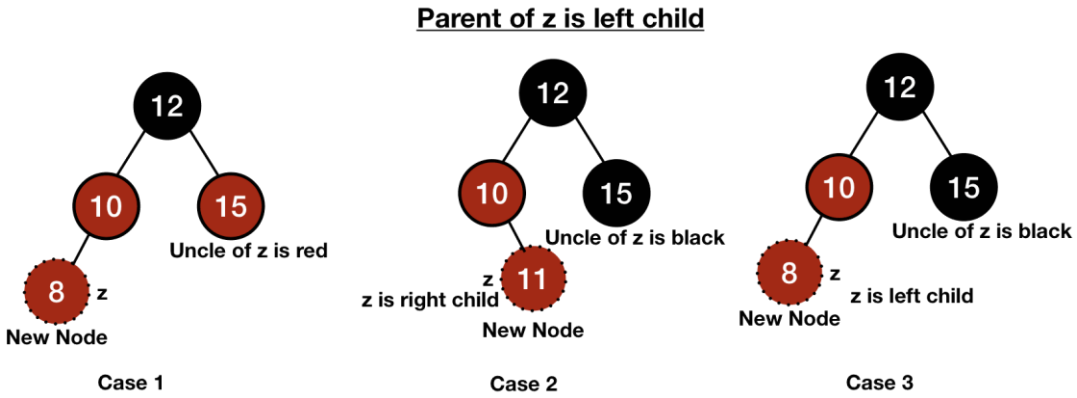
1. Objekt je vždy červený alebo čierny

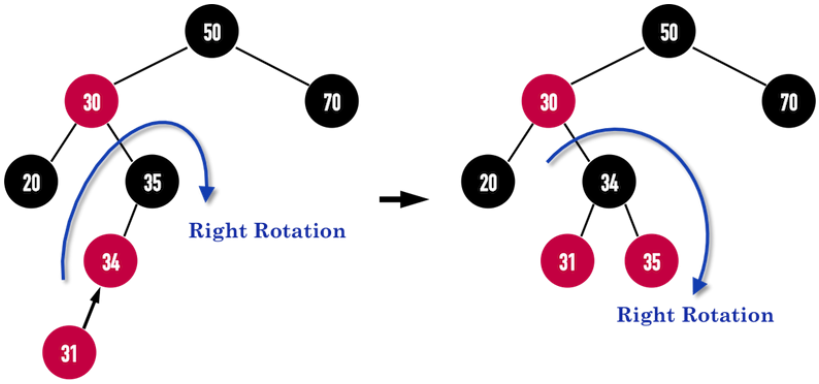
2. Koreň a listy(NIL) sú čierne

3. Ak je objekt červený tak jeho deti sú čierne

4. Všetky cesty k objektu k jeho NIL potomkom obsahuje rovnaký počet čiernych objektov

Pri vkladaní vždy vkladáme podstrom s červenou farbou a dvoma listami. Balansovať daný strom je potrebné vtedy ak prišlo k prepojeniu dvoch stromov s červenou farbou. Vtedy je potrebné vykonať rotácie.





Funkcie insert, search aj delete majú zložitosť O(log n) v priemernom aj najhoršom prípade.

3. Hashtable

Hashovacia tabuľka asociuje kľúče s hodnotami. Vďaka tejto tabuľke sme schopný veľmi rýchlo vyhľadávať a vkladať údaje asociované k danému kľúču. Pomocou hashovacej funkcie pridelí každému kľúču miesto v tabuľke kde uloží údaje alebo ukazovateľ na údaje. Pri veľkom množstve údajov alebo malej veľkosti hashovacej tabuľky môže často prichádzať ku kolíziám(jednému miestu v tabuľke pridelí hashovacia funkcia viac kľúčov). Hashovacia tabuľka rieši tieto kolízie dvoma spôsobmi:

3.1.Chaining

Na každom políčku hasovacej tabuľky je spájaný zoznam(chain).Ak vkladáme údaj s kľúčom ktorému vygeneruje hashovacia funkcia pozíciu v tabuľke, ktorá je už obsadená(nie je NULL) tak pripípaja údaj na koniec zoznamu ktorí sa na danej pozícií nachádza.

3.2.Open Addressing – Linear Probing

Tento princíp funguje tak, že ak je dané miesto v tabuľke obsadené pozerá sa na vedľajšie políčka až kým nenájde voľné a tam ho vloží. Ak nenájde žiadne voľné miesto je potrebné tabuľku zväčšiť a jednotlivé pozície prehashovať.

Insert, search a delete má v hashovacích tabuľkách konštantnú časovú zložitosť O(1) v priemernom prípade bez ohľadu na to, koľko prvkov sa v tabuľke nachádza, ale v zriedkavom najhoršom prípade ma zložitosť O(n), ak by nám hashovacia funkcia vygenerovala rovnakú pozíciu pre všetky vkladané prvky.

4. Testovanie

Čo sa týka vlastného testovania vytvorených dátových štruktúr, v tabuľkách nižšie je možné vidieť merania v milisekundách pre rôzne počty prvkov. Pri každom teste som vygeneroval daný počet čísel a textov do poľa a následne som tie isté dáta použil pre každú dátovú štruktúru. Výsledné časy sú vždy závislé aj od vygenerovaných prvkov a ich poradia, tieto faktory dokážu značne skrátiť alebo predĺžiť potrebný čas.

4.1.Tabuľky

AVL Strom

Počet prvkov	Čas vkladania	Čas hľadania	Čas mazania
50 000	60	17	65
100 000	121	59	144
250 000	251	168	307
500 000	539	312	611
1 000 000	1385	811	1709
2 500 000	4369	2454	5123
5 000 000	9593	5905	12120
10 000 000	22795	14553	29679

RB Strom

Počet prvkov	Čas vkladania	Čas hľadania
50 000	54	24
100 000	126	52
250 000	241	130
500 000	413	312
1 000 000	1079	936
2 500 000	3237	2844
5 000 000	7338	5080
10 000 000	16591	13036

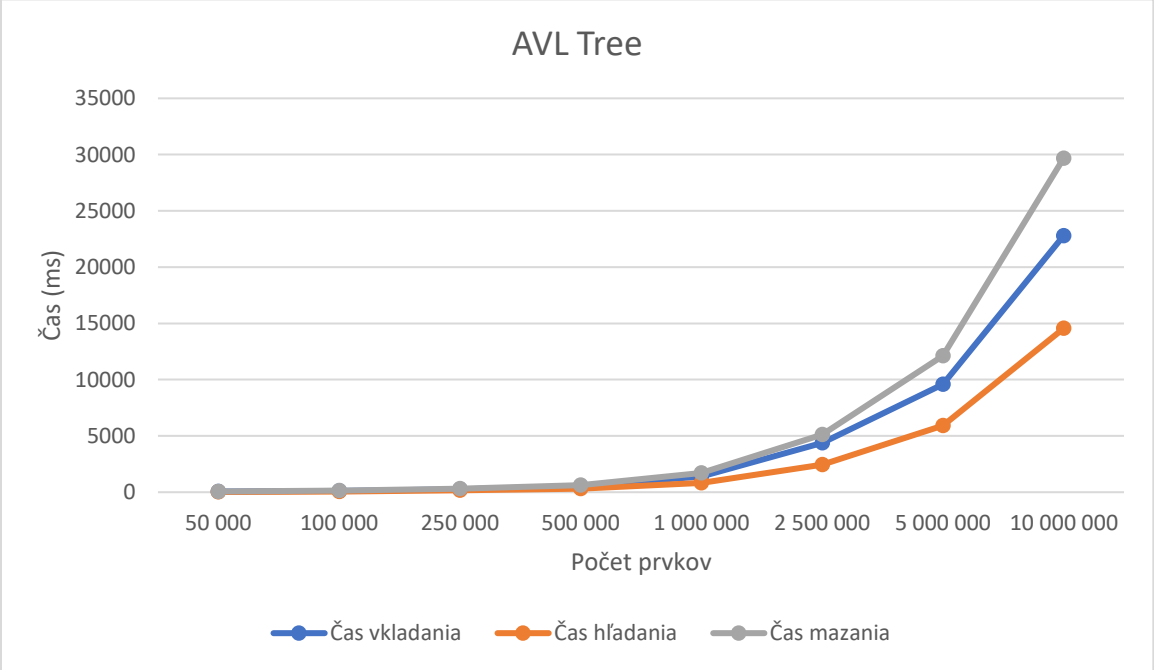
Hash Chaining

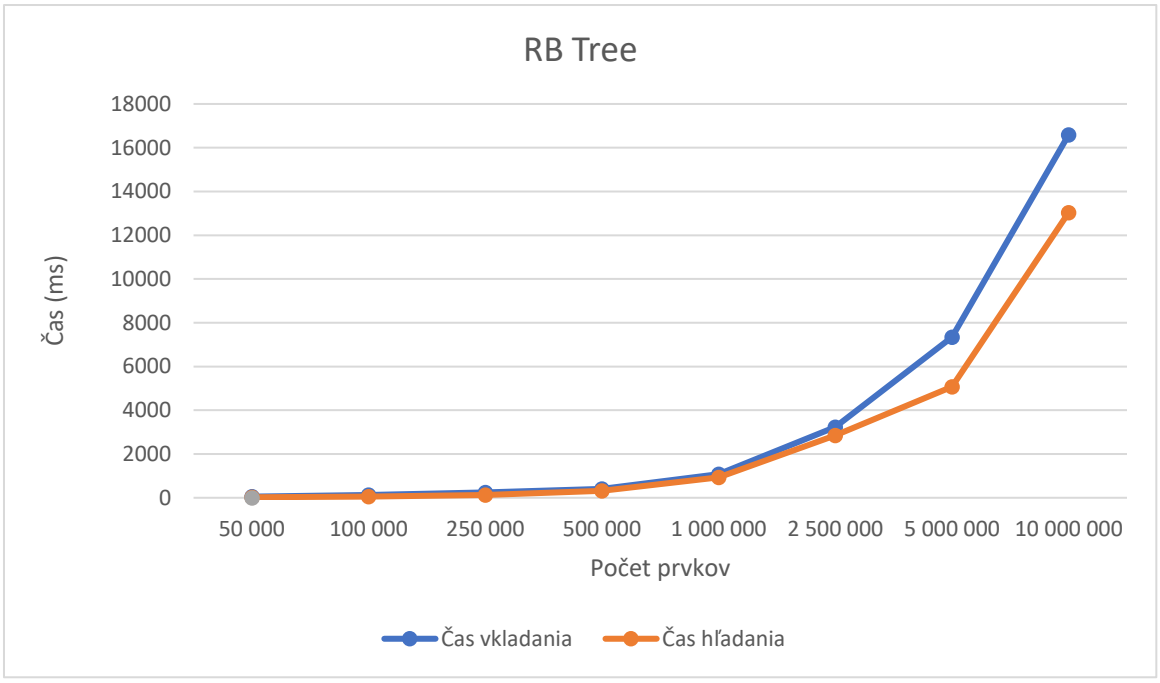
Počet prvkov	Čas vkladania	Čas hľadania	Čas mazania
50 000	28	6	6
100 000	71	10	29
250 000	119	20	62
500 000	231	35	134
1 000 000	376	71	226
2 500 000	1430	183	731
5 000 000	2355	355	1251
10 000 000	5547	745	2827

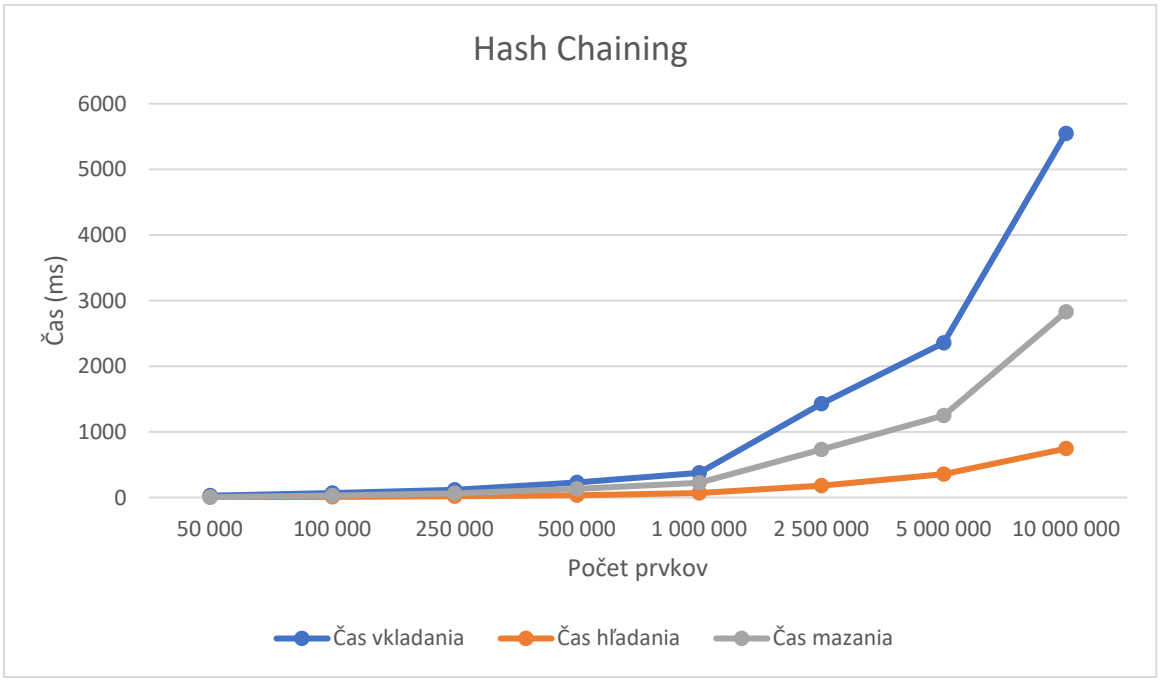
Hash Open Addressing – Linear Probing

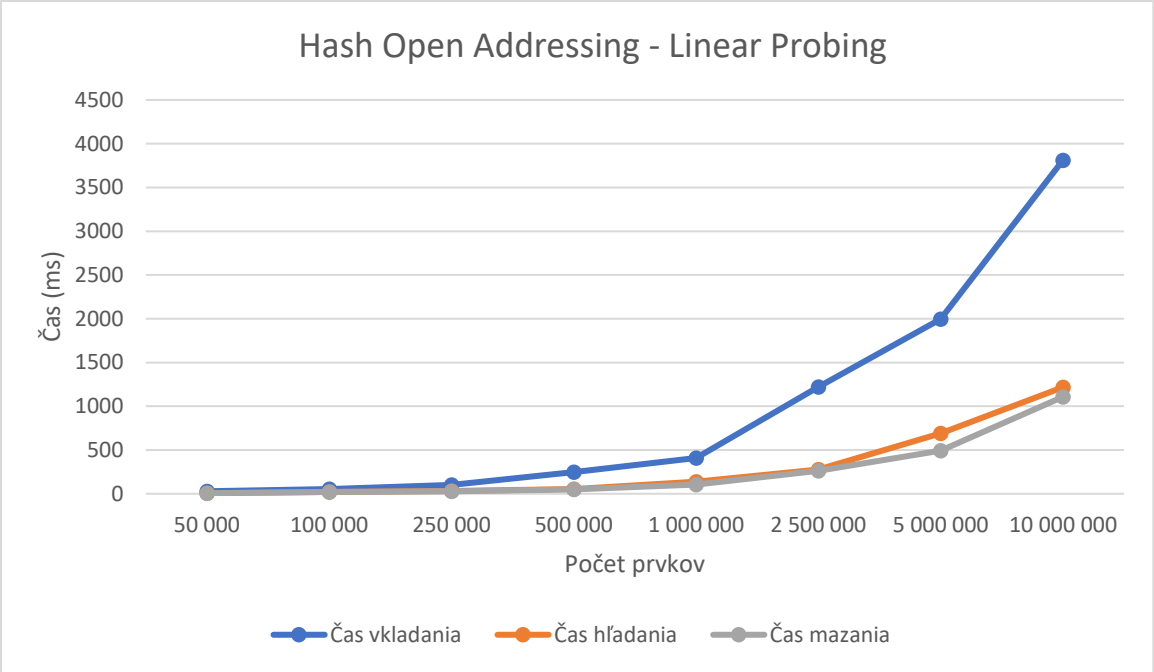
Počet prvkov	Čas vkladania	Čas hľadania	Čas mazania
50 000	29	7	6
100 000	53	22	22
250 000	102	32	29
500 000	247	54	52
1 000 000	407	138	105
2 500 000	1219	276	262
5 000 000	1996	690	494
10 000 000	3810	1217	1107

4.2. Grafy




RB Tree
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
Čas (ms)
50 000
100 000
250 000
500 000
1 000 000
2 500 000
5 000 000
10 000 000
Počet prvkov
Čas vkladania
Čas hľadania


Hash Chaining
6000
5000
4000
3000
2000
1000
0
Čas (ms)
50 000
100 000
250 000
500 000
1 000 000
2 500 000
5 000 000
10 000 000
Počet prvkov
Čas vkladania
Čas hľadania
Čas mazania



4.3. Výstup z testov

Vďaka dátam z tabuliek a grafov vidno, že dátové štruktúry pracujú relatívne efektívne do počtu prvkov 1 000 000 – 2 500 000, kedy operácie trvajú do pár sekúnd. Pri vyššom počte prvkov už operácie trvajú pomerne dlho a tieto štruktúry by neboli to najlepšie riešenie.

Čo sa týka mojich dvoch stromov, časovo sú na veľmi podobnej úrovni, časy sú dosť závislé aj od poradia prvkov, pretože jeden typ stromu môže potrebovať omnoho viac rotácii na vybalansovanie ako druhý.

Dve hashovacie metódy sú veľmi podobné čo sa týka efektívnosti ako môžeme vidieť aj z výsledkov testov. Taktiež vidíme že sú efektívnejšie ako stromy, ale to všetko závisí od toho aká je veľká tabuľka a hlavne ako často pri vkladaní dochádza ku kolíziám.

Výstup z testov jasne ukazuje, že najvýhodnejšie je použiť jednu z hashovacích funkcii: v prípade insertu sú v niektorých prípadoch 2-3x rýchlejšie ako stromy, vyhľadávanie a mazanie 10-15x rýchlejšie. Najväčší rozdiel je vidno pri veľkom počte prvkov (5 000 000 a viac), kde je hashovanie omnoho efektívnejšie.

4.4. Ukážka výpisu z testov

```
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
100000 čísel bolo vygenerovaných do poľa

Trvanie vkladania do AVL stromu pre 100000 nodov: 121 ms
Trvanie vyhľadávania v AVL strome pre 100000 nodov: 59 ms
Trvanie mazania z AVL stromu pre 100000 nodov: 144 ms

Trvanie vkladania do RB stromu pre 100000 nodov: 126 ms
Trvanie vyhľadávania v RB strome pre 100000 nodov: 52 ms

100000 textov bolo vygenerovaných do poľa

Trvanie vkladania do 2D hashovacej tabuľky s reťazením pre 100000 nodov: 71 ms
Trvanie hľadania v 2D hashovacej tabuľke s reťazením pre 100000 nodov: 10 ms
Trvanie mazania z 2D hashovacej tabuľky s reťazením pre 100000 nodov: 29 ms

Trvanie vkladania do 2D hashovacej tabuľky s otvoreným adresovaním pre 100000 nodov: 53 ms
Trvanie hľadania v 2D hashovacej tabuľke s otvoreným adresovaním pre 100000 nodov: 22 ms
Trvanie mazania z 2D hashovacej tabuľky s otvoreným adresovaním pre 100000 nodov: 22 ms


Process finished with exit code 0
```

5. Záver

Obidve dátové štruktúry – binárne vyhľadávacie stromy aj hashovanie – majú svoj zmysel a uplatnenie, na základe meraní by som si však vybral hashovanie, nakoľko operácie sa vykonávajú značne rýchlejšie ako u vyhľadávacích stromov, hlavne pri vysokom počte prvkov.