Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Dátové štruktúry a algoritmy

Zadanie 2

LS 2020/2021

Samuel Kováč

Študijný program: B-INFO4

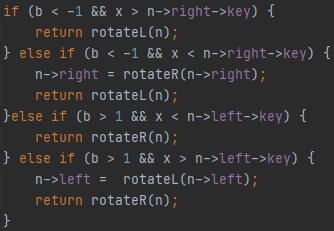
Vyučujúci: Ing. Lukáš Kohútka, PhD.

Cvičenia: Pondelok 9:00

Marec 2021

**AVL Strom (Vlastná implementácia)**

AVL stromy využívajú na balancovanie rotáciu podstromov. Po štandardnom vložení do BST (binárneho vyhľadávacieho stromu) sa pohybujeme hore až po prvý nevybalancovaný uzol. Následne máme 4 možnosti na rotáciu :

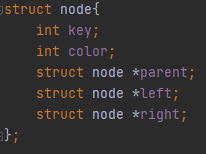


O rotácií rozhodujú 2 faktory – “vybalancovanosť“ stromu ***b*** *(získame rozdielom vo výške ľavého a pravého podstromu)* a porovnanie novo-vloženého uzla s ľavou a pravou stranou nevybalancovaného uzla. Okrem balancovania sa funkcie AVL stromu neodlišujú od klasického BST.

**Červeno-čierny strom (Prevzatá implementácia)**

Zdroj implementácie : <https://gist.github.com/aagontuk/38b4070911391dd2806f>

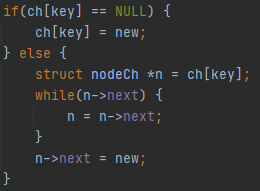
Červeno-čierne stromy využívajú na balancovanie navyše atribút uzlov – farbu.



Každý uzol môže byť červený alebo čierny. Koreň je vždy čierny. Nikdy nie sú pod sebou 2 rovnaké farby – červený uzol nesmie mať červené dieťa ani červeného rodiča. Listy tohto stromu majú hodnotu NULL – táto implementácia využíva špeciálny uzol typu NILL, pričom listy stromu sú vždy čierne. Každá cesta od koreňa až po list stromu má rovnaký počet čiernych uzlov. Červeno-čierne stromy tiež využívajú rotáciu podstromov podobne ako u AVL stromov.

**Chain hash (Vlastná implementácia)**

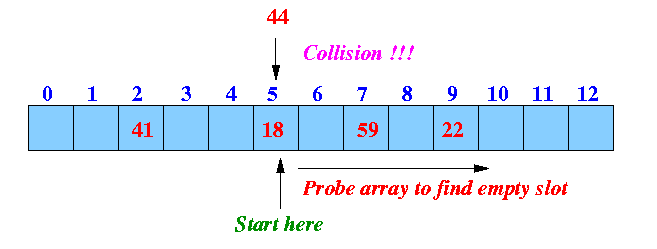
Chain hash je jednoduché riešenie kolízií pri hashovaní. Funguje na princípe spájaného zoznamu. Program najskôr zahashuje zadanú hodnotu, následne nahliadne do hashovacej tabuľky a zistí či už na danej adrese existuje hodnota. Ak áno, nastáva kolízia a program teda priradí existujúcej hodnote ukazovateľ na novú hodnotu a vytvorí tak spájaný zoznam.



**Open-Adressing hashovanie (Prevzatá implementácia)**

Zdroj implementácie : <https://steemit.com/programming/@drifter1/programming-c-hashtables-with-linear-probing>

Open-Adressing hashovanie (nazývané aj Linear Probing) pri kolízií pokračuje v prehľadávaní danej hash tabuľky až kým nenájde voľné miesto. Ak sa voľné miesto nepodarí nájsť, znamená to že je tabuľka plná.

[](https://steemit.com/programming/@drifter1/programming-c-hashtables-with-linear-probing)

**Testovanie**

Každú implementáciu som testoval dvoma testami – inkrementačným a náhodným. Časy uvedené v testoch sú celkové časy potrebné na všetky operácie insert/search.

**1. Inkrementačný test AVL stromu**

****Postupne som pridával do stromu hodnoty od 0 po 1000000, následne som vyhľadával hodnoty od 0 po 1000000.

**2. Inkrementačný test RB stromu**

Postupne som pridával do stromu hodnoty od 0 po 1000000, následne som vyhľadával hodnoty od 0 po 1000000.



**3. Inkrementačný test Chain Hash**

Postupne som pridával do hash tabuľky hodnoty od 0 po 1000000, následne som vyhľadával hodnoty od 0 po 1000000.

**4. Inkrementačný test Open Adressing Hash**

Postupne som pridával do hash tabuľky hodnoty od 0 po 1000000, následne som vyhľadával hodnoty od 0 po 1000000.

**5. Náhodný test AVL stromu**

Postupne som pridával do stromu 1000000 krát náhodné hodnoty od 0 po 10000000, následne ich všetky vyhľadal.

**6. Náhodný test RB stromu**

Postupne som pridával do stromu 1000000 krát náhodné hodnoty od 0 po 10000000, následne ich všetky vyhľadal.

**7. Náhodný test Chain Hash**

Postupne som pridával do hash tabuľky 1000000 krát náhodné hodnoty od 0 po 10000000, následne ich všetky vyhľadal.



Krátky čas vyhľadávania je spôsobený veľkým rozptylom možných hodnôt pričom klesá šanca na kolízie.

**8. Náhodný test Open Adressing Hash**

Postupne som pridával do hash tabuľky ***50000*** krát náhodné hodnoty od 0 po 10000000, následne ich všetky vyhľadal.



V tomto teste som pridal len 50000 hodnôt keďže sa algoritmus dostával do obrovských časov.

**Zhodnotenie**

Pri inkrementačných testoch vynikal RB strom, no výhody AVL stromu boli vidieť až pri náhodných testoch. AVL stromy využívajú veľa operácií rotácií na vkladanie/vyhľadávanie a preto pri častom používaní týchto funkcií je lepšie použiť RB strom.

Open Adressing Hash dosahoval lepšie výsledky pri inkrementačných testoch no pri náhodných hodnotách jednoznačne zvíťazil Chain Hash.