Vahabov Oleksandr
pondelok 16:00 – 18:00

DSA zadanie 5

Táto práca sa zameriava na porovnanie troch dátových štruktúr navrhnutých na optimalizáciu operácií vkladania, vyhľadávania a vymazávania. Pre analýzu boli vybrané váhovo vyvážený strom, červeno-čierny strom a hašovacia tabuľka s dvojitým hašovaním. Cieľom je zhodnotiť výkonnosť týchto štruktúr pri spracovaní veľkého množstva údajov.

Weight-Balanced Tree

Váhovo vyvážený strom je binárny vyhľadávací strom, ktorý udržiava rovnováhu pomocou koeficientu vyváženosti (Alfa). Po vložení nového uzla sa spustí vyvažovacia funkcia, ktorá kontroluje a opravuje vyváženosť stromu rotáciami (vľavo a vpravo). Rotácie obnovujú rovnováhu stromu, čím sa zabezpečuje logaritmická časová zložitosť operácií.

Vyhľadávanie a mazanie v tomto strome sú štandardné, podobne ako pri červeno-čiernom strome. Po odstránení uzla sa opäť spustí vyvažovacia funkcia. Faktor vyváženosti uzla sa určuje pomerom veľkostí ľavého a pravého podstromu. Ak sa strom stane nevyváženým, vykonajú sa rotácie na jeho opätovné vyváženie.

Zložitosť algoritmu:

Search O(log n)

Insertion O(log n)

Deletion O(log n)

Red-Black Tree

Červeno-čierny strom je binárny vyhľadávací strom, kde každý uzol má buď červenú, alebo čiernu farbu. Implementoval som funkciu na vytvorenie stromu, kde je koreň čierny a používam špeciálny "null" uzol, ktorý je tiež čierny. Vkladacia funkcia nájde miesto pre nový prvok a po jeho vložení sa spustí vyvažovacia funkcia, ktorá vykoná potrebné prefarbenie alebo rotácie, aby sa zachovali vlastnosti červeno-čierneho stromu. Vyhľadávanie prebieha porovnaním hodnôt a pohybom doľava alebo doprava, kým sa nenájde požadovaná hodnota alebo sa nedosiahne „null“ uzol. Funkcia na odstránenie uzla tiež vyvažuje strom po odstránení.

Červeno-čierny strom používa farby na udržanie rovnováhy, čo zabezpečuje efektívnejšie operácie ako pri nevyvážených binárnych vyhľadávacích stromoch, čím umožňuje rýchlejšie vkladanie, vyhľadávanie a mazanie.

Zložitosť algoritmu:

Vahabov Oleksandr
pondelok 16:00 – 18:00

Search O(log n)

Insertion O(log n)

Deletion O(log n)

Hash Table with Double Hashing

Hašovacia tabuľka s dvojitým hašovaním je dátová štruktúra, kde sa prvky ukladajú pomocou dvoch hašovacích funkcií na výpočet indexu v poli. Každý prvok v tabuľke má stavovú značku, ktorá môže byť 0 (voľný), -1 (vymazaný). Implementácia začína vytvorením tabuľky určenej veľkosti a následne implementovaním vkladacej metódy. Táto metóda používa dve hašovacie funkcie, čím sa minimalizujú kolízie. Pri vkladaní je zahrnutý aj mechanizmus zmeny veľkosti tabuľky, ktorý sa aktivuje, keď počet prvkov dosiahne 70 % kapacity tabuľky. Na hľadanie prvku sa používajú rovnaké hašovacie funkcie. Funkcia vymazania využíva dvojité hashovanie na správne odstránenie prvku a aktualizáciu jeho stavu. Na testovanie funkčnosti tabuľky som pridal metódu na zobrazenie obsahu tabuľky, ktorá ukazuje aktuálne prvky a ich stavy.

Hašovacia tabuľka používa hašovacie funkcie na rýchly prístup k údajom. Dvojité hashovanie pomáha minimalizovať kolízie, zlepšuje výkon vkladania, vyhľadávania a mazania tým, že zabezpečuje rýchle načítanie voľných miest.

Zložitosť algoritmu:

Search O(1)

Insertion O(1)

Deletion O(1)

Hodnotenie výkonnosti

Na vyhodnotenie výkonnosti implementovaných dátových štruktúr (váhovo vyvážený strom, červeno-čierny strom a hašovacia tabuľka) bol vytvorený samostatný súbor s testovacími prípadmi napísanými v jazyku C. V tomto súbore sa meral čas vykonávania základných operácií - vkladanie, vyhľadávanie a mazanie prvkov.

Vahabov Oleksandr
pondelok 16:00 – 18:00
Meranie času robím pomocou knižnice <time.h>, funkciou clock(). Metóda merania bola založená na výpočte rozdielu medzi počiatočným a koncovým časom pomocou nasledujúceho vzorca:

time = end - start

Odtestujem moje programy na dvoch typoch údajov

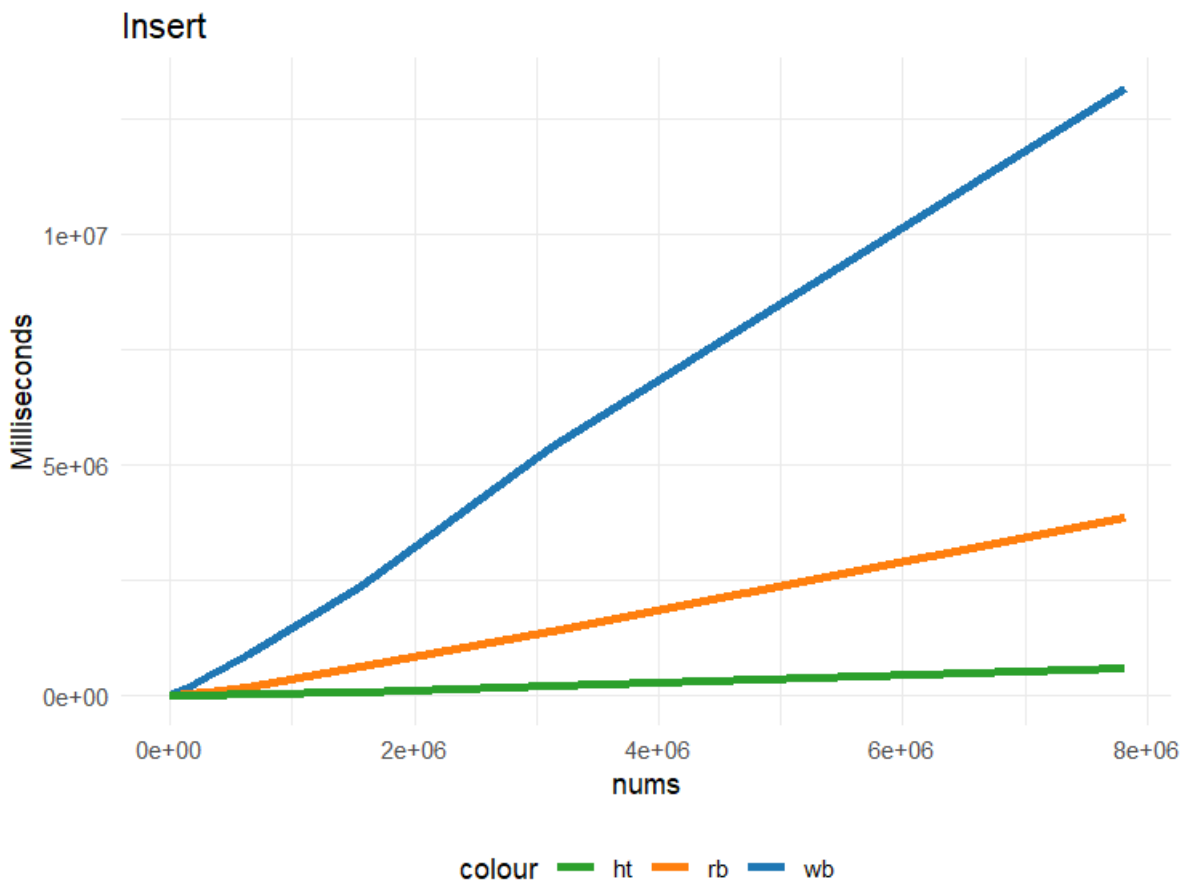
Sekvenčné hodnoty - rovnaké pole zoradené vzostupne.

Náhodné hodnoty - pole jedinečných celých čísel zamiešaných v náhodnom poradí.

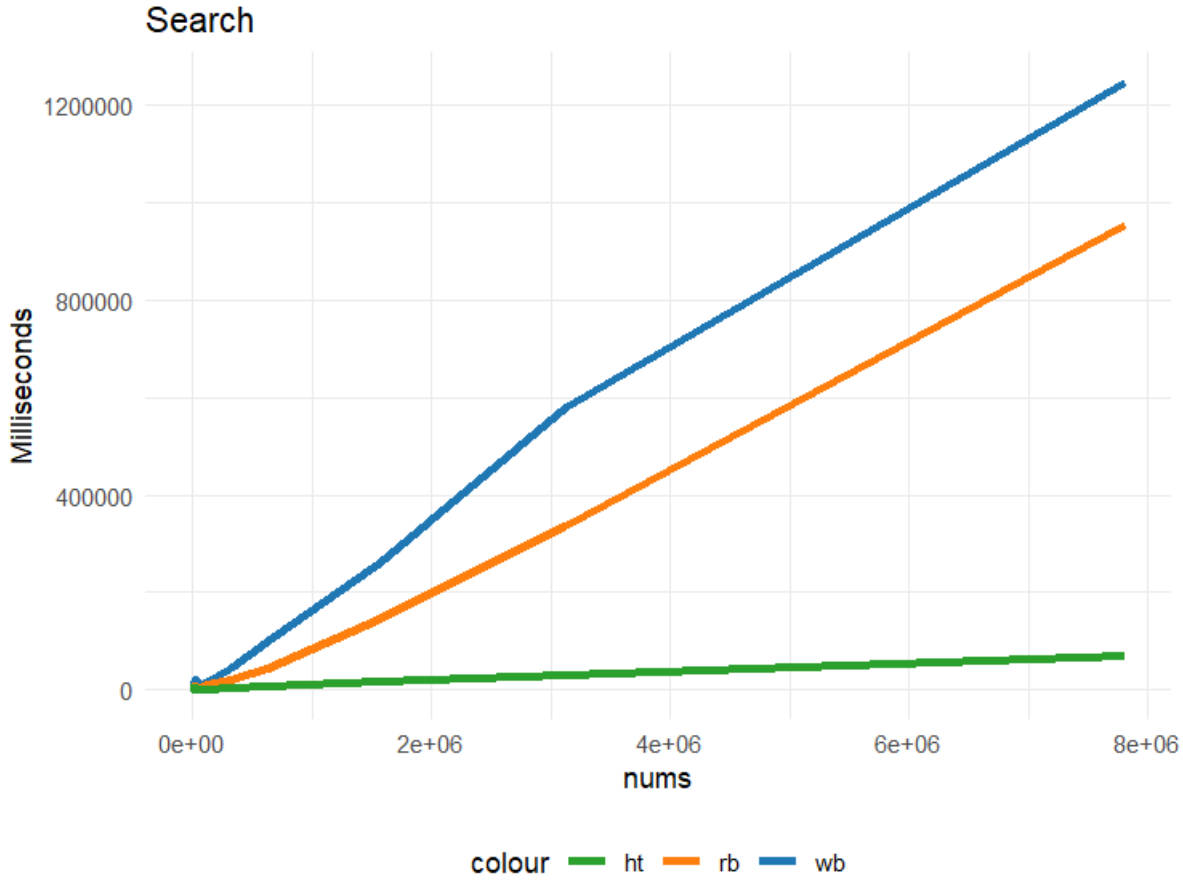
Každá dátová štruktúra bola testovaná s rovnakými vstupnými údajmi, aby sa zabezpečilo spravodlivé porovnanie výkonu.

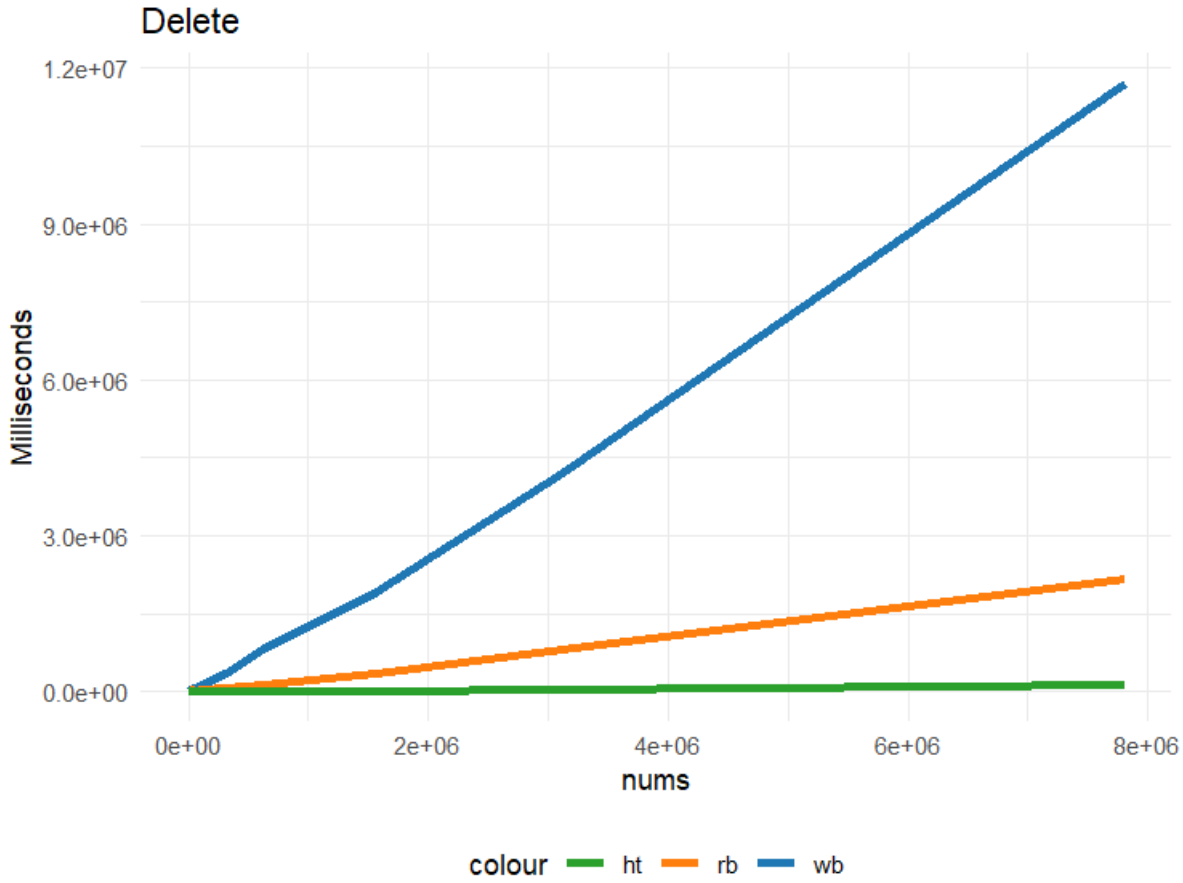
V nasledujúcom grafe môžete vidieť závislosť počtu vložených prvkov od času potrebného na ich vloženie. Z grafu najefektívnejšou štruktúrou je hašovacia tabuľka, potom červeno-čierny strom, najmenej efektívna je váhovo vyvážený strom.

os Y zobrazuje milisekundy, os X počet použitých prvkov



Vahabov Oleksandr
pondelok 16:00 – 18:00





Vahabov Oleksandr
pondelok 16:00 – 18:00

Ako vidíte v tabuľke, algoritmy na vzorke jedného milióna čísel priniesli nasledujúce výsledky (v milisekundách).

	Sekvenčné čísla			Náhodné čísla		
	Red-Black Tree	Weight-Balanced Tree	Hash table	Red-Black Tree	Weight-Balanced Tree	Hash table
Insert	262	1315	48	549	2132	12
Search	139	140	6	394	289	8
Delete	136	1071	15	256	2315	9

Výsledok:

Hašovacia tabuľka dosiahla najlepšie výsledky vďaka svojej jednoduchej architektúre. Na rozdiel od stromov, ktoré vyžadujú dodatočné operácie na udržanie rovnováhy, hašovacia tabuľka využíva priame adresovanie cez hašovacie funkcie, čo umožňuje veľmi rýchle operácie vkladania, vyhľadávania a mazania. Pri vkladaní 1 milióna prvkov trvala operácia v priemere 48 ms pre sekvenčné hodnoty. Vyhľadávanie trvalo 6 ms a mazanie 15 ms.

Červeno-čierny strom má lepšie výsledky než váhovo vyvážený strom, pričom pri vkladaní 1 milióna prvkov trval 262 ms pre sekvenčné hodnoty. Vyhľadávanie trvalo 139 ms a mazanie 136 ms pre náhodné hodnoty.

Váhovo vyvážený strom dosiahol najhoršie výsledky, pričom vkladanie trvalo 1315 ms pre sekvenčné hodnoty, vyhľadávanie 140 ms a mazanie 1071 ms

Na základe výsledkov testov sa ukázalo, že hašovacia tabuľka je najrýchlejšia a najefektívnejšia dátová štruktúra pre daný experiment, pričom červeno-čierny strom je efektívnejší ako váhovo vyvážený strom.