Empirična evalvacija urejenih slovarjev z ALGatorjem

JANI SUBAN, 89222015

MENTOR: ANDREJ BRODNIK

Kazalo

Urejen Slovar

Slovar je abstraktna podatkovna struktura

- Izbriši element iz slovarja
- Najdi element v slovarju
- Vstavi element v slovar

Urejen Slovar ohranja urejenost elementov

- Iteracija skozi slovar

Implementirane podatkovne strukture

Dvojiško iskalno drevo

AVL drevo

Rdeče – črno drevo

2-3 drevo

Zip Drevo

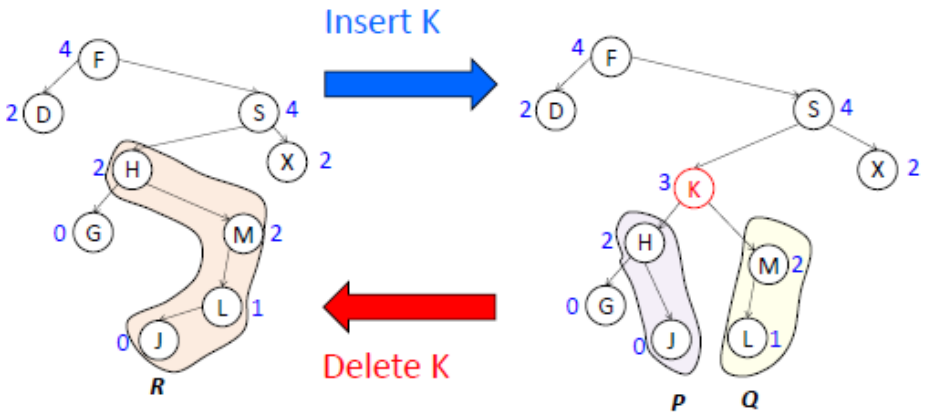
Preskočni seznam

	Vstavi	Izbriši	Najdi
Dvojiško iskalno drevo	O(n)	O(n)	O(n)
AVL drevo	O(log(n))	O(log(n))	O(log(n))
Rdeče – črno drevo	O(log(n))	O(log(n))	O(log(n))
2-3 drevo	O(log(n))	O(log(n))	O(log(n))
Zip Drevo	O(log(n))	O(log(n))	O(log(n))
Preskočni seznam	O(log(n))	O(log(n))	O(log(n))

Zip drevo

Naključna podatkovna struktura

Uravnoteženost je dosežena s pomočjo ranga



Testna metodologija

Za vsak test hranimo dva podatka

- Vektor števil operacij
- Vektor časa izvajanja operacij

Vedno moramo generirati podatkovno strukturo

- Vsak test mora vsebovati vsaj eno vstavljanje

$$N_i = (n_I, n_F, n_D),$$
$$N_i = (n_I, n_D) = (n_I, 0, n_D),$$
$$N_i = (n_I, n_F) = (n_I, n_F, 0),$$
$$N_i = (n_I) = (n_I, 0, 0),$$

$$\tau_i = (t_I, t_F, t_D)$$
$$\tau_i = (t_I, t_D) = (t_I, 0, t_D)$$
$$\tau_i = (t_I, t_F) = (t_I, t_F, 0)$$
$$\tau_i = (t_I) = (t_I, 0, 0)$$

Testiranje

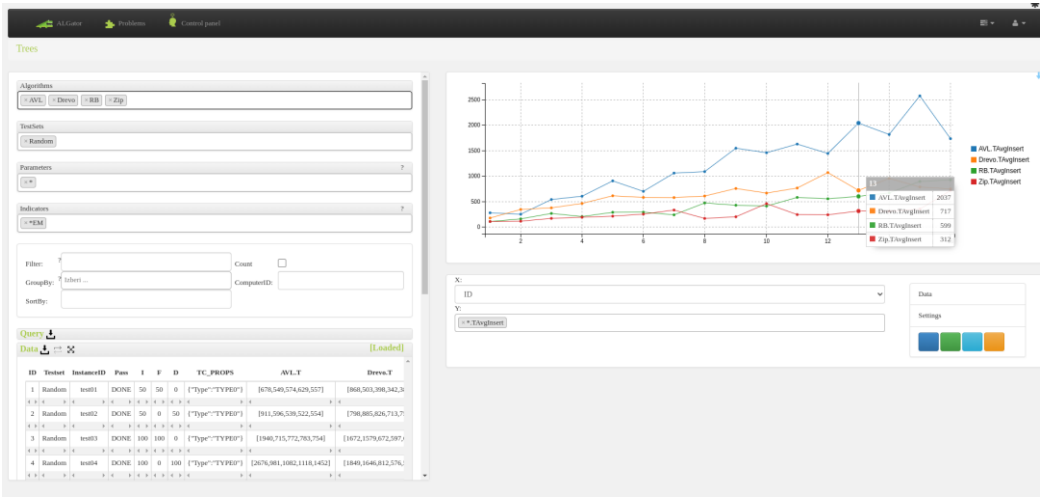
Dve vrsti testov:

1. Naključno zaporedje
   - Testiranje neoptimalnega vnašanja/iskanja/brisanja podatkov
2. Strogo naraščajoče zaporedje
   - Testiranje najslabšega možnega vnašanja/iskanja/brisanja podatkov

ALGator

Odprtokodni sistem za testiranje algoritmov in podatkovnih struktur

Testirane podatkovne strukture implementirane v Javi



Evalvacija

Sistem, na katerem so bile testirane podatkovne strukture:

- AMD Ryzen 7 2700
- 16 GB RAM
- Fedora 36, Linux kernel 6.2.14
- ALGator 0.985

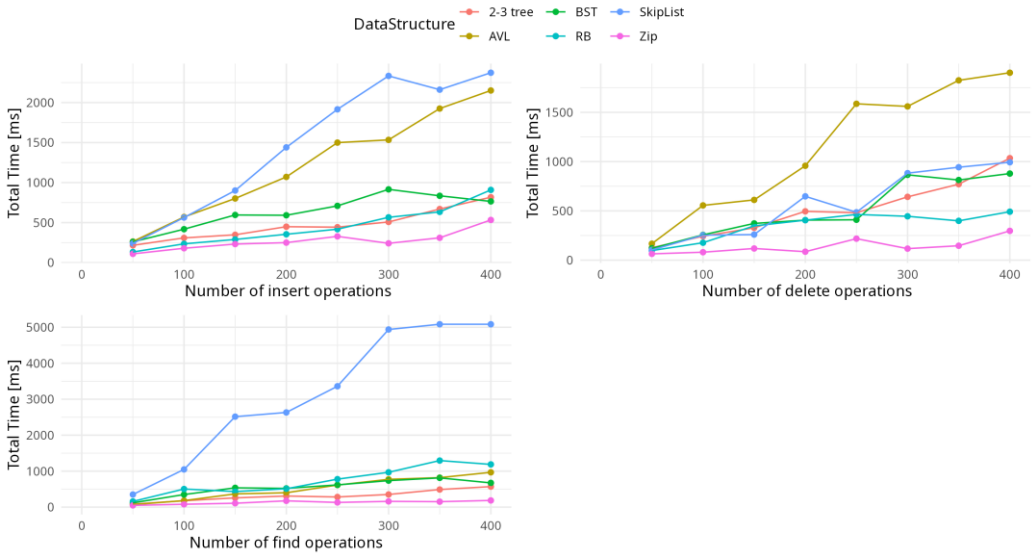
Vsak test je bil pognan 5-krat

Število elementov v podatkovni strukturi od 50 do 400

Naključni test

Implementirano s pomočjo knjižnice java.util.Random

Čas izvajanja vsakega algoritma sovpada s pričakovanim teoretičnim časom O(log(n))



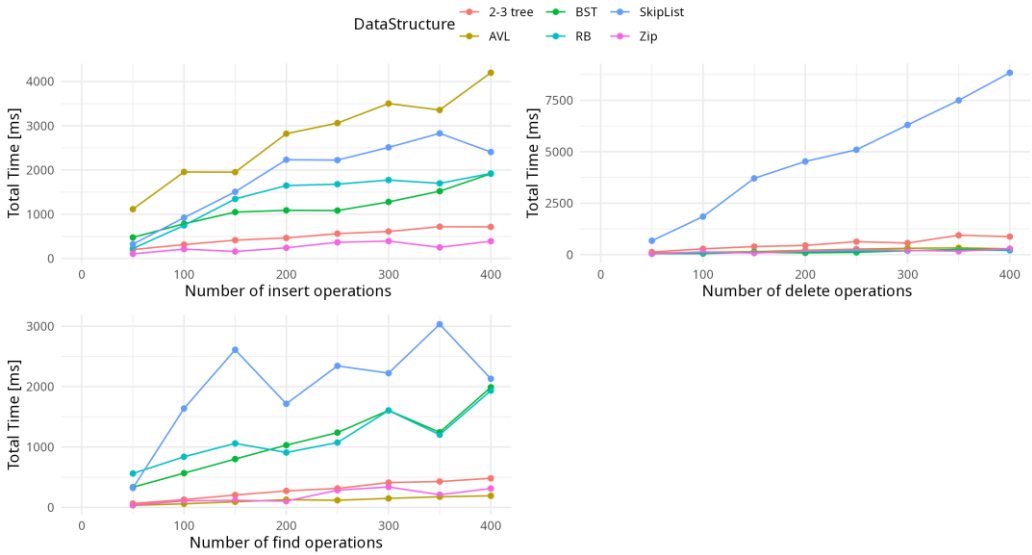
Strogo naraščajoči test

Implantiran s pomočjo števca, ki se poveča po vsaki operaciji

- Operacija izbris elementa je bila implementirana, da izbriše v obratnem vrstnem redu

Čas izvajanja vsakega algoritma sovpada s pričakovanim teoretičnim časom O(log(n))

- Dvojiško iskalno drevo ima čas izvajanja, ki sovpada s teoretičnim časom O(n)



Zaključek in prihodnja dela

Izmerjena časovna kompleksnost sovpada s teoretično časovno kompleknost

Izmeriti časovno kompleksnost tudi za večje število elementov

Posplošitev Zip drevesa na k-tiško Zip drevo

Vprašanja

Hvala za vašo pozornost