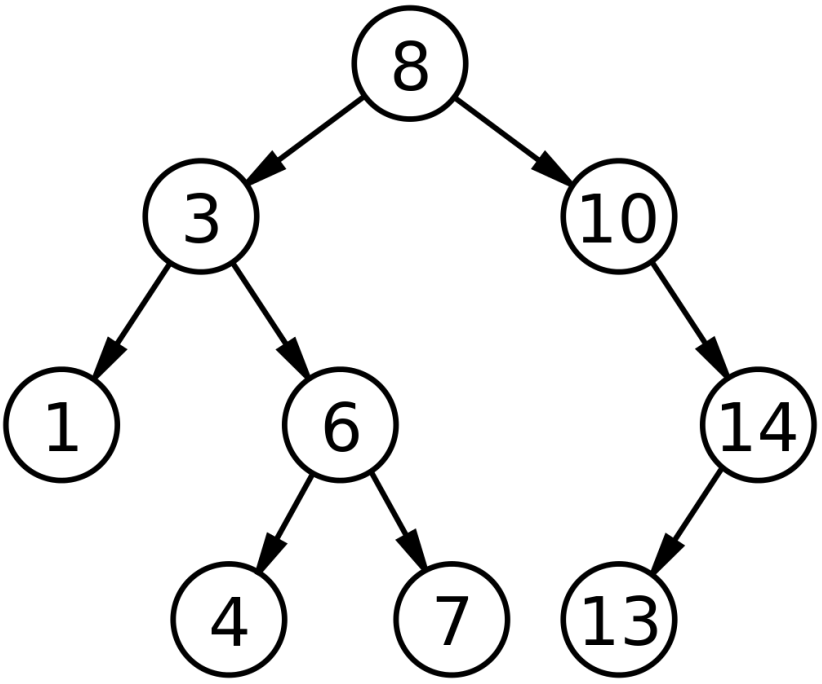
SPRAWOZDANIE
Eksperymenty na Drzewach Binarnych

autor: Daniel Drapała
kurs:Algorytmy i struktury danych

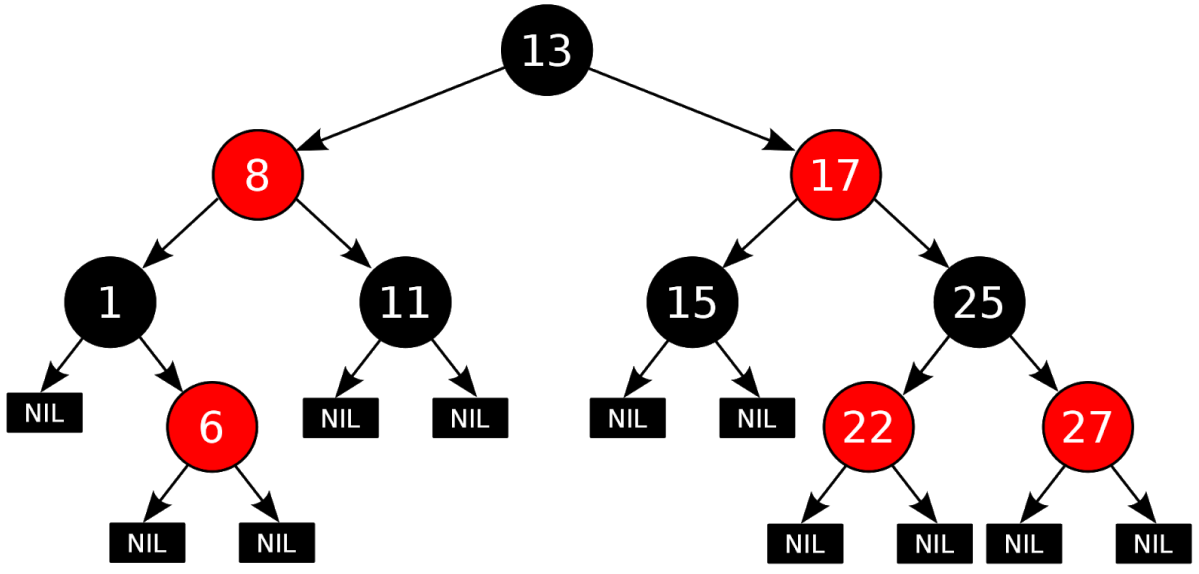
Przedstawienie struktur
Binary Search Tree (BST)-



8 jest korzeniem
1,4,7,13-liście
dziecko po lewej jest mniejsze od rodzica, a po prawej jest niemniejsze od rodzica

Jeżeli drzewo jest zrównoważone, to jego wysokość bliska jest logarytmowi dwójkowemu liczby węzłów h =log2(n), zatem dla drzewa o n węzłach optymistyczny koszt każdej z podstawowych operacji wynosi O(log n). Z drugiej strony drzewo skrajnie niezrównoważone ma wysokość porównywalną z liczbą węzłów (w skrajnym przypadku drzewa zdegenerowanego do listy wartości [1] te są równe: h=n), z tego powodu koszt pesymistyczny wzrasta do O (n).

Red Black Tree (RBT)



[1] źródło wikipedia.org

Jest to samoorganizujące się drzewo binarne, które ma dodatkowy parametr, kolor, albo czarny albo czerwony, a każdy z liści jest węzłem nil.
Zachodzą również podane właściwości:

-Każdy węzeł jest czerwony albo czarny.
-Korzeń jest czarny.
-Każdy liść jest czarny (Można traktować nil jako liść).
-Jeśli węzeł jest czerwony, to jego synowie muszą być czarni.
-Każda ścieżka z ustalonego węzła do liścia liczy tyle samo czarnych węzłów.

Wymagania te gwarantują, że najdłuższa ścieżka od korzenia do liścia będzie co najwyżej dwukrotnie dłuższa, niż najkrótsza.

Splay Tree

W porównaniu do innych drzew BST, drzewa splay zmieniają swoją strukturę również podczas wyszukiwana kluczy (a nie tylko dodawania lub usuwania), przesuwając znaleziony węzeł w kierunku korzenia, dzięki temu często wyszukiwane węzły stają się szybsze do znalezienia. Z tego powodu drzewa splay bywają wykorzystywane w systemach typu cache. Drzewa splay nie są samorównoważące, ponieważ ich wysokość nie jest ograniczona przez O(log n)– można np. tak wykonać operacje, że drzewo zdegeneruje się do listy.
Podstawowe operacje na drzewie splay, tj. wyszukiwanie elementu oraz wstawianie i usuwanie, są wykonywane w zamortyzowanym czasie O(log n) gdzie n jest liczbą elementów w drzewie. Oznacza to, że dla dowolnego ciągu m operacji na drzewie splay, łączny koszt wykonania tych operacji jest rzędu O(m log n).
SPLAY(T,x) Operacja Splay jest procedurą kluczową dla działania drzewa typu splay. Polega ona na wykonaniu sekwencji kroków, z których każdy przybliża element x do korzenia. Każdy krok polega na wykonaniu jednej lub dwóch rotacji względem krawędzi wchodzących w skład początkowej ścieżki od x do korzenia.

Eksperymenty

Korzystając z przykładów podanych na stronie będę testował wyżej wymienione struktury za pomocą dodania wszystkich ciągów z pliku, sprawdzenia wszystkich ciągów z pliku i potem usunięcia wszystkich ciągów z pliku (dla aspell_wordlist.txt, który jest przykładem posortowanym przygotowałem generator pseudolosowy i ten plik testował będę na dwa sposoby: po kolei i losowe ciągi dodawane wyszukiwane i usuwane.
Legenda rozpoznawania typu danych po nazwie plików:
aspell_wordlist.txt --unikatowe ciągi posortowane
lotr.txt -- takie ciągi , gdzie możliwe są powtórzenia
aspell_wordlist random -- przemieszane ciągi unikatowe

Wszystkie dane przerzuciłem ze zrzutów podanych poniżej do tabelki:

typ drzewa	typ danych	Typ operacji	Porównania	Modyfikacje	czas
BST	aspell_wordlist.txt 160 613 węzłów	Insert	2094490649	0	46647 ms (46s)!
		Search..	2998673381	0	11166.8 ms
		Delete	1290515	6611008	15 ms
RBT	aspell_wordlist.txt 160 613 węzłów	Insert	4730256	401200	110 ms
		Search..	4172149	0	32ms
		Delete	153724	153724	51 ms
Splay	aspell_wordlist.txt 160 613 węzłów	Insert	903845	2229699	44ms
		Search..	710155	1648626	37ms
		Delete	667000	1679162	36ms
typ drzewa	typ danych	Typ operacji	Porównania	Modyfikacje	czas
BST	aspell random 160 613 węzłów	Insert	598143618	0	42146 ms(42s)!
		Search..	5849715	0	137 ms
		Delete	4777107	456585	131 ms
RBT	aspell random 160 613 węzłów	Insert	3546504	2333660	189 ms
		Search..	5124717	0	159 ms
		Delete	4415357	141420	195 ms
Splay	aspell random 160 613 węzłów	Insert	3160613	9000153	279 ms
		Search..	3007488	8540625	267 ms
		Delete	4616559	13528425	284 ms
typ drzewa	typ danych	Typ operacji	Porównania	Modyfikacje	czas
BST	lotr.txt 190150 węzłów	Insert	157744021	0	25440ms
		Search..	7606718	0	227 ms
		Delete	4539564	720318	172 ms
RBT	lotr.txt 190150 węzłów	Insert	4332261	439892	628 ms
		Search..	3683424	0	151 ms
		Delete	3327892	193070	521 ms
Splay	lotr.txt 190150 węzłów	Insert	3164960	8924433	216 ms
		Search..	2042230	5556240	299 ms
		Delete	4089376	11887804	843 ms

Obserwacje:
Drzewo BTS- słabo radzi sobie z danymi posortowanymi (Złożoność jest wtedy przybliżona do złożoności zwykłej listy O(n))
Widzimy, że ten sam plik wstawiany losowo przyspiesza operacje wstawiania wszystkich węzłów o 4 sekundy.
Bts:
- posortowane dane zwiększają złożoność (wypełniane jest jedna strona drzewa wysokość jest równa ilości węzłów)
- nie wykonuje modyfikacji na węzłach, ponieważ nie jest drzewem samoorganizującym się. Modyfikacje wykonywane są jedynie podczas usuwania, kiedy usuwany węzeł zamienia się miejscem z następnikiem, żeby nie zgubić swoich dzieci

Drzewo RBT
Drzewo samoorganizujące się utrzymuje swoją niską złożoność poprzez odbudowywanie się po wstawianiu, usuwaniu (implementacja tego typu drzewa jest bardziej skomplikowana)

- dla testowanych plików wypada najlepiej (oprócz posortowanych ciągów)
- przy usuwaniu i wstawianiu jest nieco wolniejsze od drzewa Splay, ale modyfikuje znacznie mniej węzłów podczas wykonywania tychże operacji
- najlepsze podczas wyszukiwania (nie modyfikuje drzewa podczas operacji search() jak splayTree)

Drzewo Splay
drzewo samodostosowujące się cechuje się swoją niską złożonością i ciągłym przebudowywaniem swojego szkieletu za pomocą funkcje splay()

- z posortowanymi ciągami poradziło sobie najlepiej
- bardzo dobrze nadaje się do wyszukiwania podobnych danych parę razy pod rząd (drzewo (a raczej funkcja splay(x)) szukaną wartość podmienia za korzeń) więc szukanie wartości przybliżonej do wcześniejszej będzie znacznie szybsze
- czasowo lepsze od RBT i BST(znacznie więcej), lecz każda z operacji (nawet szukanie węzła) wykonuje nieporównywalnie dużo modyfikacji drzewa

Podstawowe operacje na drzewie splay, tj. wyszukiwanie elementu oraz wstawianie i usuwanie, są wykonywane w zamortyzowanym czasie O(log n),gdzie n jest liczbą elementów w drzewie. Oznacza to, że dla dowolnego ciągu m operacji na drzewie splay, łączny koszt wykonania tych operacji jest rzędu O(m log n).

Zrzuty ekranu przedstawiające wyniki programu dla 3 różnych plików dla drzewa BST:

```
BST for lotr.txt
Inserting 190150 nodes time: 25440.47467 ms
counterMOD number: 0
counterIF number: 157744021
Searching for 190150 nodes time: 227.420703 ms
counterMOD number: 0
counterIF number: 7606718
Deleting 190150 nodes time: 172.158774 ms
counterMOD number: 720318
counterIF number: 4539564
```

```
BST for aspell_wordlist.txt
Inserting 160613 nodes time: 46647.914266 ms
counterMOD number: 0
counterIF number: 2094490649
Searching for 160613 nodes time: 11166.878341 ms
counterMOD number: 0
counterIF number: 2998673381
Deleting 160613 nodes time: 15.806312 ms
counterMOD number: 611008
counterIF number: 1290515
```

```
BST for aspel_wordlist random
Inserting 160613 nodes time: 42146.73254 ms
counterMOD number: 0
counterIF number: 598143618
Searching for 160613 nodes time: 137.689187 ms
counterMOD number: 0
counterIF number: 5849715
Deleting 160613 nodes time: 131.210674 ms
counterMOD number: 456585
counterIF number: 4777107
```

Zrzuty ekranu przedstawiające wyniki programu dla 3 różnych plików dla drzewa RBT:

```
RBT for aspell_wordlist.txt
Inserting 160613 nodes time: 110.031485 ms
counterMOD number: 401200
counterIF number: 4730256
Searching for 160613 nodes time: 32.870153 ms
counterMOD number: 0
counterIF number: 4172149
Deleting 160613 nodes time: 51.153694 ms
counterMOD number: 153724
counterIF number: 2713081
```

```
RBT for aspel_wordlist random
Inserting 160613 nodes time: 189.770741 ms
counterMOD number: 233660
counterIF number: 3546504
Searching for 160613 nodes time: 159.471115 ms
counterMOD number: 0
counterIF number: 5124717
Deleting 160613 nodes time: 195.393075 ms
counterMOD number: 141420
counterIF number: 4415357
```

```
RBT for lotr.txt
Inserting 190150 nodes time: 628.208786 ms
counterMOD number: 439892
counterIF number: 4332261
Searching for 190150 nodes time: 151.890377 ms
counterMOD number: 0
counterIF number: 3683424
Deleting 190150 nodes time: 521.918602 ms
counterMOD number: 193070
counterIF number: 3327892
```

Zrzuty ekranu przedstawiające wyniki programu dla 3 różnych plików dla drzewa Splay:

```
Splay for lotr.txt
Inserting 190150 nodes time: 216.466166 ms
counterMOD number: 8924433
counterIF number: 3164960
Searching for 190150 nodes time: 299.919701 ms
counterMOD number: 5556240
counterIF number: 2042230
Deleting 190150 nodes time: 843.666026 ms
counterMOD number: 11887804
counterIF number: 4089376
```

```
Splay for aspel_wordlist random
Inserting 160613 nodes time: 279.424033 ms
counterMOD number: 9000153
counterIF number: 3160663
Searching for 160613 nodes time: 267.817265 ms
counterMOD number: 8540625
counterIF number: 3007488
Deleting 160613 nodes time: 284.385603 ms
counterMOD number: 13528425
counterIF number: 4616559
```

```
Splay for aspell_wordlist.txt
Inserting 160613 nodes time: 44.459571 ms
counterMOD number: 2229699
counterIF number: 903845
Searching for 160613 nodes time: 37.137963 ms
counterMOD number: 1648626
counterIF number: 710155
Deleting 160613 nodes time: 36.866033 ms
counterMOD number: 1679162
counterIF number: 667000
```