

Sprawozdanie

Algorytmy I Struktury Danych

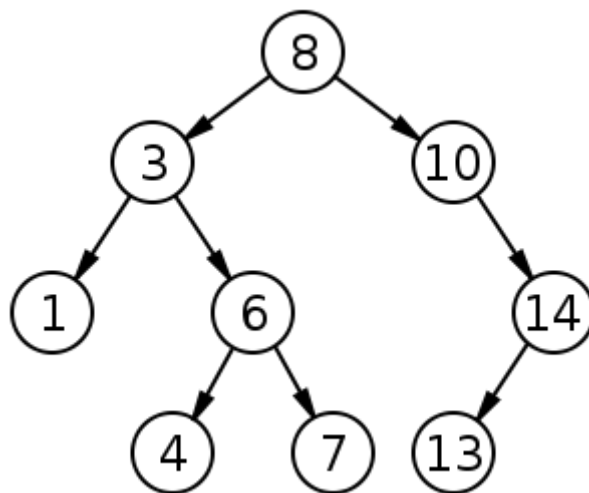
Lista 4

Krzysztof Strzała 236503

1. Wprowadzenie

1. Drzewo BST

https://en.wikipedia.org/wiki/Binary_search_tree



Drzewo wyszukiwań binarnych w którym każdy węzeł cechuje

- jeżeli jest prawym synem to ma większą wartość od ojca.
- jeżeli jest lewym synem to jego wartość jest mniejsza od ojca.

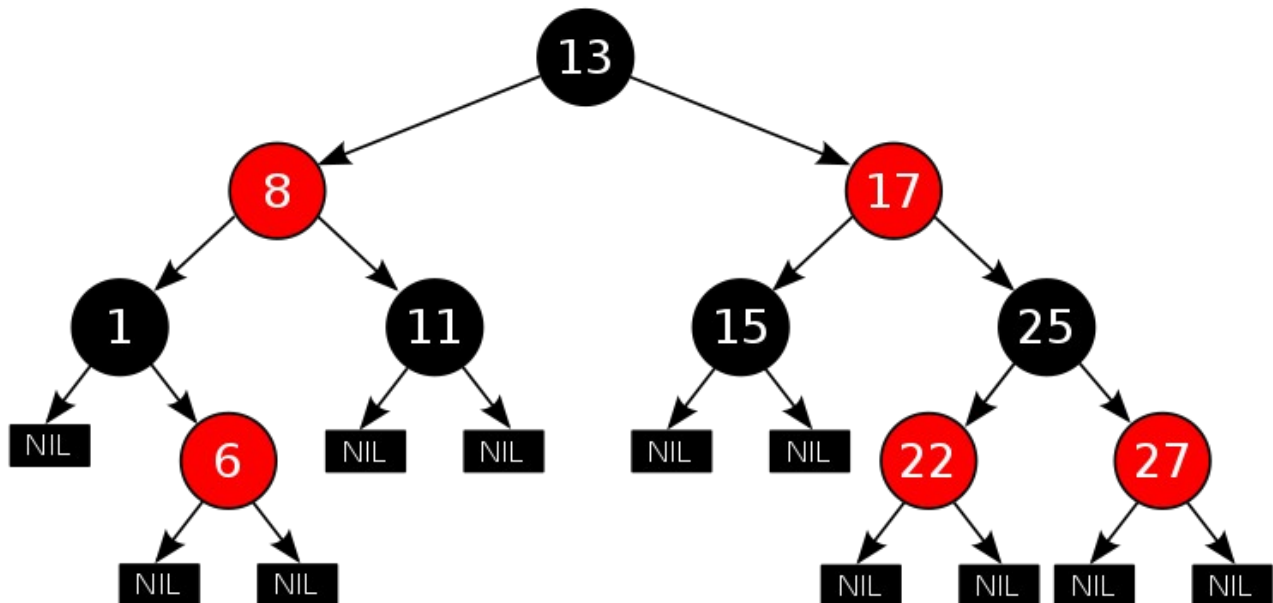
Dla zrównoważonego drzewa złożoność operacji insert , delete jak i search możemy określić rzędu $O(\log n)$. W pesymistycznym przypadku złożoność jest rzędu $O(n)$, gdy węzły drzewa ułożone są w lini.

2. Drzewo RB

https://en.wikipedia.org/wiki/Red%E2%80%93black_tree

Drzewo czerwono-czarne które różni się tylko tym że kolorujemy węzły na kolor czarny i czerwony , a jego warunkiem jest brak sąsiadujących ze sobą 2 czerwonych wierzchołków. Niesie to za sobą zwiększony czas o “naprawy drzewa” w przypadku operacji insert i delete. Porównując ze zrównoważonym drzewem BST drzewo będzie miało większą złożoność czasową , jednakże

porównując to z drzewem z drzewem BST którego wierzchołki układają się w linię, uzyskamy mniejszą złożoność.



3. Drzewo Splay

https://en.wikipedia.org/wiki/Splay_tree

Splay to rozszerzenie drzewa BST operacje splay wykonywaną przy operacji search i delete. Splay polega na umieszczeniu szukanego / usuwanego / dodawanego klucza w korzeniu przy zachowaniu porządku drzewa BST.

Drzewa tego typu przewyższają drzewa RB w przypadku zwiększonej ilości wyszukiwań podobnych kluczy nad insertami.

Poniższy przykład obrazuje jak wygląda złożoność w przypadku wyszukiwania podobnych kluczy wiele razy.

```
SPLAY DATA WITH 1000000 NUMBERS
insert time =2278
find time =481
removal time = 1254

BST TREE DATA WITH 1000000 NUMBERS
insert time =780
find time =7617
removal time =1002

REDBLACK TREE DATA WITH 1000000 NUMBERS
insert time =939
find time =821
removal time =582
```

2.Testy

Testy z plików podanych na stronie przeprowadzono w następujący sposób:

- stworzono tablice złożoną z wszystkich słów w danym liku tekstowym.
- losowe elementy z tablicy zostały dodawane do drzew , tak długo aż tablica była pusta.
- wykonano operacje search w losowej kolejności o każdy element z pliku
- usunięto w losowej kolejności każdy element

```
Splay tree data: Ins:125976 Del:0 Src:0
Capatity: 125976
If counter: 9398945
Modified counter: 5460514
Time complexity of insert: 1288ms
Time complexity of search: 1113ms
Time complexity of delete: 1083ms

BST tree data: Ins:125976 Del:0 Src:0
Capatity: 125976
If counter: 6427729
Modified counter: 125975
Time complexity of insert: 1080ms
Time complexity of search: 1084ms
Time complexity of delete: 1043ms

RB tree data: Ins:125976 Del:0 Src:0
Capatity: 125976
If counter: 5728680
Modified counter: 806043
Time complexity of insert: 1143ms
Time complexity of search: 1040ms
Time complexity of delete: 1124ms

Process finished with exit code 0
```

Dane dla unikatowych ciągów posortowanych aspell_wordlist.txt

```
Splay tree data: Ins:1152399 Del:0 Src:0
Capatity: 60107
If counter: 26502327
Modified counter: 15779610
Time complexity of insert: 308ms
Time complexity of search: 264ms
Time complexity of delete: 288ms

BST tree data: Ins:1152399 Del:0 Src:0
Capatity: 60107
If counter: 28682885
Modified counter: 60106
Time complexity of insert: 243ms
Time complexity of search: 244ms
Time complexity of delete: 307ms

RB tree data: Ins:1152399 Del:0 Src:0
Capatity: 60107
If counter: 30083605
Modified counter: 436560
Time complexity of insert: 293ms
Time complexity of search: 291ms
Time complexity of delete: 433ms

Process finished with exit code 0
```

Dane dla pliku KJB.txt

```

Splay tree data: Ins:341099 Del:0 Src:0
Capacity: 47398
If counter: 13013361
Modified counter: 7354116
Time complexity of insert: 7728ms
Time complexity of search: 7223ms
Time complexity of delete: 189ms

BST tree data: Ins:341099 Del:0 Src:0
Capacity: 47398
If counter: 9505563
Modified counter: 47397
Time complexity of insert: 7341ms
Time complexity of search: 7058ms
Time complexity of delete: 166ms

RB tree data: Ins:341099 Del:0 Src:0
Capacity: 47398
If counter: 9065368
Modified counter: 301760
Time complexity of insert: 7218ms
Time complexity of search: 7128ms
Time complexity of delete: 243ms

Process finished with exit code 0

```

Dane dla książki Anna Karenina (zamiast Łotr).

Kolejne przykłady przeprowadzono na zbiorach liczb :

oznaczenia – I – ilość insertów , S – ilość wyszukiwań kluczy , D – ilość usunąć kluczy

```

Splay tree
Insert: 618
Search :465
Remove :529

Binary search tree
Insert: 356
Search :4505
Remove :475

RedBlack tree
Insert: 401
Search :382
Remove :329

```

```

Splay tree
Insert: 15700
Search :45
Remove :1837

Binary search tree
Insert: 8715
Search :11118
Remove :1109

RedBlack tree
Insert: 9207
Search :373
Remove :695

```

```

Splay tree
Insert: 19
Search :5
Remove :9

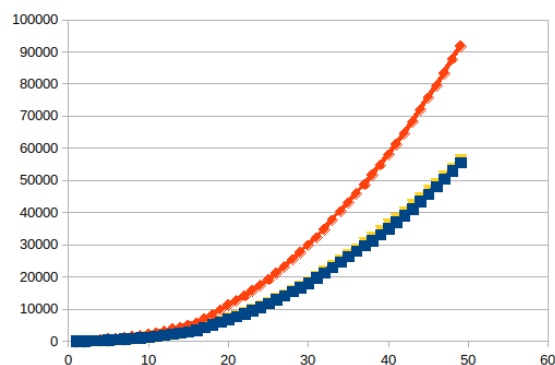
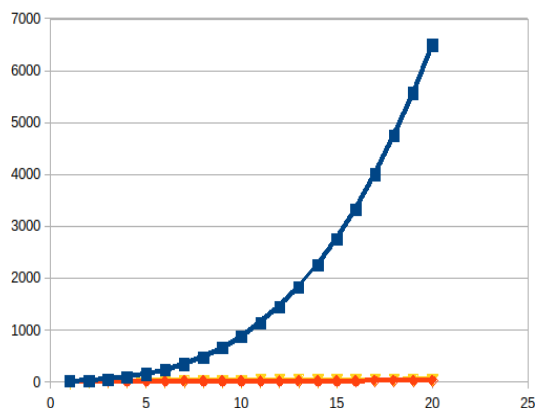
Binary search tree
Insert: 2320
Search :350
Remove :7

RedBlack tree
Insert: 20
Search :5
Remove :28

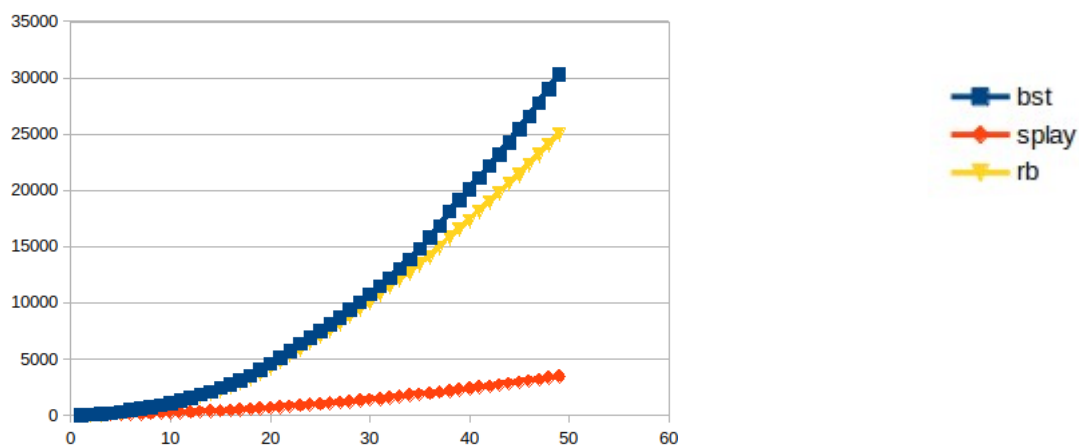
```

1. Średni czas dla przypadku gdy $10I = 10D = S$
2. Średni czas dla przypadku gdy $I = 10D = 10S$
3. Średni czas dla przypadku gdy $I = R = 10S$, lecz inserty są kolejnymi liczbami całkowitymi (1,2,3,4...)

1. Porównanie struktur dla dodawania wartości o liniowych wartościach klucza (1,2...n):
2. Porównanie struktur dla dodawania losowych wartości.



3. Porównanie struktur dla 50 razy częściej występującej operacji search.



**** Porównanie z innymi strukturami:

```
Splay tree
Insert: 4
Search :6
Remove :6

Binary search tree
Insert: 120
Search :111
Remove :113

RedBlack tree
Insert: 19
Search :3
Remove :6

Treap tree tree
Insert: 6
Search :4
Remove :3

Trie tree tree
Insert: 24
Search :23
Remove :0
```

```
Splay tree
Insert: 19
Search :4
Remove :6

Binary search tree
Insert: 5
Search :3
Remove :5

RedBlack tree
Insert: 4
Search :7
Remove :27

Treap tree tree
Insert: 25
Search :9
Remove :28

Trie tree tree
Insert: 76
Search :13
Remove :0
```

```
Splay tree
Insert: 52
Search :11
Remove :10

Binary search tree
Insert: 56
Search :16
Remove :11

RedBlack tree
Insert: 48
Search :5
Remove :7

Treap tree tree
Insert: 71
Search :15
Remove :15

Trie tree tree
Insert: 148
Search :17
Remove :0
```

```
Splay tree
Insert: 56
Search :3
Remove :63

Binary search tree
Insert: 37
Search :5
Remove :53

RedBlack tree
Insert: 37
Search :5
Remove :45

Treap tree tree
Insert: 50
Search :5
Remove :48

Trie tree tree
Insert: 115
Search :13
Remove :0
```

```

Splay tree
Insert: 997
Search :937
Remove :12

Binary search tree
Insert: 405
Search :487
Remove :9

RedBlack tree
Insert: 548
Search :401
Remove :6

Treap tree tree
Insert: 666
Search :783
Remove :7

Trie tree tree
Insert: 1275
Search :857
Remove :0

```

```

Splay tree
Insert: 1053
Search :34
Remove :12

Binary search tree
Insert: 390
Search :535
Remove :10

RedBlack tree
Insert: 537
Search :439
Remove :8

Treap tree tree
Insert: 818
Search :1048
Remove :7

Trie tree tree
Insert: 1667
Search :1008
Remove :0

```

```

Splay tree
Insert: 8
Search :261
Remove :4

Binary search tree
Insert: 26
Search :132
Remove :4

RedBlack tree
Insert: 17
Search :149
Remove :4

Treap tree tree
Insert: 14
Search :153
Remove :7

Trie tree tree
Insert: 57
Search :462
Remove :0

```

```

Splay tree
Insert: 17557
Search :18036
Remove :18262

Binary search tree
Insert: 8772
Search :9607
Remove :8747

RedBlack tree
Insert: 7528
Search :6343
Remove :7517

Treap tree tree
Insert: 10272
Search :9722
Remove :9028

Trie tree tree
Insert: 24797
Search :12848
Remove :0

```

1. Dane dodawane liniowo
2. Mała ilość danych
3. 10 razy więcej insertów niż pozostałych operacji.
4. 10 razy więcej insertów I usunięć niż pozostałych operacji.
5. 100 razy więcej insertów I wyszukiwań losowych wartości z tablic niż usunięć.
6. 100 razy więcej insertów I wyszukiwań podobnych wartości z tablic niż usunięć.
7. 100 razy więcej wyszukiwań niż innych operacji.
8. Duża ilość danych