Zadanie 7

Mateusz Wojtyna

1. Wstęp

Należało zaimplementować operacje `insert()`, `search()` i `delete()` dla drzewa splay. Następnie należało porównać wydajność tych operacji dla wszystkich omawianych typów drzew (BST, AVL, Red-Black, Splay).

2. Opis

Do implementacji BST wykorzystałem mój kod (lekko zmieniony) z zadania 6. *Próbowałem* wykorzystać gotowe implementacje drzew Red-Black i AVL z GitHuba ale nie wytrzymywały one takiej ilości danych i występował segmentation fault. Więc niestety byłem zmuszony poprosić o pomoc AI. Sprawdziłem jego implementację za pomocą testów, więc drzewa te *powinny* być poprawnie zaprogramowane.

3. Wyniki ($N = 10^6$)

```
g++ main.cpp -O1 && ./a.out
-------------------------------------------------------------------------------
Struktura      Insert            Find              Remove
-------------------------------------------------------------------------------
BST            0.283935       s | 0.293007       s | 0.275677       s
Splay          0.567029       s | 0.513584       s | 0.46978        s
AVL            0.397061       s | 0.287013       s | 0.403725       s
Red-Black      0.367322       s | 0.319483       s | 0.314633       s

g++ main.cpp -O2 && ./a.out
-------------------------------------------------------------------------------
Struktura      Insert            Find              Remove
-------------------------------------------------------------------------------
BST            0.28384        s | 3e-08          s | 0.265474       s
Splay          0.568748       s | 0.530968       s | 0.470542       s
AVL            0.383848       s | 3e-08          s | 0.404447       s
Red-Black      0.357859       s | 3e-08          s | 0.260783       s

g++ main.cpp -O3 && ./a.out
-------------------------------------------------------------------------------
Struktura      Insert            Find              Remove
-------------------------------------------------------------------------------
BST            0.293159       s | 3e-08          s | 0.289843       s
Splay          0.561522       s | 0.461544       s | 0.423451       s
AVL            0.356616       s | 2e-08          s | 0.382673       s
Red-Black      0.308063       s | 3e-08          s | 0.153408       s
```

4. Podsumowanie

- Drzewa Red-Black oraz AVL okazały się być najwydajniejsze.
- Dla losowych danych drzewa splay nie są dobrym wyborem ze względu na wykonywanie rotacji przy każdej operacji, jednak idealne są do sytuacji, gdy często wykorzystujemy pewne elementy, na przykład w pamięci cache.
- Należy zauważyć, że drzewo BST wydaje się mieć dobrą wydajność, jednakże dla posortowanych danych zdegenerowałoby się do postaci listy wiązanej.
- Co ciekawe, można zauważyć, że dla opcji optymalizacji `-O2`, `-O3` czas wykonywania procedury `find()` był rzędu jedynie 10^{-8} sekundy dla wszystkich drzew oprócz splay.