Zadanie 8

Mateusz Wojtyna

1. Wstęp

Należało zaimplementować operacje `insert()`, `search()` i `delete()` dla tablicy haszującej wykorzystującej listy i haszowanie otwarte. Następnie należało porównać wydajność tych operacji z BST, AVL, Red-Black, Splay.

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

2.1. Tablica haszująca

```
g++ hash_table.cpp -O1 && ./a.out
-------------------------------------------------------------------------------
Struktura           Insert            Find              Remove
-------------------------------------------------------------------------------
Łańcuchowanie       0.0653708      s | 0.0121954      s | 0.0922634      s
Adresowanie otwarte 0.00247679     s | 0.00190633     s | 0.00227639     s

g++ hash_table.cpp -O2 && ./a.out
-------------------------------------------------------------------------------
Struktura           Insert            Find              Remove
-------------------------------------------------------------------------------
Łańcuchowanie       0.0586574      s | 3e-08          s | 0.0979503      s
Adresowanie otwarte 0.00300628     s | 3e-08          s | 0.00211869     s

g++ hash_table.cpp -O3 && ./a.out
-------------------------------------------------------------------------------
Struktura           Insert            Find              Remove
-------------------------------------------------------------------------------
Łańcuchowanie       0.0624238      s | 3e-08          s | 0.100663       s
Adresowanie otwarte 0.00280351     s | 2e-08          s | 0.00260889     s
```

2.2. Inne struktury

```
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
```

3. Podsumowanie

- Adresowanie otwarte okazało się być szybsze pod każdym względem od łańcuchowania.
- Tablica haszująca jest lepszym rozwiązaniem problemu słownikowego od struktur drzewiastych.