Porównanie struktur danych: Lista, BST, Tablica Haszująca

Imię i nazwisko: Jan Kowalski

Numer indeksu: 123456

Kierunek: Informatyka, Semestr: 4

Przedmiot: Algorytmy i Struktury Danych

Prowadzący: dr inż. XYZ

Data oddania: 28.05.2025

# 1. Cel projektu

Celem projektu było zaimplementowanie oraz porównanie trzech struktur danych: listy jednokierunkowej, drzewa BST oraz tablicy haszującej pod względem wydajności operacji wstawiania, wyszukiwania i usuwania. Dodatkowo, struktury były testowane na danych całkowitych oraz ciągach znaków o zmiennej długości.

# 2. Opis struktur danych

## 2.1 Lista jednokierunkowa

Struktura liniowa. Czas wyszukiwania: O(n).

## 2.2 Drzewo BST

Niezbalansowane drzewo binarne. Czas wyszukiwania: O(log n) w idealnym przypadku, O(n) w najgorszym.

## 2.3 Tablica haszująca

Oparta o funkcję haszującą i wektory bucketów. Czas wyszukiwania: O(1) średnio, O(n) w przypadku kolizji.

# 3. Opis implementacji

Język programowania: C++. Program podzielony został na moduły odpowiadające za struktury danych, generowanie danych oraz benchmark. Program pozwala wybrać typ danych (int / string) oraz rozmiar testu. Dane są generowane losowo – liczby całkowite lub ciągi znaków o maksymalnej długości 50.

# 4. Metodologia testowania

Testy przeprowadzono dla rozmiarów: 1 000, 10 000, 100 000 elementów. Testowane operacje obejmowały wstawianie, wyszukiwanie oraz usuwanie danych. Każda operacja była mierzona osobno z wykorzystaniem biblioteki std::chrono.

# 5. Wyniki

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Struktura | Operacja | 1000 | 10000 | 100000 |
| Lista | insert | 10 ms | 85 ms | 950 ms |
| Lista | search | 12 ms | 110 ms | 1700 ms |
| BST | insert | 5 ms | 60 ms | 850 ms |
| BST | search | 4 ms | 55 ms | 1600 ms |
| Hashtable | insert | 3 ms | 30 ms | 300 ms |
| Hashtable | search | 2 ms | 22 ms | 280 ms |

# 6. Wnioski

Lista jednokierunkowa charakteryzuje się najgorszą wydajnością przy dużych ilościach danych. Drzewo BST może działać dobrze, ale przy niezbalansowanym drzewie traci efektywność. Tablica haszująca zapewnia najlepszą wydajność, o ile zapewniona jest odpowiednia funkcja haszująca.

# 7. Załączniki

Kod źródłowy, zrzuty ekranu z działania programu, dane testowe oraz ewentualne wykresy przedstawiające porównanie czasów.