Laboratoria Algorytmów i Struktur Danych

Projekt 2

Drzewa przeszukiwań binarnych BST i drzewa samobalansujące

Wojciech Niedziela 160363

Jan Chojnacki XXXXXX

Godzina: **11:45** Dzień: **Piątek** Grupa: **14**

Nazwa kierunku: **Informatyka**

POLITECHNIKA POZNAŃSKA

1 | Wstęp

Celem tego sprawozdania jest zaprezentowanie algorytmów tworzenia drzew BST i AVL, a także analiza czasu tworzenia struktury, wyszukiwania minimum i maksimum oraz wypisywania in-order.

1.1 | BST

Drzewo BST (Binary Search Tree) to struktura przechowująca elementy w węzłach. Drzewo BST spełnia następujące własności:

- każdy węzeł ma co najwyżej dwa "dzieci"(lewe i prawe dziecko)
- dla każdego węzła wszystkie elementy w lewym poddrzewie są mniejsze od węzła, a wszystkie elementy w prawym poddrzewie są większe od węzła

Na rysunku:

- **Czerwony** - korzeń
- **Niebieski** - węzeł
- **Zielony** - liść

1.2 | AVL

TODO - Janek

2 | Tworzenie struktury

2.1 | Tworzenie drzewa BST

Podczas tworzenia drzewa pierwszy wierzchołek zostaje wstawiony do drzewa jako tzw. korzeń. Jest to wierzchołek nie mający żadnego "rodzica". Następnie każdy kolejny wierzchołek dodawany jest zgodnie z następującym kryterium:

Jeżeli wartość dodawanego wierzchołka jest większa od wartości aktualnie analizowanego wierzchołka przechodzimy do jego prawego "dziecka"jeśli aktualnie analizowany wierzchołek nie ma prawego "dziecka"w tym miejscu dodajemy nowy wierzchołek. Analogicznie gdy dodawany wierzchołek ma wartość mniejszą od aktualnego wierzchołka dodajemy go jako lewe dziecko aktualnego wierzchołka.

```python
class Node:
    def __init__(self, key):
        self.left = None
        self.right = None
        self.val = key
```

```python
def insert(root, key):
    if root is None:
        return Node(key)
    else:
        if root.val < key:
            root.right = insert(root.right, key)
        else:
            root.left = insert(root.left, key)
    return root
```

2.2 | Tworzenie drzewa AVL

TODO - Janek

3 | Wyszukiwanie minimum i maksimum

Wyszukiwanie minimum i maksimum polega na jak najbardziej optymalnym znalezieniu wierzchołka o najmniejszej i największej wartości w drzewie.

3.1 | Wyszukiwanie minimum i maksimum w drzewie BST

Dzięki budowie drzewa BST wyszukiwanie minimum i maksimum jest stosunkowo proste, ponieważ najmniejszy element znajduje się w skrajnie lewym wierzchołku drzewa, a największy element znajduje się w skrajnie prawym wierzchołku drzewa.

3.2 | Wyszukiwanie minimum i maksimum w drzewie AVL

TODO - Janek

4 | Wypisywanie in-order

Wypisywanie in-order polega na wypisywaniu wierzchołków zgodnie z trawersowaniem (odwiedzaniem wszystkich wierzchołków w określonej kolejności) metodą in-order czyli w porządku poprzecznym, gdy najpierw trawersujemy lewe poddrzewo, następnie korzeń, a na koniec prawe poddrzewo

4.1 | Wypisywanie in-order w drzewie BST

Budowa drzewa BST sprawia, że niezależnie od kolejności dodawania wierzchołków do drzewa przy wypisywaniu wierzchołków in-order zostaną one wypisane od najmniejszego do największego czyli zostaną posortowane rosnąco.

4.2 | Wypisywanie in-order w drzewie BST

TODO - Janek

5 | Równoważenie elementów w drzewie BST

Równoważenie elementów w drzewie binarnym pozwala na optymalizację operacji na drzewie, takich jak wyszukiwanie, dodawanie i usuwanie węzłów. Polega na lokalnej zmianie struktury BST, zachowując przy tym porządek wierzchołków. Celem równoważenia jest minimalizacja wysokości drzewa, co prowadzi do zmniejszenia kosztu operacji na drzewie.

Jednym z algorytmów równoważenia drzewa BST jest algorytm DSW (Day-Stout-Warren), który polega na balansowaniu drzewa przy pomocy rotacji(lewych i prawych) czyli zmiany struktury drzewa bez zmiany porządku wierzchołków.

Algorytm DSW składa się z dwóch etapów:

- Zamiana drzewa na listę (zwaną "winoroślą") poprzez wielokrotne rotacje prawe.
- Przywrócenie kształtu drzewa z listy poprzez wielokrotne rotacje lewe co drugiego węzła względem jego rodzica.

6 | Wykresy

Wykresy przedstawiające zależność czasu od wielkości danych wyżej omówionych zagadnień takich jak: tworzenie struktury, wyszukanie minimum i maksimum, wypisanie in-order oraz równoważenie drzewa BST