Laboratoria Algorytmów i Struktur Danych

Projekt 2

Drzewa przeszukiwań binarnych BST i drzewa samobalansujące

Wojciech Niedziela 160363

Jan Chojnacki 159567

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

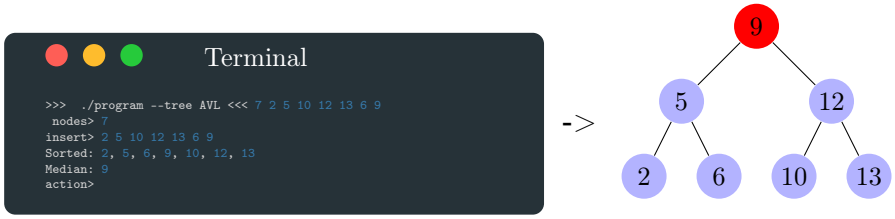
1.2 | AVL

Drzewo AVL (Adelson-Velsky and Landis Tree) to samobalansujące się drzewo binarne, które utrzymuje zrównoważony podział węzłów, aby zapewnić operacje o czasie złożoności $O(\log n)$. Drzewo AVL ma następujące cechy:

- Dla każdego węzła różnica wysokości pomiędzy lewym a prawym poddrzewem jest mniejsza lub równa 1.
- Drzewo AVL pozwala na szybkie operacje wyszukiwania, wstawiania, usuwania, a także inne funkcje, które działają efektywnie na zrównoważonym drzewie.

Na rysunku:

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



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

Drzewo AVL to rodzaj drzewa binarnego wyszukiwania (BST), w którym zachowana jest równowaga. Podczas tworzenia drzewa AVL, pierwszy wierzchołek zostaje wstawiony jako korzeń, a następnie każdy kolejny wierzchołek dodawany jest zgodnie z następującym schematem:

1. Jeśli wartość dodawanego wierzchołka jest mniejsza niż wartość aktualnie analizowanego wierzchołka, przechodzimy do lewego poddrzewa. Jeśli nie ma lewego "dziecka", dodajemy tam nowy wierzchołek. W przeciwnym razie kontynuujemy poszukiwanie w lewym poddrzewie.
2. Jeśli wartość dodawanego wierzchołka jest większa niż wartość aktualnie analizowanego wierzchołka, przechodzimy do prawego poddrzewa. Jeśli nie ma prawego "dziecka", dodajemy nowy wierzchołek w to miejsce. W przeciwnym razie kontynuujemy poszukiwanie w prawym poddrzewie.

Po dodaniu nowego wierzchołka do drzewa AVL, sprawdzana jest równowaga drzewa, aby upewnić się, że współczynnik równowagi każdego wierzchołka wynosi między -1 a 1. Jeśli współczynnik równowagi przekroczy te granice, drzewo musi zostać zbalansowane za pomocą odpowiednich rotacji. Istnieją cztery możliwe rotacje, które mogą przywrócić równowagę:

- **Rotacja lewa (LL)**: Stosowana, gdy lewe poddrzewo jest zbyt głębokie.
- **Rotacja prawa (RR)**: Stosowana, gdy prawe poddrzewo jest zbyt głębokie.
- **Rotacja lewo-prawa (LR)**: Stosowana, gdy lewe poddrzewo jest zbyt głębokie, ale problem dotyczy prawego "dziecka".
- **Rotacja prawo-lewa (RL)**: Stosowana, gdy prawe poddrzewo jest zbyt głębokie, ale problem dotyczy lewego "dziecka".

Dzięki tym mechanizmom drzewa AVL utrzymują równowagę, co zapewnia efektywne operacje dodawania, wyszukiwania i usuwania węzłów.

```python
class AVLNode:
    def __init__(self, key):
        self.val = key
        self.left = None
        self.right = None
        self.height = 1
```

```python
def insert(self, key):
    if key < self.val:
        if self.left:
            self.left = self.left.insert(key)
        else:
            self.left = AVLNode(key)
    else:
        if self.right:
            self.right = self.right.insert(key)
        else:
            self.right = AVLNode(key)

    self.update_height()
    return self.balance()
```

```python
def get_tree_size(self):
    if self is None:
        return 0
    left_size = self.left.get_tree_size() if self.left else 0
    right_size = self.right.get_tree_size() if self.right else 0
    return 1 + left_size + right_size
```

```python
def rotate_left_right(self):
    self.left = self.left.rotate_left()
    return self.rotate_right()

def rotate_right_left(self):
    self.right = self.right.rotate_right()
    return self.rotate_left()
```

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

W drzewie AVL, które jest rodzajem drzewa binarnego wyszukiwania, minimum i maksimum można znaleźć, kierując się strukturą drzewa:

3.2.1 | Wyszukiwanie Minimum

Aby znaleźć najmniejszy element, zaczynamy od korzenia i podążamy ścieżką w dół po lewym poddrzewie, dopóki nie napotkamy węzła bez lewego "dziecka". To będzie minimum drzewa.

3.2.2 | Wyszukiwanie Maksimum

Aby znaleźć największy element, podążamy ścieżką w dół po prawym poddrzewie, zaczynając od korzenia, dopóki nie natrafimy na węzeł bez prawego "dziecka". To będzie maksimum drzewa.
Dzięki zrównoważonej strukturze drzew AVL, operacje te mają złożoność rzędu $O(\log n)$, gdzie n to liczba węzłów w drzewie.

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 AVL

Aby wypisać wartości w porządku "in-order", rozpoczynamy od korzenia i wykonujemy następujące kroki:

1. Jeśli istnieje lewe poddrzewo, najpierw przechodzimy do niego i wypisujemy wartości w tym poddrzewie w porządku "in-order".
2. Następnie wypisujemy wartość bieżącego węzła.
3. Jeśli istnieje prawe poddrzewo, przechodzimy do niego i wypisujemy wartości w tym poddrzewie w porządku "in-order".

Ten proces zapewnia, że wartości w drzewie AVL są wypisywane w kolejności rosnącej. Dzięki strukturze AVL, w której każda operacja wstawiania lub usuwania jest wykonywana z zachowaniem równowagi, czas wypisywania w porządku "in-order"wynosi $O(n)$, gdzie n to liczba węzłów w drzewie.

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