Проект Самобалансуючих Дерев

Виконавці: Арсен Боцко, Мацелюх Максим, Юліан Заяць, Сиротюк Віктор

08.05.2025

Огляд

Цей проєкт реалізує кілька типів самобалансуючих деревоподібних структур даних і використовує їх як основу для створення SQL-подібної системи керування даними. Реалізовані структури дерев:

- AVL-дерево
- Червоно-чорне дерево
- Splay-дерево
- B-дерево
- 2-3-дерево

1 Алгоритми та структури даних

1.1 AVL-дерево

AVL-дерева — це збалансовані за висотою двійкові дерева пошуку, де різниця висот лівого і правого піддерев (фактор балансування) для будь-якого вузла не перевищує 1.

Ключові характеристики:

- Самобалансування через обертання
- Операції пошуку, вставки та видалення за час $O(\log n)$
- Фактор балансування = висота(ліве піддерево) − висота(праве піддерево)

Операції балансування:

- Ліве обертання
- Праве обертання
- Ліво-праве обертання (подвійне)
- Право-ліве обертання (подвійне)

1.2 Червоно-чорне дерево

Червоно-чорні дерева — це двійкові дерева пошуку з додатковим бітом для кольору (червоний або чорний), що забезпечує баланс через набір властивостей.

Ключові властивості:

- Кожен вузол або червоний, або чорний
- Корінь завжди чорний
- Усі NIL-листки чорні
- Якщо вузол червоний, обидва його нащадки чорні
- Всі шляхи від вузла до NIL-листків містять однакову кількість чорних вузлів

Операції балансування:

- Перефарбування
- Ліве обертання
- Праве обертання

1.3 Splay-дерево

Splay-дерева — це самоналаштовувані двійкові дерева пошуку, які переміщують нещодавно доступні вузли ближче до кореня завдяки операції splay.

Ключові характеристики:

- Відсутність явного фактора балансування
- Амортизований час операцій $O(\log n)$

Операції:

- Операція `splay` (переміщення вузла до кореня)
- Кроки Zig, Zig-Zig та Zig-Zag

1.4 B-дерево

B-дерева — це збалансовані дерева пошуку, оптимізовані для зберігання на дискових носіях.

Ключові характеристики:

- Усі листки на одному рівні
- Кожен вузол містить кілька ключів та дітей
- Ключі всередині вузла відсортовані
- Ефективність при роботі з повільними носіями (дисковий ввід/вивід)

Операції:

- Пошук
- Вставка (з розщепленням вузла)
- Видалення (з об'єднанням вузлів)

1.5 2-3-дерево

2-3-дерева — це збалансовані дерева пошуку, де кожен внутрішній вузол має 2 або 3 дітей.

Ключові характеристики:

- Усі листки на одному рівні
- Вузли можуть містити 1 або 2 ключі
- Завжди підтримується ідеальний баланс

Операції:

- Пошук
- Вставка (з розщепленням вузла)
- Видалення (з об'єднанням або перерозподілом вузлів)

2 Використані принципи дискретної математики

1. **Теорія графів:** Дерева як спеціалізовані графи, алгоритми обходу (inorder, preorder), властивості шляхів та обчислення глибини
2. **Рекурентні співвідношення:** Аналіз висоти збалансованих дерев та складності операцій
3. **Властивості балансування:** Фактор балансування AVL, правила кольорів червоно-чорного дерева, структурні обмеження B- та 2-3-дерев
4. **Інваріанти:** Підтримка властивостей дерев під час операцій, інваріанти циклів в алгоритмах

3 Структура проєкту

`abstract_class.py` Абстрактний базовий клас для всіх реалізацій дерев

`AVL_Tree.py` Реалізація AVL-дерева

`red_black_tree.py` Реалізація червоно-чорного дерева

`splay_tree.py` Реалізація Splay-дерева

`b_tree.py` Реалізація B-дерева

`two_three_tree.py` Реалізація 2-3-дерева

`tree_adapters.py` Адаптери для уніфікованого інтерфейсу всіх дерев

`tree_factory.py` Фабричний клас для створення екземплярів дерев

`data_manager.py` Клас для керування базами даних і таблицями

`tree_sql.py` SQL-подібний інтерфейс для роботи з системою керування даними

4 Інструкції з використання

Для запуску SQL-інтерфейсу скористайтеся наступним прикладом:

```
from tree_sql import TreeSQL

sql = TreeSQL()

sql.parse_command("CREATE DATABASE mydb")
sql.parse_command("USE mydb")
sql.parse_command("CREATE TABLE users (id, name, age) USING avl")
sql.parse_command("INSERT INTO users VALUES (1, '        ', 25)")
sql.parse_command("SELECT * FROM users")
```