Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Алгоритми та складність

Завдання №3

“Розширюване дерево”

Виконав студент 2-го курсу

Групи К-29

Пащенко Дмитро Вікторович

2020

**Завдання**:

Реалізувати розширюване дерево (splay tree), застосувавши його до моделі, описаної у Вашому варіанті.

**Варіант 15:**

Предметна область: Залізниця

Об'єкти: Дороги, Станції

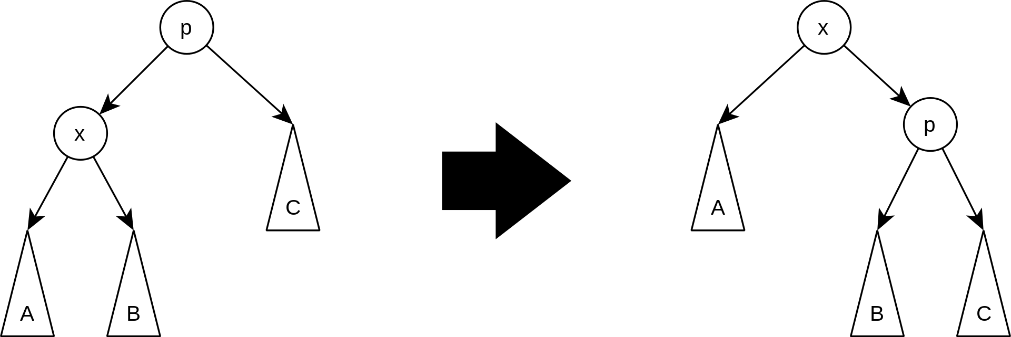
Примітка: Є безліч залізничних доріг. У відомстві кожної дороги знаходиться безліч станцій.

**Теорія:**

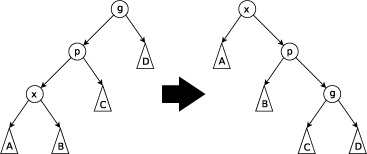
* Розширюване дерево (англ. splay tree) є двійковим деревом пошуку, у якому підтримується збалансованість.
* Дане дерево не потребує додаткової пам’яті для зберігання корисної інформації (як, наприклад, додаткові поля у червоно-чорного або АВЛ-дерева).
* Облікова вартість з розрахунку на одну операцію з деревом складає O(log n).
* **SPLAY**: Основна операція дерева. Полягає в переміщенні вершини в корінь за допомогою послідовного виконання трьох операцій: Zig, Zig-Zig і Zig-Zag.

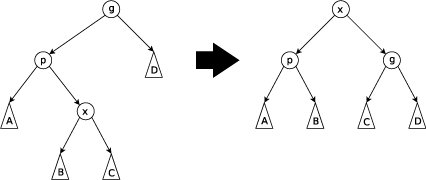
**(**Позначимо вершину, яку хочемо перемістити в корінь за x, її родича — p, а родича p (якщо існує) — g).

* **ZIG**: виконується, коли p є коренем. Дерево повертається по ребру між x та p. Існує лише для розбору крайнього випадку і виконується лише один раз в кінці, коли початкова глибина x була непарна.

****

* **ZIG-ZIG**: виконується, коли x і p є лівими (або правими) синами. Дерево повертається по ребру між p та x.

****

* **ZIG-ZAG**: виконується, коли x є правим сином, а p — лівим (чи навпаки). Дерево повертається по ребру між p та x, а потім — по ребру між x та g.
* **SEARCH (Пошук елемента)**: виконується як у звичайному двійковому дереві пошуку. При знаходженні елементу запускаємо **SPLAY** для нього.
* **INSERT (Додавання елемента)**: Запускаємо **SPLAY** від елементу, що додається, і підвішуємо дерева, що вийшли, до нього.
* **REMOVE (Видалення елемента)**: знаходимо елемент в дереві, робимо Splay для нього, робимо поточним деревом Merge його дітей.
* **MERGE (Об'єднання двох дерев)**: для злиття дерев T1 і T2, в яких всі ключі T1 менше ключів в T2, робимо Splay для максимального елементу T1, тоді біля кореня T1 не буде правого дочірнього елемента. Після цього робимо T2 правим дочірнім елементом T1.
* **SPLIT (Розділення дерева на дві частини)**: для розділення дерева знайдемо найменший елемент, більший або рівний x і зробимо для нього Splay. Після цього відрізуємо біля кореня лівого дочірнього елемента і повертаємо 2 дерева, що вийшли.

**Мова програмування:** С++.

**Інтерфейс користувача**

Введення даних відбувається через консоль.

Вхідні дані: назви залізниць, введення «end» свідчить про закінчення вводу.

Результат: дерево назв залізниць (друковане в preorder).

**Модулі програми:**

* **void rotate(node<T>\* parent, node<T>\* child)**

Здійснює звичайний поворот сина навколо батька (zig).

**Складність:** .

* **node<T>\* splay(node<T>\* vertex)**

Послідовно здійснює повороти (zig, zig-zig або zig-zag), поки вказана вершина не стане коренем.

**Складність:** , n – кількість вершин дерева.

* **node<T>\* search(T const &key, node<T>\* vertex)**

Шукає вершину з даним ключем, виносячи її в корінь. Якщо такої вершини в дереві немає, в корінь виноситься вершина з найближчим до даного ключем.

**Складність:**

* **std::pair<node<T>\*, node<T>\*> split(T const &key)**

Розділяє дерево на два піддерева в залежності від даного ключа, повертає покажчики на корені отриманих дерев.

**Складність:**

* **void insert(T const &key)**

Вставляє в дерево вершину з даним ключем, виносячи її в корінь.

**Складність:**

* **node<T>\* merge(node<T>\* left, node<T>\* right)**

Зливає два дерева в одне, не порушуючи властивостей дерева пошуку.

**Складність:**

* **void destroy(node<T>\* vertex)**

Видаляє дерево.

**Складність:**

* **void remove(T const &key)**

Видаляє вершину з даним ключем, не порушуючи властивостей дерева пошуку.

**Складність:**

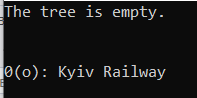
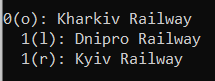
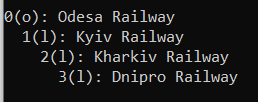
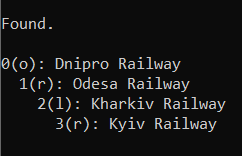
* **void print(node<T>\* vertex, unsigned level = 0, status stat = status::ROOT) const**

Друкує дерево (preorder).

**Складність:**

**Тестові приклади**

Послідовно додав чотири залізниці до дерева. Далі здійснив пошук однієї із залізниць. Відслідкував зміни покроково:

1. До порожнього дерева додав “Kyiv Railway”. Ця залізниця стала коренем.
2. Додав Dnipro Railway. Оскільки Kyiv Railway за алфавітом «більше» ніж Dnipro Railway, дерево розділилося на два піддерева: ліве з коренем nullptr і праве з коренем Kyiv Railway – які в свою чергу під’єдналися до нової кореневої вершини Dnipro Railway як лівий і правий син відповідно.
3. Додав Kharkiv Railway. Алгоритм спочатку здійснює «винесення в корінь» найближчої до ключа вершини. Kyiv Railway здійснює zig-поворот навколо Dnipro Railway, у результаті Dnipro Railway стає лівим сином нового кореня Kyiv Railway. Оскільки Kharkiv Railway менший за Kyiv Railway, то дерево ділиться на два: ліве з коренем Dnipro Railway і праве з коренем Kyiv Railway - які в свою чергу під’єднуються до нової кореневої вершини Dnipro Railway як лівий і правий син відповідно.
4. Додав Odesa Railway. Алгоритм спочатку здійснює «винесення в корінь» найближчої до ключа вершини. Kyiv Railway здійснює zig-поворот навколо Kharkiv Railway, у результаті Kyiv Railway стає коренем, Kharkiv Railway – лівим сином Kyiv Railway, Dnipro Railway – лівим сином Kharkiv Railway. Тобто на виході маємо дерево з лише лівими синами. Odesa Railway більша за Kyiv Railway, тому дерево ділиться на два дерева: ліве з коренем Kyiv Railway і праве з коренем nullptr – які в свою чергу під’єднуються до нової кореневої вершини Odesa Railway як лівий і правий син відповідно.
5. Шукаю в дереві Dnipro Railway. Dnipro Railway в процесі пошуку здійснює zig-zig-поворот навколо Kharkiv Railway та Kyiv Railway. У результаті маємо зворотну побудову: Odesa Railway (left)-> Dnipro Railway (right)-> Kharkiv Railway (right)-> Kyiv Railway. Далі відбувається звичайний zig-поворот Dnipro Railway та Kyiv Railway. Dnipro Railway стає новим коренем, Odesa Railway його правим сином, Kharkiv Railway лівим сином Odesa Railway. Відповідно повертається значення true.

**Література**

* <https://habr.com/ru/company/JetBrains-education/blog/210296/>
* <https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D1%88%D0%B8%D1%80%D1%8E%D0%B2%D0%B0%D0%BD%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>