Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Алгоритми та складність

Завдання №6

“Реалізація B+-дерева”

Виконав студент 2-го курсу

Групи К-29

Пащенко Дмитро Вікторович

2020

**Завдання**:

Реалізувати B+-дерево, 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:**

* B-деревом називають кореневе дерево, влаштоване наступним чином.
  + Кожен вузол x містить наступні поля:
    - n[x] — кількість ключів, що зберігаються у вузлі x;
    - key1[x], key2[x], …, keyn(x)[x] — самі ключі в не спадаючому порядку;
    - leaf[x] — булеве значення, істинне, коли вузол x є листом.
  + Якщо x — внутрішній вузол, то він містить покажчики c1[x], c2[x] cn(x)+1[x], на його дітей в кількості n[x]+1.
  + У листів дітей немає, і ці поля для них не визначені.
  + Усі листя знаходяться на одній і тій же глибині, що дорівнює висоті дерева.
  + Можлива кількість ключів, що зберігаються в одному вузлі, визначається параметром t≥2, яке називається мінімальним ступенем B-дерева.
  + Для кожного некореневого вузла x виконується нерівність (t-1)≤n[x]≤(2t-1). Таким чином, число дітей у будь-якого внутрішнього вузла (крім кореня) знаходиться в межах від t до 2t.
  + Ключі keyi[x] служать границями, що розділяють значення ключів в піддеревах.
* B+-дерево (англ. B+ tree) — тип дерева, яке подає відсортовані дані в вигляді, що дозволяє швидке додавання, отримання і видалення записів. В B+-дереві, на відміну від B-дерева, всі записи зберігаються на рівні листових вузлів дерева; у внутрішніх вузлах зберігаються лише ключі.
* Якщо , то для B+-дерева з n вузлами і мінімальним ступенем висота .
* B+-дерева є збалансованими, тому час виконання стандартних операцій в них пропорційний висоті, тобто .

**Мова програмування:** С++.

**Інтерфейс користувача**

Введення даних відбувається через консоль.

Вхідні дані: команди

* 0 – завершити програму
* 1 – вставити залізницю (за назвою)
* 2 – видалити залізницю (за назвою)
* 3 – вивести дерево (preorder)

Результат: B+-дерево залізниць.

**Модулі програми:**

* **bool Node<T>::contains(T const &key) const**

Перевіряє, чи даний ключ міститься у вузлі.

**Складність:** , де t – степінь дерева.

* **unsigned Node<T>::insert(T const &key)**

Вставляє у вузол ключ (якщо його ще немає), не порушуючи впорядкованості. Повертає індекс ключа або UINT\_MAX, якщо ключ уже міститься у вузлі.

**Складність:** , де t – степінь дерева.

* **bool Node<T>::remove(T const &key)**

Видаляє ключ з вузла. Повертає хибне значення, якщо вузол не містить даного ключа.

**Складність:** , де t – степінь дерева.

* **Node<T>\* BPlusTree<T>::searchLeaf(T const &key) const**

Шукає листок, у якому міститься даний ключ (або міг би міститися).

**Складність:** , де n – кількість ключів у дереві, t – степінь дерева.

* **void BPlusTree<T>::split(Node\* node)**

Розбиває переповнений вузол на два нових.

**Складність:** , де t – степінь дерева.

* **bool BPlusTree<T>::insert(T const &key)**

Вставляє ключ в дерево.

**Складність:** , де n – кількість ключів у дереві, t – степінь дерева.

* **void BPlusTree<T>::update(Node\* current, T const &key, T const &replacement)**

Оновлює вершини на шляху до кореня, замінюючи входження видаленого ключа.

**Складність:** , де n – кількість ключів у дереві, t – степінь дерева.

* **void BPlusTree<T>::deleteInNode(Node\* current, T const &key)**

Видаляє ключ з вершини, відновлює властивості дерева.

**Складність:** , де n – кількість ключів у дереві, t – степінь дерева.

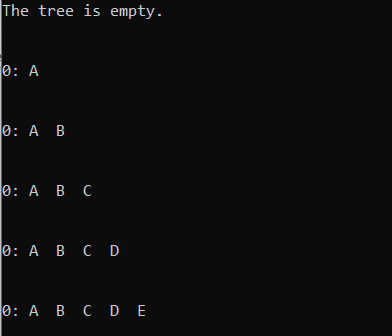
* **bool BPlusTree<T>::remove(T const &key)**

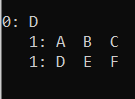
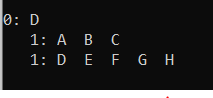
Видаляє ключ з дерева.

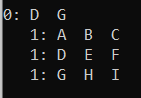
**Складність:** , де n – кількість ключів у дереві, t – степінь дерева.

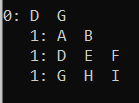
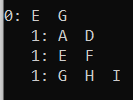
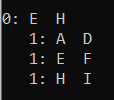
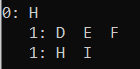
**Тестові приклади**

1. Створюю дерево степені 3. Тобто мінімальна кількість ключів у листках 2, максимальна – 5. Задля зручності та незагромадженості називатиму залізниці великими буквами латинського алфавіту. Послідовно додав до дерева ключі від «А» до «E». Бачимо, що все зберіглося в корінь.



1. Додаю ключ «F». Корінь розділився на 2 листка, в корінь відійшов середній ключ «D», причому цей ключ також знаходиться в правому листку.
2. Послідовно додаю ключі «G» і «H». Правий листок виявляється повністю заповненим.
3. Додаю ключ «І». Правий листок розщеплюється. У корінь відходить середній ключ «G». Нова конфігурація задовольняє властивостям В+-дерева.



1. Видаляю ключ «С». Властивості дерева не порушуються, тому нічого не змінюється.
2. Видаляю ключ «В». У найлівішому листку кількість ключів тепер менша за мінімальну. Ключ позичається у правого брата (мінімум). Маємо нову конфігурацію.
3. Видаляю ключ «G». Спочатку він видаляється з листка, кількість ключів тепер мінімальна, тобто злиття не відбувається. Згодом ключ «G» видаляється з кореня, на заміну стає мінімальний елемент з найправішого потомка.
4. Видаляю ключ «А». Розмір найлівішого листка менший за мінімальний, відбувається злиття його з правим братом. При цьому з кореня зникає один з ключів. Маємо нову конфігурацію.

**Література**

* Кормен, Лейзерсон, Рівест, Штайн. Алгоритми: побудова і аналіз, 2-е видання. – 2005
* <https://uk.wikipedia.org/wiki/B%2B_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>
* <https://neerc.ifmo.ru/wiki/index.php?title=B%2B-%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE>