Київський національний університет імені Тараса Шевченка

Факультет комп`ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Алгоритми та складність

Завдання №6

“ B+ - дерево”

Виконав студент 2-го курсу

Групи К-29

Циганов Аркадій Олексіійович

2021

**Предметна область**

Предметна область:  Автосалон

Об’єкти:  Виробники автомобілів, Марки

Примітка: Марки  автомобілів згруповані  по виробникам.  У  кожного

виробника є множина марок.

**Завдання**

Реалізувати B+ - дерево. Зберігати будемо інформацію про виробників.

**Теорія**

B+ - дерево є підвидом звичайних B – дерев, тому дамо спочатку визначення B- дереву.

* В-дерева – узагальнення бінарних дерев пошуку.
* Висока степінь розгалуження – вузли можуть мати

до тисяч потомків.

Дерево Т з коренем root[T] та властивостями:

1. Кожен вузол x містить поля:

* n[x] – поточна кількість ключів вузла x;
* Впорядковано збережені ключі, так що
* Логічне значення leaf[x], істинне, якщо x – лист.

1. Кожен внутрішній вузол x містить (n[x]+1) вказівник на дочірні вузли.
2. Ключі розділяють піддіапазони ключів під дерев: якщо – ключ, що зберігається у під дереві з коренем [x], то
3. Всі листи розташовані на одній глибині h, що дорівнює висоті дерева (тобто В-дерево ідеально збалансоване за висотою.)
4. Мінімальна і максимальна кількість ключів у вузлі регламентовані фіксованим цілим t ≥ 2 (мінімальна степінь, minimum degree):

* кожен вузол крім кореня містить як мінімум (t–1) ключ, тобто матиме принаймні t синів;
* непорожнє дерево має в корені хоча б один ключ;
* кожен вузол містить не більше (2t–1) ключів, тобто матиме максимум 2t синів; вузол вважається повним, якщо має рівно (2t–1) ключ.

**B+ - дерева**

• Істинні значення ключів містяться тільки в листах, внутрішні вузли містять лише ключі-роздільники діапазонів піддерев.

• Листки додатково зв’язані у список. Це дозволяє швидкий доступ до ключів в порядку зростання.

• Легко реалізується незалежність програми від структури інформаційної запису.

• Пошук обов'язково закінчується в листі. Видалення ключа завжди з листа.

• Вимагають більше пам'яті для представлення, порівняно з B-деревами.

**Алгоритми**

**Insert ключа K**

Спочатку ми просто ідем по списках листів зліва направо і як тільки ми знайдемо ключ, який більший за K, то ми K вставляємо перед цим ключем і далі перевіряємо ключі у батьківському внутрішньому вузлі на те, чи виконуються властивості В+ -дерева, тобто чи ключі зміненого листа входять в інтервал, який був до цього у батьківському внутрішньому вузлі. Якщо властивості не виконується, то ми вставляємо ключ, який стоїть посередині масиву, в який потрапив ключ К, у середину того інтервалу (утвориться двоє нових інтервалів) і змінений лист після вставки розділяємо на два інтервали.

Наприклад:

10

1 3 5 7 9

10 13 15 17 18

Вставляємо 6

10

1 3 5 6 7 9

10 13 15 17 18

Знаходимо середину масиву, це 6 => вставляємо 6 до батьківського вузла і ділим масив навпіл.

6 10

6 7 9

1 3 5

10 13 15 17 18

* **Delete**

1. Видалення ключа 6 у В+ - дереві починається з його видалення з списку листів.
2. Переходимо на рівень вище.
3. A) Якщо тут є 6, то видаляємо його, його синівські під інтервали об’єднуємо. Якщо розмір отриманого інтервалу більший ніж можливий, то ділим його навпіл, перший елемент другого масиву заміняє всі місця зверху, де ми будем зустрічати елемент 6.
4. Інакше повторюємо крок 2.

Приклади:

1)

6 10

6 7 9

1 3 5

10 13 15 17 18

10

1 3 5 7 9

10 13 15 17 18

2)

6 10

1 3 4 5

6 7 9

10 13 15 17 18

5 10

5 7 9

1 3 4

10 13 15 17 18

**Складність**

Усі операції з деревом займають де - висота дерева для якої справедлива нерівність (мінімальна кількість піддерев), то складність не перевищує

**Мова програмування**

С++

**Модулі програми**

//Class of B+ tree node

template<typename DataType>

class BPlusNode {

typedef std::shared\_ptr<BPlusNode> Node;

public:

bool leaf;

unsigned size;

std::vector<DataType> data;

std::vector<Node> children;

Node next\_leaf;

Node prev\_leaf;

Node parent;

//Constructor

BPlusNode();

};

//Class of B+ tree

template<typename DataType>

class BPlusTree {

typedef std::shared\_ptr<BPlusNode<DataType>> Node;

private:

unsigned \_minimum\_degree;

unsigned \_min\_node\_fill;

unsigned \_max\_node\_fill;

Node \_root;

//Search value in subtree

std::pair<Node, unsigned>

SubtreeSearch(Node subtree\_root, const DataType& key);

//Division tree into parts

void SplitNode(Node node);

//Insertion value into subtree

std::pair<Node, unsigned>

SubtreeInsert(Node subtree\_root, const DataType& key);

//Removing from subtree

void SubtreeRemove(Node node, unsigned index);

//Function for interactive finding elements

Node ChooseNode(Node current\_node);

public:

//Constructor

explicit BPlusTree(unsigned minimum\_degree = 2);

//Output in console

void Print();

//Check that element is at tree

bool At(const DataType& key);

//Insertion new element into tree

void Insert(const DataType& key);

//Removing element from tree

void Remove(const DataType& key);

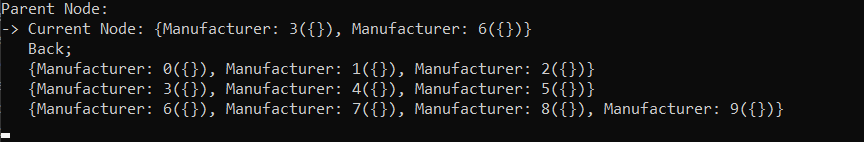
};

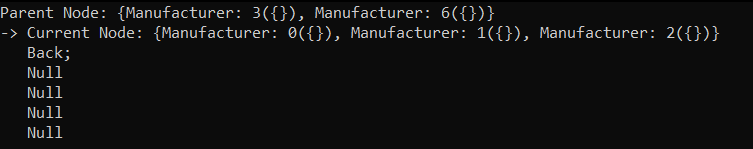
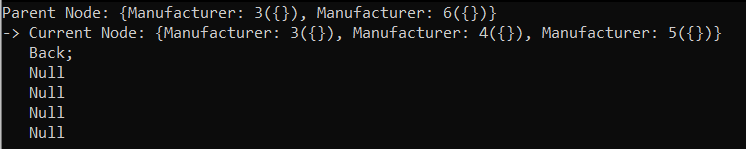
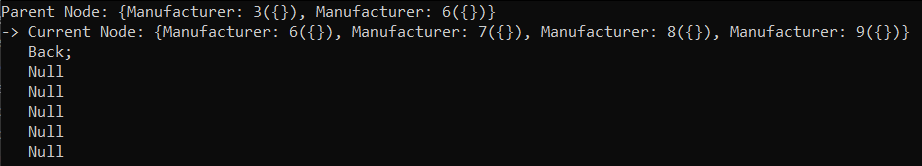
**Інтерфейс користувача**

Вхідні дані генеруються програмою і виводяться в консоль. Можливе налаштування вводу з консолі.

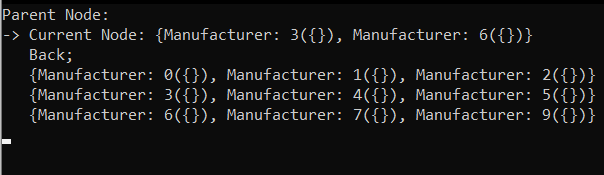
**Приклад виводу програми**

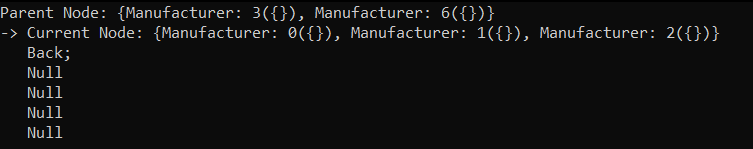
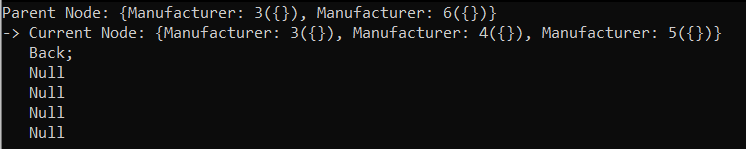
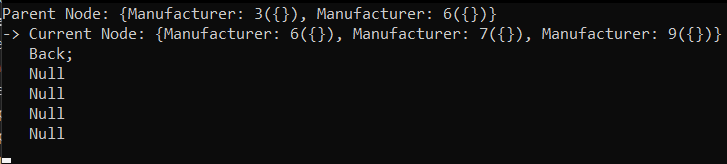
Початкове дерево:



Після видалення “8”



**Висновки**

Реалізували В+ - дерево, до мінусів можна віднести те, що реалізація дерева вимагає більше пам'яті для представлення, а до плюсів те, що листки додатково зв’язані у список. Це дозволяє швидко отримати доступ до ключів в порядку зростання порівняно з B-деревами.

**Література**

* <https://habr.com/ru/company/sberbank/blog/413749/>
* Лекція № 4