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-28
Гуща Дмитро Сергійович

2021

Предметна область

Варіант 4
Предметна область: Учбовий відділ
Об’єкти: Групи, Студенти
Примітка: Маємо множину учбових груп. Кожна група містить в собі множину студентів

Завдання

Реалізувати B+ - дерево

Теорія

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

- B-дерева – узагальнення бінарних дерев пошуку.
- Висока степінь розгалуження – вузли можуть мати до тисяч потомків.

Дерево T з коренем root[T] та властивостями :

1. Кожен вузол x містить поля:
   – n[x] – поточна кількість ключів вузла x;
   – впорядковано збережені ключі, так що $key_1[x] \le key_2[x] \le ... \le key_n[x]$ [x]; – логічне значення leaf[x], істинне, якщо x – лист.
2. Кожен внутрішній вузол x містить (n[x]+1) вказівник $c_1[x]$, ..., $c_{n[x]+1}[x]$ на дочірні вузли.
3. Ключі key i[x] розділяють піддіапазони ключів піддерев: якщо k_i – ключ, що зберігається у піддереві з коренем ci [x], то $k1 \le key[x] \le k_1 \le key_2[x] \le ... \le key_n[x][x] \le k_n[x]+1$.
4. Всі листи розташовані на одній глибині h, що дорівнює висоті дерева. (Тобто B-дерево ідеально збалансоване за висотою.)
5. Мінімальна і максимальна кількість ключів у вузлі регламентовані фіксованим цілим $t \ge 2$ (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 листках, 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.

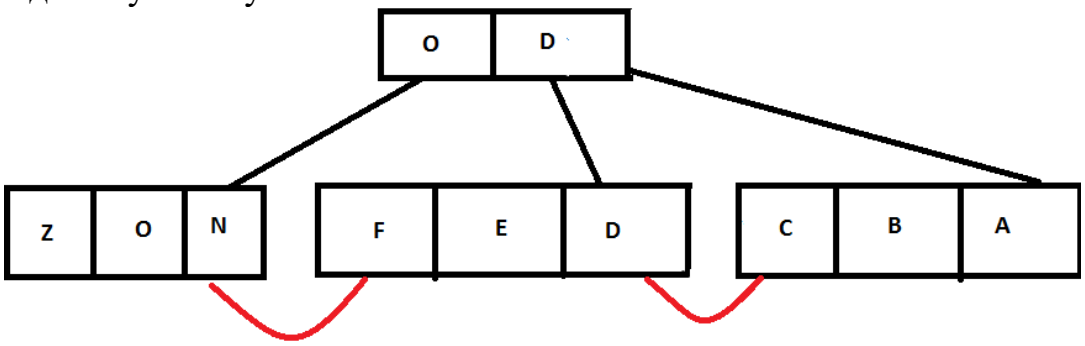
• Пошук обов'язково закінчується в листі. Видалення ключа завжди з листа.

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

Алгоритми

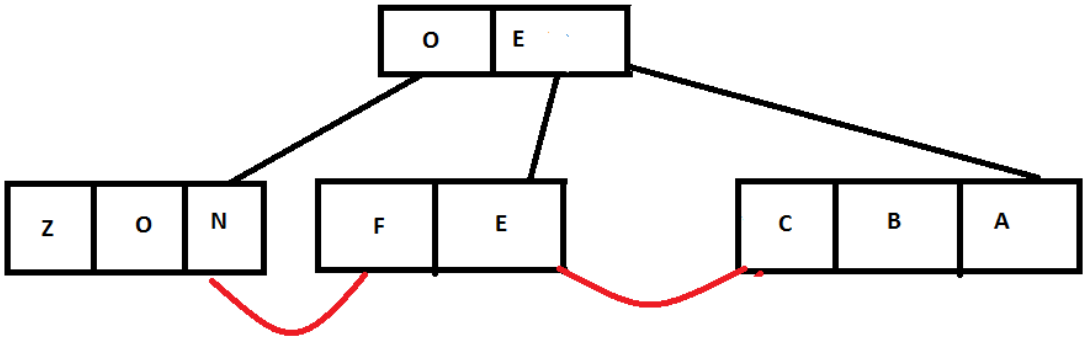
- **Delete**

Видалення ключа D у B+ - дереві починається з його видалення з списку який лежить на листі а потім його заміна у внутрішньому вузлі на таке значення K яке буде меншим за ключ I ключ який є сусідом ключа D у внутрішньому вузлі справа і більший за M (сусід зліва) K береться з того самого листа з якого ми видаляємо ключ D якщо там ключі замалі то шукаємо у сусідньому правому листі а якщо занадто великі то шукаємо у сусідньому лівому листі



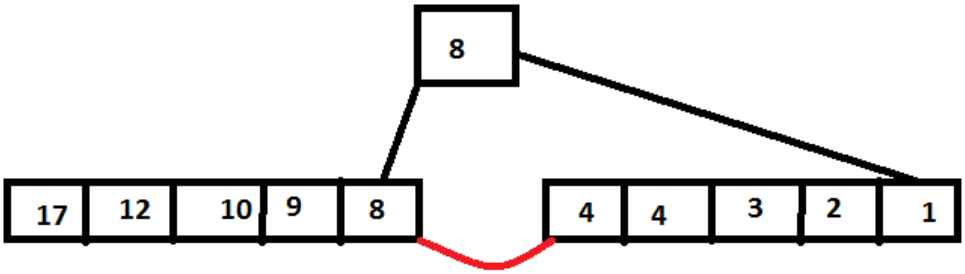
Нехай потрібно видалити елемент D , тоді видаляємо його з центрального листа. Його сусідні елементи у внутрішньому вузлі це O і null зі списку

Шукаємо такий ключ K що $O > K > null$ тоді $K = E$ і дерево після видалення D має вигляд

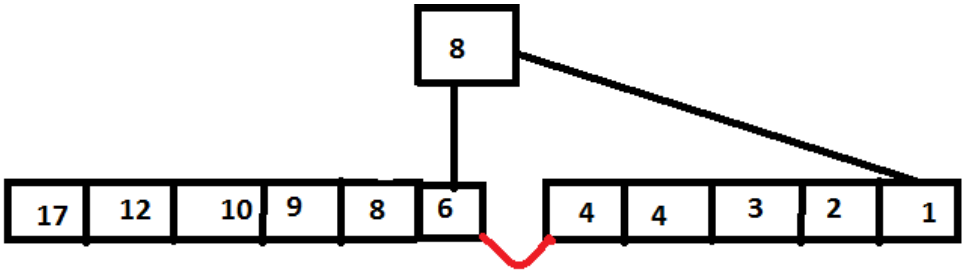


Insert ключа K

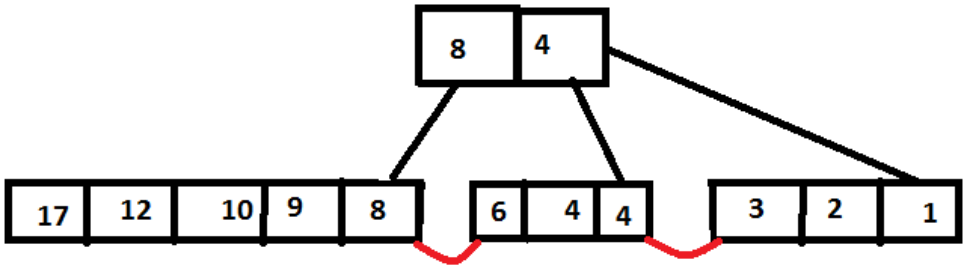
Спочатку ми просто ідем по списках листів з ліва на право і як тільки ми знайдемо ключ який більший за K то ми K вставляємо після цього ключа і далі провіряємо Ключі у батьківському внутрішньому вузлі на те чи виконуються властивості B+ -дерева тобто чи ключі зміненого листа входять в інтервал в який був до цього у батьківському внутрішньому вузлі. Якщо властивості не виконується то ми вставляємо ключ який стоїть спереду(тобто менший) K у середину того інтервалу(утвориться два нових інтервалів) і змінений лист після вставки розділяємо під ті два інтервали Наприклад:



Вставляємо 6



Так як 6 стоїть у тому листі елементи якого мають бути більші усі за 8 то вставляємо 4 у батьківський внутрішній вузол і перерозподіляєм листи



Складність

Усі операції з деревом займають $O(h)$ де h - висота дерева для якої справедлива нерівність $h \leq log_t\left(\frac{n+1}{2}\right)$ де t степінь дерева (мінімальна кількість піддерев) то складність не перевищує $O(log_t\, n)$

Мова програмування

C++

Модулі програми

student.h

```
class Student{}; //Класс опису студента
std::string getName(); // метод повертає ім’я студента
void getStudent();//метод виводить ID та ім'я студента в консоль
void setName(std::string name); //метод змінює ім'я студента
```

group.h

```
class Group {};
Group() : title("NULL"); //конструктор пустої групи
Group(std::string title); //конструктор з початковою назвою групи
Group(std::string title, Student* first_student); //конструктор з початковою назвою
групи та першим студентом
std::string getGroupTitle(); //модуль повертає назву групи
std::vector<Student*> getGroupStudents(); //модуль повертає множину студентів
void setGroupTitle(std::string title); //модуль змінює назву групи
void setGroupStudents(std::vector<Student*> students); //модуль змінює множину
студентів
void addStudent(Student* student); //додати нового студента
void printStudents(); //вивід у консоль усіх студентів групи
```

bPlusTree,h

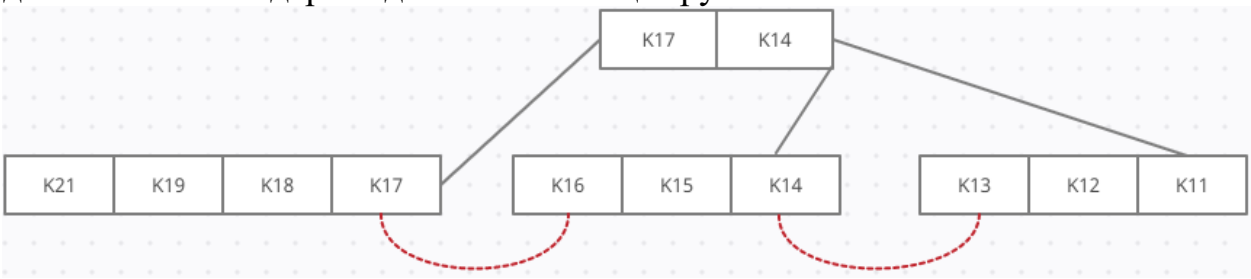
```
class BPlusTree; //клас для опису самого B+ -дерева
void _printStep(OStream& output, std::shared_ptr<BPlusNode<DataType>> node,
int level); //функція виводить деерево у консоль
std::pair<Node_ptr, unsigned>
_subtree_search(std::shared_ptr<BPlusNode<DataType>> subtree_root, const
DataType& key); //пошук ключа у піддерві
void _split_node(std::shared_ptr<BPlusNode<DataType>> node); //функція ділить
дерево на два піддерева
std::pair<Node_ptr, unsigned>
_subtree_insert(std::shared_ptr<BPlusNode<DataType>> subtree_root, const
DataType& key);//вставка ключа у піддерево
void _remove_from_node(Node_ptr node, unsigned index);//видалення ключа з
піддерева
explicit BPlusTree(unsigned minimum_degree = 2);//конструктор
void print(OStream& output);//функція виводить дерево у консоль
void printSorted(OStream& output);//функція виводить списки ключів у порядку
зростання
bool includes(const DataType& key);//функція перевіряє на знаходження ключа в
деерві
void insert(const DataType& key);//функція вставки
void remove(const DataType& key);//функція видалення
```

Інтерфейс користувача

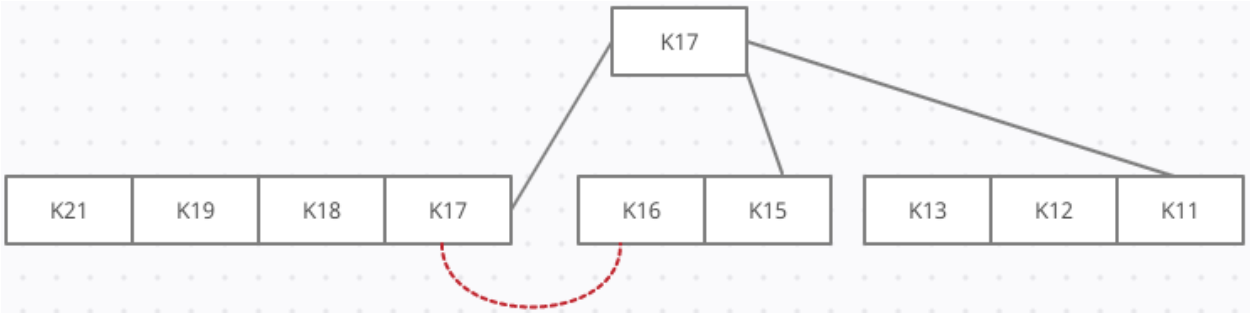
Вхідні дані вводяться з консолі користувачем і виводяться в консоль.

Тестовий приклад

Нехай Потрібно представити групи та їх назви. Це можна зробити за допомогою B +- дерева де кожен лист це група

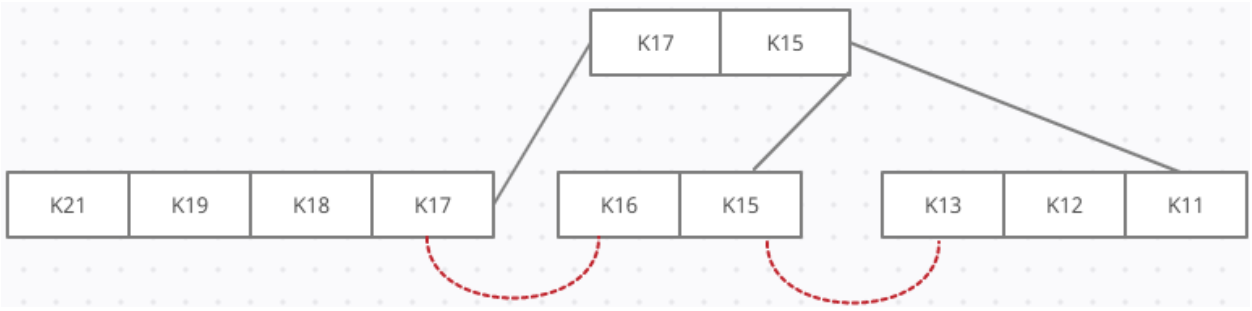


Нехай нам потрібно видалити якусь групу якого не стало у наявності наприклад К14. Спочатку за стандартним пошуком шукаєм К14 у B+ -дереві Потім видаляєм його з листа і внутрішнього вузла:



Далі зі списку 

вибираєм найменший такий ключ який буде задовольняти нерівність К17 $< K <$ $null$. К=К15 берем цей ключ вставляєм його на місце К14 у внутрішній вузол
Остаточний вигляд дерева:



Висновки

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

Література

- https://habr.com/ru/company/sberbank/blog/413749/
- Лекція № 4