# Другие виды деревьев. B-дерево

Булгаков Илья, Гусев Илья

Московский физико-технический институт

Москва, 2023

# Содержание

# Деревья поиска

Виды самобалансирующихся деревьев

- AVL-дерево
- **B-дерево**
- Splay-дерево
- Красно-черное дерево

# B-дерево

## Зачем нужно?

B-дерево - это сбалансированное, сильно ветвистое дерево. Часто используется для хранения данных во внешней памяти

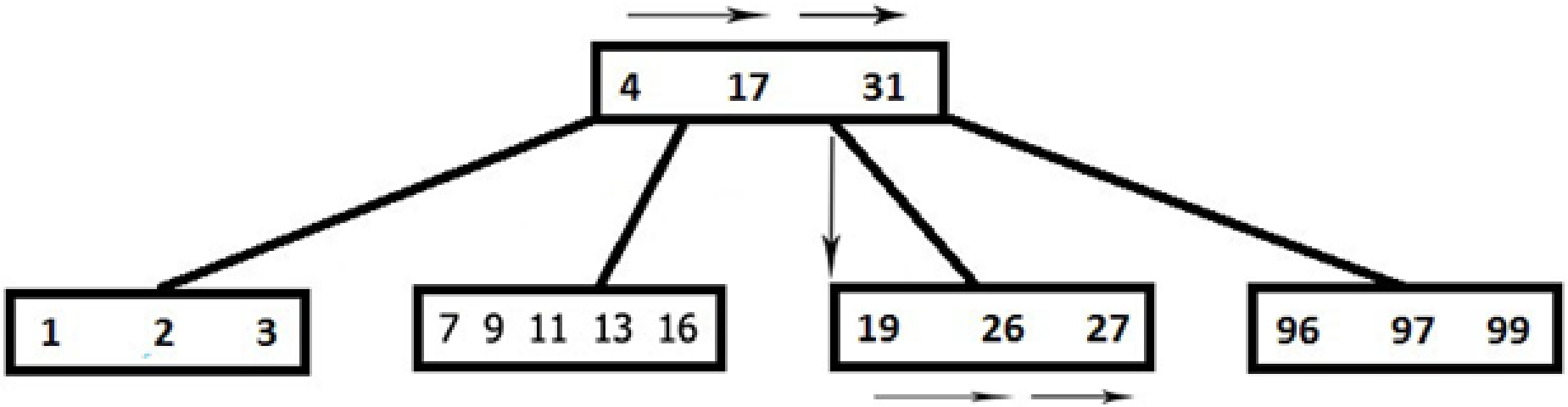
- Оптимизация для случаев деревьев, которые не помещаются в памяти. Блочная структура позволяет сократить обращения к диску - прочитать блок можно за одну операцию чтения.
- БОльшие возможности для кэширования

**Интересно:** $t$ обычно принимает значения от 50 до 2000.

# B-дерево

## Пример

На этом примере $t = 3$. Рассматриваем вершины от $t - 1$ до $2t - 1$, то есть от 2 до 5.



# B-дерево

## Определение

- Каждый узел, кроме корня, содержит не менее t-1 ключей, и каждый внутренний узел имеет по меньшей мере t дочерних узлов. Если дерево не является пустым, корень должен содержать как минимум один ключ.
- Каждый узел, кроме корня, содержит не более 2t-1 ключей и не более чем 2t сыновей во внутренних узлах.
- Корень содержит от 1 до 2t-1 ключей, если дерево не пусто и от 2 до 2t детей при высоте большей 0.
- Каждый узел дерева, кроме листьев, содержащий ключи $k_1, ..., k_n$, имеет n+1 сына. i-й сын содержит ключи из отрезка $[k_{i1}; k_i]$, $k_0 = \infty$, $k_{n+1} = \infty$.
- Ключи в каждом узле упорядочены по неубыванию.
- Все листья находятся на одном уровне.

# B-дерево

## Вставка

Дерево потомков узла - поддерево, состоящее из этого узла и его потомков.
**Добавление в дерево потомков узла X**:

- Если x — не лист: Определяем интервал, где должен находиться K. Пусть y — соответствующий потомок.
  1. Определяем интервал, где должен находиться K. Пусть y — соответствующий потомок.
  2. Рекурсивно добавляем K к дереву потомков y.
  3. Если узел y полон, то есть содержит $2t - 1$ ключей, расщепляем его на два. Узел $y_1$ получает первые $t - 1$ из ключей y и первые $t$ его потомков, а узел $y_2$ — последние $t - 1$ из ключей y и последние $t$ его потомков. Медианный из ключей узла $y$ попадает в узел $x$, а указатель на $y$ в узле $x$ заменяется указателями на узлы $y_1$ и $y_2$.
- Если x — лист, просто добавляем туда ключ K.

# B-дерево

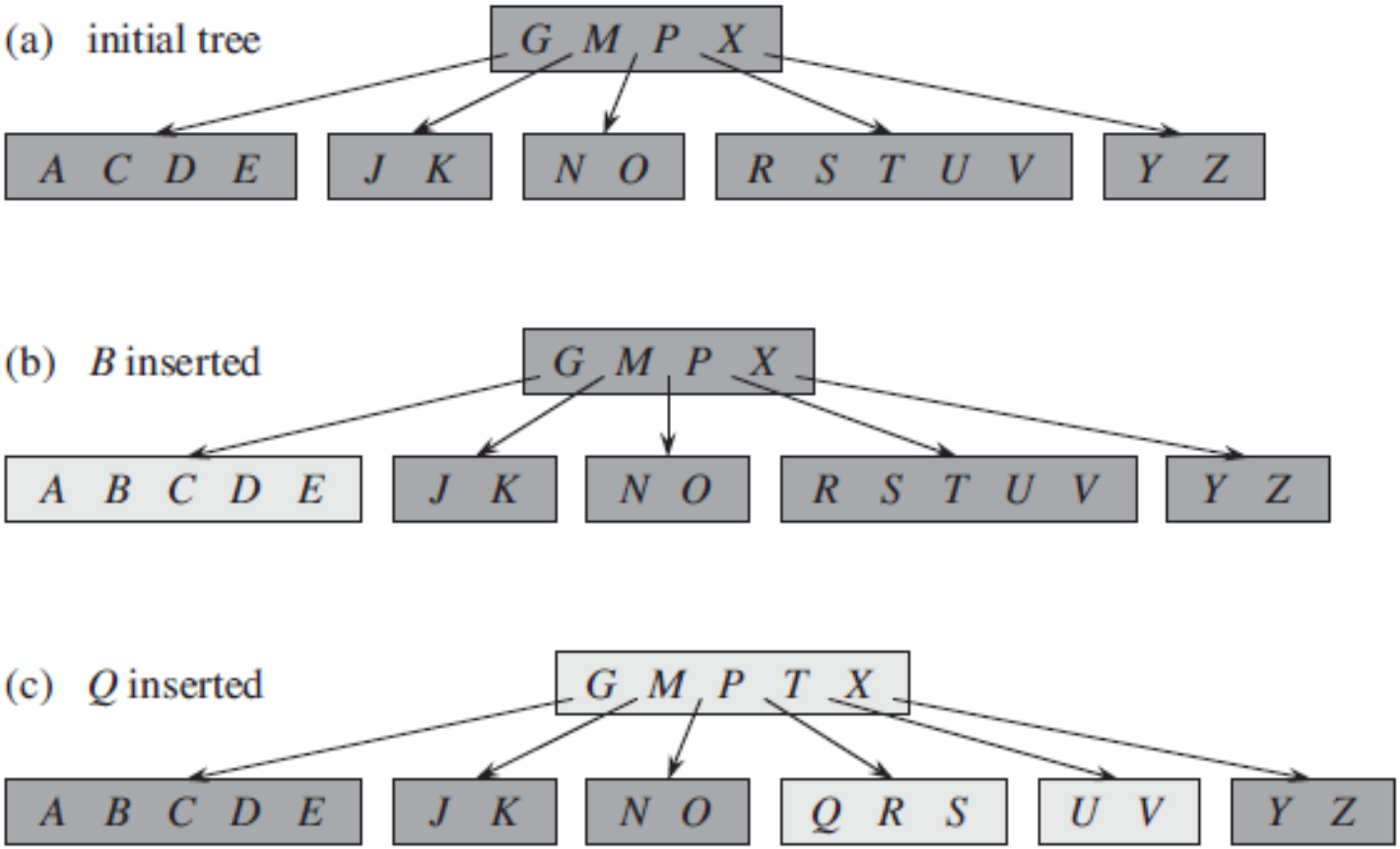
## Вставка

Добавление ключа K **ко всему дереву**. Буквой $R$ обозначается корневой узел.

- Добавим $K$ к дереву потомков $R$.
- Если R содержит теперь $2t - 1$ ключей, расщепляем его на два. Узел $R_1$ получает первые $t - 1$ из ключей $R$ и первые $t$ его потомков, а узел $R_2$ — последние $t - 1$ из ключей R и последние $t$ его потомков. Медианный из ключей узла R попадает во вновь созданный узел, который становится корневым. Узлы $R_1$ и $R_2$ становятся его потомками.

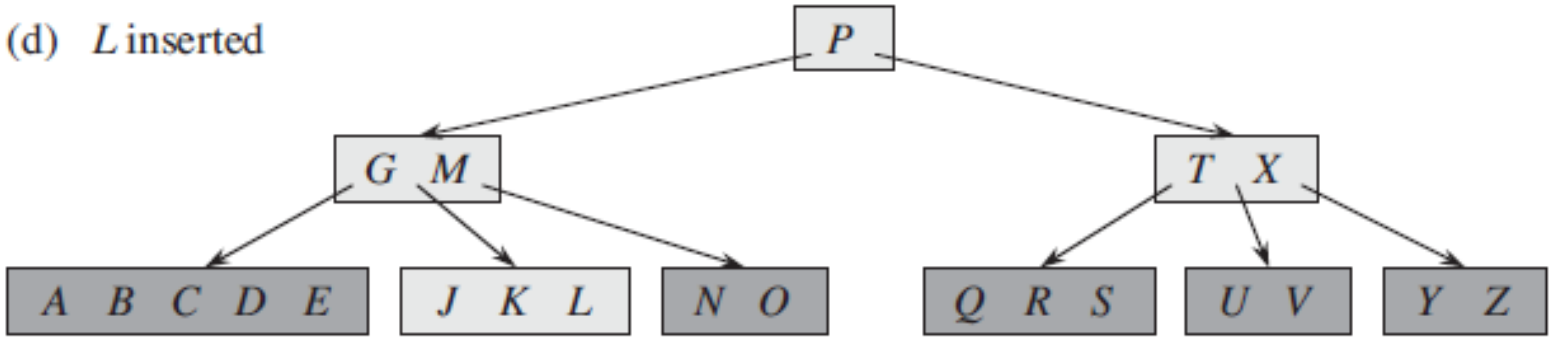
# B-дерево

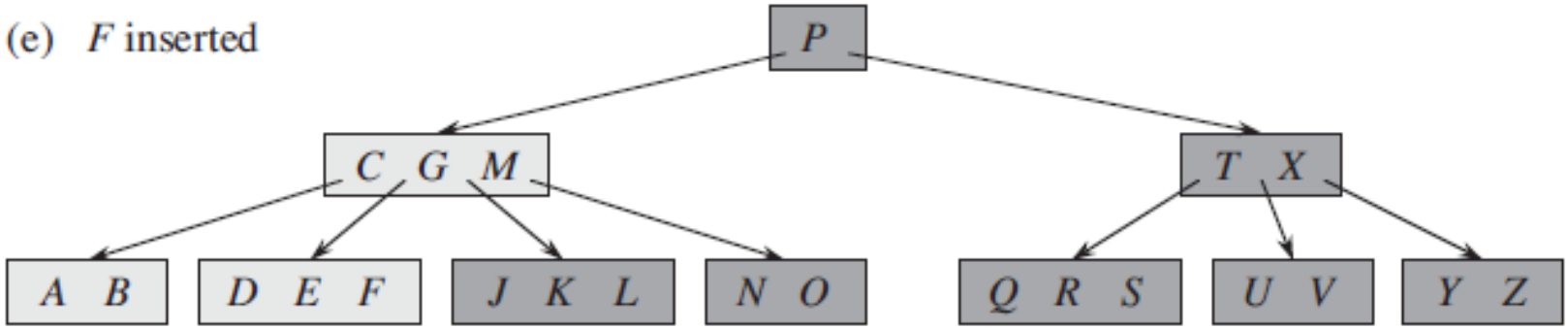
## Вставка



# B-дерево

## Вставка



# B-дерево

## Удаление из листа

- **Корень одновременно является листом**
  То есть в дереве всего один узел, мы просто удаляем ключ из этого узла.
- **Иначе**
  Сначала находим узел, содержащий ключ, запоминая путь к нему. Пусть этот узел — x.

# B-дерево

## Удаление из листа

**x — лист**

Удаляем оттуда ключ. Если в узле $x$ осталось не меньше $t-1$ ключей, мы на этом останавливаемся. Иначе мы смотрим на количество ключей в двух соседних узлах-братьях.

- **Если соседний правый узел есть, и в нём не менее $t$ ключей** мы добавляем в $x$ ключ-разделитель между ним и соседним правым узлом, а на место этого ключа ставим первый ключ соседнего правого узла, после чего останавливаемся.
- **Иначе, но есть соседний левый узел, и в нём не менее $t$ ключей** мы добавляем в $x$ ключ-разделитель между ним и соседним левым узлом, а на место этого ключа ставим последний ключ соседнего левого узла, после чего останавливаемся.
- **Если и с левым ключом не получилось**
  Мы объединяем узел $x$ с соседним левым или правым узлом, и в объединённый узел перемещаем ключ, до этого разделявший эти два узла.

# B-дерево

## Удаление из листа

В последнем случае при объединении в родительском узле может остаться только $t-2$ ключей. Тогда, если это не корень, мы выполняем аналогичную процедуру с ним. Если мы в результате дошли до корня, и в нём осталось от 1 до $t-1$ ключей, делать ничего не надо, потому что корень может иметь и меньше $t-1$ ключей. Если же в корне не осталось ни одного ключа, исключаем корневой узел, а его единственный потомок делаем новым корнем дерева.

# B-дерево

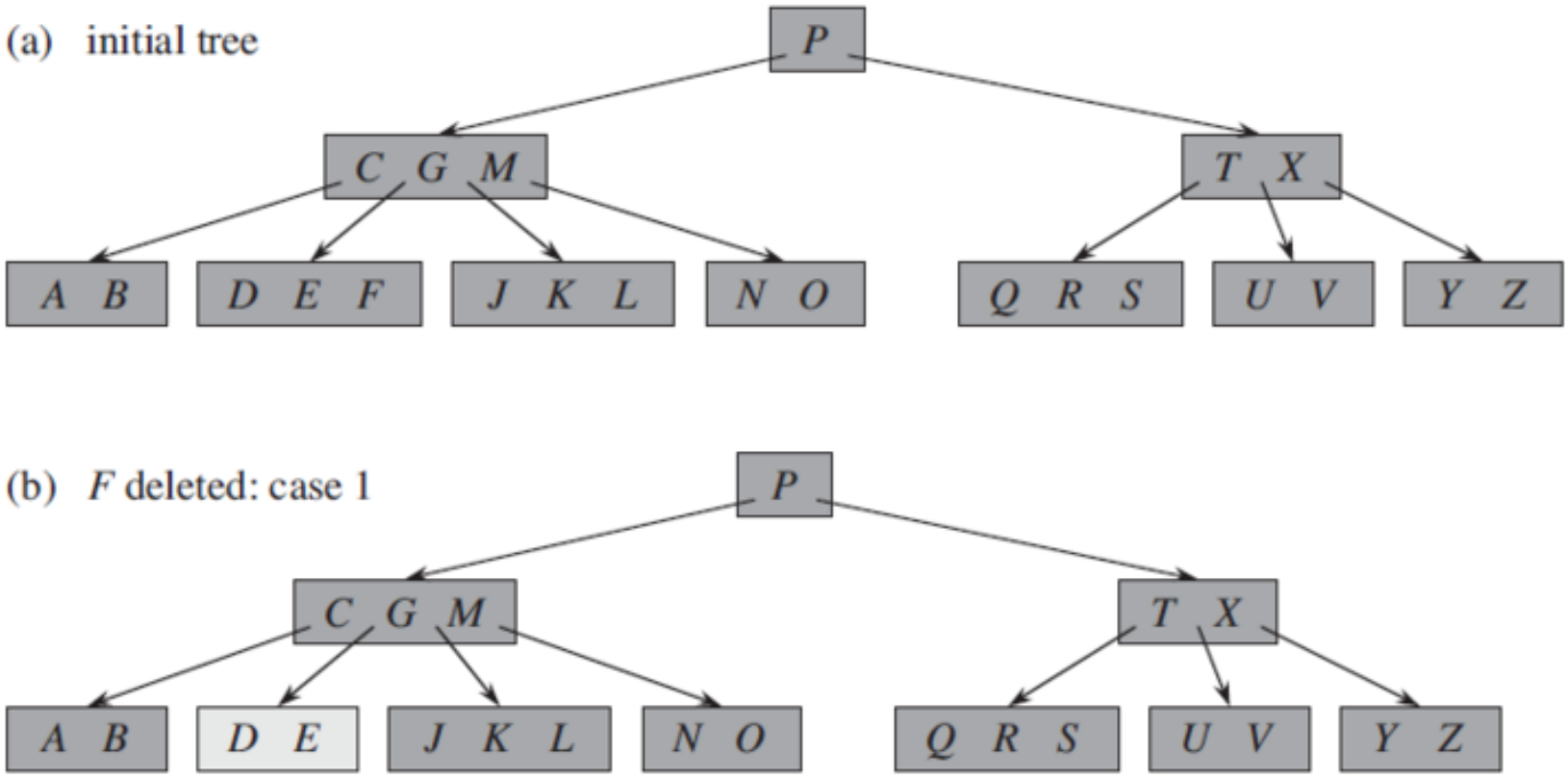
## Удаление из листа

**x — не лист**

а $K$ — его $i$-й ключ, удаляем самый правый ключ из поддерева потомков $i$-го потомка $x$, или, наоборот, самый левый ключ из поддерева потомков $i+1$-го потомка $x$. После этого заменяем ключ K удалённым ключом. Удаление ключа происходит так, как описано в предыдущем абзаце.

# B-дерево

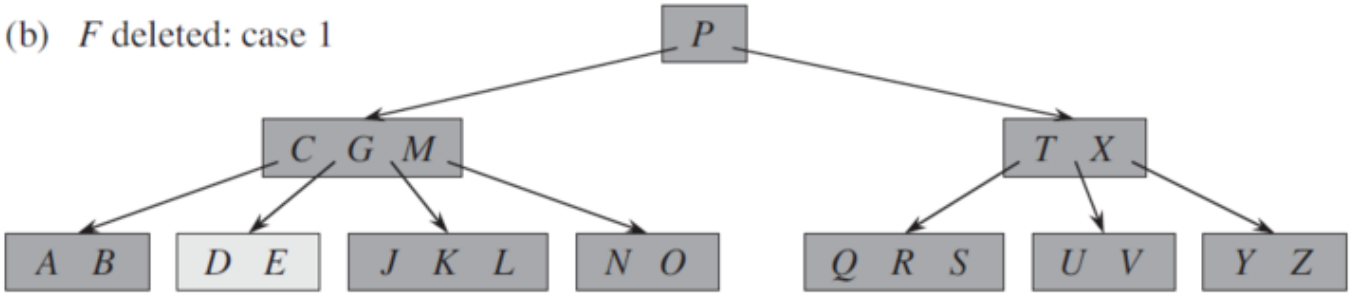
## Удаление из листа



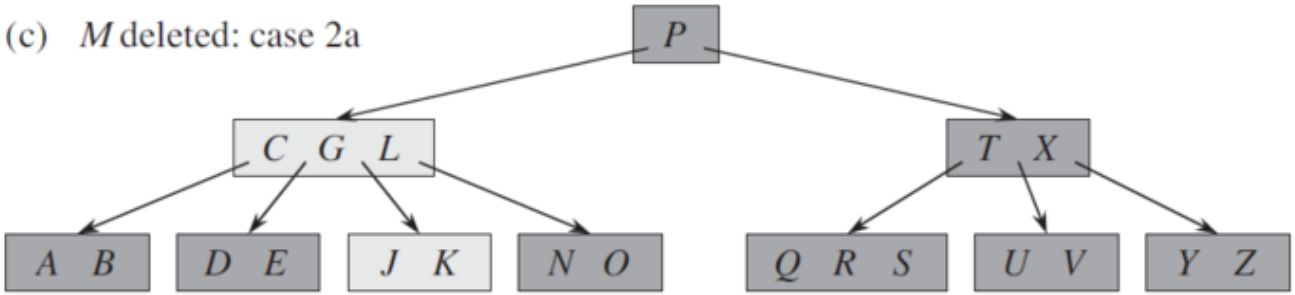
# B-дерево

## Удаление из узла

(b) *F* deleted: case 1




(c) *M* deleted: case 2a




(d) *G* deleted: case 2c