

## 2-3 Дерево

Представляет собой структуру данных, которая является сбалансированным деревом поиска, удовлетворяющее двум условиям:

- Все листья будут на одной глубине, значения будут храниться в листьях
- У всех вершин число исходящих ребер  $\deg_v \in \{0, 2, 3\}$

Также мы в каждом поддереве будем хранить максимум, а детей будем хранить упорядоченными по величине максимума.

**Операции на дереве: поиск** Спуск будем делать рекурсивно. Поскольку мы хранили максимум, то мы среди детей находим первое число, большее  $x$ , спускаемся в соответствующего сына.

**Операции на дереве: добавление** Если степень предка после добавления стала равна 4, то создадим два сына размеров 2 и 2, и рекурсивно рассмотрим отца. Если дошли до корня, то создадим новый корень степени 2.

**Операции на дереве: удаление** Рассмотрим ситуации, которые могли возникать при удалении. Заметим, что у вершины есть отец, дедушка, дядя, брат (предок, другой сын прапредка, прапредок, другой сын предка).

Если у вершины было 2 брата, то после ее удаления ничего менять не надо.

Если у вершины был 1 брат, то нарушается инвариант на степени. Посмотрим на детей дяди. Если их было 3, то можно перераспределить  $3 + 1$  как  $2 + 2$ , и инвариант не нарушится. Если же там было 2 ребенка, то склеимся в одну вершину степени 3, и уменьшим степень предка на 1. Рекурсивно запустим процесс балансировки от него.

**Оптимальное дерево поиска** Обозначим число детей за  $d$ . Тогда операции работают за  $d \cdot \log_d n$ . Найдем точку минимума:  $(d \cdot \log_d n)' = \ln n \cdot \frac{\ln d - 1}{\ln^2 d}$ . Нулевой корень производной при  $d = e$ . Таким образом, 2-3 дерево достаточно близко к оптимальному.

**$B+$  дерево** Зафиксируем константу  $T$ . Все значения опять храним в листьях. У вершин (кроме корня) степень от  $T$  до  $2 \cdot T - 1$ .  $2 \leq \deg_{root} \leq 2 \cdot T - 1$

Обычно  $T$  делают достаточно большим, чтобы дерево работало во внешней памяти.

**Операции: вставка** Если у вершины степень стала  $2T$ , то мы можем разделить ее на две вершины, подвесить их к предку, и рекурсивно запустить процесс у предка.

**Операции: удаление** Плохая ситуация при удалении — степень предка стала равна  $T - 1$ . Посмотрим на соседних братьев. Если один из них по степени больше  $T$ , то мы можем позаимствовать у него крайнего сына, чем починим свою степень. Если же у обоих братьев степень  $T$ , то мы можем смерджить себя с братом, после чего рекурсивно продолжить процесс в предке, потому что у предка степень уменьшилась на 1.

**Почему  $B+$ , а не  $B$ ?** В начале стоит сказать, что  $B$ -дерево обладает схожей структурой, но разрешает хранение значений не только в листьях. Его глубина не более чем на 1 меньше, чем у соответствующего ему  $B+$ -дерева. Но при этом элементы  $B+$ -дерева можно поддерживать в двусвязном списке, что удобно, а также можно итерироваться во внешней памяти.