

Научимся в детерминированный, так сказать, декартач.

2-3 дерево

В этом дереве есть вершины степени 2 и вершины степени 3. Элементы записываются только в листьях.

Ряд условий 2-3 дерева:

1) Все листья на одной и той же глубине. 2) Степень каждой внутренней вершины либо 2, либо 3.

В промежуточных вершинах храним ключи, которыми можно разделять вершины-сыновей.

У вершины с 3 детьми, два ключа - максимум поддеревьев двух первых детей, с 2 детьми - максимум в первом (а ещё храним максимум во всём поддереве).

Смотря на ключ, мы можем понять в какое из двух (или трёх) деревьев идти.

Вставка в 2-3 дерево.

Найдём место, в котором элемент должен быть и просто вставим его. Может случиться, что детей уже было 3, а теперь стало 4. Тогда разобьём нашу вершину (у которой стало 4 сына) и разобьём её на 2 и переподвесим их теперь к её матушке. Ну и продолжаем это процесс пока не дойдём до корня. Если вдруг мы расщепили корень, то создадим новый корень и подвесим к нему наши две вершины. После того, как мы всё сделали, можно запуститься рекурсивно вверх от только что вставленной и идём вверх. Это работает быстро, потому что уровней логарифм.

Удаление из 2-3 дерева. Найдём элемент и тупо удалим его. Если у его предка было три ребёнка - всё хорошо. Предположим, что у вершины было 2 ребёнка, а остался один (трагично, не правда ли?). Теперь начинаются боль и мучения. В общем тут перебор случаев, который можно разобрать кроме того случая, когда у нас только один брат, у которого ровно два сына. Тогда мы переподвешиваем нашего сына к брату и переходим на уровень выше. Если мы упрёмся в корень, то просто возьмём корень и удалим его за ненадобностью.

Всё 2-3 дерево можно провязать двусвязными списками, связывающие вершины на одном уровне.

Почему бы вместо 2-3 дерева не сделать бы 5-11 дерево? Оценим это так. Пусть у всех вершин степень  $d$ . Тогда стоимость операции:

$$\begin{aligned} d \cdot \log_d n &= d \cdot \frac{\ln n}{\ln d} \Rightarrow \\ (d \cdot \log_d n)' &= (d \cdot \frac{\ln n}{\ln d})' \Rightarrow \\ (d \cdot \log_d n)' &= (d \cdot \frac{\ln d - 1}{(\ln d)^2})' \end{aligned}$$

0 производной в  $d = e$ . Значит, 2 и 3 - то хорошие приближения.

$B^+$  - дерево.

Степень каждой вершины от  $t$  до  $2t - 1$ . Степень корня - от 2 до  $2t - 1$ .

Поиск вершины - также.

Такие деревья нужны для алгоритмов во внешней памяти.

Тогда асимптотика -  $O(CPU \cdot (d \log_d n) + HDD \cdot \log_d n)$ . Тогда, хочется брать достаточно большой  $d$ .

Вставка:

Ищем место для вставки и смотрим, куда вставить. Если в какой-то момент стало  $2t$  детей, то сплитим на  $t$  и  $t$  и переподвешиваем.

Удаление: Пусть в вершине теперь стало  $t - 1$  детей (иначе всё хорошо). Если у нас соседний брат - "Большой Брат" (хотя бы  $t + 1$  ребёнок), то всё - мы нашли жертву, которую будем грабить (переподвешиваем одного из сыновей брата к нашей вершине). Если нет, то вдохновившись небезызвестным произведением Кена Кизи, подкидываем брату  $t - 1$  кукушонка. Переходим на уровень выше и продолжаем процесс рекурсивно. Отдельно оговорим корень. Тут проблема только тогда, когда у него осталась 1 вершина, но тогда корень улетает в небытие, и в этом городе появляется новый корень (время для нового Жожо).

А сейчас будет нерекурсивная вставка в  $B$ -дерево. Ослабим ограничения на вершины теперь границы это  $t-1$  и  $2t-1$ . Во время поиска вершины будут разбиваться в момент спуска. Как только вершина может переполниться (её степень  $2t - 1$ ), тогда разобьём нашу вершину на  $t - 1$  и  $t$  и пойдём дальше вниз.

Ещё одна отсечка, от которой плаквится мозг - в каждой вершине нам надо хранить массив максимумов детей. Будем хранить эту вещь бинарным деревом поиска (ДД).