02-hw-CS

vladrus13rus

September 2021

1 Удаление из 2-3 дерева

Происходит все совершенно так же, как и в алгоритме добавления – нам нежелательно удалять разом две вершины рядом, поэтому сделаем как и в добавлении, сначала удалим средний элемент, потом средний из двух разбиений и так далее...

Почему нам нежелательно удалять разом две вершины? Потому что в случае, если мы целиком вырежем детей у отца, то и отец станет не нужен, то есть его нужно будет вырезать тоже. Это приведет к дисбалансу детей, а это нам очень даже не нужно. Тем более у нас иногда происходит сцепление брата удаленной вершины к брату отца, если брат остался один, что явно нам помешает в случае удаления сразу нескольких вершин.

Итого: запросим удаление у медианы, "подождем"два шага, затем у медианы двух половины, и так далее. Иногда будет случаться такое, что мы удаляем разом две вершины из трех или одну вершину из двух из какого то дерева, но поступим там так же, как поступали бы в обычном алгоритме - присоединим ее к соседу. Нам ничего не помешает так сделать.

2 Построение дерева

2.1 Подумаем

Казалось бы, было бы очень круто построить все с головы. Но зачем?

2.2 Костяк решения

Начнем строить все снизу. Массив отсортирован, мы можем просто строить с низа. Давайте воспользуемся такой стратегией:

1. При составлении дерева, либо вершина будет генерировать новую вершину (назовем таковую генерирующей), либо будет присоединяться к уже сгенерированной (назовем таковую присоединяющейся)

2. Попытаемся взять максимальное число вершин с 2 детьми. То есть, мы будем добавлять третьего ребенка только в том случае, если иначе появится вершина с одним ребенком (что, несомненно, запрещено)

3. Введем нумерацию вершин 2-3 дерева. Например, если родитель имеет номер x, то у него будут дети $3 \cdot x$, $3 \cdot x + 1$, $3 \cdot x + 2$ (при надобности). Таким образом, родитель будет знать, где находятся его дети, а дети будут знать у кого они и на каком месте они находятся (то есть, самый правый ли он ребенок) за $O(1)$. Самая верхняя вершина (корень) будет с номером 1.

4. Пусть любая генерирующая вершина будет генерировать своего предка только если она самая левая (таким образом, каждый предок будет создан только один раз)

2.3 Решение

2.3.1 Высота дерева!

Выясним высоту дерева. Зачем? Узнаем позже! Достаточно одного треда, который будет делить количество вершин на 2 (если оно будет нечетное, то не беспокоимся, просто на этом уровне последняя вершина присоединится к двум прешедствующим)

$input = n$ - количество вершин
$output = h$ - высота дерева
$work = O(log^*(n))$ - итерированный логарифм
$span = O(log^*(n))$

2.3.2 Номера листьев!

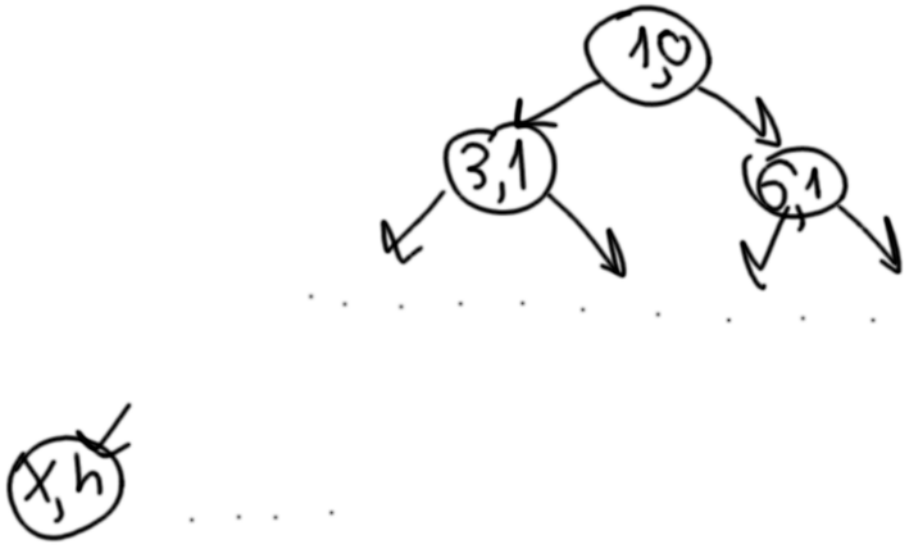

Рисунок 1. Как мы определяем номера

Просто рекурсивно спускаемся форк джойнами. Делимся каждый раз на 2 (за исключением одной вершины, она делится на 3, но там можно в 2 форка, неважно)

Высота будет равна высоте дерева, то есть $O(log(n))$
$work = O(n) + O(n/2) + O(n/4) = O(2n) = O(n)$
$span = O(h) = O(log(n))$

2.3.3 Построение!

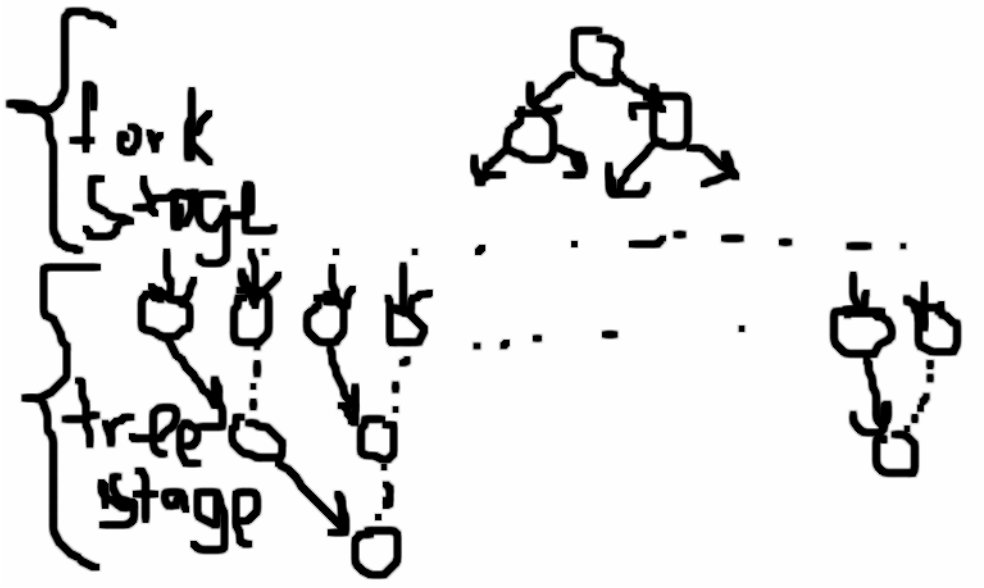

Рисунок 2. При нарисовке этой картинки был нещадно разлит горячий чай на графическое оборудование. Поэтому рисунок кривой, нарисован мышкой

Итак, распишем подробнее мною нарисованное. У нас есть fork stage, в котором мы просто делаем форки по количеству вершин. Каждая вершина уже знает свой номер. Каждая вершина либо создает себе родителя и запускает создание новой вершины уже от нее (если ее номер делится на 3, то есть она самая левая вершинка). Если же это не так, то есть различные варианты (попозже их распишем). Каждый из этих вариантов должен будет занимать $O(1)$ work и span

Теперь рассмотрим поведение тогда, когда мы оказались не самым левым ребенком.

Первый вариант - ждать, пока по номеру родителя инициализируют вершину, и привязаться к ней. Но тогда мы будем возможно довольно долго стоять, если одна очень хитрая левая вершина решит постоять и не выполняться

Рисунок 3. Threads police.
Улыбнуться на камеру - 10 рублей
Поставить главный тред алгоритма в минимальный приоритет, что бы вышел work = $O(nlog(n))$ - 1000 рублей
+0.05 баллов в табличке - бесценно

Второй вариант - как только мы решили, что мы ничего не можем инициализировать - бросить тред и уйти. Оно дальше доработает. После того, как все выполнится, мы просто пройдемся новым fork по всем n вершинам, и, если не нашлось родителя и мы не с номером 1 - просто привязать новую вершину к родителю. Этот маневр будет нам стоить O(n) work и O(log(n)), что не будет портить нам итоговую асимптотику

$work = O(n)$
$span = O(log(n))$

2.3.4 Итоги

$work = O(log^*(n)) + O(n) + O(n) = O(n)$
$span = O(log^*(n)) + O(log(n)) + O(log(n)) = O(log(n))$