2022

AVL деревья

Еремеев Тимур

Саратовский Государственный Университет им. Чернышевского

28 апреля 2022 г.

Введение

AVL-дерево — сбалансированное по высоте бинарное дерево поиска: для каждой его вершины высота её двух поддеревьев различается не более чем на 1.
AVL — аббревиатура, образованная первыми буквами создателей (советских учёных) Г. М. Адельсон-Вельского и Е. М. Ландиса.

Общие св-ва

В AVL-дереве высоты h имеется не меньше F_h узлов, где F_h — число Фибоначчи. Поскольку $F_n = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}} = \frac{\phi^n - (-\phi)^{-n}}{\phi - (-\phi)^{-1}}$, где $\frac{\phi^n - (-\phi)^{-n}}{\phi - (-\phi)^{-1}}$ — золотое сечение, то имеем оценку высоты AVL-дерева $h = Q(\lg(n))$, где n — число узлов. Следует помнить, что $Q(\lg(n))$ — мажоранта, и её можно использовать только для оценки (Например, если в дереве только два узла, значит в дереве два уровня, хотя $\lg(2) = 1$. Для точной оценки глубины дерева следует использовать пользовательскую программу.

Тип Дерева можно описать так:

Балансировка

Относительно AVL-дерева балансировкой вершины называется операция, которая в случае разницы высот левого и правого поддеревьев $= 2$, изменяет связи предок-потомок в поддереве данной вершины так, что разница становится $\leqslant 1$, иначе ничего не меняет. Указанный результат получается вращениями поддерева данной вершины.

Малое левое вращение

Используется 4 типа вращений:

Рис.: Схематическое изображение малого левого вращения

Данное вращение используется тогда, когда (высота b-поддерева; L — высота) $= 2$ и высота $\leqslant$ высота R.

Большое левое вращение

Рис.: Схематическое изображение большого левого вращения

Данное вращение используется тогда, когда (высота b-поддерева; L — высота) $= 2$ и высота c-поддерева $>$ высота R.

Малое правое вращение

Рис.: Схематическое изображение малого правого вращения

Данное вращение используется тогда, когда (высота b-поддерева — высота R) $= 2$ и высота $\leqslant$ высоты L.

Большое правое вращение

Рис.: Схематическое изображение большого правого вращения

Данное вращение используется тогда, когда (высота b-поддерева; R — высота) $= 2$ ивысота c-поддерева $>$ высота L. В каждом случае достаточно просто доказать то, что операция приводит к нужному результату и что полная высота уменьшается не более чем на 1 и не может увеличиться. Из-за условия сбалансированности высота дерева $O(\lg(N))$, где N — количество вершин, поэтому добавление элемента требует $Q(\lg(N))$ операций.

Алгоритм добавления вершины

Показатель сбалансированности в дальнейшем будем интерпретировать как разность между высотой левого и правого поддерева, а алгоритм будет основаться на типе TAVLTree, описанном выше. Непосредственно при вставке (листу) присваивается нулевой баланс. Процесс включения вершины состоит из трех частей:

1. Прохода по пути поиска, пока не убедимся, что ключа в дереве нет.
2. Включения новой вершины в дерево и определения результирующих показателей балансировки.
3. "Отступления"назад по пути поиска и проверки в каждой вершине показателя сбалансированности. Если необходимо - балансировка

Расширим список параметров обычной процедуры вставки параметром-переменной flag, означающим, что высота дерева увеличилась. Предположим, что процесс из левой ветви возвращается к родителю (рекурсия идет назад), тогда возможны три случая: h_1 — высота левого поддерева, h_r- высота правого поддерева Включение вершины в левое поддерево приведет к

1. $h_1 < h_r$: выровняется $h_1 = h_r$. Ничего делать не нужно.
2. $h_1 = h_r$: теперь левое поддерево будет больше на единицу, но балансировка пока не требуется.
3. $h_1 > h_r$: теперь $h_1 - h_r = 2$, — требуется балансировка.

В третьей ситуации требуется определить балансировку левого поддерева. Если левое поддерево этой вершины ($Tree \cdot left \cdot left$) выше правого ($Tree \cdot left \cdot right$), то требуется большое правое вращение, иначе хватит малого правого. Аналогичные (симметричные) рассуждения можно привести и для включение в правое поддерево. Процедура вставки, предложенная Н.Виртом

Алгоритм удаления вершин

Для простоты опишем рекурсивный алгоритм удаления. Если вершина — лист, то удалим её и вызовем балансировку всех её предков в порядке от родителя к корню. Иначе найдём самую близкую по значению вершину в поддереве наибольшей высоты (правом или левом) и переместим её на место удаляемой вершины, при этом вызвав процедуру её удаления.

Докажем, что данный алгоритм сохраняет балансировку. Для этого докажем по индукции по высоте дерева, что после удаления некоторой вершины из дерева и последующей балансировки высота дерева уменьшается не более, чем на 1. База индукции: Для листа очевидно верно. Шаг индукции: Либо условие балансированности в корне (после удаления корень может изменится) не нарушилось, тогда высота данного дерева не изменилась, либо уменьшилось строго меньшее из поддеревьев высота до балансировки не изменилась после уменьшится не более чем на 1.

Очевидно, в результате указанных действий процедура удаления вызывается не более 3 раз, так как у вершины, удаляемой по 2-му вызову, нет одного из поддеревьев. Но поиск ближайшего каждый раз требует $O(N)$ операций, отсюда видна очевидная оптимизация: поиск ближайшей вершины производится по краю поддерева. Отсюда количество действий $O(\lg(N))$.

Расстановка балансов при удалении

Как уже говорилось, если удаляемая вершина-"лист, то она удаляется, и обратный обход дерева происходит от родителя удалённого листа. Если не лист — ей находится «замена», и обратный обход дерева происходит от родителя «замены». Непосредственно после удаления элемента — «замена» получает баланс удаляемого узла.

При обратном обходе: если при переходе к родителю пришли слева — баланс увеличивается на 1, если же пришли справа — уменьшается на 1.

Это делается до тех пор, пока при изменении баланса он не станет равным -1 или 1 (обратите внимание на различие с вставкой элемента!): в данном случае такое изменение баланса будет гласить о неизменной дельта-высоте поддеревьев. Повороты происходят по тем же правилам, что и при вставке.

Расстановка при одинарном повороте

Обозначим:
«Current» — узел, баланс которого равен -2 или 2: то есть тот, который нужно повернуть (на схеме - элемент a)
«Pivot» — ось вращения. $+2$: левый сын Current'a, -2: правый сын Current'a (на схеме — элемент b)
Если поворот осуществляется при вставке элемента, то баланс Pivot'a равен либо 1, либо -1. В таком случае после поворота балансы обоих устанавливаются равными 0.
При удалении всё иначе: баланс Pivot'a может стать равным 0 (в этом легко убедиться).

Приведём сводную таблицу зависимости финальных балансов от направления поворота и исходного баланса узла Pivot:

[ht]

Направление поворота	Old Pivot Balance	New Current Balance	New Pivot Balance
Левый или правый	-1 или $+1$	0	0
Правый	0	-1	$+1$
Левый	0	$+1$	-1

Расстановка балансов при двойном повороте

Pivot и Current — те же самые, но добавляется третий участник поворота. Обозначим его за «Bottom»: это (при двойном правом повороте) левый сын Pivot'а, а при двойном левом — правый сын Pivot'а.
При данном повороте — Bottom в результате всегда приобретает баланс 0, но от его исходного баланса зависит расстановка балансов для Pivot и Current.

Приведём сводную таблицу зависимости финальных балансов от направления поворота и исходного баланса узла Bottom:

[ht]

Направление поворота	Old Botom Balance	New Current Balance	New Pivot Balance
Левый или правый	0	0	0
Правый	$+1$	0	-1
Правый	-1	$+1$	0
Левый	$+1$	-1	0
Левый	-1	0	$+1$

Г.М.Адельсон-Вельский и Е.М.Ландис доказали теорему, согласно которой высота AVL-дерева с N внутренними вершинами заключена между $log2(N+1)$ и $1.4404 * log2(N+2) - 0.328$, то есть высота AVL-дерева никогда не превысит высоту идеально сбалансированного дерева более,чем на 45%. Для больших N имеет место оценка $1.04 * log2(N)$. Таким образом, выполнение основных операций $1-3$ требует порядка $log2(N)$ сравнений. Экспериментально выяснено, что одна балансировка приходится на каждые два включения и на каждые пять исключений.

Источнки