Scapegoat tree

1 Введение

Scapegoat деревья – подвид сбалансированных бинарных деревьев, выполняющий операцию поиска за $O(\log n)$, а вставку и удаление за амортизированный $O(\log n)$. Дополнительным преимуществом деревьев является то, что мы не храним лишние данные в вершинах: только указатели на детей и значение.

2 Обозначения

- $d(x)$ – глубина вершины x
- $h(T)$ – высота дерева T, $h(x)$ – высота поддерева x
- $\text{size}(x)$ – размер поддерева вершины x, $\text{size}(T)$ - размер дерева T
- Пусть α – фиксированное число: $\frac{1}{2} < \alpha < 1$
- Вершина x называется α-весо-сбалансированной, если

$$\text{size}(x.\text{left}) \leq \alpha \cdot \text{size}(x) \text{ и } \text{size}(x.\text{right}) \leq \alpha \cdot \text{size}(x)$$

- Дерево T называется α-весо-сбалансированным, если все его вершины α-весо-сбалансированные
- Пусть $h_\alpha(n) = \left\lfloor \log_{\{\frac{1}{\alpha}\}}(n) \right\rfloor$. $h_\alpha(T) = h_\alpha(\text{size}(T))$.
  Дерево называется α-высото-сбалансированным, если $h(T) \leq h_\alpha(T)$.

  Интуиция: дерево является α-высото-сбалансированным, если его высота не больше максимально возможной высоты α-весо-сбалансированного дерева такого же размера. (такое дерево выглядит, например, так: будем класть в правого ребенка максимально возможное количество вершин, чтобы еще соблюдалась весо-сбалансированность родителя. Так мы получим максимальную допустимую высоту). Кроме этого нетрудно заметить, что весо-сбалансированное дерево является и высото-сбалансированным.
  (NB: в обратную сторону неверно).
- Вершина x называется глубокой, если $d(x) > h_\alpha(T)$. То есть глубокой вершиной называется та, которая нарушает α-высото-сбалансированность дерева.
- Дерево T называется почти- α-высото-сбалансированным, если $h(T) \leq h_\alpha(T) + 1$.

3 Операции

3.1 Поиск

Работает как в обычном бинарном дереве поиска: будем спускаться от корня в нужное поддерево, пока не найдем вершину или не придем в лист.

3.2 Вставка

Спускаемся от корня (как при поиске), пока не придем в лист. Подвешиваем вершину x к этому листу. Если x не глубокая вершина, то заканчиваем работу. Иначе перебалансировываем часть дерева.

3.3 Перебалансировка

Предположим, мы вставили вершину x и она оказалась глубокой. Хотим найти самого глубокого не-весо-сбалансированного предка x (такая вершина называется scapegoat). Идем от x к корню (этот путь можно запомнить при вставке и сейчас идти в обратном направлении), по пути проверяя весо-сбалансированность текущей вершины (за $O(\text{size}(v))$). Если мы нашли не-весо-сбалансированную вершину, полностью перестраиваем ее поддерево в сбалансированное бинарное (= $\frac{1}{2}$-весо-сбалансированное) дерево поиска и завершаем работу. Такое перестраивание можно сделать за линейное время: обходим дерево слева направо, кладём элементы в массив. Корнем делаем центральный элемент, делим массив на 2 части, рекурсивно вызываемся от половинок.

3.4 Удаление

Введем переменную $\text{maxSize}(T)$ – максимальный размер дерева с момента ее перестройки при удалении. Будем спускаться от корня, пока не найдем вершину с ключом, который хотим удалить. Удалим ее как в бинарном дереве поиска. Если $\text{size}(T) < \alpha * \text{maxSize}(T)$, то перестроим дерево полностью и обновим $\text{maxSize}(T) = \text{size}(T)$. Перестраивать будем в сбалансированное бинарное дерево поиска.

3.5 Замечание

При вставке может перебалансировываться часть дерева, в то время как при удалении всегда происходит полная перебалансировка.

4 Корректность

4.1 Вставка

Теорема 4.1. Scapegoat вершина на пути от глубокой до корня обязательно существует

Proof. От противного: пусть это не так. Тогда все предки x весо-сбалансированные.
Получается
$\text{size}(x) \leq \alpha * \text{size}(x.\text{prnt}) \leq \alpha^2 * \text{size}(x.\text{prnt.prnt}) \leq ... \leq \alpha^{d(x)} * \text{size}(\text{root}) = \alpha^{d(x)} * \text{size}(T)$
$\text{size(x)} \leq \alpha^{d(x)} * \text{size}(T)$
$\text{d(x)} \leq \log_{\frac{1}{\alpha}} \text{size}(T)$. Однако по условию x глубокая. Противоречие. □

Теорема 4.2. Принимая высото-сбалансированное дерево, вставка сохраняет его высото-сбалансированность

Для доказательства этого утверждения рассмотрим несколько вспомогательных фактов

Лемма 4.3. Перебалансировка поддереве не увеличивает его высоту

Proof. Перебалансировав дерево, мы получим бинарное сбалансированное дерево, а у него высота минимальная. □

Лемма 4.4. Если корень дерева T не весо-сбалансирован и вставляемая вершина x - единственная на глубине $h(T) + 1$, то перебалансировка уменьшит высоту дерева.

Proof. Рассмотрим T_l - легкое поддерево корня.
Пусть $T_l' = T_l \backslash x$. Оно не может быть полным деревом высоты $h(T)$.
От противного: T_l' - полное дерево высоты $h(T)$. Тогда T_h - также полное дерево высоты $h(T)$(так как оно тяжелее) + возможно x на глубине $h(T) + 1$. В таком случае размеры левого и правого сына корня отличаются на 1 вершину x, и из этого следует, что корень весо-сбалансирован. Противоречие.

Из этого следует, что при перебалансировке в легком поддереве будет место для x и на глубине $h(T)+1$ не будет вершин.

Данное утверждение применяем для scapegoat поддерева при вставке. □

Следствие 4.5. Если поддерживать только операции вставки и поиска, то выполняется инвариант, что дерево всегда высото-сбалансированное.

4.2 Удаление

Теорема 4.6. При операциях удаления и вставки дерево всегда почти-высото-сбалансированное

Для доказательства этого утверждения рассмотрим несколько вспомогательных фактов

Лемма 4.7. Рассмотрим произвольное дерево T. Пусть мы вставили элемент x и получили T'. Тогда $h(T') \leq \max(h_\alpha(T'), h(T))$

Proof. Рассмотрим несколько случаев:
- вставка не привела к перебалансировке → глубина $x \leq h_\alpha(T)$
- вставка привела к перебалансировке и x единственная на такой глубине → по лемме 4.4 перебалансировка уменьшит высоту T' до $h(T)$
- вставка привела к перебалансировку и x не единственная вершина на такой глубине → по лемме 4.3, перебалансировка не увеличивает высоту. Значит, $h(T') \leq h(T)$

□

Лемма 4.8. Если $h_\alpha(T)$ не меняется во время последовательности вставок и удалений, то $\max(h_\alpha(T), h(T))$ не выросло за это время.

Proof. Отдельно рассмотрим вставку и удаление:
- удаление не увеличивает высоту дерева
- для вставки по предыдущей лемме знаем $h(T') \leq \max(h_\alpha(T'), h(T)) = \max(h_\alpha(T'), h(T')) = \max(h_\alpha(T), h(T'))$ по условию.

□

Лемма 4.9. Пусть $T' = \text{Insert}(x, T)$. Если T было почти-высото-сбалансированным, но не высото-сбалансированным и $h_\alpha(T') = h_{\alpha(T)} + 1$, то T' – высото-сбалансированное.

Proof.
- По определению высото-сбалансированностей $h(T) = h_\alpha(T) + 1$
- По условию $h_\alpha(T) + 1 = h_\alpha(T')$
- Получили $h(T) = h_\alpha(T')$

□

Следствие 4.10. Доказательство инварианта

Рассмотрим последовательность вставок и удалений до первой операции удаления, приведшей к перестройке дерева.
- Рассмотрим операции, не меняющие $h_\alpha(T)$:
  - удаление не увеличивает высоту и по условию не меняет $h_\alpha(T)$. Следовательно, после него высото-сбалансированное дерево им и остается
  - вставка сохраняет высотосбалансированность дерева (по пункту 4.3 $h(T)$ в худшем случае дорастет до $h_\alpha(T)$).
- Рассмотрим операции, меняющие $h_\alpha(T)$:

- вставка оставляет дерево высото-сбалансированным, если оно им было. Если же дерево было почти-высото-сбалансированным, то по пункту 4.5, вставка сделает его высото-сбалансированным.
- заметим, что при удалении $h_\alpha(T)$ мог только уменьшиться. Кроме того, такое удаление могло быть лишь одно в последовательности: если бы их было два, то $h_\alpha(T)$ уменьшился бы хотя бы в $\alpha + \varepsilon$ раз (см определение $h_\alpha(T)$), а тогда бы выполнилось условие перестройки при удалении($\text{size}(T) < \alpha \cdot \text{maxSize}(T)$). Значит, такая операция была всего одна и на вход она получила высото-сбалансированное дерево $\rightarrow$уменьшив $h_\alpha(T)$ на 1, получим в худшем случае почти-высото-сбалансированное дерево.

5 Асимптотика

5.1 Поиск

По доказательству выше дерево почти-высото-сбалансированное, следовательно $h(T) \leq h_{\alpha(T)} + 1$. Таким образом оценили сверху высоту логарифмом. Тогда асимптотика поиска составляет $O\log(\text{size } T)$.

5.2 Вставка

Пусть $\triangle x = |\text{size}(x.\text{left}) - \text{size}(x.\text{right})|$
Потенциал$(x) = 0$, если $\triangle x < 2$ и $\triangle x$ иначе
Тогда потенциал $\frac{1}{2}$-весо-сбалансированного дерева $= 0$
Потенциал scapegoat или любой не весо-сбалансированно вершины $x = \theta(\text{size}(x))$

$(\triangle x = (\alpha + \varepsilon) \cdot \text{size}(x) - (1 - \alpha - \varepsilon) \cdot \text{size}(x) = (2 \cdot \alpha + 2 \cdot \varepsilon - 1) \cdot \text{size}(x) \geq 2\varepsilon * \text{size}(x))$

- Таким образом, потенциал scapegoat-вершины до перебалансировки $= \theta(\text{size}(x))$, а после $= 0$.
- Заметим, что на поиск scapegoat вершины мы тратим $O(\text{size}(x))$ (каждую вершину мы спросим о ее размере один раз), на перебалансировку также уходит $O(\text{size}(x))$ (обходим поддерево scapegoat от самых левых к самым правым, записываем в массив. В массиве делаем середину корнем, от двух половин вызываемся рекурсивно).
- Оплатим поиск scapegoata и перебалансировку этим потенциалом - получаем амортизированно $O(1)$ на операцию.
- Остальные операции, касающиеся вставки (поиск позиции, куда вставим), всегда выполняются за $O\log(\text{size}(T))$.

Итоговая асимптотика: амортизированный $O\log(\text{size}(T))$.

5.3 Удаление

Заметим, что перестройка от удаления происходит не чаще чем 1 в $\alpha \cdot \text{size}(T)$ раз. На перестройку дерева уходит $O(\text{size}(T))$ времени. Получается амортизированно $O(1)$ на операцию. Остальные операции, касающиеся удаления, всегда выполняются за $O\log(\text{size}(T))$.

Итоговая асимптотика: амортизированный $O\log(\text{size}(T))$.