Worst-Case Optimal Priority Queues

Е.Р. Лебедева

1 Введение

Существует миллион реализаций приоритетных очередей, разберем одну из самых оптимальных. Очередь реализует все базовые операции, а именно:

1) **FindMin** (нахождение минимума) $O(1)$
2) **Insert** (вставка) $O(1)$
3) **Decrease** (уменьшение) $O(1)$
4) **Meld** (слияние) $O(1)$
5) **DeleteMin** (удаление минимума) $O(\lg n)$
6) **Delete** (удаление) $O(\lg n)$

Можно заметить, что почти все операции константные, очень хорошая асимптотика. Изначально такого успеха достиг Brodal, однако его реализация была слишком сложна. Авторы текущей структуры упростили очередь Бродала, хоть и оставили многие его идеи.

2 Устройство

Структура состоит из двух деревьев T_1 и T_2. В корне первого хранится минимум. T_2 может быть пустым, а если нет, то в его корне следующий по минимальности элемент. Каждый узел содержит:

1) **Значение:** Обычное число.
2) **Ранг:** Ранг узла на один больше, чем ранг его последнего ребенка. Вершина без детей имеет ранг 0.
3) **Указатель на правого соседа:** Если такового нет, указатель на родителя. Порядок вершин определяется рангом, справа наибольший ранг.
4) **Указатель на левого соседа**

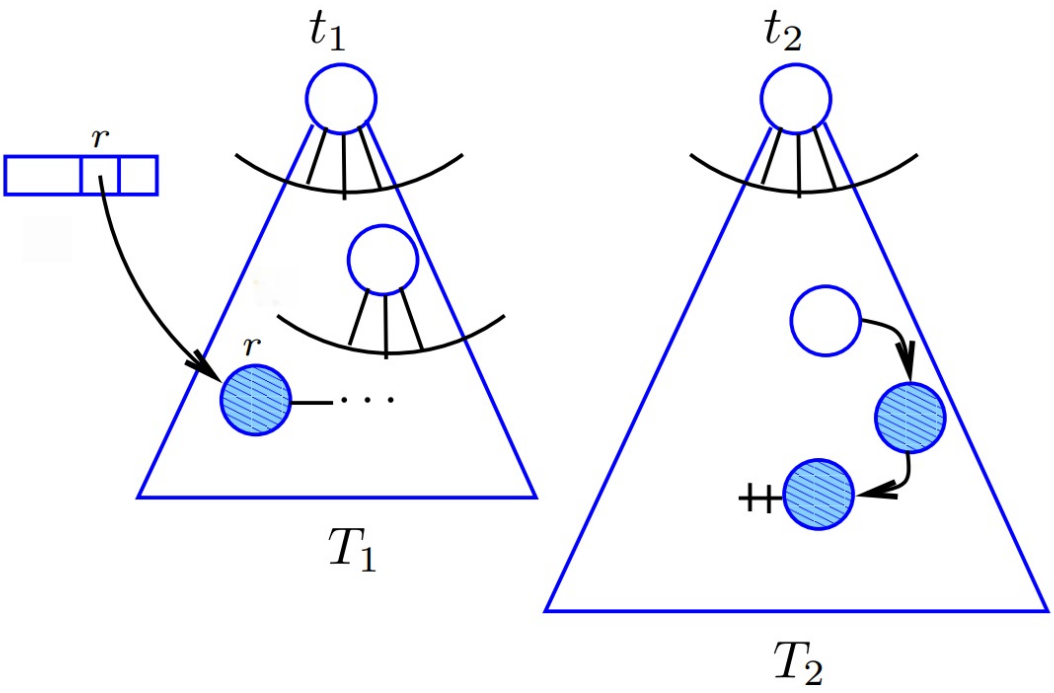
5) **Указатель на последнего ребенка:** Дети вершины хранятся в двусвязном списке, для этого нужны указатель на соседей, а крайний правый узел этого списка ведет к родителю.

6) **Ранговая последовательность:** Чтобы отслеживать количество детей каждого ранга, используется расширенный двоичный счетчик.

7) **Указатель на начало списка нарушений:** Можно догадаться, что это тоже двусвязный список. Конкретнее что это и с чем едят рассмотрим чуть позже.

8) **Указатель на преемника в списке нарушений**

9) **Указатель на предшественника в списке нарушений**



Поговорим про структуру нарушений. Мы хотим, чтобы значения родительских вершин были меньше значений их детей, если вершина может нарушать это правило, она заносится в список нарушений. В основном нарушения активные, но если ранг нарушения больше размера массива нарушений на момент его возникновения, он заносится в список неактивных, такие есть только у t_1. В список нарушений конкретного узла мы заносим те, которые возникли пока текущий узел держал минимум своей приоритетной очереди. Для подсчета количества нарушений каждого ранга используется двоичный счетчик. Так же записи этого счетчика, которые хотя бы два, соединены двусвязным списком. Возникающие нарушения стараются убирать при любой возможности, а именно когда нарушений одного ранга становится хотя бы 3. Механизм сокращения нарушения достаточно слложен из-за большой вариативности, но выполняется быстро. В итоге размер массива нарушений не больше $O(\lg n)$.

Что такое расширенный регулярный счетчик? Это по факту массив чисел от 0 до 3, на который накладываются условия: Между двумя 3 должна быть 0 или 1, Между

двумя 0 должна быть 2 или 3. Значения счетчика хранят количество нарушений каждого ранга. Чтобы поддерживать условия на счетчик, выполняются сокращения нарушений. Такое устройство обеспечивает быстрые операции изменения элементов счетчика.

Все устройство нужно чтобы поддерживать три инварианта:

1) Минимум хранится в t_1.

2) Второе наименьшее значение хранится либо в t_2, либо в одном из дочерних узлов t_1, либо в одном из узлов нарушений, связанных с t_1.

3) Для очереди размера n размер списка нарушений не превышает $O(\lg n)$

Сформулируем **Лемму**: Ранг и количество детей любого узла в нашей структуре данных равны $O(\lg n)$, где n - размер очереди приоритетов.
Доказывается она через прикидку что размер поддерева равен примерно числу Фибоначчи.

3 Операции

Разберем, как выполняются операции в этой приоритетной очереди.

1) **FindMin** Исходя из структуры очереди, минимум находится в вершине t_1.

2) **Insert** Делаем новый узел x со значением e. Если e меньше значения в t_1, то ставим x на место t_1, а дерево с корнем в t_1 подвешиваем к x.

3) **Meld** В этой операции участвуют не более четырех деревьев T_1, T_2, T_1' и T_2', с корнями t_1, t_2, t_1' и t_2' соответственно. НУО $value(t_1) \leqslant value(t_1')$. Дерево T_1 становится первым деревом объединенной очереди приоритетов. Массив нарушений t_1' сбрасывается. Если T_1 имеет максимальный ранг, то остальные деревья подвешиваются к t_1, в результате чего второе дерево для новой очереди приоритетов не образуется. В противном случае дерево с максимальным рангом среди оставшихся становится вторым деревом новой приоритетной очереди. Остальные деревья подвешиваются к корню этого дерева, а корни добавленных деревьев заносятся в массив нарушений. Чтобы число активных нарушений не превышало порогового значения, по возможности выполняется два сокращения нарушений. Наконец, регулярные счетчики, которые больше не соответствуют корням, удаляются.

4) **Decrease** Значение в узле x заменяется значением e. Если e меньше значения в t_1, то роли x и t_1 меняются местами, не меняя их списки нарушений. Если x является либо t_1, либо t_2, либо дочерней вершиной $t1$, то ничего не делаем. В противном случае x помечается как нарушение и добавляется в структуру нарушений t_1; если x уже был нарушающим, то он удаляется из структуры нарушений, в которой он находился. Чтобы удержать количество активных нарушений в пределах порога, по возможности выполняется сокращение нарушений.

5) **DeleteMin** Согласно структуре дерева, минимум находится в вершине t_1. Узел t_2 и все поддеревья, укорененные в его дочерних вершинах, подвешиваются к t_1. Это сопровождается расширением массива нарушений T_1, и удалением регулярного счетчика T_2. Согласно второму инварианту, новый минимум теперь хранится в одном из дочерних или нарушающих узлов t_1. Согласно лемме 1 и третьему инварианту, число кандидатов в минимумы равно $O(\lg n)$. Пусть x - узел с новым минимумом. Если x находится среди узлов нарушений t_1, то дерево, имеющее тот же ранг, что и x, удаляется из-под t_1, его корень становится нарушающим и присоединяется вместо поддерева с корнем в x. Если x находится среди детей t_1, то дерево с корнем в x удаляется из-под t_1. Неактивные нарушения t_1 записываются в массив нарушений. Нарушения x также записываются в этот массив. Список нарушений t_1 добавляется к списку нарушений x. Затем узел t_1 удаляется и заменяется узлом x. Старые поддеревья x добавляются, за другим, под новым корнем. Чтобы количество нарушений не превышало порогового значения, сокращения нарушений выполняются столько раз, сколько возможно. Согласно третьему инварианту, должно быть выполнено не более $O(\lg n)$ сокращений нарушений.

6) **Delete** Вершина x заменяется на t_1, которая затем становится нарушающей. Чтобы удалить текущий корень x, выполняются те же действия, что и в **DeleteMin**.