**Приоритетная очередь со свойством временных указателей**

Абстрактное описание  
Мы представляем приоритетную очередь, которая поддерживает операцию вставки в худшем случае за постоянное время, удаление минимального значения, доступ к наименьшему, удаление элемента и уменьшение элемента х в худшем случае за время O(log(min{wх, qх})), где wх (соответственно qx) это количество элементов ,к которым был получен доступ после (соответственно до) последнего обращения к х, и которые все еще находятся в приоритетной очереди, пока выполняется соответствующая операция. (Доступ к элементу учитывается любой операцией приоритетной очереди, которая включает в себя этот элемент.) Наша приоритетная очередь имеет и рабочее множество, и свойство очередности; и еще больше она удовлетворяет этим свойствам в значении худшего случая. По результатам [1, 2] наша приоритетная очередь также удовлетворяет свойствам статического указателя, статической оптимальности и единым ограничениям. Более того, мы модифицировали нашу очередь, чтобы она могла реализовывать новое объединяющее свойство-временных указателей-которое инкапсулирует рабочее множество и свойство очередности.

**1.Введение**

Чувствительные к распределению структуры данных - это структуры, для которых временные ограничения выполнения операций изменяются в зависимости от последовательности выполняемых операций [1]. Эти структуры данных обычно выполняют как и их нечувствительные к распределению аналоги на случайной последовательности операций в амортизированном значении; однако, когда последовательность операций следует определенному распределению (например, имеет временную или пространственную локальность), чувствительные к распределению структуры данных работают значительно лучше.

Наиболее типичная структура данных, чувствительная к распределению, представляет собой сплэй (splay) деревья [4]. Сплэй деревья, кажется, выполняются очень эффективно над несколькими естественными последовательностями операций, теоретическими [4] (асимптотически быстрее θ(log n) времени поиска на наборе из n элементов) и практически [5]. До сих пор не существует всестороннего анализа, чувствительного к распределению, для сплэй деревьев. Вместо этого существуют теоремы и предположения, характеризующие их свойства, чувствительные к распределению; они включают: статический указатель, статическую оптимальность, рабочее множество [4], последовательный доступ

[6, 7], единые границы [4], динамический указатель [8] и единые гипотезы. Мы обращаемся к читателю [4, 9] для подробного описания и обсуждения этих свойств.

Рассмотрим достаточно длинную последовательность операций доступа A = <a1; a2; …; am> (Сортированную в хронологическом порядке), выполняемую на наборе из n элементов {x1; x2; …; xn}. Пусть ei - индекс элемента, к которому обращается операция ai. Для реализации структуры данных, которая достигает чувствительных к распределению границ (даже в амортизированном смысле), длина такой последовательности доступа должна удовлетворять m = Ω(n log n).

Свойство статического указателя [4] указывает, что для любого фиксированного элемента индекса f (указатель) амортизированное время выполнения ai равно O (log (dA (ei; f)), где dA (ei; f) разница в порядке между xei и xf, когда элементы сортируются по значению. Более конкретно, для доступа к последовательности А общее время доступа для структуры со свойством static- finger равно O ())

Свойство статической оптимальности (энтропийная граница) [4] указывает, что для доступа к последовательности A, где элемент xei получает доступ hA (ei) раз, общее время доступа равно O()=O().

Размер wA (i) рабочего множества для операции ai в последовательности A определяется как число различных элементов, к которым осуществляется доступ с момента последнего доступа к элементу xei, или с начала последовательности, если это первый доступ к xei.

Свойство рабочего множества [4] указывает, что общее время доступа для последовательности A равно O()). Неофициально свойство "рабочего множества" подразумевает, что элементы, которые в последнее время использовались, работают быстрее по сравнению с элементами, доступ к которым в недавнем прошлом не осуществлялся.

Унифицированная граница [4] указывает на очевидное, но не более того [2], сильное свойство. Унифицируемая граница - это минимальное значение для каждой операции между статическим указателем, статической оптимальностью и границами рабочего множества. Точнее, чтобы иметь унифицированную привязку, общее время доступа для последовательности A и любого фиксированного указателя f равно O().

Между вышеупомянутыми чувствительными к распределению свойствами существуют взаимосвязи. Iacono [1] заметил, что свойство рабочего множества подразумевает статическую оптимальность и свойства статического указателя. Мы недавно доказали, что унитарная граница связана с рабочей границей [2], что указывает на то, что обе границы асимптотически эквивалентны. То есть, в то время как Iacono [1] показал, что свойство рабочего множества гарантирует минимум из трех границ, мы [2] показали, что он даже гарантирует минимум на операцию этих границ.

Структуры данных, чувствительные к распределению, не ограничиваются деревьями поиска. Также приоритетные очереди были разработаны и проанализированы в контексте чувствительности распределения [10, 11, 12, 13]. В модели сравнения легко заметить, что приоритетная очередь с постоянным временем вставки не может иметь последовательную характеристику доступа и, следовательно, не может также иметь свойство динамического указателя; В противном случае последовательность вставок, за которой следует последовательность минимальных удалений, перечисляет элементы в отсортированном порядке за линейное время (противоречащем нижней границе Ω[n log n) для сортировки на основе сравнения. В качестве альтернативы, свойство рабочего множества представляет основной интерес для приоритетных очередей. Iacono [12] доказал, что парные пирамиды [14] удовлетворяет свойству рабочего множества следующим образом: в пирамиде максимального размера n они занимают O (log (min {ox,n})) амортизированного времени для удаления минимального значения x, где ox - количество операций, выполненных с момента вставки x. Воронкообразные пирамиды [10] представляют собой эффективные пирамиды ввода/вывода, для которых требуется O (log (min {ix,n})) времени для удаления минимального значения x, где ix - количество вставок, выполненных с момента вставки x. Elmasry [11] предоставил приоритетную очередь, поддерживающую удаление минимального x в наихудшем случае времени O (log ωx), где ωx - количество элементов, вставленных после вставки x и все еще присутствующих в приоритетной очереди, когда x удаляется (обратите внимание, что ωx≤ ix≤ ox); Мы кратко рассмотрим эту приоритетную очередь в Разделе 2. Ни одна из вышеупомянутых приоритетных очередей не поддерживает удаление (как показано, они поддерживают только удаление минимального элемента) в пределах ограничения рабочего множества. В разделе 3 представлена приоритетная очередь, которая поддерживает вставку в наихудшем случае постоянного времени, и удаление минимального, доступ к минимальному элементу, удаление и уменьшение в наихудшем времени O (log ωx). Для удаления и уменьшения предполагается, что указатель на назначенный узел легко задается.

Одна естественная последовательность операций это тип FIFO. Структуры данных, чувствительные к этим последовательностям, должны быстро работать с элементами, доступ к которым был получен наименее недавно. Это чувствительное к распределению свойство называется свойством "очередности" в [13]. В контексте приоритетной очереди такое свойство указывает, что время удаления минимального x равно O (log qх), где qх - количество элементов, вставленных перед x и всё ещё присутствующих в очереди приоритета при удалении x. Обратите внимание, что qx = n –ωх, где n - количество элементов, присутствующих в очереди приоритетов. Приоритетная очередь со свойством очередности представлена в [13]; Она также не поддерживает удаление. В [13] также показано, что ни одно двоичное дерево поиска не может быть чувствительным к этому свойству.

Все еще оставалось неясным, существует ли приоритетная очередь, чувствительная как к свойствам рабочего множества, так и к свойствам очереди. Мы отвечаем на этот вопрос утвердительно, предоставляя такую приоритетную очередь в разделе 4. В разделе 5 вводится свойство временных указателей. Это свойство инкапсулирует свойства очередности и рабочего множества и, таким образом, захватывает свойства статического указателя, статической оптимальности и единых границ. Поскольку эти свойства являются полным списком чувствительных к распределению свойств, известных для приоритетной очереди, мы называем свойство временных указателей "объединяющим свойством для приоритетной очереди ". В результате мы изменяем нашу приоритетную очередь, чтобы удовлетворить этому объединяющему свойству.

Мы опираемся на понятие числовых систем в нашей конструкции наихудшего случая. Система чисел представляет собой способ представления чисел с правилами, регулирующими выполняемые над ними операции. Существует связь между цифровыми системами и конструкцией данных [15, 16]. Идея состоит в том, чтобы связать количество объектов определенного типа в структуре данных со значением цифры. Основные операции, которые мы используем, это приращение и уменьшение любой цифры. Двоичное число расширенного типа [15] использует цифры {0; 1; 2; 3} с ограничениями, что между любыми двумя 3-ми имеется цифра, отличная от 2, а между любыми двумя 0-ми имеется цифра, отличная от 1. Расширенно-регулярная система цифр позволяет увеличивать и уменьшать произвольные цифры с постоянным числом изменений цифр за операцию (см. [17, 18] для получения подробной информации о реализации этой системы цифр).

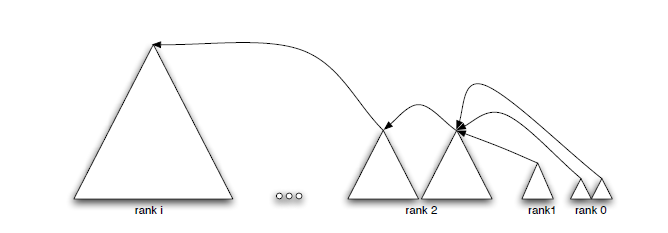


Рис. 1: (2, 3) -биномиальные деревья составляют приоритетную очередь: Ранги деревьев не уменьшаются справа налево. Предварительно фиксированные минимальные указатели поддерживаются у корней деревьев. Каждый корень дерева указывает на минимальный корень слева от него, включая его самого.

**2.Приоритетная очередь со свойством рабочего множества**

Наша приоритетная очередь строится на очереди приоритетов из [11], которая поддерживает вставку за постоянное время и удаление минимального значения в пределах ограничения рабочего множества. Преимущество очереди приоритетов из [11] по сравнению с очередями из [10, 12, 14] состоит в том, что она более удовлетворяет свойствам рабочего множества, в котором удаляемые элементы не засчитываются в рабочее множество. Далее мы обрисовываем структуру этой приоритетной очереди.

Приоритетная очередь в [11] состоит из упорядоченной пирамиды (2, 3) -биномиальных деревьев. Как указано в [11], поддеревья корня (2, 3) -биномиального дерева ранга r представляют собой (2, 3) биномиальные деревья; Есть один или два потомка, имеющих ранги 0; 1; ... ; R - 1, упорядоченные в не уменьшаемом значении ранга справа налево. (2, 3) -биномиальное дерево ранга 0 является единственным узлом. Легко проверить, что ранг (2, 3) -биноминального дерева, которое содержит n узлов является θ (log n). Ранги (2, 3) -биномиальных деревьев очереди приоритета также не уменьшаются справа налево. Для амортизированного решения существует не более двух (возможно, нулевых) деревьев на ранг. Для наихудшего решения число деревьев на ранг соответствует вариации расширенной-регулярной численной системы [15], которая гарантирует, что ранги любых двух соседних деревьев отличаются максимум на два. Корень каждого (2, 3) -биномиального дерева имеет указатель на корень с минимальным значением среди корней слева от него, включая себя. Такие указатели предварительного фиксированного минимума позволяют найти общий минимальный элемент за постоянное время с возможностью поддерживать такие указатели после удаления минимума за время, пропорциональное рангу удаленного узла. Обратите внимание, что указатель предварительного фиксированного минимума корня может быть обновлен за постоянное время путем сравнения значения, сохраненного в этом корне, со значением, на которое ссылается указатель в корне слева от него (при условии, что указатель в этом последнем корне обновлен). На рис. 1 показана структура этой очереди приоритетов.

Используются две примитивные операции: разделение и соединение. (2, 3) -биномиальное дерево ранга r может быть разделено на два или три дерева ранга r-1; Это выполняется путем отделения одного или двух потомков корня, который имеет высший ранг, т.е. с рангом r-1. С другой стороны, два или три (2, 3) -биноминальных дерева ранга r-1 могут быть соединены с образованием (2, 3) -биномиального дерева ранга r; Это делается путем присоединения корню с большим значением крайнего левого нижестоящего потомка с наименьшим значением. Другой особой формой операции соединения, необходимой для построения, является присоединение дерева ранга r -1 и дерева ранга r -2. Чтобы выполнить эту задачу, первое дерево разделяется, затем все полученные деревья соединяются: если после разделения существуют три дерева ранга r - 2, они соединяются в дерево ранга r -1. В противном случае разбиение получается в виде четырех деревьев ранга r - 2; каждая пара соединяется, а затем получающиеся два дерева соединяются, оканчиваясь деревом ранга r.

Общий хронологический порядок поддерживается в соответствии со временем вставки элементов. Если биномиальное дерево T1 находится справа от другого дерева T2, то все элементы T1 должны быть вставлены после элементов T2. Кроме того, в пределах отдельного дерева цель заключалась в том, что переход по порядку элементов с приоритетом справа налево по отношению к поддеревьям будет указывать порядок вставки этих элементов. Однако при выполнении операций соединения этот порядок иногда не соблюдается путем возможного изменения порядка двух целых поддеревьев. Для отслеживания правильного порядка достаточно поддерживать обратный бит с каждым узлом x; Такой обратный бит указывает, были ли элементы в поддереве x вставлены до или после элемента в родительском элементе x плюс элементы в потомках правых братьев и сестер x. При разделении результирующие деревья позиционируются в соответствии с обратным битом отделенных нижестоящих элементов корня.

С помощью операций соединения и разделения можно отсоединить корень a (2, 3) -биномиального дерева ранга r и снова реконструировать дерево в виде (2, 3) - биномиального дерева с рангом r - 1 или r при сохранении хронологического порядка; это выполняется путем повторяющихся соединений и разделений, начиная с крайних правых поддеревьев удаленного корня и заканчивая крайними левыми (подробнее см. [11]).

Для выполнения операции вставки новый одиночный узел, содержащий вставленный элемент, добавляется в качестве крайнего правого дерева в очереди приоритетов. Это может вызвать повторяющиеся присоединения до тех пор, пока есть три дерева с одинаковым рангом; Количество таких соединений амортизируется постоянно, что приводит к постоянной амортизируемой стоимости на вставку. После выполнения соединений обновляется указатель предварительного фиксированного минимума оставшегося корня. Для наихудшего решения нижележащая расширенно-регулярная численная система гарантирует максимум два соединения на вставку [15, 17, 18].

Для выполнения операции удаление минимального элемента дерево T минимального корня идентифицируется с помощью указателя предварительного фиксированного минимума крайнего правого дерева. Дерево T реконструируется как биномиальное дерево (2, 3) после отсоединения его корня. За этим может последовать разделение и соединение, если ранг T теперь на один меньше своего первоначального ранга. Наконец, обновляются предварительно фиксированные минимальные указатели деревьев справа от Т и включая Т. Для амортизированного решения, несколько разделений T могут получиться так: пусть T' будет деревом справа от дерева T; Начиная с T, крайнее правое дерево, получающееся в результате предыдущих разделений, многократно разделяется до тех пор, пока результирующие деревья разделений T' не будут иметь последовательные ранги. Такое разделение не требуется для наихудшего решения. Нетрудно сделать вывод, что стоимость операции удаление минимального элемента равна O (r), где r - ранг удаленного узла. В деревьях справа от T имеются элементы O (ωx), и отсюда количество таких деревьев равно O (log ωx). Из наихудшего решения следует, что рангом удаленного узла x является O (log ωx). Для амортизированного решения дополнительная лемма [11, Теорема 1] устанавливает такую же связь в амортизированном смысле.

**3. Поддержка удаления, доступа к минимальному и увеличения элемента**

Существующие очереди приоритетов, чувствительные к распределению [10, 11, 12, 13], не поддерживают удаление в пределах ограничения рабочего набора. В этом разделе мы изменяем конструкцию, описанную в разделе 2, для поддержки удаления в пределах ограничения рабочего набора. Включить удаление в репертуар операций не трудно, но нужно делать осторожно. Главная задача - поддерживать элементы в хронологическом порядке.

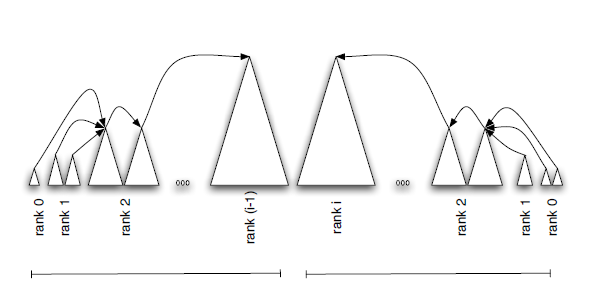
Пусть x является удаляемым узлом. Мы проходим предков x вверх через родительские указатели, пока не достигнем корня дерева. Мы используем два стека; Правый стек и левый стек. Начиная с корня, мы многократно разделяем текущее поддерево на два-три поддерева. Продолжая разделять поддерево, которое содержит x, другой или два поддерева проталкиваются на стеки, пока мы не получим поддерево, корнем которого является x. (В зависимости от обратного бита и относительного положения корня поддерева, которое содержит x, мы вставляем поддерево либо в правый, либо в левый стек.) На этом этапе мы удаляем x аналогично операции удаление минимального элемента: узел x отсоединяется, а поддеревья, являющиеся результатом удаления x, постепенно соединяются справа налево, при этом возможно выполнение одного разделения перед каждым соединением. Далее мы должны прорабатывать свой путь до корня и присоединяться ко всем поддеревьям, которые мы ввели путем расщепления на пути вниз от корня. Один или два поддерева, которые имеют одинаковый ранг, многократно выскакивают из стеков и соединяются с текущим поддеревом, при этом возможно выполнение одного разделения перед каждым соединением. Как только два стека пусты, может быть выполнено разделение и соединение, если результирующее дерево имеет ранг на один меньше своего первоначального ранга.

Общий порядок корректно поддерживается, отмечая, что единственными используемыми операциями являются разделение и соединение, которые гарантированно правильно устанавливают и используют обратные биты. С тех пор как высота биномиального (2, 3) дерева это один плюс его ранг, время для удаления - это O (r), где r - ранг дерева, содержащего удаленный узел. При этом устанавливается тот же временной предел, что и для удаление минимального элемента.

Используя указатели предварительного фиксированного минимума, возможно минимальное нахождение за постоянное время. Однако версия операции, которая считается доступом, не может поддерживаться во времени асимптотически меньше, чем у удаление минимального элемента; В противном случае следующая операция удаление минимального должна поддерживаться за постоянное время в соответствии со свойством рабочего набора. Операция доступа к минимальному для элемента x может быть выполнена за время O (log ωx) путем выполнения операции удаление минимального с последующим повторным выполнением.

Также операция уменьшения, которая считается доступом, не может быть выполнена за асимптомно меньшее время, чем удаление; В противном случае операция удаления может быть выполнена путем выполнения операции уменьшения со значением уменьшения, равным нулю, которая по существу выводит элемент на передний план рабочего набора, после чего выполняется операция удаления, которая теперь должна выполняться в постоянное время. Операция уменьшения для элемента x может быть выполнена через время O (log ωx) путем выполнения операции удаления, за которой следует повторная вставка.

**Теорема 1.** Приоритетная очередь, представленная в этом разделе, поддерживает вставку за постоянное время, и удаление минимального, доступ к минимальному, удаление и уменьшение элемента x за время O (log ωx), где ωx - количество элементов, к которым осуществляется доступ после последнего доступа к x и которые все еще присутствуют в очереди приоритетов при текущей операции.



Левая очередь Правая очередь

Рис. 2: Приоритетная очередь, удовлетворяющая свойству очереди: приоритетная очередь состоит из двух очередей, одна из которых имеет ранги дерева, увеличивающиеся справа налево (правая очередь), а другая - слева направо (левая очередь). Ранги крупнейших деревьев с двух сторон должны отличаться не более чем на один. Указатели предварительного x-минимума являются отдельными для каждой стороны.

**4. Включение свойства очередности**

Свойство очередности для приоритетных очередей указывает, что время выполнения операции доступа (удаление, доступ к минимальному элементу, удаление или уменьшения) для элемента x = O (log (n-ωx)), где n - количество элементов, присутствующих в настоящее время в очереди приоритетов, а ωx - количество элементов, к которым осуществляется доступ после последнего доступа к x и которые все еще присутствуют в очереди приоритетов. Другими словами, свойство очередности указывает, что время выполнения одной из вышеупомянутых операций над элементом x равно O (log qx), где qx = n-ωx - количество элементов, к которым последний раз обращались до x и которые все еще присутствуют в очереди приоритетов. Очереди [13] есть очередность приоритетной очереди, которые поддерживают вставку за амортизируемое постоянное время и поддерживают любые другие операции на x за амортизируемое время O (log qx).

Мы расширяем нашу приоритетную очередь из раздела 3, в дополнение к поддержке привязанного рабочего набора, чтобы также поддерживать операции внутри ограничений очередности. Вместо не уменьшения рангов деревьев очереди справа налево, мы делим очередь на две стороны, правую очередь и левую очередь. Ранги деревьев правой очереди монотонно не уменьшаются справа налево, а ранги левой очереди монотонно не уменьшаются слева направо. Мы также накладываем ограничение, чтобы разница в ранге между самым большим деревом с каждой стороны была не более одного. На рис. 2 показана новая приоритетная очередь.

Указатели предварительного фиксированного минимума в левой и правой очередях сохраняются неотделимо. В правой очереди корень каждого дерева сохраняет указатель на корень с минимальным значением среди тех, которые находятся в правой очереди слева от него. И наоборот, в левой очереди корень каждого дерева сохраняет указатель на корень с минимальным значением среди тех, которые находятся в левой очереди справа от него. Для обеспечения общего минимального значения выполняется зондирование как левой, так и правой очередей.

Вставки выполняются точно так же, как и ранее в правой очереди. Операция удаление минимального элемента выполняется в левой или правой очереди в зависимости от того, где лежит минимум. Остальные операции выполняются, как мы уже говорили ранее.

Мы должны сохранять инвариант в том, что разница в ранге между самым большим деревом с левой и правой сторон не более одного. Поскольку хронологический порядок сохраняется среди наших деревьев, эта инвариантность гарантирует, что рангом дерева элемента x является O (log (min {ωx; qx})). В результате вставки или удаления эта разница в рангах может быть равна двум. Раз высший ранг с одной стороны на два больше, чем с другой стороны, нам нужно перенести некоторые деревья между двумя сторонами. Предположим, что наивысший ранг на стороне "A" - r, а на стороне "B" это r - 2. Для поддержания инварианта мы делаем следующее:

Случай 1. Если есть два дерева ранга r на стороне "A", то переместите соответствующий (для поддержания хронологического порядка) одно из двух на сторону "B".

Случай 2. В противном случае разделите единственное дерево ранга r на два или три дерева ранга r -1. Теперь у нас есть k ∈ [2…5] деревьев ранга r - 1 на стороне "A".

(а) Если 2 ≤ k≤ 4, переместите соответствующие [k / 2] деревья на сторону "B", и остановитесь.

(b) В противном случае (k = 5) переместите соответствующие два дерева на сторону "B", а затем присоедините соответствующие два дерева среди оставшихся трёх на стороне "A", чтобы сформировать дерево ранга r.

Для наихудшего случая решения, используя вышеописанную процедуру, ранги деревьев на каждой стороне все равно будут подчиняться расширенной-регулярной численной системе. После перемещения деревьев с одной стороны на другую может потребоваться обновление всех указателей предварительного фиксированного минимума на обеих сторонах очереди приоритетов. Однако для амортизированного решения следующая лемма указывает, что эта работа не влияет на заявленные амортизированные границы для каждой операции.

**Лемма 1.** Рассмотрим двустороннюю приоритетную очередь, представленную в этом разделе. Для амортизируемого решения обновление указателей предварительного фиксированного минимума учитывает только дополнительный постоянный коэффициент амортизируемой стоимости на операцию.

**Доказательство.** Рассмотрим момент сразу после перемещения деревьев между двумя сторонами очереди приоритетов и обновления указателей предварительного фиксированного минимума. Чтобы это действие повторилось, высшие ранги с одной или обеих сторон должны измениться. Одна из возможностей состоит в том, что такой ранг с одной стороны уменьшается в результате операции удаления-минимиального или удаления. В этом случае удаление займет θ(log n) времени; этого времени достаточно для обновления указателей предварительного фиксированного минимума в пределах той же асимптотической связи, если необходимо. Другая возможность заключается в том, что наивысший ранг с одной стороны увеличивается в результате операций вставки. Далее показано, что в таком случае обновление указателей выполняется только после многих операций вставки.

Например, можно проанализировать Случай 2 (a). После обновления указателя наивысшим рангом с обеих сторон является r -1. Высший ранг может стать r довольно скоро, даже после одной вставки, но это не оставит ни одно из деревьев с рангом меньше r. Другими словами, эта сторона очереди приоритета будет состоять только из одного дерева, рангом которого является r. Чтобы высший ранг стал r +1, должно быть достаточно вставленных элементов для получения ещё двух деревьев ранга r. Так как размер каждого такого дерева, по крайней мере, 2r ,необходимо выполнить, по меньшей мере, еще 2r+1 операций вставки перед обновлением следующего указателя. Поскольку число обновлений указателя равно O (r) и каждое обновление требует постоянного объема работы, постоянная амортизированная стоимость на вставку легко покрывает эти затраты. Остальные случаи довольно похожи на этот, и к ним применимы те же аргументы.

Чтобы гарантировать затраты в худшем случае, обновление этих указателей предварительного фиксированного минимума должно выполняться поэтапно с предстоящими вставками следующим образом. При каждой операции вставки мы обновляем постоянное число указателей предварительного -минимума; Этот процесс обновления начинается с корня с самым высоким рангом до корня с наименьшим рангом с каждой стороны. Как следствие, в любой момент времени возможно, что у корней с более высокими рангами указатели предварительного -минимума обновлены, в то время как у других корней нет. Сохраняется ссылка на последний корень x, указатель предварительного -минимума которого обновлен. В соответствии с этим, для операции удаление минимального, чтобы обеспечить минимум на одной стороне очереди приоритета, он обращается к указателю предварительного -минимума корня с наименьшим рангом, а также к указателю корня X. Предыдущая лемма показывает, что будет достаточно вставок, чтобы опустить этот процесс обновления, прежде чем необходимо будет инициировать новый.

Удаление узла x в дереве ранга r все равно стоило бы О (r) времени, но теперь r = О (log (min {ωx; qx})) в амортизированном смысле (для амортизированного решения) или в наихудшем смысле (для наихудшего решения).

**Теорема 2.** Двусторонняя приоритетная очередь, представленная в этом разделе, поддерживает вставку за постоянное время и удаление минимального, доступ к минимальному, удаление и уменьшение элемента x за время (log (min {ωx; qx})), где ωx и qx - количество элементов, к которым был получен доступ после, и соответственно, до x и которые все еще присутствуют в очереди приоритетов при текущей операции.

**5. Поддержка многократных временных указателей**

Мы определили временные указатели t1, t2 ,…,tc как экземпляры времени в последовательности операций, которые свободно устанавливаются исполнителем. Мы определяем рабочее множество элемента x по отношению к временному указателю ti, ωx (ti), как количество элементов, которые были в последний раз доступны во время между последним доступом к x и ti, и все еще присутствуют в очереди приоритетов. Мы говорим, что приоритетная очередь удовлетворяет свойству нескольких временных указателей, если время доступа x равно O (c + log ( = { ωx (ti)})). Нетрудно заметить, что свойство рабочего множества эквивалентно одному временному указателю t1 = +∞, а свойство очередности эквивалентно одному временному указателю t1 = 0. Приоритетная очередь, представленная в разделе 4, которая поддерживает как свойства рабочего множества, так и свойства очередности, имеет два временных вывода t1 = 0; t2 = 1. В этом разделе представлена приоритетная очередь, которая удовлетворяет свойству для любого количества временных указателей.

Структура состоит из нескольких двусторонних очередей приоритетов, как разработанные в разделе 4. Мы начинаем с одной двухсторонней очереди приоритетов PQ0, и в каждый момент, когда вводится новый временной указатель, мы завершаем очередь и запускаем новую. Поэтому, соответствуя c временные указатели t1 = 0,…,tc = ∞, у нас есть c - 1 двухсторонних приоритетных очередей PQ1,…, PQс-1.

Вставки выполняются в последней (в момент выполнения вставки) очереди приоритетов и занимают постоянное время каждая (по Теореме 2). Для операций удаления предоставляется ссылка на элемент x для удаления. Мы определяем, к какой очереди приоритета PQj принадлежит элемент, и удаляем его. Это требует времени O (log (min {ωx (tj ); ωx (tj+1 ) })) (по теореме 2). Поскольку x принадлежит PQj, для любого i < j, ωx (tj ) ≤ ωx (ti ),а для любого i > j+1, ωx (tj+1 ) ≤ ωx (ti ). Из этого следует, что log (min {ωx (tj ); ωx (tj+1 ) }) = log ( = { ωx (ti)}). Для операции удаление минимального достаточно отметить, что поиск минимального элемента на очередь занимает постоянное время. Поэтому мы можем определить в какое время O (c) приоритетная очередь, содержит минимум, и выполняет операцию. Аргумент времени выполнения аналогичен аргументу для операции удаления.

**Теорема 3.** Для заданных временных указателей t1, t2 ,…,tc, приоритетная очередь, представленная в этом разделе, поддерживает вставку за постоянное время, а операции удаление минимального, доступ к минимальному, удаление и уменьшение элемента x за время O(c + log ( = { ωx (ti)})),где ωx (ti )- количество элементов, которые были доступны во время между последним доступом к x и ti ,и все еще присутствуют в очереди приоритетов при текущей операции.

**6. Итог**

Наше внимание было сосредоточено на приоритетных очередях, чувствительных к распределению. Как предполагается, очереди приоритетов не могут удовлетворять свойству последовательного доступа и в соответствии ни динамического указателя, ни единых гипотез. Поэтому мы рассмотрели другие чувствительные к распределению свойства, а именно: рабочего множества и свойство очередности; Мы представили приоритетную очередь, которая удовлетворяет обоим свойствам. Наша приоритетная очередь строится на очереди приоритетов [11], которая поддерживает вставку за постоянное время и удаление-минимального во временном ограничении рабочего набора. Мы показали, что одна и та же структура может также поддерживать удаление в пределах рабочего множества. Затем мы модифицировали структуру, чтобы удовлетворить также и свойству очереди. Стоит заметить, что наша приоритетная очередь поддерживает более точное определение рабочего набора и свойств очередности, в которых удаленные элементы не учитываются в пределах времени. Наш недавний результат о асимптотической эквивалентности ограничения рабочего набора и ограничения единых границ [3] подразумевает, что наша приоритетная очередь также удовлетворяет статической-оптимальности, статическому указателю и единым границам. Мы взяли за основу понятие временных указателей, которое инкапсулирует рабочий набор и свойство очередности. Описанная до сих пор приоритетная очередь имеет два временных указателя. Мы обобщили нашу приоритетную очередь, чтобы было возможно поддерживать любое количество временных указателей. Указанные границы амортизируются. Однако мы показали, что временные рамки для свойств рабочего набора и очереди также могут работать в худшем случае. В целом, границы нескольких временных указателей могут работать в худшем случае. Однако временные границы для других свойств- статической оптимальности, статического указателя и единых границ-естественно остаются амортизированными.

**Список литературы.**

[1] J. Iacono, Distribution-sensitive data structures, Ph.D. thesis, Rutgers, The

state University of New Jersey, New Brunswick, New Jersey, 2001.

[2] A. Elmasry, A. Farzan, J. Iacono, On the hierarchy of distribution-sensitive

properties for data structures, accepted under revision to Acta Informatica

(2012).

[3] A. Elmasry, A. Farzan, J. Iacono, A unifying bound for distribution-

sensitive priority queues, in: Proceedings of the 22nd International Work-

shop on Combinatorial Algorithms, volume 7056 of Lecture Notes in Com-

puter Science, Springer-Verlag, 2011, pp. 209{222.

[4] D. D. Sleator, R. E. Tarjan, Self-adjusting binary search trees, Journal of

the ACM 32 (1985) 652-686.

[5] H. E. Williams, J. Zobel, S. Heinz, Self-adjusting trees in practice for large

text collections, Software|Practice and Experience 31 (2001) 925-939.

[6] A. Elmasry, On the sequential access theorem and dequeue conjecture for

splay trees, Theoretical Computer Science 314 (2004) 459-466.

[7] R. E. Tarjan, Sequential access in splay trees takes linear time, Combinatorica 5 (1985) 367-378.

[8] R. Cole, On the dynamic finger conjecture for splay trees. part II: Finger

searching, SIAM Journal on Computing 30 (2000) 44-85.

[9] M. Badoiu, R. Cole, E. D. Demaine, J. Iacono, A unified access bound on

comparison-based dynamic dictionaries, Theoretical Computer Science 382

(2007) 86-96.

[10] G. S. Brodal, R. Fagerberg, Funnel heap - a cache oblivious priority queue,

in: Proceedings of the 13th International Symposium on Algorithms and

Computation, volume 2518 of Lecture Notes in Computer Science, Springer-

Verlag, 2002, pp. 219-228.

[11] A. Elmasry, A priority queue with the working-set property, International

Journal of Foundation of Computer Science 17 (2006) 1455-1466.

[12] J. Iacono, Improved upper bounds for pairing heaps, in: Proceedings of the

7th Scandinavian Workshop on Algorithm Theory, volume 1851 of Lecture

Notes in Computer Science, Springer-Verlag, 2000, pp. 32-45.

[13] J. Iacono, S. Langerman, Queaps, Algorithmica 42 (2005) 49{56.

[14] M. L. Fredman, R. Sedgewick, D. D. Sleator, R. E. Tarjan, The pairing

heap: a new form of self-adjusting heap, Algorithmica 1 (1986) 111-129.

[15] M. J. Clancy, D. E. Knuth, A programming and problem-solving seminar, Technical Report STAN-CS-77-606, Department of Computer Science,Stanford University, 1977.

[16] J. Vuillemin, A data structure for manipulating priority queues, Communications of the ACM 21 (1978) 309-315.

[17] A. Elmasry, J. Katajainen, Worst-case optimal priority queues via extended

regular counters, in: proceedings of the 7th International Computer Science

Symposium in Russia, to appear.

[18] H. Kaplan, N. Shafrir, R. E. Tarjan, Meldable heaps and Boolean union-

find, in: 34th Annual ACM Symposium on Theory of Computing, pp.

573-582.