Лекция 1.7. Приоритетные очереди

Добрый день. На этой лекции мы познакомимся еще с одним интересным типом данных, с приоритетной очередью.

Дадим определение.

Приоритетной очередью называется совокупность однотипных элементов, для которой определены две основные операции:

- добавление элемента в очередь;
- извлечение из очереди наименьшего по значению элемента.

В некоторых случаях удобнее считать, наоборот, что извлекается всегда наибольший элемент, но это неважно для реализации.

Как и в случаях стека и обычной очереди, тип элементов может быть различным (но в данном случае допускающим сравнение элементов), а размер очереди теоретически неограничен. Принципиальное отличие от обычной очереди и стека заключается в том, что порядок извлечения зависит не от порядка добавления, а от значения элементов. Как нетрудно догадаться, это отличие требует совершенно иной реализации.

Давайте сначала обсудим простые и очевидные варианты реализации приоритетной очереди, которые никуда не годятся с точки зрения эффективности.

Если использовать массив и добавлять элементы в конец массива, то для извлечения придется пройти по всему массиву в поисках минимального элемента, а потом еще и сдвинуть к началу элементы, стоявшие после извлеченного.

Если поддерживать в массиве порядок элементов по возрастанию, то извлекаемый элемент будет всегда на первом месте, но при добавлении придется искать место для вставки м отодвигать последующие элементы.

В любом из этих вариантов время выполнения операций пропорциональны количеству элементов в очереди.

Использование списка вместо массива ненамного улучшает дело, придется проходить по списку либо при добавлении элемента, либо при его извлечении.

Хочется реализовать более быстрое добавление и извлечение элементов. Но как это сделать?

Будем использовать для приоритетной очереди структуру данных, известную как пирамида.

Определение 1. Пусть дан массив A[N]. Будем называть сыновьями элемента A[k] элементы A[2k+1] и A[2k+2] (если они существуют).

Соответственно, A[k] называется отцом A[2k+1] и A[2k+2].

В частности, сыновья A[0] – это A[1] и A[2], сыновья A[1] – это A[3] и A[4] и т.д.

При $k \geq N/2$ элементы A[k] не имеют сыновей, потому что значения индексов 2k+1и 2k+2 выходят за пределы массива. В случае четного N элемент A[N/2–1] имеет только одного сына с индексом N–1.

Определение 2. Массив A[N] называется пирамидой, если значения сыновей каждого элемента не меньше, чем значение отца.

Рассмотрим пример массива из 10 элементов. Нетрудно убедиться, что этот массив является, согласно определению, пирамидой.

Очевидно, что первый элемент массива-пирамиды всегда наименьший, что имеет место и в данном примере. Это удобно для извлечения из приоритетной очереди.

Но возникают два вопроса, которые надо решить:

- как сделать, чтобы при добавлении элемента в пирамиду она оставалась пирамидой?

- как восстановить пирамиду после извлечения из нее элемента A[0]?

И как сделать то и другое за минимальное время?.

Рассмотрим, как делается добавление элемента в пирамиду.

Мы добавляем в пирамиду новый элемент – число 15. Сначала мы помещаем это число в конец массива. Конечно, нужно проверить, не нарушается ли правило пирамиды, то есть, не будет ли новый элемент меньше своего отца?

Новый элемент имеет индекс 10, стало быть, его отец согласно известной нам формуле имеет индекс 4. Значение отца равно 88.

Поскольку значение отца больше, чем значение сына, поменяем их местами. Теперь отец нового элемента имеет индекс 1, его значение – 20.

Опять значение отца больше, чем значение сына; меняем их местами. В полученном массиве новый элемент 15 является сыном элемента 12, то есть отец теперь меньше сына. Массив снова является пирамидой.

Теперь рассмотрим извлечение элемента из приоритетной очереди.

Извлекаем из очереди начальный элемент – число 12.

На освободившееся место помещаем последний элемент массива – число 88.

Проверим, не нарушились ли после этого свойства пирамиды. Для этого из двух сыновей начального элемента, имеющих значения 15 и 14, выбираем меньшее значение 14 и сравниваем его со значением отца 88. Поскольку $14 < 88$, меняем отца и сына местами.

Снова выбираем меньшего из сыновей 88. Это 40. Поскольку $40 < 88$, опять меняем отца и сына местами.

И массив снова становится пирамидой, на первом месте опять наименьший элемент.

Очень интересный вопрос, какова трудоемкость операций добавления и извлечения элемента приоритетной очереди, реализованной как пирамида?

Наиболее трудоемкая часть обоих алгоритмов – выполняемый в цикле обмен значений отца с одним из сыновей. Поскольку при каждой итерации цикла индекс текущего элемента уменьшается или увеличивается более, чем в два раза, всего таких обменов может быть не больше, чем $\log2(n) + 1$, где n – текущее число элементов очереди. Это так, поскольку двоичный логарифм n (округленный вверх до целого числа) как раз и указывает, сколько раз надо нацело разделить число n на 2, чтобы получить 0. Таким образом, операции добавления и извлечения имеют логарифмическую трудоемкость, что

значительно меньше, чем у рассмотренных ранее наивных реализаций на базе обычного массива.

А почему массив с такими свойствами в русскоязычной литературе называется пирамидой?

Если мы представим такой массив в виде бинарного дерева, то получим почти треугольную картинку, где, правда, самый нижний ярус может быть неполным. При этом все перестановки элементов при добавлении и извлечении выполняются вдоль одной из ветвей бинарного дерева.

Некоторое сходство с пирамидой имеется. Кстати, в англоязычной литературе вместо слова «пирамида используется слово «heap», то есть «куча», что гораздо менее выразительно, да и термин слишком затасканный.

Но, раз мы используем бинарное дерево для наглядной иллюстрации, то, может быть, есть смысл использовать его вместо массива для практической реализации приоритетной очереди?

Конечно, это можно сделать. Преимущество перед массивом будет обычное: возможность неограниченного роста очереди. Но это единственное преимущество. Представление в виде массива значительно компактнее по памяти и несколько проще по выполнению операций.

А почему этот тип данных называется приоритетной очередью?

Это можно показать на таком примере. Допустим, элементы очереди описывают запросы на какое-то обслуживание, какую-то обработку. Добавление элемента означает поступление нового запроса. Извлечение элемента – передачу очередного запроса из очереди на обслуживание.

Значение элемента может означать приоритет запроса (чем меньше значение, тем выше приоритет, тем раньше надо обслужить этот запрос). Тогда на обслуживание всегда будет выбираться запрос с самым высоким приоритетом.

Итак, на этой лекции мы познакомились еще с одним полезным типом данных – приоритетной очередью. Ее принципиальное отличие от обычной очереди – извлечение элементов не по порядку добавления, а по значению элементов. В качестве структуры данных можно выбрать массив или бинарное дерево. Но важно, чтобы размещение элементов в массиве или в дереве соответствовало понятию «пирамиды». В таком случае время выполнения операций добавления и извлечения элементов будет примерно пропорционально логарифму размера очереди. Это намного быстрее, чем при использовании списка или обычного массива (не пирамиды).

На этом лекция закончена. Благодарю за внимание.