Приоритетные очереди

Приоритетная очередь (англ. *priority queue*) — это абстрактная структура данных наподобие стека или очереди, где у каждого элемента есть приоритет. Элемент с более высоким приоритетом находится перед элементом с более низким приоритетом. Если у элементов одинаковые приоритеты, они располагаются в зависимости от своей позиции в очереди. Обычно приоритетные очереди реализуются с помощью **куч** (англ. *heap*).

Содержание

Операции

Приоритетные очереди поддерживают следующие операции:

- findMin или findMax — поиск элемента с наибольшим приоритетом,
- insert или push — вставка нового элемента,
- extractMin или extractMax — извлечь элемент с наибольшим приоритетом,
- deleteMin или deleteMax — удалить элемент с наибольшим приоритетом,
- increaseKey или decreaseKey — обновить значение элемента,
- merge — объединение двух приоритетных очередей, сохраняя оригинальные очереди,
- meld — объединение двух приоритетных очередей, разрушая оригинальные очереди,
- split — разбить приоритную очередь на две части.

Реализации

Наивная

В качестве наивной реализации мы можем взять обычный список и при добавлении нового элемента класть его в конец, а при запросе элемента с максимальным приоритетом проходить по всему списку. Тогда операция insert будет выполняться за $O(1)$, а extractMin или extractMax за $O(n)$.

Обычная

Для лучшей производительности приоритетные очереди реализуют с помощью куч, что позволяет выполнять операции вставки и удаления за $O(\log n)$. Использование специальных куч, таких как Фибоначчиева куча и спаренная куча, позволяет еще больше улучшить асимптотику некоторый операций.

Виды приоритетных очередей

Название	Операции				Описание
	insert	extractMin	decreaseKey	merge	
Наивная реализация (неотсортированный список)	$O(1)$	$O(n)$	$O(n)$	$O(1)$	Наивная реализация с использованием списка.
Наивная реализация (отсортированный массив)	$O(n)$	$O(1)$	$O(\log n)$	$O(n+m)$	Наивная реализация с использованием отсортированного массива.
Двуродительская куча	$O(\sqrt{n})$	$O(\sqrt{n})$			Двуродительская куча (англ. *bi-parental heap* или *beap*) — такая куча, где у каждого элемента обычно есть два ребенка (если это не последний уровень) и два родителя (если это не первый уровень). Структура позволяет производить сублиненый поиск.
Двоичная куча	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(n+m)$	Двоичная куча (англ. *binary heap*) — такое двоичное дерево, для которого выполнены три условия: ▪ Значение в любой вершине не меньше, чем значения её потомков. ▪ Глубина листьев (расстояние до корня) отличается не более чем на 1 слой. ▪ Последний слой заполняется слева направо.
d-арная куча	$O(\log_d n)$	$O(d \log_d n)$	$O(d \log_d n)$	$O(n+m)$	d-арная куча (англ. d-*ary heap*) — двоичная куча, в которой у каждого элемента d детей вместо 2.
Левосторонняя куча	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$	Левосторонняя куча (англ. *leftist heap*) — двоичное левостороннее дерево (не обязательно

					сбалансированное), но с соблюдением порядка кучи.
Биномиальная куча	$O(1)$	$O(\log n)$	$O(\log n)$	$O(1)$	Биномиальная куча (англ. *binomial heap*) — структура данных, реализующая приоритетную очередь, которая представляет собой набор биномиальных деревьев с двумя свойствами: ▪ ключ каждой вершины не меньше ключа ее родителя ▪ все биномиальные деревья имеют разный размер
Спаренная куча	$O(1)$	$O(\log n)$	$O(\log n)$	$O(1)$	Спаренная куча (англ. *pairing heap*) — куча с относительно простой реализацией и хорошей производительностью, может быть рассмотрена как упрощенная Фибоначчиева куча.
Толстая куча	$O(1)$	$O(\log n)$	$O(1)$	$O(\log n)$	Толстая куча — это почти кучеобразный нагруженный пёс.
2-3 куча	$O(1)$	$O(\log n)$	$O(1)$	$O(1)$	Структура похожа на Фибоначчиеву кучу и использует в своей реализации 2-3 дерево.
Тонкая куча	$O(1)$	$O(\log n)$	$O(1)$	$O(1)$	Тонкая куча (англ. *thin heap*) — это структура данных, реализующая приоритетную очередь с теми же асимптотическими оценками, что и фибоначчиева куча, но имеющая большую практическую ценность из-за меньших констант.
Сонная куча	$O(1)$	$O(\log n)$	$O(1)$	$O(1)$	Куча, построенная на основе Фибоначчиева дерева. Фибоначчиево дерево (англ. *Fibonacci tree*) — биномиальное дерево, где у каждой вершины удалено не более одного ребенка.
Куча Бродала-Окасаки	$O(1)$	$O(\log n)$	$O(1)$	$O(1)$	Куча Бродала-Окасаки (англ. *Brodal's and Okasaki's Priority Queue*) — основана на использовании биномиальной кучи без каскадных ссылок, добавлении минимального элемента и на идеи Data-

					structural bootstrapping.

Применение

Приоритетные очереди используются в следующих алгоритмах: алгоритм Дейкстры, алгоритм Прима, дискретно-событийное моделирование (англ. *discrete-event simulation, DES*)[1], алгоритм Хаффмана, поиск по первому наилучшему совпадению, управление полосой пропускания.

Реализации в языках программирования

- Стандартная библиотека шаблонов[2] (англ. *STL*) в C++ предоставляет методы управления кучей make_heap, push_heap и pop_heap (обычно реализуются бинарные кучи), которые оперируют с итераторами произвольного случайного доступа. Методы используют итераторы как ссылки на массивы и выполняют преобразование массив-куча.
- Библиотека Boost[3] для C++ включает в себя библиотеку для работу с кучами. В отличии от STL, поддерживает операции decrease-key и increase-key, а также имеет поддержку дополнительных видов куч, таких как фибоначчиева куча, биномиальная куча и спаренная куча.
- В Java 2 (начиная с версии 1.5) предоставляется реализация бинарной кучи в классе java.util.PriorityQueue<E>[4], который не поддерживает операции decrease-key и increase-key.
- Python имеет модуль heapq[5], который реализует очереди с приоритетами с помощью бинарной кучи.
- PHP имеет поддержку кучи на максимум SplMaxHeap[6] и кучи на минимум SplMinHeap[7], как часть Standard PHP Library начиная с версии 5.3.
- В Perl имеются реализации[8] бинарной, биномиальной и фибоначчиевой куч во всеобъемлющей сети архивов.
- Go имеет пакет heap[9], в котором реализованы алгоритмы для работы с кучами.

См. также

- Сортировка
- Поисковые структуры данных
- Поиск подстроки в строке

Примечания

1. Wikipedia — Дискретно-событийное моделирование (http://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BA%D1%80%D0%B5%D1%82%D0%BD%D0%BE-%D1%81%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)
2. C++ — Стандартная библиотека шаблонов (http://en.cppreference.com/w/cpp/container)
3. C++ — Boost (http://www.boost.org/)
4. Java — java.util.PriorityQueue<E> (http://docs.oracle.com/javase/6/docs/api/java/util/PriorityQueue.html)
5. Python — heapq (http://docs.python.org/2/library/heapq.html)
6. PHP — SplMaxHeap (http://php.net/manual/ru/class.splmaxheap.php)

7. PHP — SplMinHeap (http://php.net/manual/ru/class.splminheap.php)
8. Perl — Heap (http://metacpan.org/pod/Heap)
9. Go — package heap (http://golang.org/pkg/container/heap/)

Источники информации

- Wikipedia — Heap (data structure) (http://en.wikipedia.org/wiki/Heap_(data_structure))
- Wikipedia — 2-3 heap (http://en.wikipedia.org/wiki/2%E2%80%933_heap)
- Wikipedia — Beap (http://en.wikipedia.org/wiki/Beap)
- Wikipedia — Binary heap (http://en.wikipedia.org/wiki/Binary_heap)
- Wikipedia — Binomial heap (http://en.wikipedia.org/wiki/Binomial_heap)
- Wikipedia — Brodal queue (http://en.wikipedia.org/wiki/Brodal_queue)
- Wikipedia — d-ary heap (http://en.wikipedia.org/wiki/D-ary_heap)
- Wikipedia — Fibonacci heap (http://en.wikipedia.org/wiki/Fibonacci_heap)

Источник — «http://neerc.ifmo.ru/wiki/index.php?title=Приоритетные_очереди&oldid=67382»

- Эта страница последний раз была отредактирована 30 ноября 2018 в 01:12.