

## 背景：

Heap 也是 ADT, Heap 是个性质 (max heap, min heap)

PQ 可以用 heap 来实现 (改 priority, 按 priority 排序)

Heap 可以用树来实现, 也可用指针来实现

优先队列(PriorityQueue) 是一种抽象数据结构 ADT, 队列内的元素都有对应的优先级, 并根据其优先级出队。

## Priority Queue 优先队列能解决的问题：

优先队列是一种队列, 它满足优先级别最高的先出队列。它的一个重要的应用时海量数据的排序, 当我们需要找出10亿数据中的Top10项, 不实际的解决方案是对所有数据排序, 取Top10项。使用优先队列可以在很小的辅助空间内找出Top10项。具体思路如下:

- (1) 建立大小为k的容器
- (2) 每次从大数据集中读取一个数据项
- (3) 如果容器没有满, 则把当前数据项加入容器
- (4) 如果容器已经满, 找出容器中最小的元素 (利用优先队列特性) 如果当前数据项大于最小元素, 则最小元素出队, 当前数据项入队
- (5) 重复 (2) 到 (3) 直到读取完大数据集

上述的方案可以以并行的方式运行, 最后合并结果。

优先度列的五种实现方式, 分别如下

1. 无序数组
2. 有序数组
3. 链表
4. 堆
5. 二叉搜索树

### 【优先队列的数组实现】

用数组实现优先队列: (insert 已经进行有序操作。pop 操作即为 peekMax, remove 操作)

### 无序数组的实现

无序数组实现方式的入队操作, 直接把入队元素加到数组尾部。出队需要遍历数组, 找出优先级别最高的出队, 空缺的位置由后面元素依次补上。因此, 入队的时间复杂度为 $O(1)$ , 出队为 $O(N)$ 。

### 有序数组的实现

由于要求数组有序, 因此在插入的时候需要保存有序, 插入操作需要找到适合的位置, 然后在该位置插入, 位置后面的元素依次往后移动, 时间复杂度为 $O(n)$ 。而出队, 由于序列是有序, 可以在 $O(1)$ 内出队。

Data Structures for Priority Queues: Lists

**Unsorted List:**

- $Insert(PQ, x, p): \Theta(1)$
- $FindMax(PQ): \Theta(n)$
- $ExtractMax(PQ): \Theta(n)$
- $IncreaseKey(PQ, x, k): \Theta(1)$

CSC263 | University of Toronto
4

Data Structures for Priority Queues: Lists

**Sorted List (by priorities):**

- $Insert(PQ, x, p): \Theta(n)$
- $FindMax(PQ): \Theta(1)$
- $ExtractMax(PQ): \Theta(1)$
- $IncreaseKey(PQ, x, k): \Theta(n)$

csc263 lecture2 ppt pg4-5

Queue(priority+value)

Priority Queue

Complete Binary Tree

Max-Heap

Min-Heap

FindMax(PQ) : 返回最大 pv 值的结点

HeapMaximum(A): 返回 root

IncreaseKey(A,i,k) : 增加堆 A 的结点 i 的 pv 值至 k (改)

HeapIncreaseKey(A,i,k) : 将 Array A 第 i 个的 pv 值设置成 k (改)

Insert(PQ,x,p) : 在堆 PQ 里插入结点 x 伴随 pv 值 p

MaxHeapInsert(A,x) : 在堆 A 里插入结点伴随 pv 值 x

MaxHeapify(B,i) :

ExtractMax(PQ):返回并删除最大 pv 值的结点

HeapExtractMax(H): H 是 heap 堆/array 数组, 返回 root, (并删除 root) , 并重新修复堆/Array - H

HeapSort(A) : 对数组 array A 进行 heap sort 排序

BuildMaxHeap(A)

CSC263 第三周

复习 BST Insert 和 Delete 操作