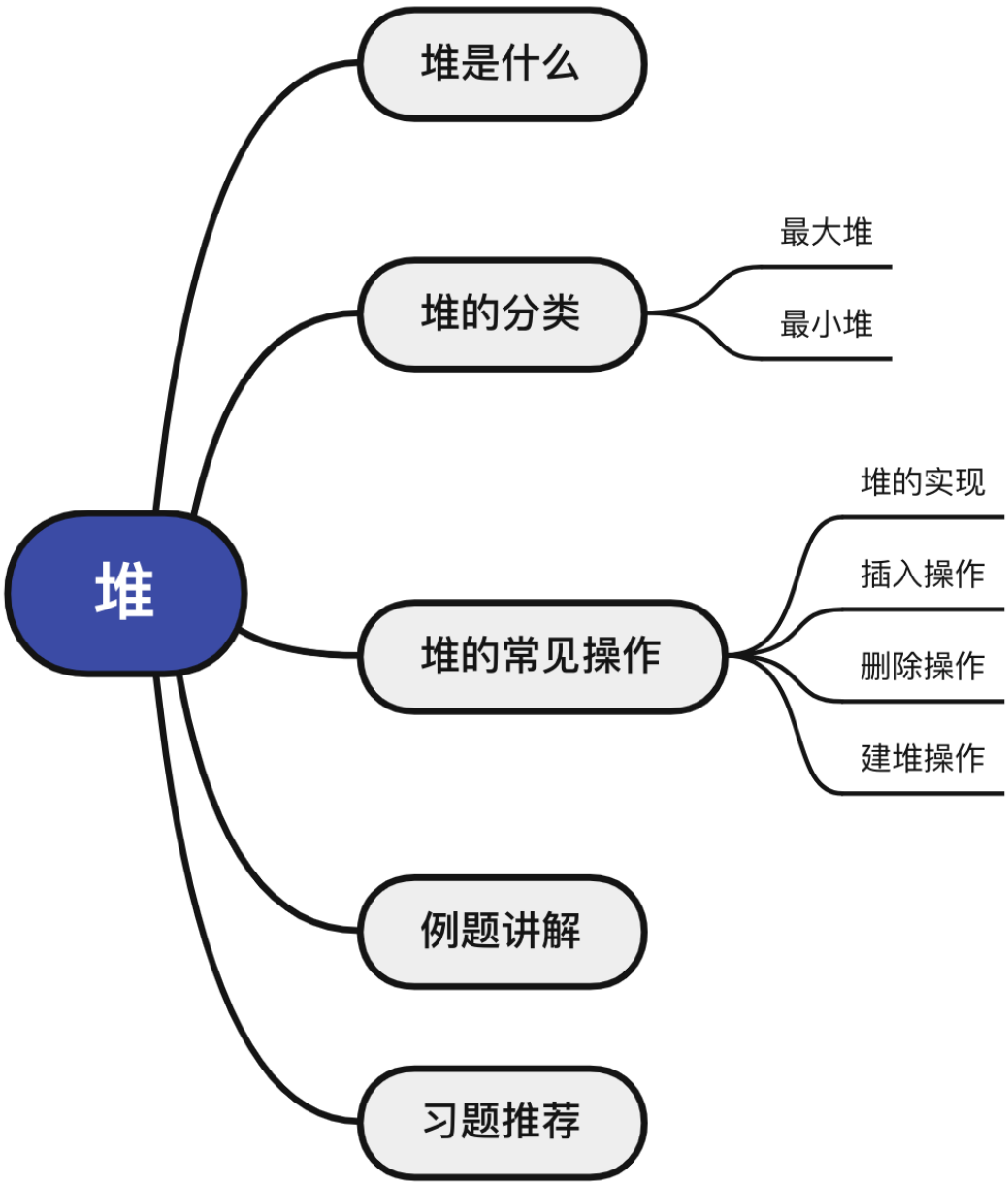
数据结构之堆

面试、刷题过程中，经常 被问到/使用的 优先队列，底层的数据结构是什么，如何实现的？

本篇文章通过介绍 优先队列 的底层数据结构「堆」，从知识到应用，相信看完的你肯定有所收获。

本文将从以下几个方面展开，目录如下，接下来是正文。



一、堆是什么

堆是一种特殊的完全二叉树。其特殊点在于，父节点的值 大于或小于 子节点的值，根据大于和小于也分成了二种，下面会分别讲解两种的特点。

我们经常使用的优先队列常用堆实现，优先队列可以O(1)时间复杂度获取最大值，O(logn)时间复杂度插入一个新的节点、删除堆顶节点。

因此常使用优先队列作为辅助数据结构，加速查找数据速度。

二、堆的分类

堆根据父节点和子节点的值的关系，分为两种。

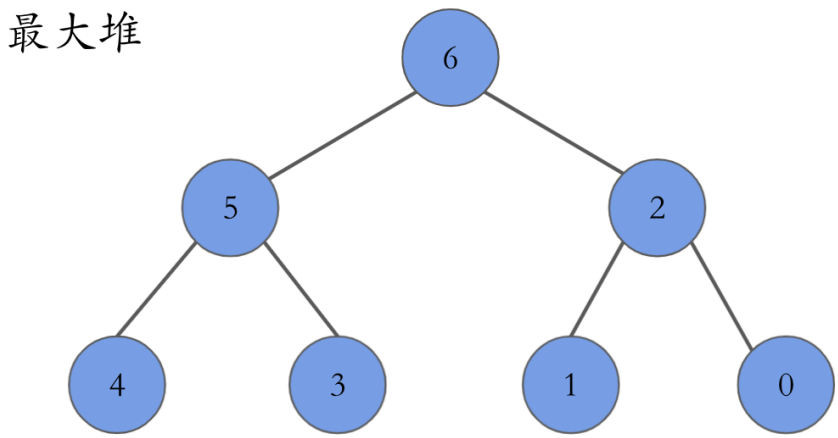
- 最大堆：父节点的值 大于等于 子节点的值。
- 最小堆：父节点的值 小于等于 子节点的值。

1）最大堆

最大堆满足两个特性。

- 首先它要是一颗完全二叉树。
- 其次要满足父节点的值 大于等于 子节点的值，因此堆顶整个堆的最大元素。

下图演示了一个常见的最大堆。

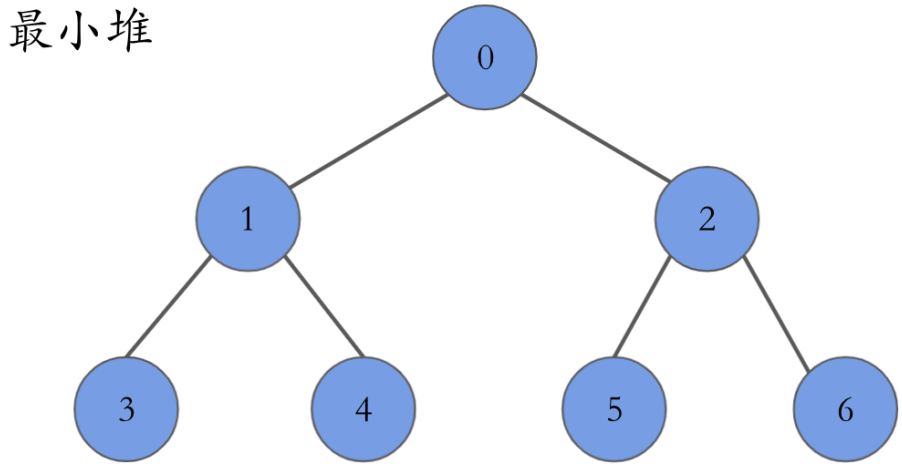


2）最小堆

最小堆满足两个特性。

- 首先它要是一颗完全二叉树。
- 其次要满足父节点的值 小于等于 子节点的值，因此堆顶整个堆的最小元素。

下图演示了一个常见的最小堆。



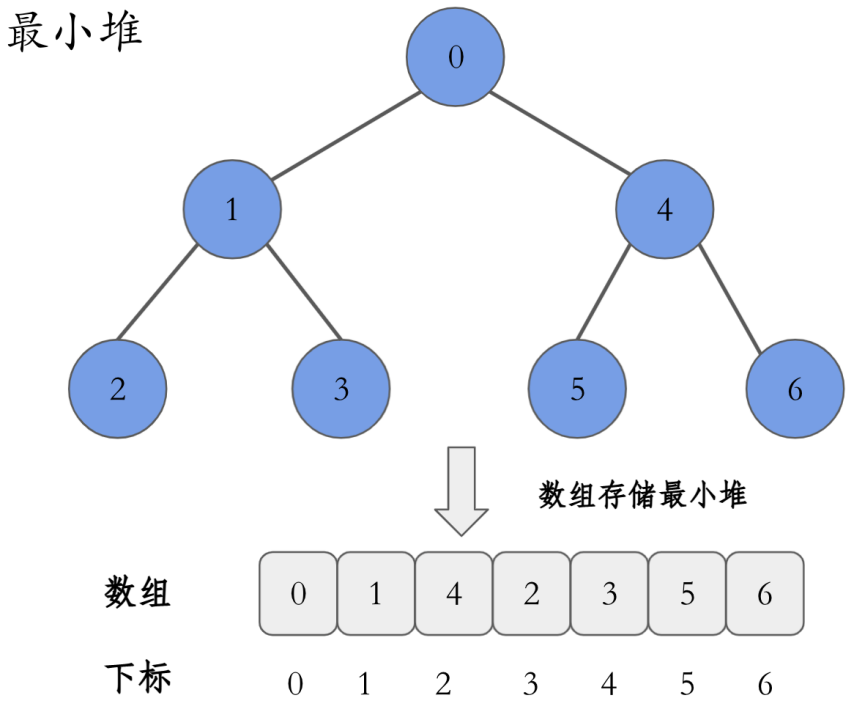
三、堆的常见操作

1）堆的实现

堆是特殊的完全二叉树，可以使用指针建立树，但通常使用更为简便的数组存储。

因为堆满足完全二叉树的性质，以数组下标从0开始举例，对于位置数组下标为 `i` 的节点，其父节点为 `(i-1)/2`，左右儿子分别为 `2i + 1` 和 `2i + 2`。

下图演示了如何将 堆，这种「特殊的完全二叉树」，转化为数组，以最小堆为例。



2）插入操作

插入操作是指，向二叉堆插入一个新的节点，并且保证插入后仍满足二叉堆的性质。

插入一个新的节点的方法是，直接插入到最下面一层最右边的叶子之后插入，对应到数组，即是当前数组的最后一个数字后面的位置。

如果最后一层已经满了，就新增一层，然后插到最左边，对应到数组，即是当前数组的最后一个数字后面的位置。

如何保证插入新的节点的二叉堆，仍满足二叉堆的性质？**答案是向上调整法！**

向上调整法

如果当前插入的节点的值大于它父节点的值，则交换，重复此过程直至不满足该条件 或者到根。

向上调整法的时间复杂度 O(logn)，因为最多只会修改一条链。

代码

```
void up(int x) {
  // 数组从下标 1 开始
  while (x > 1 && a[x] > a[x / 2]) {
    swap(a[x], a[x / 2]);
    x /= 2;
  }
}
```

3）删除操作

删除操作是指，删除堆顶元素。

如何删除堆顶元素呢？

1、方法一，直接删除堆顶，会将树分两个堆，不好合并起来。

2、方法二，插入操作的逆操作，将堆顶元素向下调整到最后一个叶子节点，然后就可以直接删除了。但你会发现，调整的过程比较复杂。

3、方法三，向下调整法，直接将根节点和最有一个叶子节点交换，然后将根节点删除，然后将暂时在根节点的叶子向下调整到正确的位置。

方法三中，如何将暂时在根节点的叶子向下调整到正确的位置？

在该节点中找到左右儿子较大的一个，然后与它交换，重复此过程直到不满足该条件 或者最底层。

方法三代码

```
void remove() {
  swap(a[1], a[n]);
  n--;
  down(1);
}

void down(int x) {
  while (x * 2 <= n) {
    t = x * 2;
    if (t + 1 <= n && a[t + 1] > a[t]) t++;
    if (a[t] <= a[x]) break;
    swap(a[x], a[t]);
    x = t;
  }
}
```

4）建堆操作

建堆操作是指，给一个无序的数组，建立一个二叉堆。

方法一，对数组排序，如果是最大堆是从大到小的顺序；如果是最小堆是从小到大的顺序，时间复杂度 O(nlogn)。

方法二，利用堆的插入操作，一个一个插入，时间复杂度 O(nlogn)。

方法三，从叶子开始，每个节点向下调整。
理解方法，可以看做每次合并两个已经建立好的堆，时间复杂度O(n)。

方法三代码

```
void build_heap() {
  for (int i = 1; i <= n; i++)
      up(i);
}
```

四、例题讲解

LeetCode 703. 数据流中的第 K 大元素

题意

给定一个数组和一个数字k，动态向数组中增加新的数字，每次增加后，返回第 k 大的数字。

示例

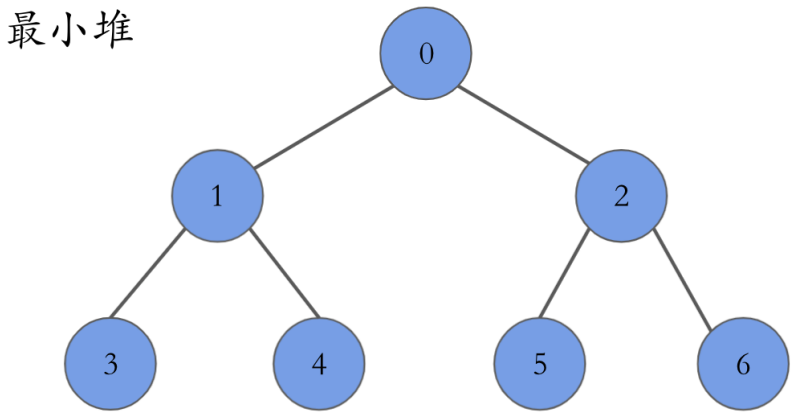
```
输入：
["KthLargest", "add", "add", "add", "add", "add"]
[[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]
输出：
[null, 4, 5, 5, 8, 8]

解释：
KthLargest kthLargest = new KthLargest(3, [4, 5, 8, 2]);
kthLargest.add(3);   // return 4
kthLargest.add(5);   // return 5
kthLargest.add(10);  // return 5
kthLargest.add(9);   // return 8
```

题解

维护一个**最小堆**，堆顶元素就是最小的元素，如果新增后堆内元素数量大于K，则弹出堆顶元素即可。

时间复杂度O(nlogn)。



代码

```
class KthLargest {
public:
    priority_queue<int, vector<int>, greater<int> > q;
    int K;
    KthLargest(int k, vector<int>& nums) {
        K = k;
        for (int i = 0; i < int(nums.size()); i++) {
            add(nums[i]);
        }
    }

    int add(int val) {
        q.push(val);
        if (q.size() > K) {
            q.pop();
```

```
        }
        return q.top();
    }
};
```

五、习题推荐

LeetCode 1046. 最后一块石头的重量

最后

大家好，我是编程熊，字节跳动、旷视科技前员工，ACM亚洲区域赛金牌，欢迎加入 LeetCode组队刷题群，群里有多位 **ACM亚洲区域赛金牌选手**，一起学习刷题打卡，**欢迎所有伙伴加入一起学习。**