C2-D 简单的堆 题解

题目描述

本题是堆的模板题，你的任务是实现一个大根堆，即根节点为所有元素最大值的堆，需要支持插入元素、查询堆顶元素、删除堆顶元素三个操作。

输入

第一行一个正整数n $(1 \leq n \leq 10^5)$，表示操作的数量。

接下来n行，每行为一个操作，格式如下：

- $1\ x$：向堆中插入元素x $(1 \leq x \leq 10^9)$。
- 2：删除堆顶元素。
- 3：查询堆顶元素。

数据保证后两种操作时堆非空。

输出

对于每次查询堆顶元素时，输出一行一个正整数，表示此时堆顶元素的值。

在所有操作结束后，将堆中的元素从大到小依次输出到一行中。

输入样例

```
8
1 2
1 4
3
2
1 1
3
1 3
3
```

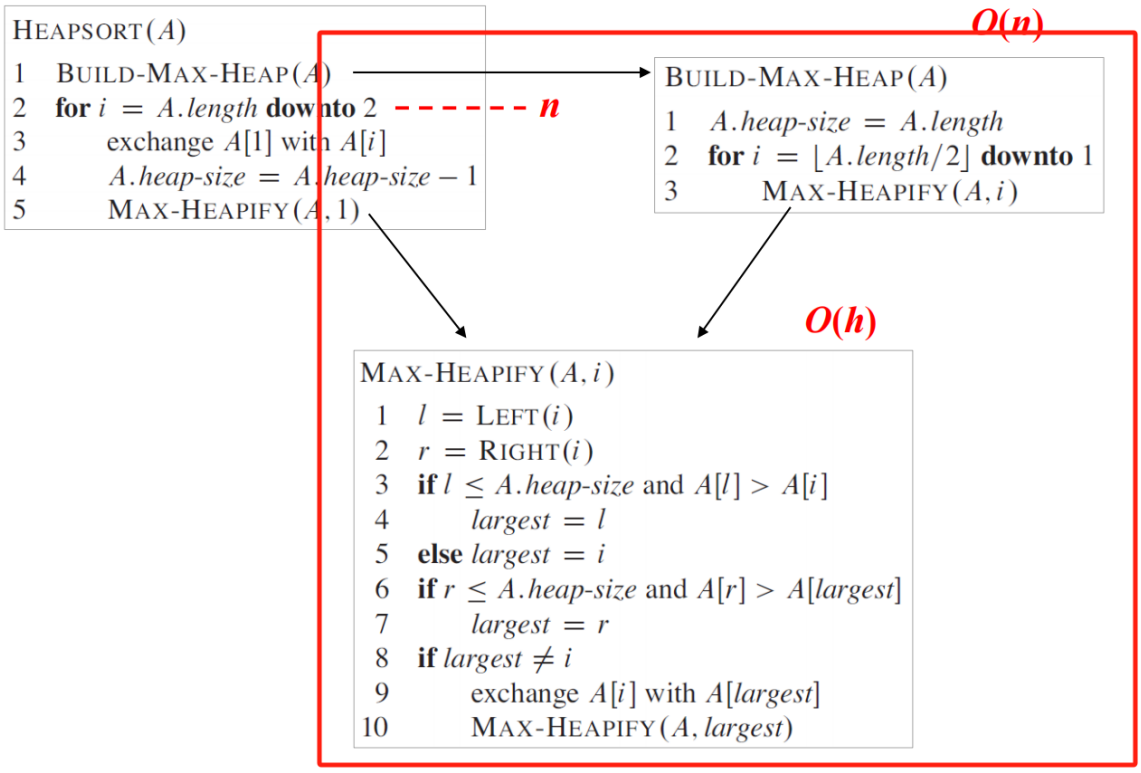
输出样例

```
4
2
3
3 2 1
```

解题思路

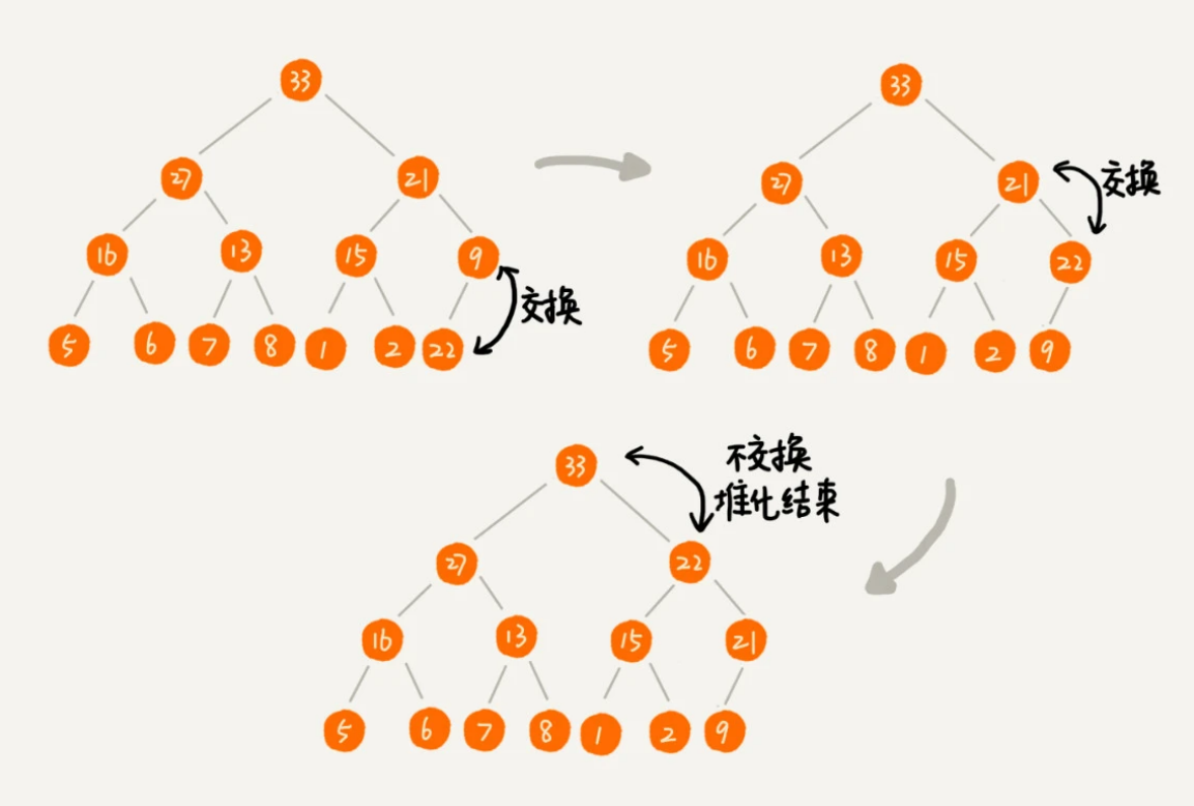
本题可以分为两个部分，前半部分涉及到堆的元素插入，元素删除等维护操作，后半部分则是堆排序内容。

当向一个堆中插入一个元素时，会部分破坏堆的结构，参考《算法导论》及课件，运用`Build_Max_Heap`函数和`Max_Heapify`这两个函数可以维护堆，使其成为大顶堆。



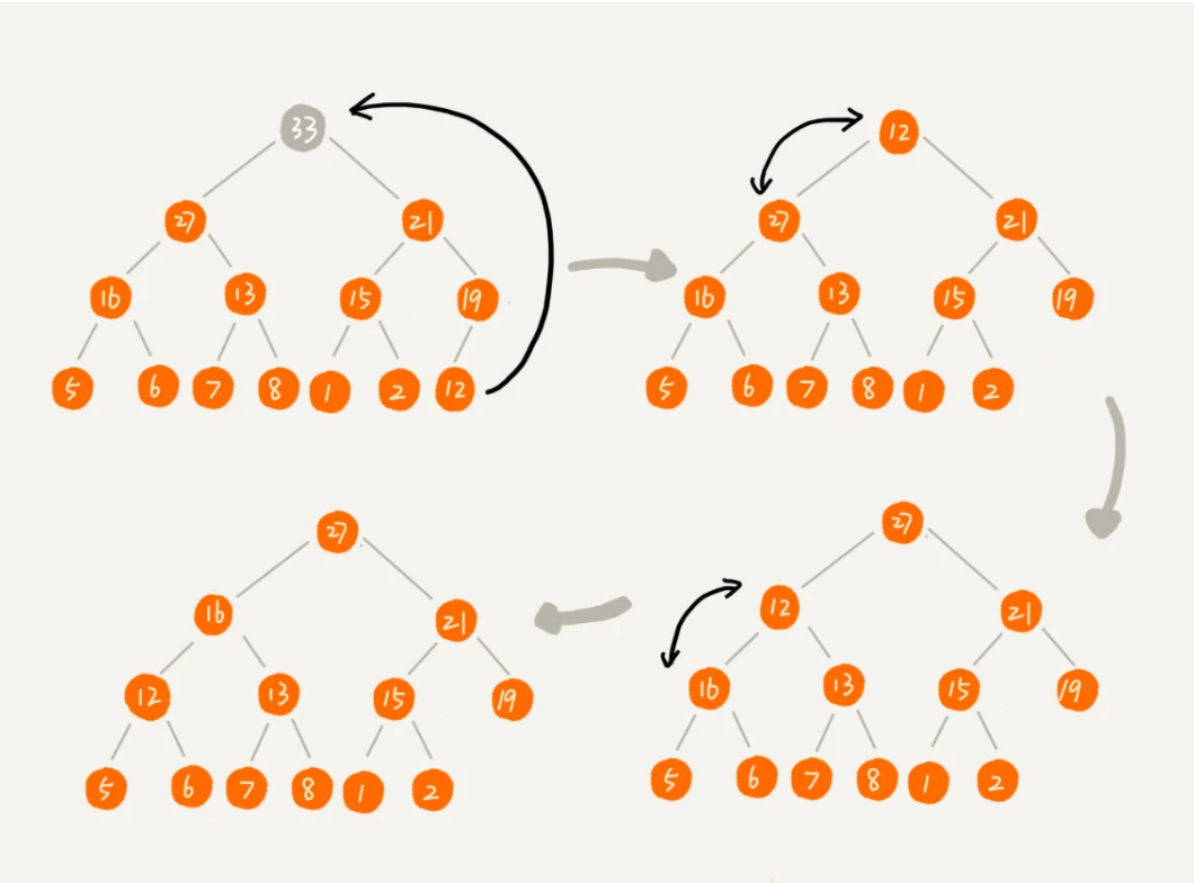
但`Build_Max_Heap`函数时间复杂度为$O(n)$，`Max_Heapify`函数时间复杂度为$O(h)$，而题目中数据量级为10^5，如果每次都调用`Build_Max_Heap`函数来维护堆，会`TLE`。

因为`Build_Max_Heap`函数是用于维护一个完全无序的堆的，如果用于维护本来就部分有序的堆，会浪费很多时间用于比较。因此，对于插入后维护堆的操作，由于原堆是大顶堆，可以在数组末尾插入元素，再判断是否符合堆的特点，即是否小于父节点，如果不小于，跟父节点交换，交换直到小于父节点。 **（自下而上堆化）**



这样就可以在$O(h)$时间内完成维护。

而对于删除堆顶元素，由于堆的二叉树特点限制，如果将后$n-1$个元素依次前移1位，会打乱堆，所以采用的方法是用最后一个元素替换掉堆顶元素，再效仿上述堆化方法**自上而下堆化**。



这就完成了前半部分的堆元素插入和删除及维护操作

后半部分的堆排序内容可以用以下伪代码表示

```
for i from n to 2:
    swap(h[1],h[i])
    adjust(i)//自上而下堆化
```

最后可得到从小到大排序的数组，逆序输出即得从大到小排序的数组

代码实现

主要用到三个函数：

- 插入&自下而上堆化：

```
void insert(int data, int* lenptr) {
    (* lenptr)++;
    int len = *lenptr;
    heap[len] = data;
    for (int i = len; i / 2 > 0 && heap[i] > heap[i / 2]; i /= 2) {
        int temp = heap[i];
        heap[i] = heap[i / 2];
        heap[i / 2] = temp;
    }
}
```

- 自上而下堆化

```
void adjust(int len) {
    int j = 2;
    int temp = heap[1];
    while (j <= len) {
        if (j < len && heap[j] < heap[j + 1]) {
            j++;
        }
        if (temp >= heap[j]) {//退出while循环
            break;
        }
        heap[j / 2] = heap[j];
        j = j * 2;
    }
    heap[j / 2] = temp;
}
```

- 堆排序

```
void heapsort(int len) {
    int temp;
    for (int i = len; i >= 2; i--) {
        temp = heap[1];
        heap[1] = heap[i];
        heap[i] = temp;
        len -= 1;
        adjust(len);
    }
}
```