

C2: D题题解

D 简单的堆

时间限制: 1000ms 内存限制: 65536kb

通过率: 89/143 (62.24%) 正确率: 89/448 (19.87%)

本题必须使用 C 语言提交。

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。

C2: D题题解

解题思路：

本题是标准的模板题，实现题目描述的功能只需要两个函数，分别负责插入元素操作与弹出堆顶元素操作。

插入操作insert(x)，将元素x加入堆中，并保持大根堆的性质

弹出操作pop()，弹出堆顶元素并维持大根堆的性质

访问堆顶元素直接调用存储数组heap[1]即可，无需单独实现

因为是模板题而且课上详细讲解过，这里不再复述各操作的具体实现原理，接下来直接上代码，实现过程和课件中大同小异。

C2: D题题解

insert函数:

```
void insert(int x)
{
    int temp = heap[cnt];
    cnt++;
    int i;
    for (i = cnt; heap[i / 2] < x; i /= 2)
    {
        heap[i] = heap[i / 2];
    }
    heap[i] = x;
}
```

pop函数:

```
int pop(void)
{
    int parent;
    int child;
    int temp = heap[cnt];
    int x = heap[1];
    cnt--;
    for (parent = 1; parent * 2 <= cnt; parent = child)
    {
        child = parent * 2;
        if (heap[child] < heap[child + 1]) { child++; }
        if (temp < heap[child]) { heap[parent] = heap[child]; }
        else { break; }
    }
    heap[parent] = temp;
    return x;
}
```

这里并没有像算法课件一样，专门编写进行堆化操作的函数MAX-HEAPIFY，而是在插入和弹出元素的同时进行复杂度为 $O(\log n)$ 的操作，保证了大根堆的性质，而实际原理其实和课件中的堆化操作类似，不再细讲。

C2: D题题解

作为一道模板题，实在没有什么可讲，还是建议大家参考算法课程课件，把堆的板子写出来，功能也更加齐全。最后附上本人完整代码：

```
2  #include<stdio.h>
3  #include<stdlib.h>
4
5  int heap[100000];
6  int cnt, n;
7
8  void insert(int x)
9  {
10     int temp = heap[cnt];
11     cnt++;
12     int i;
13     for (i = cnt; heap[i / 2] < x; i /= 2)
14     {
15         heap[i] = heap[i / 2];
16     }
17     heap[i] = x;
18 }
19
20 int pop(void)
21 {
22     int parent;
23     int child;
24     int temp = heap[cnt];
25     int x = heap[1];
26     cnt--;
27     for (parent = 1; parent * 2 <= cnt; parent = child)
28     {
29         child = parent * 2;
30         if (heap[child] < heap[child + 1]) { child++; }
31         if (temp < heap[child]) { heap[parent] = heap[child]; }
32         else { break; }
33     }
34     heap[parent] = temp;
35     return x;
36 }
```

```
38  int main()
39  {
40     heap[0] = 0x7fffffff;
41     scanf("%d", &n);
42     while (n--)
43     {
44         int op, x;
45         scanf("%d", &op);
46         if (op == 1)
47         {
48             scanf("%d", &x);
49             insert(x);
50         }
51         else if (op == 2)
52         {
53             pop();
54         }
55         else if (op == 3)
56         {
57             printf("%d\n", heap[1]);
58         }
59     }
60     int C = cnt;
61     for (int i = 1; i <= C; i++)
62     {
63         printf("%d ", pop());
64     }
65 }
```