E2-A problem

题目描述

本题是堆的一种拓展应用，你的任务是实现一个小根堆，即根节点为所有元素最小值的堆，需要支持插入元素、查询堆顶元素、删除堆顶元素、删除堆中任意元素四个操作。

数据规模&输入格式

第一行一个正整数 n（1≤n≤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 xx（1≤x≤10^9）。
- 2：删除堆顶元素。
- 3：查询堆顶元素。
- 4 x：删除堆中元素 x（1≤x≤10^9），若有多个相同的元素，只需删除一个。

数据保证进行操作 2,3 时堆非空，进行操作 4 时 x 必为堆中元素。

题解思路

- 首先编写维护堆性质的函数min_heapify，可以实现对于堆中任意一点作为堆顶的子堆的维护。创建数组heap[MAX]用于存储堆的全部元素。
- 对于插入元素，由于插入元素后元素变多，需要将堆的size++，然后将新元素插入到堆的最后。之后根据堆在数组中的索引一路遍历该元素的父结点，若父结点小于自己则交换父子结点，直到堆顶。
- 对于删除堆顶元素，将堆顶元素设置为heap[size]，即为堆中的最后一个元素。之后size--，然后对对顶调用min_heapify，维护堆的性质。【注意此处不能直接将堆顶元素改为0，对堆顶维护堆的形式，再将size--。因为维护堆性质后0不一定移动都堆中的最后一个元素的位置。】
- 对于查询堆顶元素，直接返回heap[1]即可。
- 对于删除堆中元素 x，首先遍历堆数组，找到需要删除元素第一次出现的下标。之后将该位置元素设置为heap[size]，即为堆中最后一个元素，之后size--，然后对此位置为堆顶的子堆调用min_heapify。之后根据堆在数组中的索引一路遍历此位置元素的父结点，若父结点大于自己则交换父子结点，直到堆顶。

代码

```
#include <stdio.h>

#define MAX 100005

int heap[MAX] = {0};

void exchange(int *a, int *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
}

void min_heapify(int i, int size) {
```

```
    if (i > size) {
        return;
    }
    int l = 2 * i;
    int r = 2 * i + 1;
    int smallest;
    if (l <= size && heap[l] < heap[i]) {
        smallest = l;
    }
    else {
        smallest = i;
    }
    if (r <= size && heap[r] < heap[smallest]) {
        smallest = r;
    }
    if (smallest != i) {
        exchange(&heap[i], &heap[smallest]);
        min_heapify(smallest, size);
    }
}

int main() {
    int t;
    scanf("%d", &t);
    int size = 0;
    while (t--) {
        int op;
        scanf("%d", &op);
        if (op == 1) {
            size++;
            int key, i = size;
            scanf("%d", &key);
            heap[size] = key;
            while (i > 1 && heap[i/2] > heap[i]) {
                exchange(&heap[i], &heap[i/2]);
                i = i / 2;
            }
        }
        else if (op == 2) {
            heap[1] = heap[size];
            heap[size] = 0;
            size--;
            min_heapify(1, size);
        }
        else if (op == 3) {
            printf("%d\n", heap[1]);
        }
        else if (op == 4) {
            int key = 0;
            scanf("%d", &key);
            for (int i = 1; i <= size; i++) {
                if (heap[i] == key) {
                    heap[i] = heap[size];
                    size--;
                    min_heapify(i, size);
```

```
                    while (i > 1 && heap[i/2] > heap[i]) {
                        exchange(&heap[i], &heap[i/2]);
                        i = i / 2;
                    }
                    break;
                }
            }
        }
    }
    int size_store = size;
    for (int i = size; i > 1; i--) {
        exchange(&heap[1], &heap[i]);
        size--;
        min_heapify(1, size);
    }
    for (int i = size_store; i >= 2; i--) {
        printf("%d ", heap[i]);
    }
    printf("%d", heap[1]);
    return 0;
}
```