大作业

一、小蓝鲸的奇妙冒险–第一季

1、解题思路

数据结构：数组

数据存储方式：顺序存储

本题是求一个数组中第k大的数，AC的时间复杂度要求为O(n)，于是采用快排中的划分思想，将数组中按照一个数划分为两半，并按照m的大小相应递归调用，时间复杂度近似为O(n)。

2、核心代码+注释

A C++ 复制代码

```cpp
#include <iostream>
using namespace std;

int partition(int *array, int start, int end, int m){
    // 采用快排中的划分思想，将数组按照一个数划分为两半，左大右小，
    // 按照m的大小递归左边的部分或右边的部分，或是直接输出
    if(start == end){ // base case
        return array[start];
    }
    int pat = array[end];
    int index = start;
    for(int i = start; i < end; i++){
        if(array[i] >= pat){
            int temp = array[i];
            array[i] = array[index];
            array[index] = temp;
            index ++;
        }
    }
    array[end] = array[index];
    array[index] = pat;
    if(index == m){
        return pat;
    }
    else if(index > m){
        return partition(array, start, index - 1, m);
    }
    else{
        return partition(array, index + 1, end, m);
    }
}

int main(){
    int m, n;
    cin >> m >> n;
    m --;  // 程序从0开始计数，因此m自减1
    int array[n];
    for(int i = 0; i < n; i ++){
        cin >> array[i];
    }
    int result;
    result = partition(array, 0, n - 1, m);
    cout << result << endl;
}
```

3、OJ运行结果截图

Test #1:	score: 10	Accepted	time: 2626ms	memory: 42528kb
Test #2:	score: 10	Accepted	time: 3ms	memory: 3496kb
Test #3:	score: 10	Accepted	time: 3ms	memory: 3380kb
Test #4:	score: 10	Accepted	time: 3ms	memory: 3420kb
Test #5:	score: 10	Accepted	time: 2ms	memory: 3328kb
Test #6:	score: 10	Accepted	time: 2ms	memory: 3372kb
Test #7:	score: 10	Accepted	time: 2ms	memory: 3448kb
Test #8:	score: 10	Accepted	time: 4ms	memory: 3460kb
Test #9:	score: 10	Accepted	time: 0ms	memory: 3408kb
Test #10:	score: 10	Accepted	time: 5ms	memory: 3420kb

二、小蓝鲸的奇妙冒险–第二季

1、解题思路

数据结构：二叉树

数据存储方式：顺序存储

本题是求一个数组前i个数中的第[x/m]大的数，AC的时间复杂度要求为O(nlgn)，由于要输出n个数，所以O(n)的循环无法避免，只能要求在平均为O(lgn)的时间内找出第[x/m]大的数，这要求我们必须利用前面已比较获取的信息，因此采用堆结构来存储信息。由于题目特性，每隔m次，要求的第[x/m]大的数向后移动一位，因此建立一个最大堆，每隔m次循环删除掉堆顶的最大数，即可实现输出第[x/m]大的数。

但这样做有一个小问题，即如果后面循环中输入的数大于最大堆的堆顶数值，则需要将刚删除的数重新加回来，因此，再维护一个最小堆，将最大堆删除掉的数值存入最小堆，即可保存第[x/m]–1大的数，可以快速弥补第[x/m]大的数的空缺。

由于最大最小堆堆插入，删除操作时间复杂度均为O(lgn)，查询复杂度为O(1)，所以可以在O(lgn)的时间内找出第[x/m]大堆数。

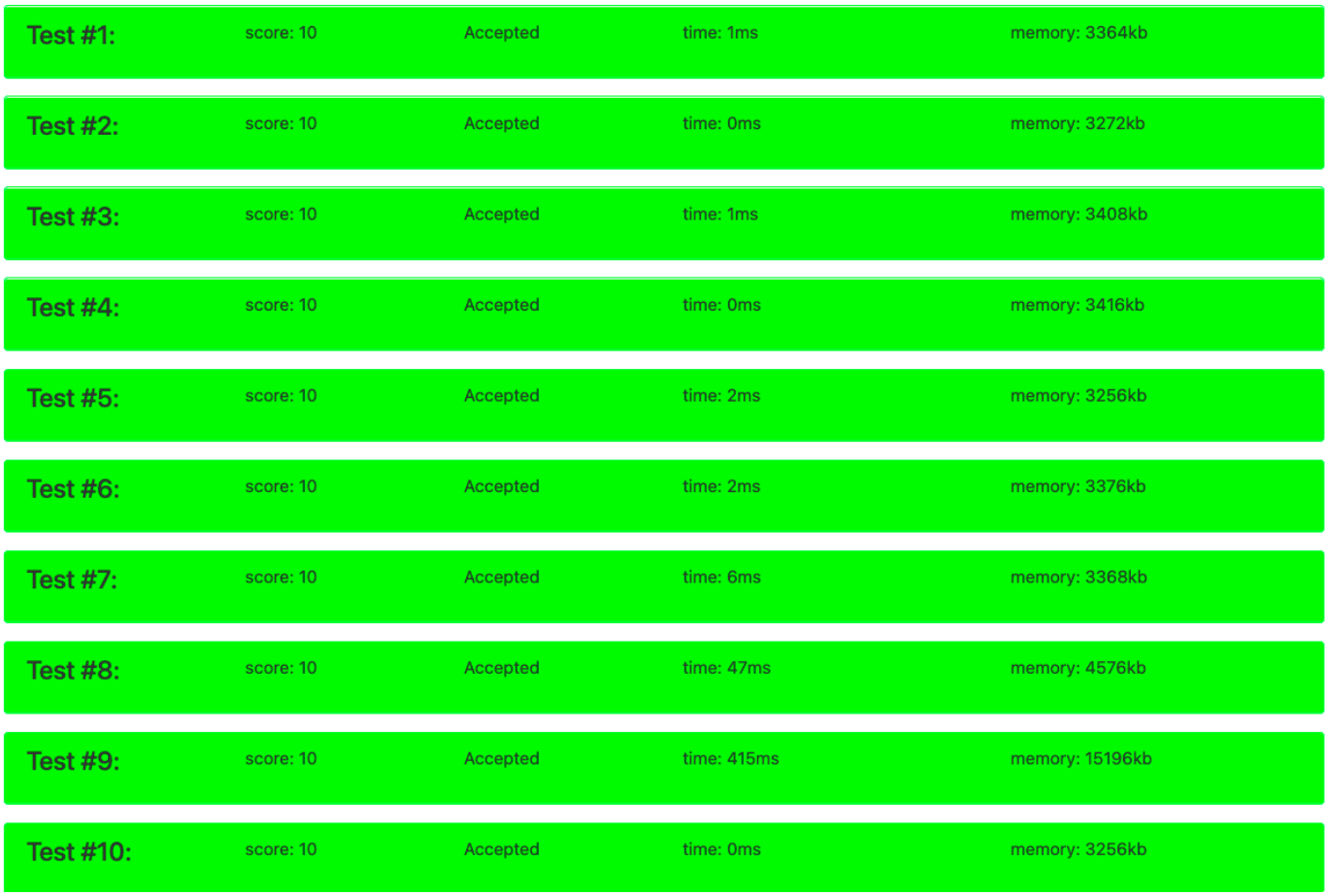
2、核心代码+注释

C++ | 复制代码

```cpp
#include <iostream>
using namespace std;

// 最大堆与最小堆的建立由于篇幅过长，故省略
// 大致结构如下
class maxHeap{
    int *array;
    int len;
public:
    maxHeap(int n);
    void add(int number);
    void pop();
    int output();
private:
    void downward_adjust(int position);
    void upward_adjust(int position);
};

int main(){
    int m, n;
    cin >> m >> n;
    int array[n];
    for(int i = 0; i < n; i ++){
        cin >> array[i];
    }
    // 分别建立最大堆与最小堆
    maxHeap *maxheap = new maxHeap(n);
    minHeap *minheap = new class minHeap(n);
    int t = 0, tt = 0;
    // 最大堆的堆顶永远是第[x/m]大的数
    // 最小堆用来存放目前大于第[x/m]大的数

    for(int i = 0; i < n; i ++){
        // 每经过[x/m]次循环，都将最大堆的堆顶拿出存入最小堆
        if(tt == 1 && i % m == 0){
            minheap->add(maxheap->output());
            maxheap->pop();
            t = 1;
        }
        if(t == 1 && array[i] > minheap->output()){
            // 新来的数先和最小堆比较，如果大于最小堆的堆顶
            // 则将最小堆的堆顶拿出存入最大堆，新来的数放入最小堆
            minheap->add(array[i]);
            maxheap->add(minheap->output());
            minheap->pop();
```

```
46              }
47              else
48                  // 否则将新来的数放入最大堆
49                  maxheap->add(array[i]);
50              // 每次都输出最大堆的堆顶
51              cout << maxheap->output() << " ";
52              tt = 1;
53          }
54          cout << endl;
55
56      }
```

3、OJ运行结果截图



三、小蓝鲸的奇妙冒险-第三季

1、解题思路

数据结构：数组

数据存储方式：顺序存储

本题是求单源最短路径，使用Dijkstra算法即可解决问题，时间复杂度为O(n^2)。

2、核心代码+注释

C C++ 复制代码

```cpp
#include <iostream>
using namespace std;

#define maxV 10000001

int main(){
    int n, m, s, e;
    cin >> n >> m >> s >> e;
    int country[n][n];
    for(int i = 0; i < n; i ++){
        for(int j = 0; j < n; j ++){
            if(i == j){
                country[i][j] = 0;
            }
            else
                country[i][j] = maxV;
        }
    }
    bool S[n];     // S用来判断各点与起点之间是否有道路连通
    int record[n]; // record用来存放起点s距各点的距离
    int path[n];   // path用来记录各点到起点的路径
    for(int i = 0; i < m; i ++){
        int p;
        cin >> p;
        int temp[p];
        int value[p - 1];
        for(int j = 0; j < p; j ++){
            cin >> temp[j];
        }
        for(int j = 0; j < p - 1; j ++){
            cin >> value[j];
            country[temp[j]][temp[j + 1]] = value[j];
            country[temp[j + 1]][temp[j]] = value[j];
        }
    }
    // 采用Dijkstra算法
    for(int i = 0; i < n; i ++){
        record[i] = country[s][i]; // 用起点到各点的直接距离初始化record数组
        S[i] = false;
        if(i != s && record[i] < maxV)
            path[i] = s;           // 如果record有值则说明该点与起点有路径相连
        else
            path[i] = -1;          // 否则说明目前并无路径相连
    }
    S[s] = true;
```

```
46        int min;
47        for(int i = 0; i < n; i ++){
48            min = maxV;
49            int u = s;
50            for(int j = 0; j < n; j ++){ // 从剩余节点中找出离目前连通集合最短的点u
51                if(S[j] == false && record[j] < min){
52                    u = j; min = record[j];
53                }
54            }
55            S[u] = true;
56            for(int j = 0; j < n; j ++){
57                // 比较加入点u后剩余节点与集合间点最短距离有无变化，用最小值替换
58                int w = country[u][j];
59                if(S[j] == false && w < maxV && record[u] + w < record[j]){
60                    record[j] = record[u] + w;
61                    path[j] = u;
62                }
63            }
64        }
65        if(record[e] == maxV){
66            cout << -1 << endl;
67        }
68        else{
69            cout << record[e] << endl;
70        }
71    }
```

3、OJ运行结果截图

Test #1:	score: 10	Accepted	time: 1ms	memory: 3376kb
Test #2:	score: 10	Accepted	time: 0ms	memory: 3304kb
Test #3:	score: 10	Accepted	time: 0ms	memory: 3332kb
Test #4:	score: 10	Accepted	time: 0ms	memory: 3468kb
Test #5:	score: 10	Accepted	time: 0ms	memory: 3460kb
Test #6:	score: 10	Accepted	time: 4ms	memory: 3472kb
Test #7:	score: 10	Accepted	time: 5ms	memory: 3736kb
Test #8:	score: 10	Accepted	time: 6ms	memory: 4204kb
Test #9:	score: 10	Accepted	time: 4ms	memory: 5728kb
Test #10:	score: 10	Accepted	time: 9ms	memory: 7128kb