Exercise 02

201918008629001 肖阳

1、

Solution:

思路：①对于给定的多重集 S，先对其进行快速排序，时间复杂度为O(nlogn)。

②对于排序后的数组取中位数，并找到与中位数等值的数的下标范围，并将这个数组分为左、中、右三段，记录中位数的值及其重数。例如：

1 2 2 3 3 3 5 7 7 9

的中位数为 3，与其等值的数的下标范围是[3,6)。故该数组分为三段

[0,3)、 [3,6)、[6,10)

③如果左段长度大于中位数的重数，则对左段数组递归执行上述过程，否则不对左段进行搜索；如果右段长度大于中位数的重数，则对右段数组递归执行上述过程，否则不对右段进行搜索。

④比较左、右、中三段的重数，重数最大的数即为众数。时间复杂度为O(nlogn)。

代码实现：

```
#include <iostream>
#include <vector>
using namespace std;
class ModeSolution {
public:
    vector<int> multiset;
    int N;
    int Mode;
    int MaxCnt = 0;

    ModeSolution(int n);
    ~ModeSolution();

    void Split(int start,int end, int &left, int &right);
    void GetMode(int start, int end);
    void QuickSort(vector<int> &v,int start,int end);
};
ModeSolution::ModeSolution(int n) {
    int temp;
    N = n;
    for (int i = 0; i < n; i++) {
        temp = rand() % 20 + 1;
        multiset.push_back(temp);
    }
    multiset.push_back(-1);
}
ModeSolution::~ModeSolution() {
```

```
    vector<int>().swap(multiset);
}
void ModeSolution::QuickSort(vector<int> &v, int start, int end) {
    if (v.size() == 0)
        return;
    if (start >= end)
        return;
    int cardinal = v[start];
    int i = start, j = end;
    while (i != j) {
        while (v[j] >= cardinal && i<j) {
            --j;
        }
        while (v[i] <= cardinal && i<j) {
            ++i;
        }
        if (i<j) {
            int temp;
            temp = v[i];
            v[i] = v[j];
            v[j] = temp;
        }

    }
        v[start] = v[i];
        v[i] = cardinal;
        QuickSort(v, start, i-1);
        QuickSort(v, i+1, end);

}
void ModeSolution::Split(int start,int end, int &left, int &right) {
    int mid = (start + end) / 2;
    for (left = 0; left < N; left++) {
        if (multiset[left] == multiset[mid]) {
            break;
        }
    }
    for (right = left + 1; right < N; right++) {
        if (multiset[right] != multiset[mid]) {
            break;
        }
    }
}
void ModeSolution::GetMode(int start,int end) {
```

```
    if (start >= end)
        return;
    int left, right;
    Split(start, end, left, right);
    int mid = (end + start) / 2;
    int cnt = right - left;
    if (cnt > MaxCnt) {
        MaxCnt = cnt;
        Mode = multiset[mid];
    }
    if (left - start + 1 > cnt) {
        GetMode(start, left - 1);
    }
    if (end - right + 1>cnt){
        GetMode(right, end);
    }
}
int main(int argc,char **argv) {
    ModeSolution s1(20);
    s1.QuickSort(s1.multiset,0,s1.N-1);
    s1.GetMode(0, s1.N-1);
    cout << "众数为：" << s1.Mode<<"重数为："<<s1.MaxCnt<<endl;
    system("pause");
    return 0;
}
```