Assignment 2

> CHD 24 fall Advanced Algorithm Design and Analysis

2-3

> 给定一个由 n 个互不相同的数组成的集合 S，及一个正整数 $k \leq n$，试设计一个 $O(n)$ 时间算法找出 S 中最接近 S 的中位数的 k 个数。
> Given a set S consisting of n distinct numbers and a positive integer $k \leq n$, try to design an $O(n)$ time algorithm to find the k numbers closest to the median of S in S.

Solution:

计算中位数，接下来计算每个数和中位数的差，按照差的绝对值做topK选择$O(kn)$，输出结果

> 随机化快速排序
>
> 概括： 为避免快速排序算法在整体有序的数据上出现退化现象，随机选取轴值而不是直接取用第一个
>
> 由于随机化，基本不会出现退化，对于topK排序问题，平均时间复杂度 $O(n)$，最坏 $O(n^2)$

> BFPRT线性查找算法 Blum et al. (1973)
>
> 概括：类似快速排序，每次选择轴值分块查找，对于轴值的选择，在给定的区间中每5个分组分别取中位数（插入排序），对于各组中位数的结果继续取中位数（递归调用中位数算法），直到取出这些中位数们55一组的中位数，这个数跟真正的中位数八九不离十
>
> 算法的时间最坏情况下也是 $O(n)$

快速选择算法的实现请见源码, 这里我实现了多种轴值选择方案，请在`QuickSelect`函数调用时指定

完整版代码见2_3

```cpp
//
// Created by tianq on 24-11-29.
//
#include <algorithm>
#include <iostream>
#include <vector>
using namespace std;

constexpr int groupSize = 5;

/**
 * pick median of given array
 * @param arr the array to be parsed
 * @return median of given array, when arr size is odd, bigger one is chosen
 * @attention not a mathematical median,
 */
template<typename NumType>
NumType BruteMedian(vector<NumType> arr) {
    ranges::sort(arr);
    return arr[arr.size() / 2];
}
```

```cpp
/**
 * pick a likely median element from vector,used by BFPRT
 * @param arr the array to be parsed
 * @return element from array
 */
template<typename NumType>
NumType PickBfprt(const vector<NumType> &arr) {
    if (arr.size() <= groupSize) return BruteMedian(arr);

    auto it = arr.begin();
    vector<NumType> medians;
    while (it != arr.end()) {
        vector<NumType> group;
        while (group.size() < groupSize && it != arr.end()) {
            group.push_back(*it);
            ++it;
        }
        medians.push_back(BruteMedian(group));
    }

    return PickBfprt(medians);
}

/**
 * get nth element in given array
 * @param arr the array you want to determine its nth element
 * @param targetIndex [0,arr.size()-1], return nth smallest
 * @param PickPivot Pivot Picking strategy, PickRandom should be good enough
 * @return nth smallest element
 */
template<typename NumType>
NumType QuickSelect(vector<NumType> arr, const unsigned long long targetIndex,
NumType (*PickPivot)(const vector<NumType>&) = PickBfprt) {
    if (arr.size() == 1) return arr.front();

    const NumType pivot = PickPivot(arr);

    vector<NumType> left, mid, right;
    for (NumType &num: arr) {
        if (num < pivot) left.push_back(num);
        else if (num > pivot) right.push_back(num);
        else mid.push_back(num);
    }

    if (targetIndex < left.size())
        return QuickSelect(left, targetIndex, PickPivot);
    if (targetIndex < left.size() + mid.size())
        return mid.front();
    return QuickSelect(right, targetIndex - left.size() - mid.size(), PickPivot);
}

int main (){

    int k = 0, N = 0;
    vector<double> arr;  // 考虑到中位数出现0.5的情况，double更合适
```

```
    cin >> k >> N;
    for (int i = 0; i < N; i++) {
        int tmp = 0;
        cin >> tmp;
        arr.push_back (tmp);
    }

    const double lmid = QuickSelect (arr, (arr.size () - 1) / 2);
    const double rmid = QuickSelect (arr, arr.size () / 2);
    const double mid = (lmid + rmid) / 2;

    vector<double> variations;
    for (const double &i : arr)
        variations.push_back (abs (i - mid));

    double maxVar = QuickSelect (variations, k-1);

    vector<double> ans, ge;
    for (double &i : arr) {
        if (abs (i - mid) < maxVar)
            ans.push_back (i);
        else
            ge.push_back (i);
    }
    for (double &i : ge) { // deal with numbers equal to boundaries
        if (abs (i - mid) <= maxVar && ans.size () < k)
            ans.push_back (i);
    }
    for (const auto &i : ans)
        cout << i << " ";
    cout << endl;
}
```

2-4

> 在一个由元素组成的表中，出现的次数最多的元素称为众数。试写一个寻找众数的算法，并分析其计算复杂性。
>
> In a table composed of elements, the element that appears the most often is called the mode. Write an algorithm to find the mode and analyze its computational complexity

Solution:

使用哈希表统计每一个元素出现的次数，接下来，输出哈希表中个数最多的一项

时间复杂度$O(n)$

完整版代码见2_4

```
//
// Created by tianq on 24-11-29.
//
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;
```

```cpp
int main() {
    cout.precision(4);
    int N = 0;
    vector<int> arr;

    cin >> N;
    for (int i = 0; i < N; i++) {
        double tmp;
        cin >> tmp;
        arr.push_back(tmp);
    }

    unordered_map<int, int> map;
    for (const int &i: arr) {
        if (!map.contains(i))
            map[i] = 1;
        else
            map[i] = map[i] + 1;
    }
    int modeValue = 0, modeCount = 0;
    for (const auto &[fst, snd]: map) {
        if (snd <= modeCount) continue;
        modeCount = snd;
        modeValue = fst;
    }

    cout << modeValue << endl;
    return 0;
}
```