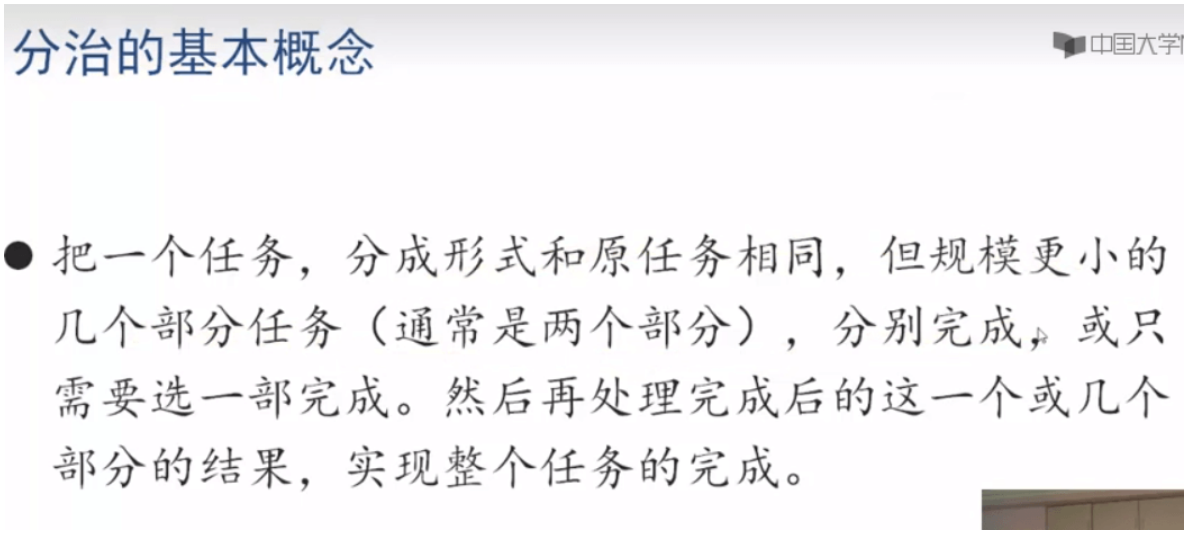
算法

分治算法

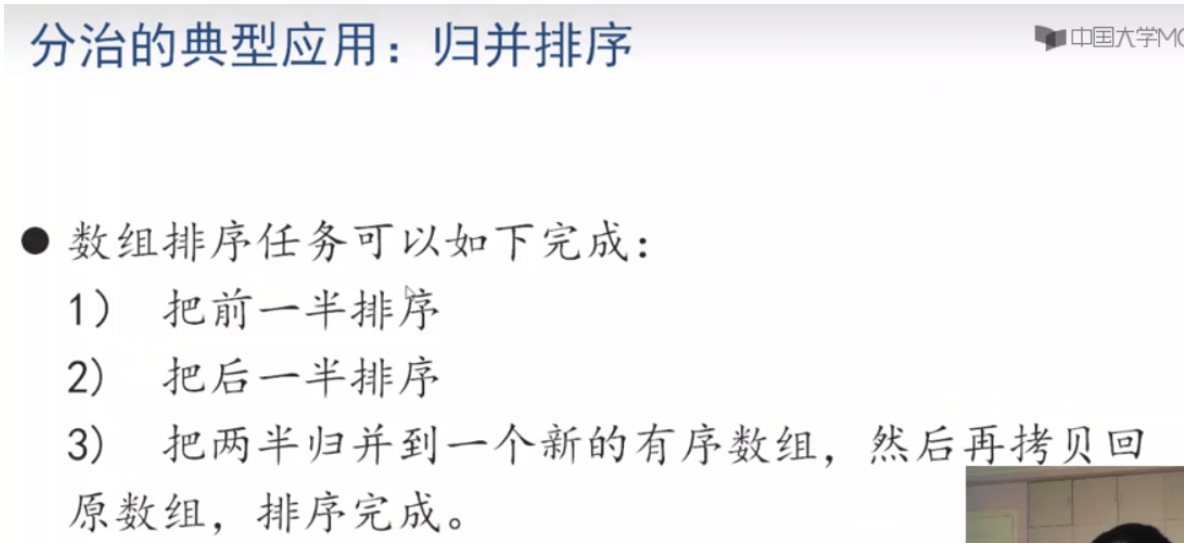
分治算法很多是用递归进行实现的



归并排序

归并排序是稳定的排序.即相等的元素的顺序不会改变，时间复杂度nlogn,最坏时间复杂度也是nlogn，平均时间仅次于快速排序,采用分治思想。

题解



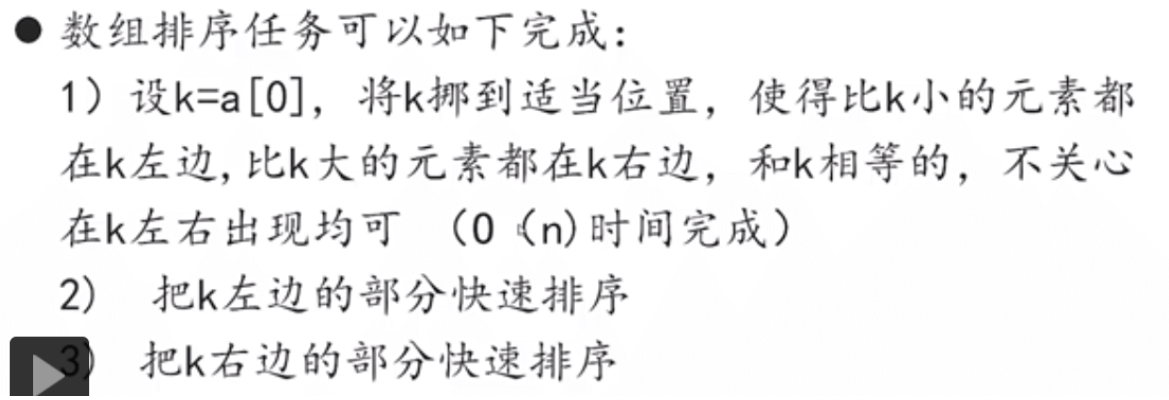
code c++

```
#include<iostream>
#include<vector>
using namespace std;
void Merge(vector<int>&a, int l, int m, int r, vector<int>&b){
    //归并的结果先放在b里面，然后放在a里面
    int p = 0;
    int p1 = l, p2 = m + 1;
```

```
    while (p1 <= m&&p2 <= r){//归并
        if (a[p1] < a[p2])b[p++] = a[p1++];
        else b[p++] = a[p2++];
    }
    while (p1 <= m){ b[p++] = a[p1++]; }
    while (p2 <= r){ b[p++] = a[p2++]; }
    int c = 0;
    for (int i = l; i <= r; ++i)a[i] = b[c++];//拷贝到a里面
}
void MergeSort(vector<int>&a, int l, int r, vector<int>&b){
    if (l < r){
        int m = l + (r - l) / 2;
        MergeSort(a, l, m, b);//分治 一半一半来排序
        MergeSort(a, m+1, r, b);// 递归的终止条件  只剩下一个元素 也就是l=r
        Merge(a, l, m, r, b);//将排序好的 归并在一起
    }
}
void main(){
    vector<int>a = { 1, 58, 98, 254, 0, 4, 56, 1, 2, 35, 898, 45, 1568, 335, 5
};
    int r = a.size()-1;
    vector<int>b(a.size());
    for (int i : a)cout << i << " ";
    MergeSort(a, 0, r, b);
    for (int i : a)cout << i << " ";
    system("pause");
}
```

快速排序

题解



code c++

```
#include<iostream>
#include<vector>
using namespace std;
void QuickSort(vector<int>&a, int l, int r){
    if (l>=r)return;//递归终止条件
    int t = a[l];
    int s = l, e = r;
    while (l<r){
        while (a[r] >= t)--r;
        swap(a[l], a[r]);
```

```
        while (r>l&&a[l] < t)++l;//因为r的值减少了 需要判断是否l<r
        swap(a[l], a[r]);
    }
    QuickSort(a, s, l - 1);
    QuickSort(a, l + 1, e);
}
void main(){
    vector<int>a = { 1, 58, 98, 254, 0, 4, 56, 1, 2, 35, 898, 45, 1568, 335, 5
};
    int r = a.size() - 1;
    for (int i : a)cout << i << " ";
    QuickSort(a, 0, r);
    for (int i : a)cout << i << " ";
    system("pause");
}
```

前m大数

题解

例题：输出前m大的数

中国大学MOOC

引入操作 arrangeRight(k)：把数组(或数组的一部分）前k大的都弄到最右边

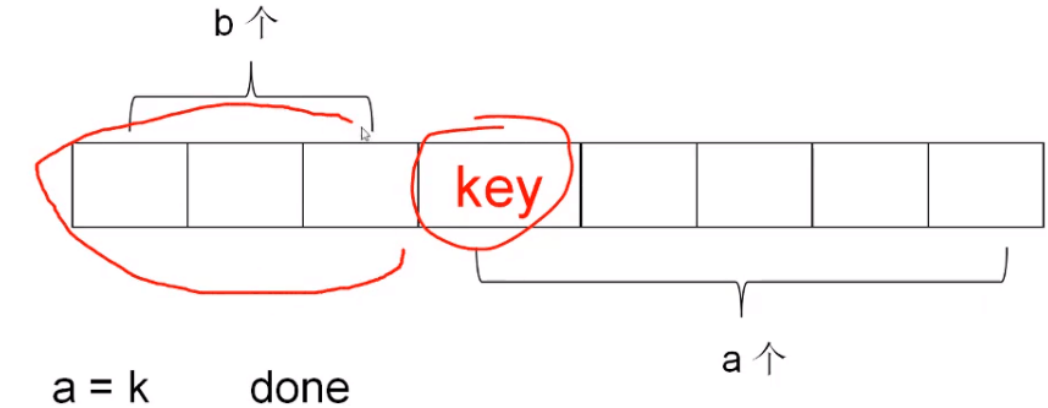
如何将前k大的都弄到最右边

1）设key=a[0]，将key挪到适当位置，使得比key小的元素都在key左边，比key大的元素都在key右边（线性时间完成）

2）选择数组的前部或后部再进行 arrangeRight操作

例题：输出前m大的数

2）选择数组的前部或后部再进行 arrangeRight操作



a = k　done

a > k　对右边a-1个元素再进行arrangeRight(k)

a < k　对左边b个元素再进行arrangeRight(k-a)

例题：输出前m大的数

排序后再输出，复杂度 O(nlogn)

用分治处理：复杂度 O(n+mlogm)

思路：把前m大的都弄到数组最右边，然后对这最右边m个元素排序，再输出

关键 ：O(n)时间内实现把前m大的都弄到数组最右边

code c++

```
#include<iostream>
#include<vector>
#include<queue>
#include<functional>//greater&less的头文件
using namespace std;

void BigM(vector<int>&a, int l, int r, int m) {
    //快排分治思想,时间复杂度 n+mlogm
    int s = l, e = r;
    int p = a[l];
    while (l < r){
        while (a[r] >= p)--r;
        swap(a[l], a[r]);
        while (l < r&&a[l] < p)++l;
        swap(a[l], a[r]);
    }
    int c = e - l + 1;// 排在p右面的个数
    if (c == m)return;//m个 返回
    else if (c<m)BigM(a, s, l - 1, m - c);//小于m ，在左面继续找剩下的
    else BigM(a, l + 1, e, m);//大于 在右面找m个
}
void BigM1(vector<int>&a, int l, int r, int m){
    //优先级队列 堆排序
    priority_queue <int, vector<int>, greater<int>>res;
    for (int i = l; i <= r; ++i){
        if (res.size() >= m){
            int min = res.top();
            if (a[i]>min){
                res.pop();
                res.push(a[i]);
            }
        }
        else res.push(a[i]);
    }
    while (!res.empty()){
        cout << res.top() << " ";
        res.pop();
```

```
    }

}
void main(){
    vector<int>a = { 1, 58, 98, 254, 0, 4, 56, 1, 2, 35, 898, 45, 1568, 335, 5
};
    int r = a.size()-1;
    int m = 5;
    for (int i : a)cout << i << " ";
    cout << endl;
    BigM(a, 0, r, m);
    for (int i : a)cout << i << " ";
    cout << endl;
    for (int i = 0; i < m; ++i)cout << a[r - i] << " ";
    cout << endl;
    BigM1(a, 0, r, m);
    system("pause");
}
```

逆序数

题解

例题：求排列的逆序数

中国大学MOOC

考虑1, 2, …, n (n <= 100000)的排列i_1, i_2, …, i_n, 如果其中存在j, k, 满足j < k 且 i_j > i_k, 那么就称(i_j, i_k)是这个排列的一个逆序。

一个排列含有逆序的个数称为这个排列的逆序数。例如排列 263451 含有8个逆序(2, 1), (6, 3), (6, 4), (6, 5), (6, 1), (3, 1), (4, 1), (5, 1), 因此该排列的逆序数就是8。

现给定1, 2, …, n的一个排列, 求它的逆序数。

例题：求排列的逆序数

中国大学MOOC

笨办法：$O(n^2)$

分治O(nlogn)：

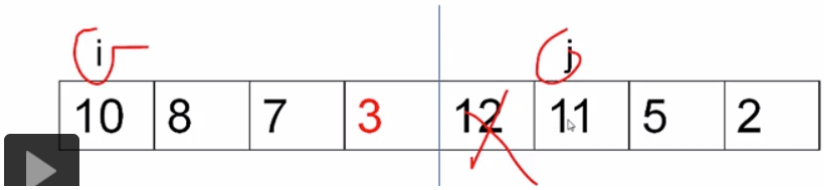
1) 将数组分成两半，分别求出左半边的逆序数和右半边的逆序数

2) 再算有多少逆序是由左半边取一个数和右半边取一个数构成(要求O(n)实现)

例题：求排列的逆序数

中国大学MOOC

2）的关键：左半边和右半边都是排好序的。比如，都是从大到小排序的。这样，左右半边只需要从头到尾各扫一遍，就可以找出由两边各取一个数构成的逆序个数



总结：

由归并排序改进得到，加上计算逆序的步骤

MergeSortAndCount：归并排序并计算逆序数

code c++

```
#include<iostream>
#include<vector>
using namespace std;
int n = 0;

void Merge(vector<int>&a, int l, int m, int r, vector<int>&b){
    //归并的结果先放在b里面，然后放在a里面
    int p = 0;
    int p1 = l, p2 = m + 1;
    while (p1 <= m&&p2 <= r){//归并
        if (a[p1] > a[p2]){
            b[p++] = a[p1++];
            n += r - p2 + 1;
        }
        else b[p++] = a[p2++];
    }
    while (p1 <= m){ b[p++] = a[p1++]; }
    while (p2 <= r){ b[p++] = a[p2++]; }
    int c = 0;
    for (int i = l; i <= r; ++i)a[i] = b[c++];//拷贝到a里面
}
void MergeSort(vector<int>&a, int l, int r, vector<int>&b){
    if (l >= r)return;//递归终止条件
    int m = l + (r - l) / 2;
    MergeSort(a, l, m, b);//分治 一半一半来排序
```

```
    MergeSort(a, m+1, r, b);// 递归的终止条件  只剩下一个元素 也就是l=r
    Merge(a, l, m, r, b);//将排序好的 归并在一起
}
void main(){
    vector<int>a = { 1, 7, 2, 9, 6, 4, 5, 3 };
    int r = a.size()-1;
    vector<int>b(a.size());
    for (int i : a)cout << i << " ";
    cout << endl;
    MergeSort(a, 0, r, b);
    for (int i : a)cout << i << " ";
    cout << endl;
    cout << n % 1000000007;
    system("pause");
}
```