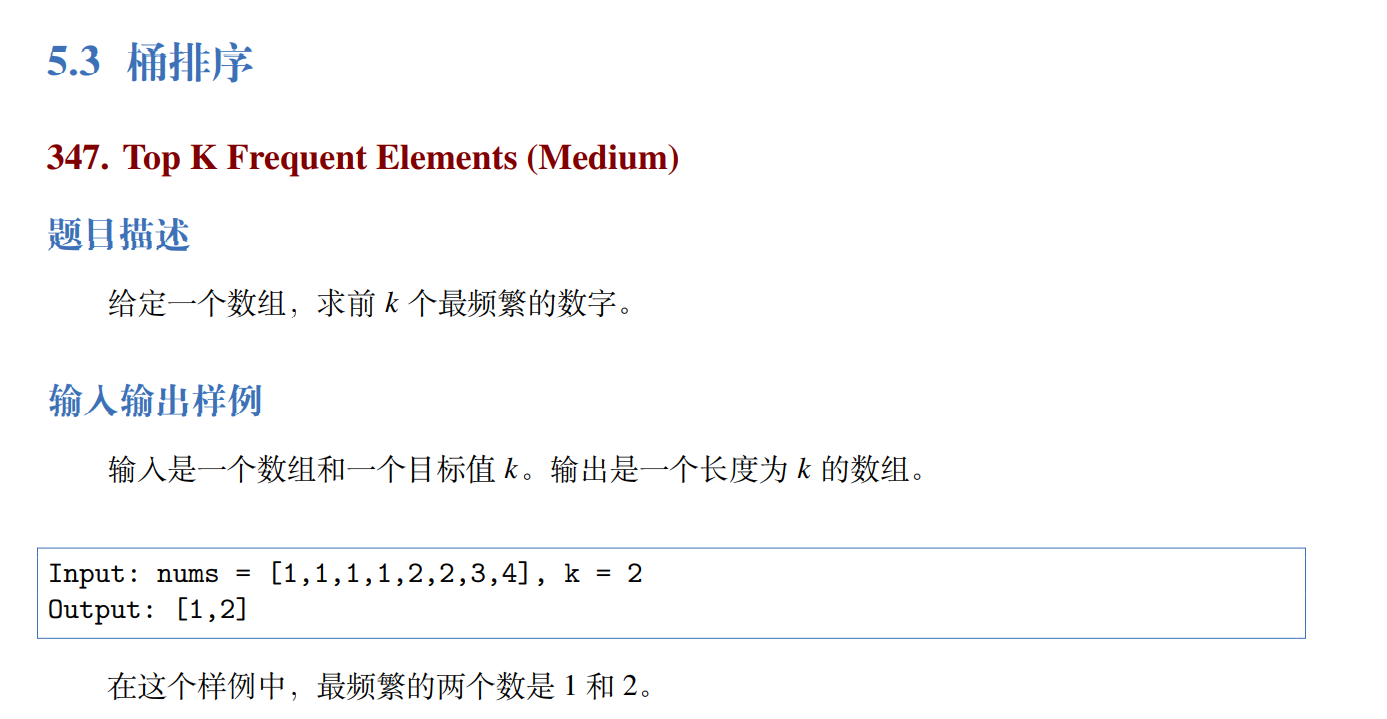
# 经典排序算法：

## 快速排序：

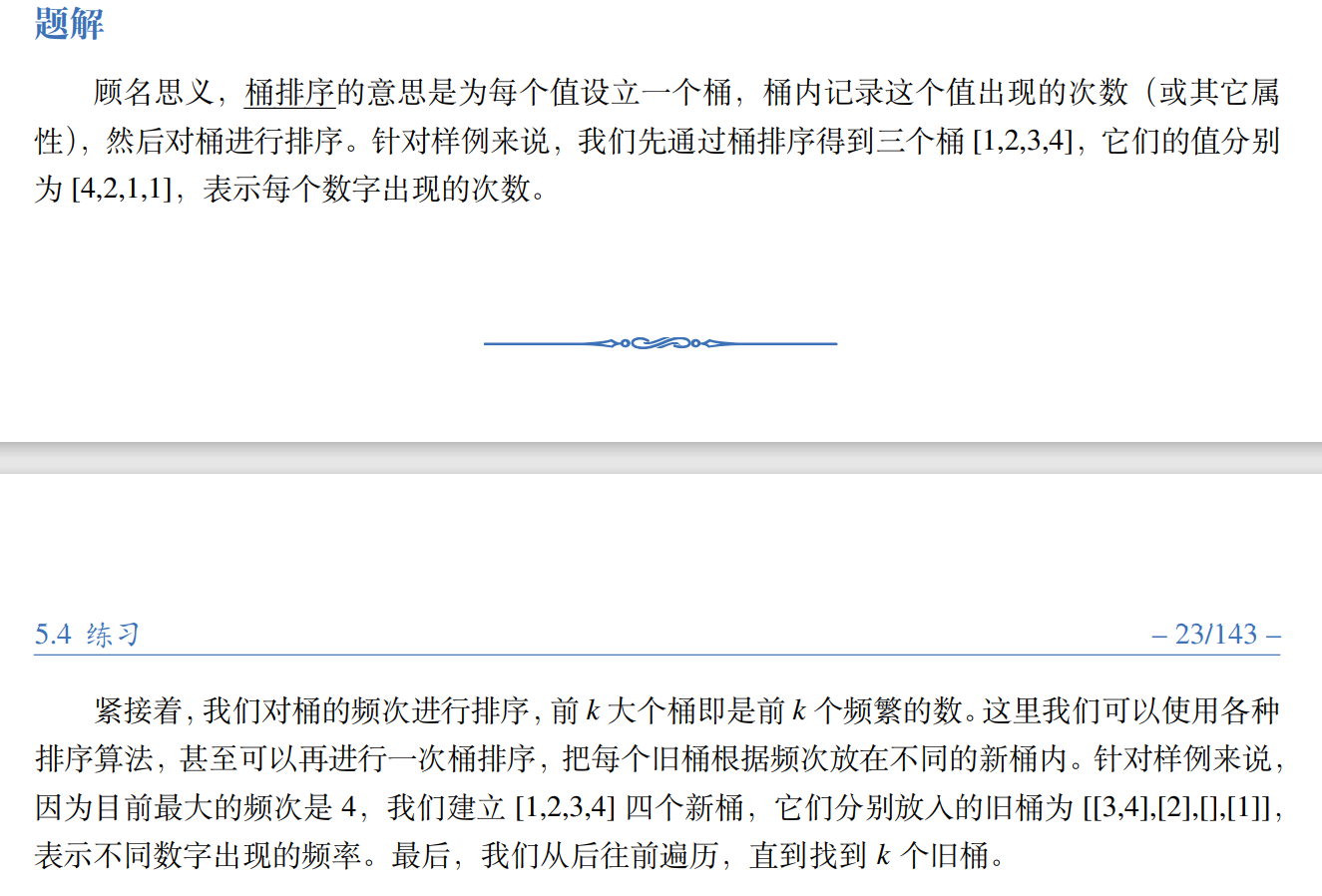
* **快速排序函数是个递归函数；要设置递归函数的终止条件：当左边界begin大于有边界end时终止**
* **该递归函数右三个参数：1：要排序的数组以引用的方式传入；2：传入要排序数组的边界；**
* **理解依次快排的作用：将大于枢轴的数和小于枢轴的数分别放在枢轴数的两边**
* **循环条件设置为left<right:这样终止时的情况就是left==right，而此时的位置正是枢轴数所在的位置**
* **通过现场演示一遍来确定具体的过程：**
* **从左边扫描，直到有数小于枢轴；交换**
* **从右往左扫描，直到有数大于枢轴；交换**

#include<iostream>  
#include<vector>  
#include <iomanip>  
#include<fstream>  
#include<bitset>  
#include<unordered\_map>  
using namespace std;  
  
  
class Solution {  
private:  
public:  
 //掌握快速排序  
 //快排的函数是递归函数  
 void quickSort(vector<int>&nums,int begin,int end) {  
 //设置递归函数的终止条件  
 //此处等于的时候不能return -1  
 //等于的时候直接return分界线  
 if (begin > end)return;  
 //1:边界参数，left、right;  
 int left = begin, right = end;  
 int pivot = nums[left];//设置第一个数为枢轴  
 while (left < right) {  
 //此次快排的作用是将大于枢轴的数放在枢轴右边；小于枢轴的数放在枢轴左边  
 while (left < right && nums[right] >= pivot)--right;  
 //循环后，nums[right]的值小于枢轴处的值  
 nums[left] = nums[right];  
 while (left < right && nums[left] <=pivot)++left;  
 //循坏后，nums[left]的值大于枢轴处的值  
 nums[right] = nums[left];  
 }  
 //出循环时：left==right  
 nums[left] = pivot;  
 //递归  
 //此处由于是闭区间，分界线出要进行加减1操作  
 quickSort(nums, begin, left-1);  
 quickSort(nums, left + 1, end);  
 }  
};  
  
int main() {  
   
 vector<int>nums = { 10,9,2,5,3,7,101,18 };  
 Solution A;  
 A.quickSort(nums, 0, nums.size() - 1);  
 for (auto tp : nums) {  
 cout << tp << " ";  
 }  
 return 0;  
}

## 桶排序：



### 桶排序思想：



### 代码实现：

