**插入排序**

**void insertionSort(vector<int>&a,int n) {//vector不加&不会带出操作？//int不加&也可以操作**

**//n可以外部导入也可以内部写，但是在不加&的时候函数内算n算的是指针，n=1**

**//数组中数字索引为1~n**

**for (int i = 2; i <= n; i++) {**

**a[0] = a[i];//监视哨，可以省去内循环中每次对j>=0的判断，相当于是临时变量**

**int j = i - 1;**

**while (a[j] > a[0]) {**

**a[j + 1] = a[j]; j--;**

**}**

**a[j + 1] = a[0];**

**}**

**}**

**归并排序**

**void merge( int low, int mid,int high) {**

**int i=low, j=mid+1, k=low;//对闭区间[low<=mid]和[mid+1<=high]进行合并**

**while (i <= mid && j <= high) {**

**if (a[i] <= a[j]) b[k++] = a[i++];**

**else b[k++] = a[j++];**

**}**

**if (i > mid) while (j <= high) b[k++] = a[j++];**

**else while (i <= mid) b[k++] = a[i++];**

**for (int i = low; i <= high; i++)a[i] = b[i];//将辅助数组b的值copy到a的相应位置**

**}**

**void mergeSort(int low,int high) {//排序的下标为开区间[low,high]**

**if (low < high) {**

**//递归是：若划分区间超过1个元素则继续划分，若只有一个元素则返回向上进行两个区间的合并**

**int mid = (low + high) / 2;**

**mergeSort(low, mid);**

**mergeSort(mid + 1, high);**

**merge(low, mid, high);**

**}//之前用while break了，不就是if么，递归都用if**

**}**

**快排**

**int partition(int m,int n) {//划分闭区间[m,n]**

**//划分元素先视为a[m]**

**int t = a[m], i = m, j = n+1;//下面的j--决定初始值为n+1，如果划分元素不是a[m]的话m要变成m-1?**

**do {**

**do { i++; } while (a[i] < t);//都无等号**

**do { j--; } while (a[j] > t);**

**if (i < j) { int k; k=a[i]; a[i] = a[j]; a[j] = k; }//换位，这一步没写对过**

**} while (i < j);//无等号**

**a[m] = a[j]; a[j] = t;**

**return j;//返回划分元素最终的下标位置**

**}**

**void quickSortRec(int m,int n) {**

**if (m < n) {//无等号**

**int j = partition(m, n);**

**quickSortRec(m, j-1);**

**quickSortRec(j + 1, n);**

**}**

**}**

**二分搜索**

**//x is the num to be searched, and n is the lenth of a[]**

**//return index of x, or -1 if not exist**

**int binarySearch(int \*a, int x, int n) {**

**int low = 0, high = n, mid = n / 2;**

**while (low <= high) {**

**if (a[mid] == x) return mid;**

**else if (x < a[mid]) { high = mid-1; mid = (low + high) / 2; }//low+high may overflow→mid = low+(high-low)/2**

**else if (x > a[mid]) { low = mid+1; mid = (low + high) / 2; }**

**}**

**return -1;**

**}**