

SZU-ACM周训6. 0

排序: 冒泡, 选择, 插入, 快排, 归并

Kaslen_

sort函数

- 传入参数为数组需要排序的区间首地址和尾地址
- 传入的排序区间为左闭右开，即 $[l, r)$
- 时间复杂度为 $O(n \log n)$ ，其中 n 为区间长度
- 使用格式为



```
1  sort(/*首地址*/, /*尾地址*/, /*排序函数*/)
```

sort函数

- 排序默认为从小到大，可通过使用`greater< (数据类型) >()`函数从大到小排序

```
1 void solve(){
2     int a[] = {1,2,4,2,5};
3     sort(a, a + 5);
4     for(auto x:a){
5         cout << x << " ";
6     }
7     // 输出为1 2 2 4 5
8 }
```

```
1 void solve(){
2     int a[] = {1,2,4,2,5};
3     sort(a, a + 5, greater<int>());
4     for(auto x:a){
5         cout << x << " ";
6     }
7     // 输出为5 4 2 2 1
8 }
```

sort函数

- 排序默认为从小到大，可通过使用`greater< (数据类型) >()`函数从大到小排序
- 若遇到未定义比较函数的数据结构（例如结构体），可自行传入比较函数

sort函数

```
1 struct node{
2     int x, y;
3 };
4
5 bool cmp(node a, node b){
6     if(a.x != a.y) return a.x < b.x;
7     else return a.y < b.y;
8 }
9
10 void solve(){
11     vector<node> a(10);
12     sort(a.begin(), a.end(), cmp);
13 }
14
```

```
1 //简便写法
2 struct node{
3     int x, y;
4 };
5
6 void solve(){
7     vector<node> a(10);
8     sort(a.begin(), a.end(), [&](auto x, auto y){
9         if(x.x != y.x) return x.x < y.x;
10        else return x.y < y.y;
11    });
12 }
13
```

目录

➤ 冒泡排序

➤ 选择排序

➤ 插入排序

➤ 快速排序

➤ 归并排序

冒泡排序

冒泡排序

- 每次检查相邻两个元素，如果前面的元素大于后面的元素，就交换相邻两个元素。

初始数组



第一趟排序(升序)



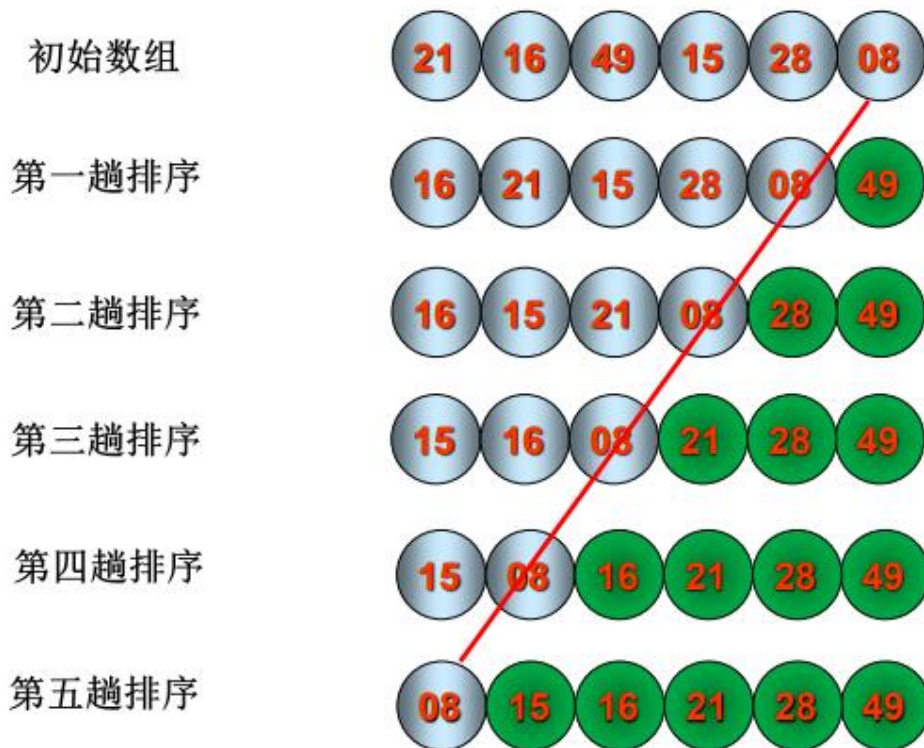
第一趟排序完成



可以看到最大的元素
来到了最后

冒牌排序

- 每次检查相邻两个元素，如果前面的元素大于后面的元素，就交换相邻两个元素。



冒泡排序的时间复杂度为 $O(n^2)$;

冒泡排序



```
1 void bubbleSort(){
2     for(int i = 1; i <= n - 1; i++){
3         for(int j = 1; j <= n - i; j++){
4             if(a[j] > a[j + 1]) swap(a[j], a[j + 1]);
5         }
6     }
7 }
8
```

冒泡排序

- 例题：<https://www.luogu.com.cn/problem/P1116>

题目描述

 复制Markdown  展开

在一个旧式的火车站旁边有一座桥，其桥面可以绕河中心的桥墩水平旋转。一个车站的职工发现桥的长度最多能容纳两节车厢，如果将桥旋转 180 度，则可以把相邻两节车厢的位置交换，用这种方法可以重新排列车厢的顺序。于是他就负责用这座桥将进站的车厢按车厢号从小到大排列。他退休后，火车站决定将这一工作自动化，其中一项重要的工作是编一个程序，输入初始的车厢顺序，计算最少用多少步就能将车厢排序。

冒泡排序

- 例题： <https://www.luogu.com.cn/problem/P1116>

```
1 void solve(){
2     int n; cin >> n;
3     vector<int> a(n + 1);
4     for(int i = 1; i <= n; i++) cin >> a[i];
5     int cnt = 0;
6     for(int i = 1; i <= n - 1; i++){
7         for(int j = 1; j <= n - i; j++){
8             if(a[j] > a[j + 1]){
9                 swap(a[j], a[j + 1]);
10                cnt += 1;
11            }
12        }
13    }
14    cout << cnt << endl;
15 }
```

本质上是求解逆序对
存在 $O(n)$ 求法

选择排序

选择排序

- 找出第 i 小的元素，然后将这个元素与数组第 i 个位置上的元素交换

	1	2	3	4	5	6	
$i = 1$	21	16	49	15	28	08	最小者 08 交换21, 08
$i = 2$	08	16	49	15	28	21	最小者 15 交换15, 16
$i = 3$	08	15	49	16	25	21	最小者 16 交换49, 16
$i = 4$	08	15	16	49	25	21	最小者 21 交换49, 21
$i = 5$	08	15	16	21	25	49	最小者 25 交换25, 25
	08	15	16	21	25	49	

选择排序

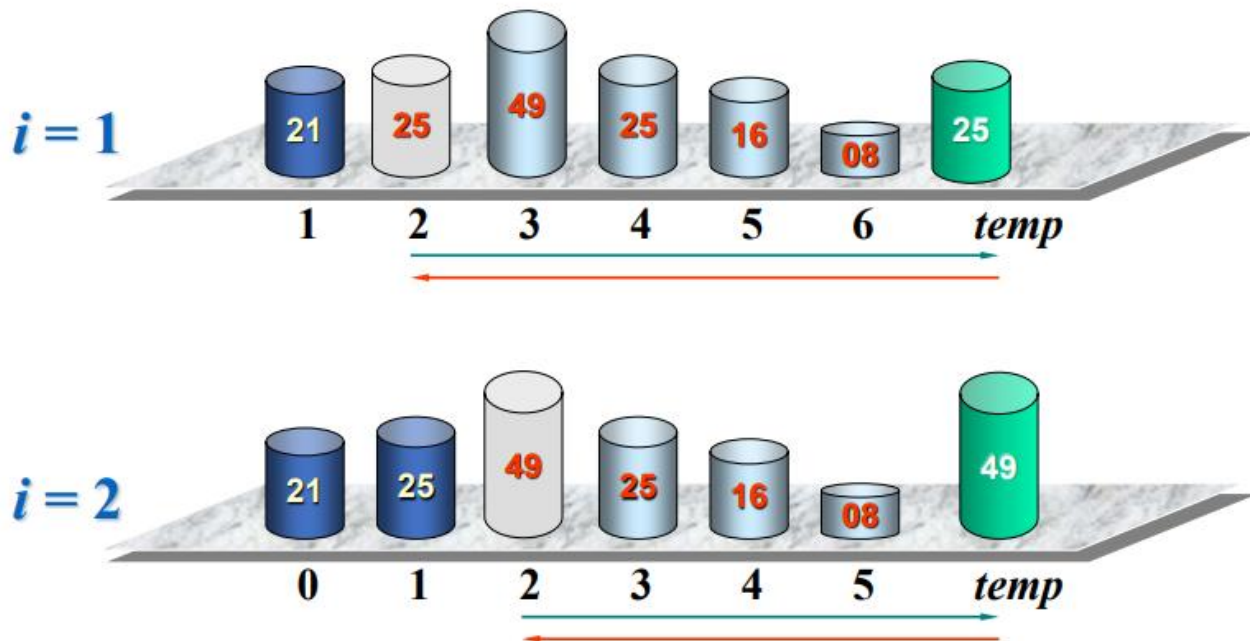


```
1 void selectSort(){
2     for(int i = 1; i <= n - 1; i ++){
3         int minn = i;
4         for(int j = i + 1; j <= n; j++){
5             if(a[j] < a[minn]) minn = j;
6         }
7         swap(a[minn], a[i]);
8     }
9 }
10
```

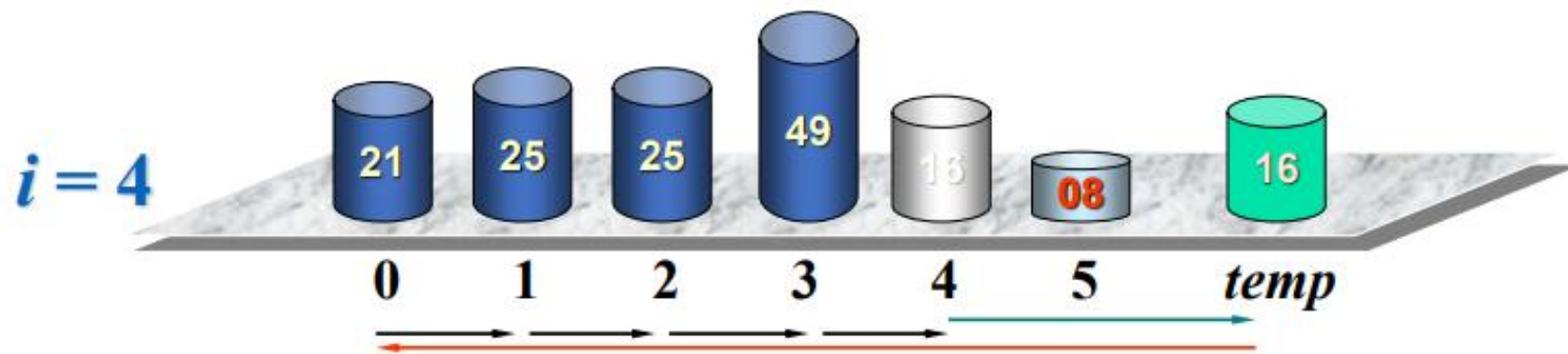
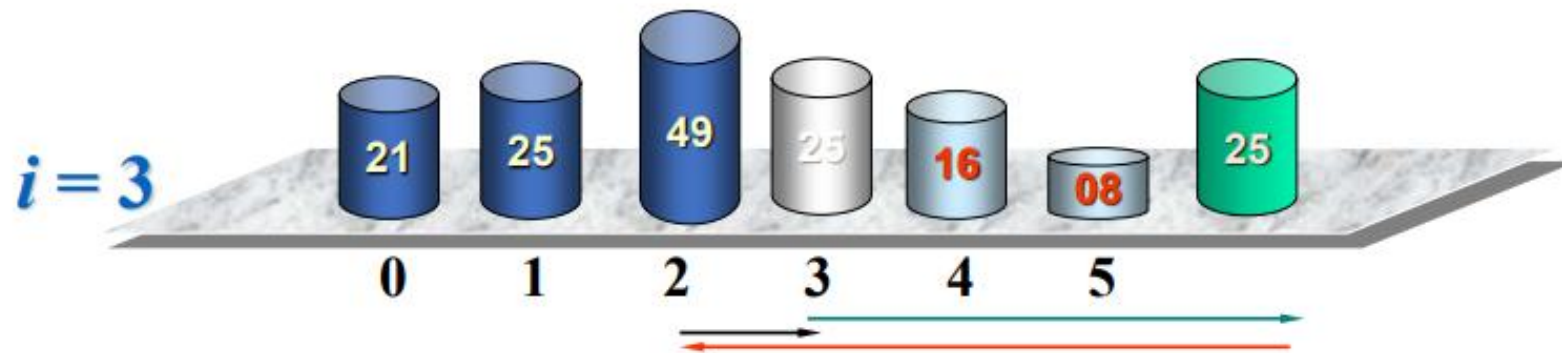
插入排序

插入排序

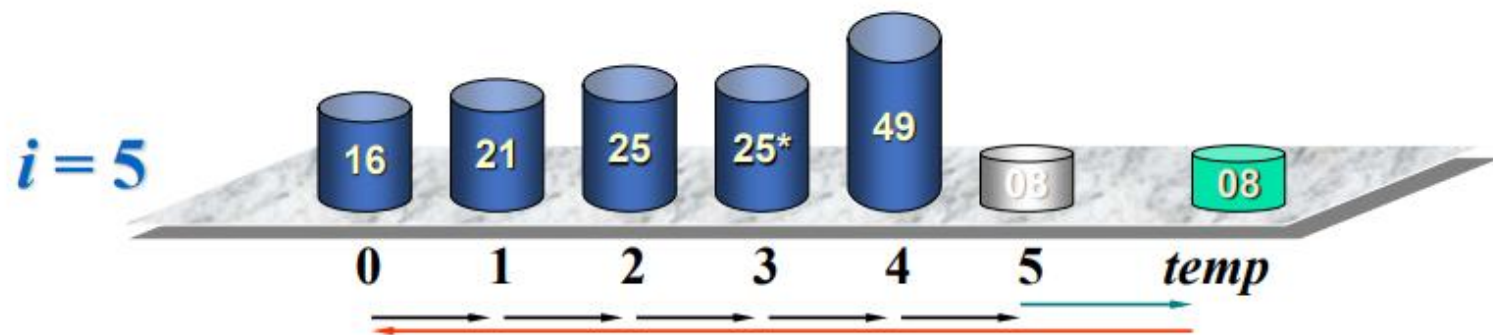
- 从头到尾依次扫描未排序的序列，每次把未排序的元素插入到已排序的元素中的正确位置。



插入排序



插入排序



完成

插入排序

```
1 void InsertSort(int a[],int l)
2 {
3     int temp, j;
4     for(int i = 1; i < l; i++){
5         if(a[i] < a[i-1]){
6             temp = a[i];
7             for(j = i - 1; j >= 0 && temp < a[j]; j--) {
8                 a[j+1] = a[j];
9             }
10            a[j+1] = temp;
11        }
12    }
13 }
```

复杂度为 $O(n^2)$

快速排序

快速排序

取序列第一个记录为枢轴记录，其关键字为Pivotkey

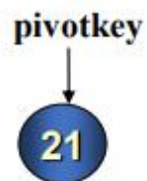
指针low指向序列第一个记录位置

指针high指向序列最后一个记录位置

一趟排序(某个子序列)过程

1. 从high指向的记录开始, 向前找到第一个关键字的值小于Pivotkey的记录, 将其放到low指向的位置, 然后low++
2. 从low指向的记录开始, 向后找到第一个关键字的值大于Pivotkey的记录, 将其放到high指向的位置, 然后high--
3. 重复1, 2, 直到low=high, 将枢轴记录放在low(high)指向的位置

快速排序



初始关键字



一次交换



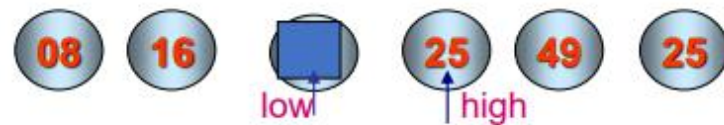
二次交换



三次交换



high-1



完成一趟排序



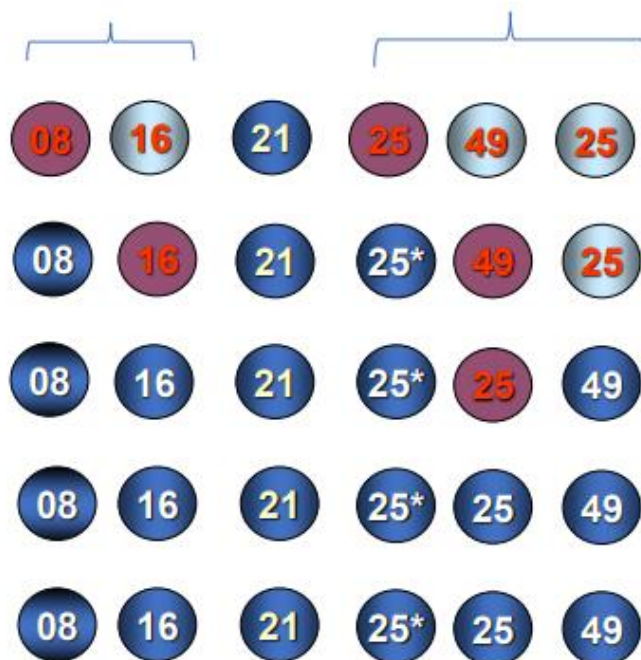
快速排序

完成一趟排序



21在正确的地方上，并且21前面的数字比21小，21后面的数字比21大。

分别进行快速排序



有序序列

快速排序

```
int quicks(int low,int high) {
    int key=a[low];
    while(low<high) {
        while(low<high&& a[high]>=key)
            high--;
        a[low]=a[high];
        while(low<high&& a[low]<=key)
            low++;
        a[high]=a[low];
    }
    a[low]=key;
    return low; //return shuzhou pos
}

void Qsort(int low,int high) {
    if(low<high) {
        int mid=quicks(low,high);
        Qsort(low,mid-1),Qsort(mid+1,high);
    }
}
```

可以证明，快速排序的时间复杂度是 $O(n\log_2 n)$ 。

实验结果表明：就平均计算时间而言，快速排序是所有内排序方法中最好的一个。

归并排序

归并排序

- 问题：如何合并两个有序数组？

归并排序

- 问题：如何合并两个有序数组？
- 新建一个数组，比较两个数组的第一个元素，把小的那个元素放到新建数组的后面

```
1 void solve(){
2     int n , m;
3     vector<int> a(n + 1),b(m + 1);
4     int p = 1, q = 1, cnt = 1;
5     vector<int> c(n + m + 1);
6     while (p <= n && q <= m){
7         if(a[p] < b[q]) c[cnt++] = a[p++];
8         else c[cnt++] = b[q++];
9     }
10    while(p <= n) c[cnt++] = a[p++];
11    while(q <= m) c[cnt++] = b[q++];
12 }
```

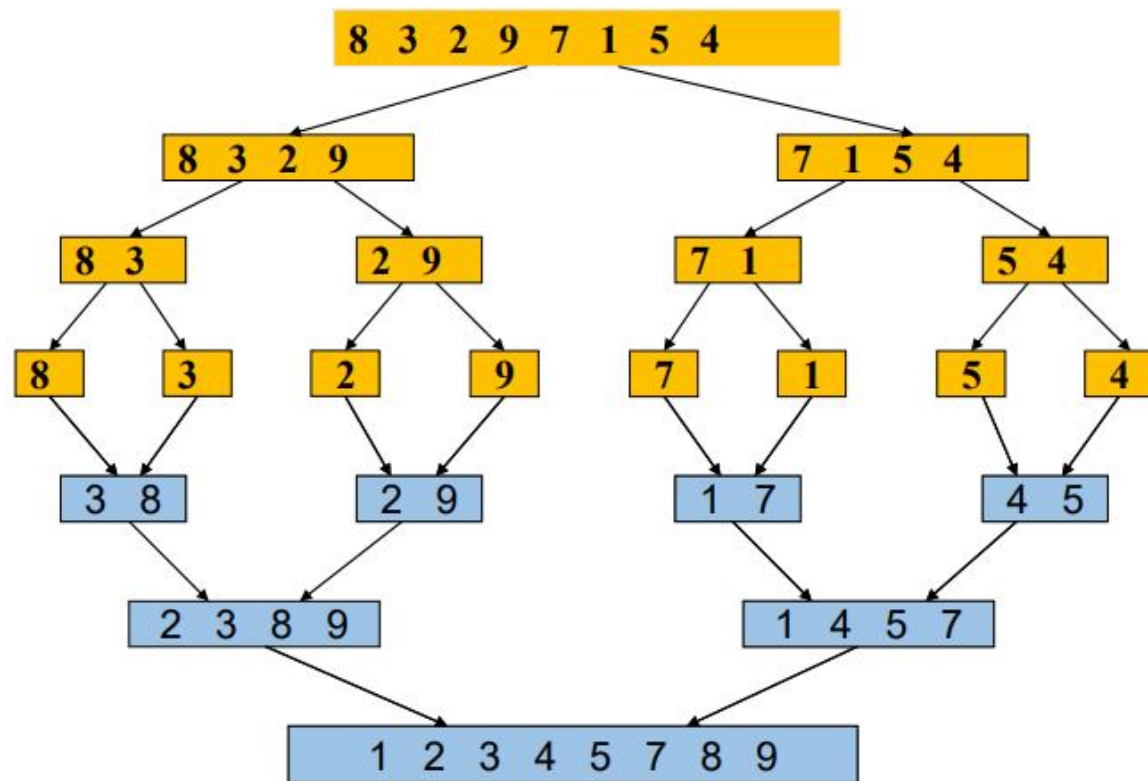
归并排序

- 那如果我们只想对一个数组进行排序，我们要怎么把合并两个数组的方法推广过来？

归并排序

- 那如果我们只想对一个数组进行排序，我们要怎么把合并两个数组的方法推广过来？
- 提示：如果数组中都只有一个数字，那这个数组一定是有序的

归并排序



归并排序

```
1 void merge_sort(int q[], int l, int r){
2     if(l >= r) return;
3     int mid = (l + r) / 2;
4     merge_sort(q, l, mid), merge_sort(q, mid + 1, r);
5     int k = 0, i = l, j = mid + 1;
6     while(i <= mid && j <= r){
7         if(q[i] <= q[j]) temp[k++] = q[i++];
8         else temp[k++] = q[j++];
9     }
10
11     while(i <= mid) temp[k++] = q[i++];
12     while(j <= r) temp[k++] = q[j++];
13
14     for(int i = l, j = 0; i <= r; i++, j++)
15         q[i] = temp[j];
16 }
17
```

复杂度是 $O(n\log n)$

例题

- 链接: <https://codeforces.com/contest/1833/problem/B>

题意翻译

本题有 t 组测试数据。

对于每组测试数据, 给定整数 n, k 与两个长度为 n 的数组 a 与 b 。

要求重新排列 b 使得对于每一个 $1 \leq i \leq n$, 满足 $|a_i - b_i| \leq k$

保证一定有解。

Example

input

```
3
5 2
1 3 5 3 9
2 5 11 2 4
6 1
-1 3 -2 0 -5 -1
-4 0 -1 4 0 0
3 3
7 7 7
9 4 8
```

output

```
2 2 5 4 11
0 4 -1 0 -4 0
8 4 9
```

例题

- 显然当a数组中的第i大的元素，对应的是b数组中的第i大元素时，总体的 $|a_i - b_i|$ 一定会是最小值
- 所以只需要先把两个数组排序，然后把a的编号给b，在按顺序输出b数组就行

例题

```
struct NODE{
    int data, id;
}a[N];
NODE b[N];
bool cmp(NODE x, NODE y){
    return x.data<y.data;
}
bool cmpid(NODE x, NODE y){
    return x.id<y.id;
}
int main() {
    int T; cin>>T;
    while(T--){
        int n, m; cin>>n>>m;
        for(int i=1; i<=n; i++) cin>>a[i].data, a[i].id = i;
        for(int i=1; i<=n; i++) cin>>b[i].data;
        sort(b+1, b+n+1, cmp);
        sort(a+1, a+n+1, cmp);
        for(int i=1; i<=n; i++){
            b[i].id = a[i].id;
        }
        sort(b+1, b+n+1, cmpid);
        for(int i=1; i<=n; i++) cout<<b[i].data<<' ';
        puts("");
    }
}
```

先都升序排序，再把A的编号赋给B，最后按照编号还原B数组即可。