2025-2026 秋《数据结构与算法》第二次实验题解

助教-陈孙一硕

Difficulties:

- Easy: A
- Medium-Easy: B,C,D
- Medium-Hard: E,F
- Hard: G

A h0279. 逆序对

归并排序求逆序对板子题。习题反馈没认真看的同学有难了，之前的习题反馈里面明确说了万一把数据量加到 10^5 卡掉了暴力做法呢。

时间复杂度 $O(n \log n)$.

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
int q[100005], temp[100005], cnt = 0;
void merge_sort(int l, int r) {
    if (l >= r) return ;
    int mid = (l + r) / 2, i = l, j = mid + 1, k = 0;
    merge_sort(l, mid);
    merge_sort(mid + 1, r);
    while (i <= mid && j <= r) {
        if (q[i] <= q[j]) temp[k++] = q[i++];
        else temp[k++] = q[j++], cnt += mid - i + 1;
    }
    while (i <= mid) temp[k++] = q[i++];
```

```
    while (j <= r) temp[k++] = q[j++];
    for (i = l, j = 0; i <= r; i++, j++) q[i] = temp[j];
}
signed main() {
    int n;cin >> n;
    for (int i = 0; i < n; i++) cin >> q[i];
    merge_sort(0, n - 1);
    cout << cnt << endl;
    return 0;
}
```

B 社团纳新（easy）

首先考虑一个简单版，如果仍然是每个班级只能有一个人去，但是每个人去的班级数量没有限制，那应该怎么做？很显然应该是对于某一个班级，谁去的收益高就派谁去。

但是现在有了限制，那应该怎么办？我们做一个简单的假设，假设所有班级最开始都是 L 同学去的，那么陈同学应当要从 L 同学手中抢走一半的班级。怎么抢呢？当然是优先拿走那些陈同学去了之后，增加的收益更多的班级。这就是按照二者差值进行排序的原理了。

时间复杂度 $O(n \log n)$。

```
#include<iostream>
#include<algorithm>
using namespace std;
#define int long long
int n,ans=0;
struct node{
    int x,y;
};
bool cmp(node n1,node n2){
    return n1.x-n1.y>n2.x-n2.y;
}
node lst[100005];
void solve(){
    cin>>n;
    for (int i=1;i<=n;i++) cin>>lst[i].x>>lst[i].y;
    sort(lst+1,lst+n+1,cmp);
```

```
    for (int i=1;i<=n/2;i++) ans+=lst[i].x;
    for (int i=n/2+1;i<=n;i++) ans+=lst[i].y;
    cout<<ans<<endl;
}
signed main(){
    solve();
    return 0;
}
```

C 德才论

这道题是上学期弘深班期末考试的第二题，考试时大部分同学都通过了本题，难度不大。先将考生封装为结构体，将三类考生分别放到三个 vector 里，并对三个 vector 都按照题目给定的排序规则，重载小于运算符后排序即可。

时间复杂度 $O(n \log n)$。

```
#include<bits/stdc++.h>
using namespace std;
struct node{
    string id;
    int d,c;
    bool operator < (const node n1) const{
        if (d+c!=n1.d+n1.c) return d+c>n1.d+n1.c;
        else if (d!=n1.d) return d>n1.d;
        return id<n1.id;
    }
};
const int N=2e5+101;
node a[N];
int main(){
    int n,l,h; cin>>n>>l>>h;
    for (int i=1;i<=n;i++) cin>>a[i].id>>a[i].d>>a[i].c;
    vector<node> v1,v2,v3,v4;
    for (int i=1;i<=n;i++){
        int dd=a[i].d,cc=a[i].c;
        if (dd<l||cc<l) continue;
```

```
        else if (dd>=h&&cc>=h) v1.push_back(a[i]);
        else if (dd>=h&&cc<h) v2.push_back(a[i]);
        else if (dd<h&&cc<h&&dd>=cc) v3.push_back(a[i]);
        else v4.push_back(a[i]);
    }
    sort(v1.begin(),v1.end());
    sort(v2.begin(),v2.end());
    sort(v3.begin(),v3.end());
    sort(v4.begin(),v4.end());
    cout<<v1.size()+v2.size()+v3.size()+v4.size()<<endl;
    for (auto u:v1) cout<<u.id<<" "<<u.d<<" "<<u.c<<endl;
    for (auto u:v2) cout<<u.id<<" "<<u.d<<" "<<u.c<<endl;
    for (auto u:v3) cout<<u.id<<" "<<u.d<<" "<<u.c<<endl;
    for (auto u:v4) cout<<u.id<<" "<<u.d<<" "<<u.c<<endl;
    return 0;
}
```

D 最远的距离

不妨设 $i < j$。那么只有 $a_i < a_j$ 和 $a_i \geq a_j$ 两种情况。

复制两份数组。对第一份数组按照 $i^2 + a_i^2$ 进行排序，第二份数组按照 $i^2 - a_i^2$ 进行排序。假设第一份数组中，两端的数分别为 $x^2 + a_x^2, y^2 + a_y^2$，且 $x < y$。

- 如果 $a_x < a_y$，那么说明这两个位置之间的 CQU 距离就是二者之差，按照 $i^2 + a_i^2$ 来排序是正确的；
- 否则，我们得出的 CQU 距离为 $|x^2 + a_x^2 - (y^2 + a_y^2)|$，而这两者之间的 CQU 距离实际上应该是 $|x^2 - a_x^2 - (y^2 - a_y^2)|$，显然后者是大于前者的，所以取 max 时，前者不会影响答案。

对于按照 $i^2 - a_i^2$ 排序的情况，分析同理。

因此，只要按照两种情况分别排序一遍，分别计算头尾差的绝对值，取最大即可。

时间复杂度 $O(n \log n)$，主要来源于排序的复杂度。

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
```

```
#define endl '\n'
const int N=2e6+101;
int a[N];
void solve(){
    int n; cin>>n;
    vector<int> f,b;
    for(int i=1;i<=n;i++) cin>>a[i];
    for(int i=1;i<=n;i++){
        f.push_back(a[i]*a[i]+i*i);
        b.push_back(a[i]*a[i]-i*i);
    }
    sort(f.begin(),f.end());
    sort(b.begin(),b.end());
    cout<<max(abs(f[0]-f[n-1]),abs(b[0]-b[n-1]))<<endl;
}
signed main(){
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int T=1; //cin>>T;
    while (T--) solve();
    return 0;
}
```

E 杂货店店主的烦恼

出这道题是为了向大家介绍一个方法：微扰法。

假设 $a_1, a_2, \ldots, a_n$ 从下到上放置。类似冒泡排序的思想，我们分析何时 a_i 排在 a_{i+1} 的下面，是比将它们交换位置之后更优的。

由于每个物品只会影响放在它下面的物品的实际体积，因此，考虑 a_i 和 a_{i+1} 对体积压缩的贡献。

在不交换位置时，a_i 对体积压缩的贡献为：

$$\sum_{j=1}^{i-1} c_j w_i$$

a_{i+1} 对体积压缩的贡献为：

$$\sum_{j=1}^{i-1} c_j w_{i+1} + c_i w_{i+1}$$

如果将二者交换位置，同理可得到二者对压缩的总贡献为

$$\sum_{j=1}^{i-1} c_j w_{i+1} + \sum_{j=1}^{i-1} c_j w_i + c_{i+1} w_i$$

考虑交换前后的贡献差大于 0，得到

$$c_i w_{i+1} > c_{i+1} w_i$$

依照此规则重载小于运算符，对所有物品进行排序，随后用一个后缀记录，线性时间模拟计算答案即可。

时间复杂度 $O(n \log n)$，主要来源于排序。

F 跳排序

为了防止大家爆零，给了一半的暴力分。对于每次询问，都花费 $O(n)$ 的时间去判断是可行的，但是会超时。我们需要尝试 $O(1)$ 回答每个询问。

从宏观的角度上看，如果 $a[i] = j$，那么这个数应当回到下标为 j 的位置，需要移动的距离就是 $|i - j| = |a[i] - i|$。但是对于某个 x，每个数一次只能移动 x 个位置。那么如果这个数字要归位，x 必须是 $|a[i] - i|$ 的因数。这意味着，想要让所有数全部归位，给定的 x 必须是所有 $a[i] - i, i = 1, 2, \cdots n$ 的因数，即它们最大公约数的因数。

我们需要注意一种特殊情况，就是如果整个序列已经有序了，那对于每个 x 都应当输出 YES（因为题目说了可以不操作）。一个办法是把 gcd 的初始值设置成 0，这样对于每次询问，$g\%tmp$ 的值都是 0，都会输出 YES. 没想到这个办法的同学，特判一下也不困难。

时间复杂度 $O(n \log n + k)$。

```
#include<iostream>
using namespace std;
#define int long long
int n,k,lst[100005],g=0,tmp;
int gcd(int x,int y){
    if (x<y) swap(x,y);
    if (x%y==0) return y;
    return gcd(y,x%y);
}
```

```
void solve(){
    cin>>tmp;
    if (g%tmp==0) cout<<"YES"<<endl;
    else cout<<"NO"<<endl;
}
signed main(){
    cin>>n>>k;
    bool flag=0;
    for (int i=1;i<=n;i++){
        cin>>lst[i];
        if (lst[i]-i==0) continue;
        if (!flag) g=abs(lst[i]-i),flag=1;
        else g=gcd(g,abs(lst[i]-i));
    }
    while (k--) solve();
    return 0;
}
```

G 四数交换排序

下面提出“交换环”：离散数学应该已经学过划分的概念。所有交换环恰好构成的是整个序列的一个划分。

先找交换环。如果是大小为 2 的环，一次操作可以解决 2 个；大小为 3 或 4 的环，一次操作解决一个；大小在 5 及以上的环，每次操作可以让环的大小减少 3。按照这个思路，先处理大小大于 5 的环，直到其大小变为 2 或 3 或 4 为止。然后再处理大小为 2，3，4 的环即可. 时间复杂度 $O(n)$。

```
#include<iostream>
using namespace std;
#define int long long
int lst[1000005],cir[1000005],ans=0;
bool vis[1000005];
signed main(){
    int n;cin>>n;
    for (int i=1;i<=n;i++) cin>>lst[i];
    for (int i=1;i<=n;i++){
```

```
        if (lst[i]==i||vis[i]) continue;
        int res=0,tmp=i;
        while (!vis[tmp]) res++,vis[tmp]=1,tmp=lst[tmp];
        cir[res]++;
    }
    for (int i=5;i<=n;i++){
        ans+=i*cir[i]/3;
        if (i%3==2) cir[2]+=cir[i];
    }
    ans+=(cir[2]+1)/2+cir[3]+cir[4];
    cout<<ans<<endl;
    return 0;
}
```