第五届GXCPC广西大学生程序设计竞赛 部分题解（无CDK）

A、Arrangement

自定义排序。

最终难度低的放前面，难度一样则思维难度低放前面，都一样就题目序号小的放前面

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;

bool cmp(vector<int>a,vector<int>b)
{
    if(a[0]!=b[0])return a[0]<b[0];
    if(a[1]!=b[1])return a[1]<b[1];
    return a[2]<b[2];
}

signed main()
{

    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n;
    cin>>n;
    vector<int>a(n),b(n);
    vector<vector<int>>c(n);
    for(int i=0;i<n;i++)
        cin>>a[i];
    for(int i=0;i<n;i++)
    {
        cin>>b[i];
        c[i]={a[i]*b[i],a[i],i};
    }
    sort(c.begin(),c.end(),cmp);
    for(int i=0;i<n;i++)
    {
```

```
        cout<<(char)(c[i][2]+'A')<<" ";
    }
    return 0;
}
```

B、Bus

简单的除法，注意只用一个bus的时候输出的语句不同。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;

signed main()
{

    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n,m;
    cin>>n>>m;
    if(n<=m)
    {
        cout<<"We need one bus."<<endl;
    }
    else
    {
        cout<<"We need "<<(n%m==0?n/m:n/m+1)<<" buses."<<endl;
    }
    return 0;
}
```

E、Envy-freeness

这题题目很长，但看完题目就能发现这是个签到。

记录一下两边自己物品在自己眼里的最大值和在对面眼里的最大值，以及对面认为我的价值最大和

最小的两个物品。

• 只要两边都认为自己的东西价值最大，就输出EF。

• 如果对面物品减去最小的货物后价值比我的小，输出EFX。

• 如果对面物品减去最大的货物后价值比我的小，输出EF！。

• 其它情况输出E。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;

signed main()
{

    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    //max表示对面认为我的价值最高的物品，min表示对面认为我的价值最低的物品
    int n, a, b, c, maxx = 0, maxy = 0, minx = 1e9, miny = 1e9;
    //first表示自己拥有的自己眼里物品总价值，second表示自己拥有的对面眼里物品总价值
    PII x, y;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        cin >> a >> b >> c;
        if (c == 0)
        {
            x.first += a;
            x.second += b;
            maxx = max(maxx, b);
            minx = min(minx, b);
        }
        else
        {
            y.first += b;
            y.second += a;
            maxy = max(maxy, a);
```

```
            miny = min(miny, a);
        }

        if (x.first >= y.second && y.first >= x.second)
        {
            cout << "EF" << endl;
        }
        else if (x.first >= y.second - miny && y.first >= x.second - minx)
        {
            cout << "EFX" << endl;
        }
        else if (x.first >= y.second - maxy && y.first >= x.second - maxx)
        {
            cout << "EF1" << endl;
        }
        else cout << "E" << endl;
    }
    return 0;
}
```

F、Finding Stars

我们可以用线段树的单点修改+区间查询模板来写。

每个线段树都有一个bitset<1000>，如果一个数x的出现次数是奇数，那么bitset上第x个位置我们设为1，偶数设为0。

那么查询一个区间有多少出现次数为偶数，只要看bitset<1000>有多少个1就行。至于合并两个bitset，我们可以用异或运算（全为1或全为0时为0，一边为0一边为1时为1，正好符合我们的要求）。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>
#include<bitset>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;
bitset<1001>f[4 * N];
int a[N], n, m;
```

```
void build_tree(int k, int l, int r)
{
    if (l == r)
    {
        f[k].set(a[l], 1);
        return;
    }
    int mid = (l + r) / 2;
    build_tree(k + k, l, mid);
    build_tree(k + k + 1, mid + 1, r);
    f[k] = f[k + k] ^ f[k + k + 1];
}

void revise(int k, int l, int r, int x, int y)
{
    if (l == r)
    {
        f[k].reset();
        f[k].set(y, 1);
        return;
    }
    int mid = (l + r) / 2;
    if (x <= mid)revise(k + k, l, mid, x, y);
    else revise(k + k + 1, mid + 1, r, x, y);
    f[k] = f[k + k] ^ f[k + k + 1];
}

bitset<1001> query(int k, int l, int r, int x, int y)
{
    if (l == x && r == y)return f[k];
    int mid = (l + r) / 2;
    if (y <= mid)return query(k + k, l, mid, x, y);
    else
        if (x > mid)return query(k + k + 1, mid + 1, r, x, y);
        else return query(k + k, l, mid, x, mid) ^ query(k + k + 1, mid + 1, r, mid +
1, y);
}

signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    cin >> n >> m;
    for (int i = 1; i <= n; i++)cin >> a[i];
    build_tree(1, 1, n);
    int st, x, y;
    while (m--)
    {
        cin >> st >> x >> y;
        if (st == 0)
        {
            revise(1, 1, n, x, y);
            a[x] = y;
        }
        else if (st == 1)
        {
            auto c=query(1, 1, n, x, y);
            if (c.none() || c.flip().none())cout << 1 << endl;
            else cout << 0 << endl;
        }
```

```
        else
        {
            auto c = query(1, 1, n, x, y);
            cout << c.count() << endl;
        }
    }
    return 0;
}
```

G、Gambling

如果我们不换题，那么就有：

• 抽到会的题，概率1/n，期望分数就是：1/n*100.

• 没抽到会的题，概率（n-1）/n，期望分数就是：（n-1）/n*50.

总期望是：(50+50*n)/n

如果我们换题，那么有：

• 一次抽到会的题，概率1/n，期望分数：1/n*100.

• 没抽到会的题，换题后抽到，期望分数：（n-1)/n*(1/n) *100 *0.8

• 没抽到会的题，换题后也没抽到，期望分数：(n-1)/n*(n-2)/n *50 *0.8.

总期望是（100+40*n)/n

我们只要根据n，看是50+50*n大还是100+40 *n大。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;

signed main()
{
```

```
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n;
    cin>>n;
    cout<<max(50*n+50,40*n+100);
    return 0;
}
```

H、Homework

首先我们肯定至少要写最多的那一份的数量的字，其余的我们都可以用复制粘贴来解决。

但不是说我们先把最多的那一份作业写完，比如三个作业子数是100，200，300对应的复制时间是99，199，1。那么显然我们先写完200的字，再复制给作业1和3，最后补足作业三缺少的字更省时间。

既然我们要写的字数已经固定，那初步时间就可以知道了，然后只有第一份是不能粘贴的，所以我们第一份作业选的就是复制时间用时最多的那一个。

即总时间是：最大字数+除去一份最大复制时间后其它复制时间的总和。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;

signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n;
    cin>>n;
    vector<int>a(n),b(n);
    int cnt=0,ans=0;
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
```

```
        cnt=max(cnt,a[i]);
    }
    for(int i=0;i<n;i++)
    {
        cin>>b[i];
        cnt+=b[i];
        ans=max(ans,b[i]);
    }

    cout<<cnt-ans;
    return 0;
}
```

I、Infinity

因为如果x可以被表示，那么x+b也能被表示。

如果a^k%b可以得到{2,b-1}的所有数（不用0和1因为初始就是0和1），那么就可以通过a^k+b*n得到所有的数。

由于题目范围只有 10e6，所以直接枚举 $k \in [1, b)$，判断 $a^k \bmod b$ 是否遍历 $2 \cdots b-1$ 即可。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;

signed main()
{

    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n, m;
    cin >> n >> m;
    int cnt = m-2,num=1;
    unordered_map<int,int>mymap;
    for (int i = 1; i <= m; i++)
    {
        num *= n;
```

```
        num%=m;
        if (num != 0&&num!=1 && mymap[num] == 0)mymap[num] = 1, cnt--;
    }
    if (cnt <= 0)cout << "YES";
    else cout << "NO";
    return 0;
}
```

J、Jump game

初步想法，对所有数质因数分解，比如42分解成2、3、6，那只要在前面找所有含有质因数2或3或6的位置，把他们的方案数加起来就是我们42的方案数了。但是这样的话时间复杂度是n^2。

所以要用到容斥定理，在这里简单来说就是，加满足奇数条件的-满足偶数条件的。

用一个sum数组，sum[i]表示含有因数i的位置的方案数。

例如序列：2 3 5 9 10 18 30 120，sum[3]=f[2]+f[4]+f[6]+f[7]+f[8] (f表示到达第i个位置的方案数)

那么f[7]=sum[2]+sum[3]+sum[5]-sum[2*3]-sum[2 *5]-sum[3 *5]+sum[2 * 3 *5]。

可以先求出所有数的质因数，然后根据他们的质因数数量用01枚举来求出所有可能的组合数（最多2^5种），然后根据用了奇数个质因数或偶数个质因数来决定是加还是减。

每次算完f后要更新sum，只要是f能影响到的sum全部都要更新。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>
#include<bitset>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 1e5 + 50, MOD = 1e9 + 7;
vector<int>prime[N];
int sum[N], f[N];

void get_prime(int x)
{
    int num = x;
    for (int i = 2; i <= x / i; i++)
```

```
    {
        if (num % i == 0)
        {
            prime[x].push_back(i);
            while (num % i == 0)
                num /= i;
        }

    }
    if (num > 1)prime[x].push_back(num);
}

void get_f(int x, int pos)
{
    //x有n个质因数
    int n = prime[x].size(), num = 1 << n;
    for (int i = 1; i < num; i++)
    {
        int cnt = 0, res = 1;
        for (int j = 0; j < n; j++)
        {
            if ((i >> j) & 1)
            {
                cnt++;
                res *= prime[x][j];
            }
        }
        //用了奇数个质因数，按照容斥定理，要加
        if (cnt % 2 == 1)
            f[pos] = (f[pos] + sum[res]) % MOD;
        //偶数是减
        else
            f[pos] = (f[pos] - sum[res] + MOD) % MOD;
    }
}

void get_sum(int x, int pos)
{

    int n = prime[x].size(), num = 1 << n;
    for (int i = 1; i < num; i++)
    {
        int res = 1;
        for (int j = 0; j < n; j++)
        {
            if ((i >> j) & 1)
            {
                res *= prime[x][j];
            }
        }
        //看x能影响到的sum全部都要更新
        sum[res] = (sum[res] + f[pos]) % MOD;
    }
}

signed main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n;
```

```
    cin >> n;
    vector<int>v(n);
    for (int i = 0; i < n; i++)
    {
        cin >> v[i];
        if (prime[v[i]].empty())
            get_prime(v[i]);
    }
    f[0] = 1;
    for (int i = 0; i < n; i++)
    {
        if (i > 0)
            get_f(v[i], i);
        get_sum(v[i], i);
    }
    cout << f[n - 1] << endl;
    return 0;
}
```

L、Lowbit

一开始可能想的，如果只有单独一个1就用减，如果有连续一串1就先加再减。

但是如果是1110111这样的，其实先加两次后再减才是最优解。

我们除了判断单个1和一串1，还要看两串1直接隔一个0的情况。

```
#include<iostream>
using namespace std;
#include<vector>
#include<algorithm>
#include<math.h>
#include<set>
#include<numeric>
#include<string>
#include<string.h>
#include<iterator>
#include<map>
#include<unordered_map>
#include<stack>
#include<list>
#include<queue>
#include<iomanip>

#define endl '\n'
#define int ll
typedef long long ll;
typedef unsigned long long ull;
typedef pair<ll, ll>PII;
const int N = 2e5 + 50, MOD = 1e9 + 7;

signed main()
{

    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    string s;
    cin >> s;
```

```
    reverse(s.begin(), s.end());
    s += "00";
    int cnt = 0, n = s.size();
    for (int i = 0; i < n; i++)
    {
        if (s[i] == '1')
        {
            if (i == n - 1 || s[i + 1] == '0')cnt++;
            else
            {
                while (i < n - 1 && s[i] == '1')
                    s[i++] = '0';
                s[i] = '1';
                i--;
                cnt++;
            }
        }
    }
    cout << cnt << endl;
    return 0;
}
```