谢罪谢罪谢罪

两位小菜鸡第一次出题，题目质量有待提高，希望各位大佬多多担待

杨占宇：A、B、C、D、F

吕占捷：E、G、H

I题不知道是哪混进来的

A、点我进行签到

思路

这题是真签到题，判断奇偶输出就好啦！

代码

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll n;

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    cin >> n;
    if(n & 1) cout << "odd" << endl;
    else cout << "even" << endl;

    return 0;
}
```

B、吕老师爱刷题

思路

因为 $1 \leq a_i \leq 10^5$ ，所以可以利用桶排序思想，记录当前数字出现的次数，最后从小到大统计出现次数超过1次的数字即可

代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll maxn = 1e5 + 5;
ll n, a[maxn], b[maxn];
bool f;

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    cin >> n;
    for(ll i = 1; i <= n; i++) cin >> a[i], b[a[i]]++;
    for(ll i = 1; i <= 100000; i++) if(b[i] > 1) cout << i << " ", f = true;
    if(!f) cout << -1 << endl;

    return 0;
}
```

C、占宇哥的难题

思路

首先用并查集预处理 n 个限制条件，将答案一样的放进一个集合里，然后开始dfs枚举所有情况，统计答案的时候判断一下是否满足预处理的集合即可

代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 1e3 + 5;
int a[5], m, ans, cnt[5], f[13], ret[13], vis[13];

inline int read(){
    int s = 0, w = 1; char ch = getchar();
    while(ch < '0' || ch > '9'){ if(ch == '-') w = -1; ch = getchar();}
    while(ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch = getchar();
    return s * w;
}

inline void print(int x){
    if(x < 0) x = ~x + 1, putchar('-');
    if(x > 9) print(x / 10);
    putchar(x % 10 + '0');
}

int father(int x){
    return x == f[x] ? x : f[x] = father(f[x]);
}

void marge(int u, int v){
    f[father(u)] = father(v);
}

bool check(){
    for(int i = 1; i <= 12; i++)
        if(ret[i] != ret[f[i]]) return false;
    return true;
}

void dfs(int num){
    if(num == 13){
        if(check()) ans++;
        return;
    }
    for(int i = 1; i <= 4; i++){
        if(cnt[i] < a[i]){
            cnt[i]++;
            ret[num] = i;
            dfs(num + 1);
            cnt[i]--;
        }
    }
}
```

```
int main(){
    for(int i = 1; i <= 12; i++) f[i] = i;
    a[1] = read(), a[2] = read(), a[3] = read(), a[4] = read(), m = read();
    for(int i = 0; i < m; i++){
        int x = read(), y = read();
        marge(x, y);
    }
    for(int i = 1; i <= 12; i++) f[i] = father(f[i]);
    dfs(1);
    print(ans);

    return 0;
}
```

D、吕老师的01串 I

思路

令 a_i 表示以 i 为结尾的最长全0子串的长度，则当 s_i 为0时有公式 $a_i = a_{i-1} + 1$；当 i 不为0时 $a_i = 0$，最后遍历一下 a_i 取最大值即可

代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll maxn = 1e5 + 5;
ll a[maxn], ans;
string s;

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    cin >> s; if(s[0] == '0') a[0] = 1;
    for(ll i = 1; i < s.size(); i++)
        if(s[i] == '0') a[i] = a[i - 1] + 1;
    for(ll i = 0; i < s.size(); i++)
        ans = max(ans, a[i]);
    cout << ans << endl;

    return 0;
}
```

E、吕老师的01串II

思路

考虑前缀和，将 s_i 为0的位置当成-1，这样我们取前缀和数组中（下标从0开始）的最大值和最小值做差取绝对值就是答案

代码

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll maxn = 1e5 + 5;
ll a[maxn], len, mx = -1e18, mn = 1e18;
char s[maxn];

int main(){
    scanf("%s", s + 1); len = strlen(s + 1);
    for(ll i = 1; i <= len; i++){
        if(s[i] == '0') a[i] = a[i - 1] - 1;
        else a[i] = a[i - 1] + 1;
    }
    for(ll i = 0; i <= len; i++){
        mx = max(mx, a[i]);
        mn = min(mn, a[i]);
    }
    cout << abs(mx - mn) << endl;

    return 0;
}
```

F、占宇哥的01串 I

思路

二分答案，首先预处理前缀0和1的个数，然后二分答案，判断是否可以在小于 k 次操作时操作成功

代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 1e6 + 5;
int n, k;
char s[maxn];
int sum0[maxn],sum1[maxn];

bool check(int m){
    for(int i = m; i <= n; i++)
        if(sum0[i] - sum0[i - m] <= k || sum1[i] - sum1[i - m] <= k) return
true;
    return false;
}

int work(){
    int l = 1, r = n, ans;
    while(l <= r){
        int mid = (l + r) >> 1;
        if(check(mid)) l = mid + 1, ans = mid;
        else r = mid - 1;
    }
    return ans;
}

int main(){
    scanf("%d%s", &k, s + 1); n = strlen(s + 1);
    for(int i = 1; i <= n; i++){
        sum0[i] = sum0[i - 1] + (s[i] == '0');
        sum1[i] = sum1[i - 1] + (s[i] == '1');
    }
    printf("%d\n", work());

    return 0;
}
```

G、占宇哥的01串II

思路

dp+矩阵快速幂
首先，两个的时候有四种情况，分别是00，01，10，11。设dp[i][j]表示在第i位放的数为j是的方案数。当第i位放1时，第i-1位可以选0，第i-1位也可以选1，所以dp[i][1]=dp[i-1][0]+dp[i-1][1],当第i位放0时，因为不能存在010串，所以第i-1位可以放0，但是如果第i-1位放1的话，第i-2位就不能放0，第i-2位必须放1才行。所以dp[i][0]=dp[i-1][0]+dp[i-2][1],设f[n]为长度为n的非01串方案数，经整理得，f[n]=f[n-1]+f[n-2]+f[n-4],但是因为数据范围是1e15,所以使用矩阵快速幂求解。此题难度大概为省赛金牌难度

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int mod=1e9+7;
const int N=4;
ll n;
void mul(ll f[][N],ll a[][N],ll b[][N])
{
    ll t[N][N];
    memset(t,0,sizeof(t));
    for(int i=0;i<N;i++)
    {
        for(int j=0;j<N;j++)
        {
            for(int k=0;k<N;k++)
            {
                t[i][j]=(t[i][j]+a[i][k]*b[k][j])%mod;
            }
        }
    }
    memcpy(f,t,sizeof(t));
}
int main()
{
    ll f[4][4]={1,2,4,7};
    ll a[4][4]={
    {0,0,0,1},
    {1,0,0,0},
    {0,1,0,1},
    {0,0,1,1}
    };
    ll sum=0;
    cin>>n;
    if(n==1)
    {
        cout<<2;
        return 0;
    }
    else if(n==2)
    {
        cout<<4;
        return 0;
    }
```

```
    n-=3;
    while(n)
    {
        if(n&1) mul(f,f,a);
        n>>=1;
        mul(a,a,a);
    }
    cout<<f[0][3];
    return 0;
}
```

H、占宇哥的矩阵

思路

欧拉筛+单调队列
首先欧拉筛算出矩阵中每个元素的值，然后对每一行用单调队列算出每k个数中的最大值，再对每一列用单调队列算出每k个数中的最大值，最后遍历整个矩阵将每一个小矩阵的最大值加起来即为答案，此题难度大概为省赛金牌难度

代码

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1e6+5;
int primes[N],st[N],cnt,phi[N];
int a[1005][1005],b[1005][1005];
deque<int> q;
ll sum=0;
void ola(int n)
{
    phi[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(!st[i])
        {
            primes[cnt++]=i;
            phi[i]=i-1;
        }
        for(int j=0;primes[j]<=n/i;j++)
        {
            st[primes[j]*i]=1;
            if(i%primes[j]==0)
            {
                phi[i*primes[j]]=primes[j]*phi[i];
                break;
            }
            phi[i*primes[j]]=phi[i]*(primes[j]-1);
        }
    }
}
int main()
{
    ola(1000000);
    int n,m,k;
    cin>>m>>n>>k;
    for(int i=1;i<=m;i++)
    {
        for(int j=1;j<=n;j++)
        {
            a[i][j]=phi[i*j/__gcd(i,j)];
        }
    }
    for(int i=1;i<=m;i++)
    {
        q.clear();
```

```
        for(int j=1;j<=n;j++)
        {
            if(q.empty()) q.push_back(j);
            if(q.front()+k==j) q.pop_front();
            while(q.size()&&a[i][j]>=a[i][q.back()]) q.pop_back();
            q.push_back(j);
            if(j>=k) b[i][j]=a[i][q.front()];
        }
    }
    for(int i=k;i<=n;i++)
    {
        q.clear();
        for(int j=1;j<=m;j++)
        {
            if(q.empty()) q.push_back(j);
            if(q.front()+k==j) q.pop_front();
            while(q.size()&&b[j][i]>=b[q.back()][i]) q.pop_back();
            q.push_back(j);
            if(j>=k) sum+=b[q.front()][i];
        }
    }
    cout<<sum;
    return 0;
}
```

I、风暴风暴风暴

请参看https://www.nowcoder.com/discuss/618825?type=101 最后一题