2016 Multi-University Training Contest 2

比赛情况

题号	A	B	C	D	E	F	G	H	I	J	K	L	M
状态	O	-	-	-	O	O	-	-	O	-	O	Ø	-

O 在比赛中通过 Ø 赛后通过 ! 尝试了但是失败了 - 没有尝试

比赛时间

2020-05-27 13:00-18:00

提交记录

```
A: Accepted 2020-05-27 13:28:13

K: Accepted 2020-05-27 13:33:18

I: Accepted 2020-05-27 13:41:58

L: Time Limit Exceeded 2020-05-27 15:09:42

F: Wrong Answer 2020-05-27 15:56:01

F: Accepted 2020-05-27 16:24:15

E: Accepted 2020-05-27 17:30:56 （这个E写不过来了）
```

题解

F - Fantasia

题目大意

给定一个无向图G□定义G_i为该图去掉i点后的权重，图的权重定义如下

如果图连通，那么图的权重就是所有点权重的乘积

如果图不连通，那么图的权重就是图中若干连通块权重的加和。

现在求所有的G_i

数据范围

$n\leq 10^5$□$m\leq 2\times 10^5$

题解

意识到对于一个无向图，之后删除割点的时候才会产生新的连通块，那么我们可以对所有点双联通分量缩点，在森林上进行dp和处理即可。

这样写起来比较麻烦，据说官方题解是block forest data□可以等有时间了学习一手。

代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <vector>
#include <algorithm>
using namespace std;
typedef long long ll;
const ll mod = 1e9+7;
const int N = 1e5+5;
const int M = 2e5+5;
vector<int>edge[N];
vector<int>son[N];
void add(int x,int y) {
    edge[x].push_back(y);
    edge[y].push_back(x);
}
int n,m;
int w[N];
int dfn[N],low[N];
ll val[N],treeval[N];
int tree[N],root[N];
ll ans,ans1,ans2;
int treecount,clock;
ll quick_pow(ll x,ll y) {
    ll ans = 1;
    x = x%mod;
    while (y) {
        if (y&1)
            ans = ans*x%mod;
        y >>= 1;
        x = x*x%mod;
    }
    return ans;
}
ll Inv(ll x) {
    return quick_pow(x,mod-2);
}
int tarjan(int x,int fa) {
    int lowx = dfn[x] = clock++;
```

```
    tree[x] = treecount;
    ll sum = w[x];
    int sz = edge[x].size();
    for (int i = 0;i < sz;i++) {
        int to = edge[x][i];
        if (!dfn[to]) {
            int lowto = tarjan(to,x);
            lowx = min(lowx,lowto);
            son[x].push_back(to);
            sum = sum*val[to]%mod;
        } else {
            lowx = min(lowx, dfn[to]);
        }
    }
    val[x] = sum;
    return low[x] = lowx;
}
void init() {
    clock = 1;
    memset(dfn,0,sizeof(dfn));
    memset(root,0,sizeof(root));
    treecount = 0;
    scanf("%d%d",&n,&m);
    for (int i = 1;i <= n;i++)scanf("%d",&w[i]);
    for (int i = 1;i <= n;i++) {
        edge[i].clear();
        son[i].clear();
    }
    int x,y;
    for (int i = 1;i <= m;i++) {
        scanf("%d%d",&x,&y);
        add(x,y);
    }
}
int main()
{
    int cas;
    scanf("%d",&cas);
    while (cas--) {
        init();
        for (int i = 1;i<= n;i++) {
            if (!dfn[i]) {
                treecount++;
                tarjan(i,-1);
                treeval[treecount] = val[i];
                root[i] = 1;
            }
        }
        ll sum = 0;
        for (int i = 1;i <= treecount;i++)
            sum+= treeval[i];
```

```
        ans = 0;
        for (int i = 1;i<= n;i++) {
            ans2 = 0;
            ans1 = treeval[tree[i]];
            ll othersum = sum-ans1;
            ans1 = ans1*Inv(w[i])%mod;
            int sz = son[i].size();
            for (int j = 0;j < sz;j++) {
                int to = son[i][j];
                if (low[to] >= dfn[i]) {
                    ans1 = ans1*Inv(val[to])%mod;
                    ans2 = (ans2+val[to])%mod;
                }
            }
            if (root[i]) ans1--;
            (ans += (ans1 + ans2 + othersum)%mod*i%mod)%=mod;
        }
        printf("%lld\n",ans);
    }
}
```

I - It's All In The Mind

题目大意

假设有这样一个序列，其长度为n满足下述三个条件

1.对于任意的i有$0\leq a_i\leq 100$

2.序列是不上升的$a_n \leq a_{n-1} \leq $.....$\leq a_2 \leq a_1$

3.序列的和不为0

现在让你来构造这样一个序列，使得$\frac{a_1+a_2}{\sum_{i=1}^na_i}$最大

数据范围

$n\leq 100$

题解

贪心构造就好a_1和a_2尽量大，余下的数字尽量小。

代码

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <algorithm>
using namespace std;
const int N = 105;
int a[N];
int gcd(int x,int y) {
    if (y == 0) { return x; }
    return gcd(y,x%y);
}
int main()
{
    int cas;
    scanf("%d",&cas);
    while (cas--) {
        int n,m;
        scanf("%d%d",&n,&m);
        for(int i = 1;i<= n;i++)a[i] = -1;
        int x,y;
        for (int i = 1;i<= m;i++) {
            scanf("%d%d",&x,&y);
            a[x] = y;
        }
        int sum = 0;
        int ans = 0;
        for(int i = n;i>= 3;i--) {
            if (a[i] == -1)
                a[i] = ans;
            else
                ans = a[i];
            sum+=a[i];
        }
        ans = 100;
        for (int i = 1;i<= 2;i++) {
            if (a[i] == -1)
                a[i] = ans;
            else
                ans = a[i];
            sum+=a[i];
        }
        int q = gcd(a[1]+a[2],sum);
        printf("%d/%d\n",(a[1]+a[2])/q,sum/q);
    }
    return 0;
}
```

K-Keep on Movin

有 n 种字符，每种字符有 a_i 个，用所有字符组成多个回文串，问最短的回文串的最大值。

贪心的构造尽量少的回文串然后让长度尽可能平均，容易发现最少的回文串的个数等于个数为奇数的字符数，然后尽可能平均分配每个回文串即可。

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 2e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}
int n,a[maxn];
int main()
{
    fastio();
    int _;
    for(cin>>_;_;_--){
        cin>>n;
        rep(i,1,n) cin>>a[i];
        int cnt = 0;
        ll sum = 0,tot = 0;
        rep(i,1,n){
            if(a[i]&1) cnt++;
            sum += (ll)a[i]/2;
            tot += a[i];
        }
        if(cnt<=1){
            cout<<tot<<endl;
            continue;
        }
        ll ans = 1 + (sum/cnt)*2LL;
        cout<<ans<<endl;
    }
    return 0;
}
```

L-La Vie en rose

给两个长度分别为 n 和 m 的串 s,p，问对于$1\le i\le n$ p 在经过某种变换之后是否能完全匹配 $s_i s_{i+1} \ldots s_{i+m-1}$ ，这种变换定义为，任选 $1,2,..,m$ 中 k 个不相邻的下标 i_k，交换 p_{i_k} 和 p_{i_k+1}。

本来暴力 $nm=5e8$ 可以过的，好像这个赛后加强了数据，得用 bitset 优化一下才能过。

```
#include<bits/stdc++.h>
#define ll long long
#define pii_ pair<int,int>
#define mp_ make_pair
#define pb push_back
#define fi first
#define se second
#define rep(i,a,b) for(int i=(a);i<=(b);i++)
#define show1(a) cout<<#a<<" = "<<a<<endl
#define show2(a,b) cout<<#a<<" = "<<a<<"; "<<#b<<" = "<<b<<endl
using namespace std;
const ll INF = 1LL<<60;
const int inf = 1<<30;
const int maxn = 1e5+5;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

bitset<maxn> dp[2][3],f[30];
char s[maxn],p[5005];
int n,m;

int main()
{
    fastio();
    int _;
    for(cin>>_;_;_--){
        cin>>n>>m;
        cin>>s>>p;
        rep(i,0,26) f[i].reset();
        rep(i,0,2) dp[0][i].reset(),dp[1][i].reset();
        rep(i,0,n-1) f[s[i]-'a'][i] = 1;
        dp[0][1] = f[p[0]-'a'];
        if(m>1) dp[0][2] = f[p[1]-'a'];
        int now = 0;
        rep(i,1,m-1){
            now ^= 1;
            dp[now][0] = (dp[now^1][2]<<1) & f[p[i-1]-'a'];
            dp[now][1] = ((dp[now^1][1] | dp[now^1][0])<<1) & f[p[i]-'a'];
            if(i<m-1) dp[now][2] = ((dp[now^1][0] | dp[now^1][1])<<1) &
f[p[i+1]-'a'];
        }
        rep(i,0,n-1){
            if(dp[now][0][i+m-1] || dp[now][1][i+m-1]) cout<<1;
```

```
            else cout<<0;
        }
        cout<<endl;
    }
    return 0;
}
```