2020牛客暑期多校训练营（第九场）

比赛情况

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

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

2020-08-08 12:00-17:00

题解

A - Groundhog and 2-Power Representation

用栈模拟一下，并且用 py 水一下高精度

```
s = input().strip()

stack = []

ans = 0

for ch in s:
    if ch == ')':
        now = 0
        while 1:
            r = stack[-1]
            stack.pop()
            if r == '(':
                break
            if r != '+':
                now += int(r)
        stack.pop()
        stack.append(2**now)

    else:
        stack.append(ch)
for each in stack:
    if each != '+':
        ans += int(each)
print(ans)
```

F - Groundhog Looking Dowdy

我们考虑把所有衣服放到一起，然后把衣服按照邋遢值排序，然后做一个类似滑动窗口的东西，每次滑动保证窗口内有m天及以上可以穿的衣服，然后用当前的最大值和最小值相减一下更新答案即可。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6+5;
struct Node {
    int val,bel;
}clo[N<<1];
bool cmp(const Node &x,const Node &y) { return x.val < y.val; }
int buk[N],cnt,n,m;
int main()
{
    scanf("%d%d",&n,&m);
    int ki;
    for (int i = 1;i<= n;i++) {
        scanf("%d",&ki);
        for (int j = 1;j<= ki;j++) {
            cnt++;
            scanf("%d",&clo[cnt].val);
            clo[cnt].bel = i;
        }
    }
    sort(clo+1,clo+cnt+1,cmp);
    for (int i = 1;i<= n;i++)buk[i] = 0;
    int tail = 0;
    int ans = 1e9+5;
    int cntbel = 0;
    for (int i = 1;i<= cnt;i++) {
        while (tail+1<=cnt) {
            tail++;
            buk[clo[tail].bel]++;
            if (buk[clo[tail].bel] == 1)cntbel++;
            if (cntbel >= m) {ans = min(ans,clo[tail].val-
clo[i].val);break;}
        }
        buk[clo[i].bel]--;
        if (buk[clo[i].bel]==0)cntbel--;
    }
    printf("%d\n",ans);
    return 0;
}
```

==== G - Groundhog Chasing Death 求 $\product_{i=a}^b \product_{j=c}^d gcd(x^i,y^j)$

只有 x,y 的公共质因子会有贡献，枚举这些质因子，然后枚举 i可以用等差数列求出 j 的贡献。

```
#pragma GCC optimize(2)
#pragma GCC optimize(3,"Ofast","inline")
#include<bits/stdc++.h>
#define ALL(x) (x).begin(),(x).end()
#define ll long long
#define db double
#define ull unsigned 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 per(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;
const int M = 998244353;
inline void fastio() {ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);}

ll qpow(ll a,ll b) {ll s = 1;while(b){if(b&1)s=(s*a)%M;a=(a*a)%M;b>>=1;
}return s;}
vector<int> cop,X,Y;
int main()
{
    fastio();
    ll a,b,c,d,x,y;
    cin>>a>>b>>c>>d>>x>>y;
    //if(!a) a++; if(!b) b++;
    //if(!c) c++; if(!d) d++;
    int sqn = 1e5;
    rep(i,2,sqn){
        int t1 = 0,t2 = 0;
        if(i > x && i > y) break;
        while(x%i==0) x/=i,t1++;
        while(y%i==0) y/=i,t2++;
        if(t1 && t2){
            cop.pb(i);
            X.pb(t1);
            Y.pb(t2);
        }
    }
    if(x>1 && y>1 && x==y){
        cop.pb(x);
        X.pb(1);
```

```
        Y.pb(1);
    }
    int sz = X.size();
    ll ans = 1;
    ll M1 = M-1;
    rep(k,0,sz-1){
        ll u = X[k],v = Y[k];
        ll sum = 0;
        rep(i,a,b){
            ll un = u*i;
            ll lim = un / v;
            if(lim>=d){
                sum += (c+d)*(d-c+1)/2 * v;
                sum %= M1;
            }else if(lim<c){
                sum += (d-c+1) * un;
                sum %= M1;
            }else{
                sum += (c+lim)*(lim-c+1)/2 * v + (d-lim)*un;
                sum %=M1;
            }
        }
        ans = ans * qpow(cop[k],sum) % M;
    }
    cout<<ans<<endl;
    return 0;
}
/*
0 3000000 0 3000000 223092870 223092870
*/
```

K - The Flee Plan of Groundhog

因为被追的人可以停在某个位置不动，我们知道被追的人应该会跑到一个不会在中途被追到，并且离当前位置最远的地方躲起来，所以我们可以考虑先预处理出来追的人和被追的人到树上每一个点的距离，然后遍历一下每个点，判断一下能否在中途被追上，不会的更新一下答案即可。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5+5;
struct E {
    int nxt,to;
}e[N<<1];
int head[N],tot;
void add(int x,int y) {
    e[++tot].nxt = head[x];
```

```
    head[x] =  tot;
    e[tot].to = y;
}
int dis1[N],dis2[N],fa[N];
void dfs1(int x,int f) {
    fa[x] = f;
    if (x!=f)dis1[x] = dis1[f]+1;
    for (int i = head[x];i;i=e[i].nxt)
        if (e[i].to!=f)
            dfs1(e[i].to,x);
}
void dfs2(int x,int f) {
    if (x!=f)dis2[x] = dis2[f]+1;
    for (int i = head[x];i;i=e[i].nxt)
        if (e[i].to!=f)
            dfs2(e[i].to,x);
}
int main() {
    int n,t;
    scanf("%d%d",&n,&t);
    int x,y;
    for (int i = 1;i< n;i++) {
        scanf("%d%d",&x,&y);
        add(x,y);add(y,x);
    }
    dfs1(1,1);
    x = n;
    while (dis1[x] > t)
        x = fa[x];
    int ans = 0;
    dis1[x] = 0;
    dfs1(x,x);
    dis2[n] = 0;
    dfs2(n,n);
    for (int i = 1;i<= n;i++)
        if (2*dis1[i] <= dis2[i])
            ans = max(ans,((dis2[i]+1)>>1));
    printf("%d\n",ans);
    return 0;
}
```