2020.07.23codeforces加训

比赛情况

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

2020-07-23 12:00-17:00

题解

F - Empty Vessels

可以把最大的桶作为中介构造目标，之后的东西就是搜索并记录路径就好了。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e4+5;
int a[15],maxi;
bool vis[N];
int from[N];
queue<int>Q;
void print(int x,int dep) {
    if (x==0) {
        printf("%d\n",dep);
        return ;
    }
    if (a[from[x]]<=x) {
        print(x-a[from[x]],dep+2);
        printf("1 %d\n3 %d %d\n",from[x],from[x],maxi);
    } else {
        print(x-a[from[x]]+a[maxi],dep+4);
        printf("1 %d\n3 %d %d\n2 %d\n3 %d
%d\n",from[x],from[x],maxi,maxi,from[x],maxi);
    }
}
int main()
{
    int n,A;
    scanf("%d%d",&n,&A);
    maxi = 1;
```

```
    for (int i = 1;i<= n;i++)
    {
        scanf("%d",&a[i]);
        if (a[i]>a[maxi])maxi = i;
    }
    vis[0] = true;
    Q.push(0);
    while (!Q.empty()) {
        int x = Q.front();
        Q.pop();
        for (int i = 1;i<= n;i++)
            if (!vis[(x+a[i])%a[maxi]]) {
                vis[(x+a[i])%a[maxi]] = true;
                from[(x+a[i])%a[maxi]] = i;
                Q.push((x+a[i])%a[maxi]);
            }
    }
    if (!vis[A]) {
        if (A==a[maxi]) printf("1\n1 %d\n",maxi);
        else printf("-1");
        return 0;
    }
    print(A,0);
    return 0;
}
```

G - Maximum Product

可以意识到，乘积最大的必然是在区间L,R内并且有若干个9的，我们考虑找一个小于R的，最后若干位为9的数字，然后看这个数字是否大于L□如果大于则可以更新答案。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll calc(ll a) {
    ll ans = 1;
    while (a) {
        ans = ans*(a%10);
        a = a/10;
    }
    return ans;
}
ll Pow10(int x) {
    ll ans = 1;
    for (int i = 1;i<= x;i++)
        ans = ans*10;
```

```
    return ans;
}
int main()
{
    ll a,b;
    scanf("%lld%lld",&a,&b);
    ll ans = calc(b);
    ll tans = b;
    for (int i = 1;i<= 18;i++) {
        ll now = b;
        int newn[20]={0},newlen=0;
        for (int j = 1;j<= i && now;j++) {
            if (now >= 10) {
                if (now%10!=9) now -= 10;
                newn[++newlen] = 9;
            } else {
                newn[++newlen] = now;
            }
            now /= 10;
        }
        ll tnewn = 0;
        for (int i = newlen;i>=1;i--)
            tnewn = tnewn*10+newn[i];
        tnewn = now*Pow10(newlen)+tnewn;
        if (tnewn >= a && calc(tnewn) > ans) {
            ans = calc(tnewn);
            tans = tnewn;
        }
    }
    printf("%lld\n",tans);
    return 0;
}
```

I - Archaeological Research

这道题的题意可以转换为要构造一个字符串，给定一些限制条件，这个限制条件是要求i和在区间$(j+1,i-1)$之间的字符不能重复，求一个字典序最小的字符串。

我们考虑贪心构造，我们对于i位置要选择的字符就是出现在了$1,j$但没有出现在$j+1,i-1$中的字符最小的那一个，这样我们可以维护一个线段树，只保留每种字符最后一次出现的位置，其他出现的都在线段树上置为inf即可。

```
#include<bits/stdc++.h>
using namespace std;
const int inf = 1e9+5;
const int N = 3e5+5;
int tr[N<<2],last[N],pos[N],tans[N];
```

```
void build(int p,int l,int r) {
    tr[p] = inf;
    if (l==r) { return;}
    int mid = (l+r)>>1;
    build(p<<1,l,mid);
    build(p<<1|1,mid+1,r);
}
void Update(int p,int l,int r,int a,int b) {
    if (l==r) {
        tr[p] = b;
        return ;
    }
    int mid = (l+r)>>1;
    if (a<=mid)Update(p<<1,l,mid,a,b);
    else Update(p<<1|1,mid+1,r,a,b);
    tr[p] = min(tr[p<<1],tr[p<<1|1]);
}
int Getans(int p,int l,int r,int a,int b) {
    if (l>=a&&r<=b) return tr[p];
    int ans = inf;
    int mid = (l+r)>>1;
    if (a<=mid)ans = min(ans,Getans(p<<1,l,mid,a,b));
    if (b>mid)ans = min(ans,Getans(p<<1|1,mid+1,r,a,b));
    return ans;
}
int main()
{
    int n;
    scanf("%d",&n);
    int m,x;
    for (int i = 1;i<= n;i++)pos[i] = i;
    for (int i = 1;i<= n;i++) {
        scanf("%d",&m);
        for (int j = 1;j<= m;j++) {
            scanf("%d",&x);
            pos[x] = min(pos[x],i+1);
        }
    }
    build(1,1,n);
    tans[1] = 1;
    last[1] = 1;
    int c = 1;
    Update(1,1,n,1,1);
    for (int i = 2;i<= n;i++) {
        int tmp = Getans(1,1,n,1,pos[i]-1);
        if (tmp == inf) tans[i] = ++c;
        else tans[i] = tmp;
        if (last[tans[i]]) Update(1,1,n,last[tans[i]],inf);
        Update(1,1,n,i,tans[i]);
        last[tans[i]] = i;
```

```
    }
    for (int i = 1;i<= n;i++)
        printf("%d ",tans[i]);
    return 0;
}
```

K - Toll Roads

一个比较麻烦的树上dp□因为数据范围很小，边权置零的还是一条路径，所以我们可以枚举这条路径的一点，然后以这个点作为根开始dp□先与处理好每个子树中最长的链和子树直径，然后对所有深度小于k的点作为路径的另一端点，假设根为root□当前点为x□这时这个树的直径可能是x的子树直径，可能是root到x路径上延展出去的另外的子树的直径，可能是root到x路径上延展出去的另外的两个子树的最长链拼凑，还可能是root到x路径上延展出去的另外的子树的链和当前链的拼凑。多种情况讨论一下就行了。

很多 很多细节。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 5010;
const int inf = 1e9+5;
struct E
{int nxt,to;}e[N<<1];
int head[N],tot,n,k,ans,ansdp,ans1,ans2,root;
void add(int x,int y) {
    e[++tot].nxt = head[x];head[x] = tot;e[tot].to = y;
    e[++tot].nxt = head[y];head[y] = tot;e[tot].to = x;
}
int zj[N],lc[N],dep[N];
void dfs(int x,int fa) {
    dep[x] = dep[fa]+1;
    zj[x] = lc[x] = 0;
    for (int i= head[x];i;i=e[i].nxt) if (e[i].to!=fa) {
        dfs(e[i].to,x);
        zj[x] = max(max(zj[x],zj[e[i].to]),lc[e[i].to]+lc[x]+1);
        lc[x] = max(lc[e[i].to]+1,lc[x]);
    }
}
void dfs2(int x,int fa,int ZJ,int LC) {
    if (dep[x] > k) return;
    int tmp = max(max(ZJ,zj[x]),LC+lc[x]);
    if (tmp < ans || (tmp == ans && dep[x] < ansdp)) {
        ansdp = dep[x];
        ans = tmp;
        ans1 = root-1;ans2 = x-1;
    }
    int zj1,zj2,lc1,lc2,lc3;
    zj1 = zj2 = 0;
```

```
    lc1 = lc2 = lc3 = -1;
    for (int i = head[x];i;i=e[i].nxt) if (e[i].to!=fa) {
        int y = e[i].to;
        if (zj[y] > zj2) zj2 = zj[y];
        if (zj2 > zj1) swap(zj1,zj2);
        if (lc[y] > lc3) lc3 = lc[y];
        if (lc3 > lc2) swap(lc3,lc2);
        if (lc2 > lc1) swap(lc2,lc1);
    }
    for (int i = head[x];i;i=e[i].nxt) if (e[i].to!=fa) {
        int nzj1 = lc1+lc2+2;
        int y = e[i].to;
        if (lc[y] == lc1) nzj1 = lc2+lc3+2;
        if (lc[y] == lc2) nzj1 = lc1+lc3+2;
        int nzj2 = zj1;
        if (zj[y] == zj1) nzj2 = zj2;
        int nlc = lc1;
        if (lc[y] == lc1) nlc = lc2;
        dfs2(y,x,max(max(nzj2,ZJ),max(nzj1,LC+1+nlc)),max(LC,nlc+1));
    }
}
int main()
{
    scanf("%d%d",&n,&k);
    int x,y;
    for (int i = 1;i<n;i++) {
        scanf("%d%d",&x,&y);
        x++;y++;
        add(x,y);
    }
    dep[0] = -1;
    ans = inf;
    for (root = 1;root<=n;root++) {
        dfs(root,0);
        dfs2(root,0,0,0);
    }
    printf("%d\n%d\n",ans,ansdp);
    if (ansdp!=0) printf("%d %d\n",ans1,ans2);
    return 0;
}
```

比赛总结与反思