2020牛客暑期多校训练营（第五场）

比赛网址

训练结果

- 时间:2020-07-25 12:00~17:00
- rank:181/1116
- 完成情况：4/7/11

题解

A.Portal

题意

有一个n个点m条边的带权图，你一开始在1号点，你要按顺序完成k个任务，第i个任务是先去a_i再走到b_i当你走到一个点上的时候，你可以在这个点创建一个传送门。当同时存在两个传送门的时候， 你可以在传送门之间不耗代价地传送。如果已经存在了两个传送门，你想再创建一个，就必须选择之前 的一个传送门关掉（关掉这个操作不耗时间，并且是远程操作，不需要走过去）。问完成所有任务的最 短总行走距离。

题解

一道很有NOIP模拟赛风格的题，直接把题意翻译成依次经过$2k$个点。

先考虑最暴力的DP$f[i][u][a][b]$表示当前已经完成了前i个结点，当前在u俩传送门分别在ab$的最小距离，这个玩意得通过Dijkstra转移。然后发现我们没有必要把俩传送门位置都记录，因为剩下一个我们就扔在脚底下就能完全包含，还得通过Dijkstra转移。继续想，直接把u扔掉，当前已经完成了前i个节点，传送门在a,那么我们考虑从i走到$i+1$有几种方式呢？我们可以直接走过去，还可以在脚下丢个传送门再瞬移到另一个传送门再去$i+1$还可以考虑转移到$i+1$的时候传送门到b了，可以从i瞬移到a,然后走到b,在b处放下传送门，最后再走到$i+1$,还可以直接从i走到b放下传送门，然后再瞬移到a再去$i+1$或者直接走去$i+1$复杂度$O(2k*N^2)$

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
```

```
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=310;
int n,m,K,u,v,w,pos[maxn];
LL dis[maxn][maxn],dp[maxn<<1][maxn],ans=1ll<<40;
int main()
{
    mem(dis,42);mem(dp,42);
    n=read();m=read();K=read();
    for(int i=1;i<=n;i++)dis[i][i]=0;
    while(m--
)u=read(),v=read(),w=read(),dis[u][v]=min(dis[u][v],(LL)w),dis[v][u]=min(dis
[v][u],(LL)w);
    for(int k=1;k<=n;k++)
        for(int i=1;i<=n;i++)
            for(int j=1;j<=n;j++)
                dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j]);
    pos[0]=1;dp[0][1]=0;
    for(int i=1;i<=2*K;i++)
    {
        pos[i]=read();
        for(int j=1;j<=n;j++)
        {
            dp[i][j]=min(dp[i][j],dp[i-1][j]+dis[pos[i-1]][pos[i]]);
            dp[i][j]=min(dp[i][j],dp[i-1][j]+dis[j][pos[i]]);
            for(int k=1;k<=n;k++)
                if(k!=j)
                {
dp[i][k]=min(dp[i][k],dp[i-1][j]+dis[j][k]+dis[k][pos[i]]);
dp[i][k]=min(dp[i][k],dp[i-1][j]+dis[pos[i-1]][k]+min(dis[k][pos[i]],dis[j][
pos[i]]));
                }
        }
    }
    for(int i=1;i<=n;i++)ans=min(ans,dp[2*K][i]);
    printf("%lld\n",ans);
    return 0;
}
```

B.Graph

题意

给你一棵树，你可以对这树上进行一些修改，修改要求和限制如下：

- 全程整个图要联通

- 可以加上或者删去一条边
- 若图中出现环，必须保证这个环的一圈异或和为0

问最终这个图的最小权值和是多少。

题解

由于异或的神奇性质，可以发现按照上述操作的话任意两点的边权都是由他们到根上的异或路径再异或得到的，也就是说是固定的，于是我们把到根的异或和变成点的点权，这个问题就变成了很经典的Xor-MST问题了。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=100010;
int n,a,b,c,ce=-1,v[maxn],tot;
int first[maxn];
struct Edge
{
    int u,v,w,next;
    Edge() {}
    Edge(int _1,int _2,int _3,int _4):u(_1),v(_2),w(_3),next(_4) {}
}e[maxn<<2];
void addEdge(int a,int b,int c)
{
    e[++ce]=Edge(a,b,c,first[a]);first[a]=ce;
    e[++ce]=Edge(b,a,c,first[b]);first[b]=ce;
}
void Dfs(int now,int fa)
{
    for(int i=first[now];i!=-1;i=e[i].next)
        if(e[i].v!=fa)
            v[e[i].v]=v[now]^e[i].w,Dfs(e[i].v,now);
}
struct Node{int ch[2];}t[maxn<<5];
void insert(int &x,int w,int p)
{
    if(!x)x=++tot,t[x].ch[0]=t[x].ch[1]=0;
    if(p==-1)return;
```

```
    insert(t[x].ch[(w>>p)&1],w,p-1);
}
int Query(int x,int w,int p)
{
    if(p==-1)return 0;int c=(w>>p)&1;
    if(t[x].ch[c])return Query(t[x].ch[c],w,p-1);
    else return Query(t[x].ch[c^1],w,p-1)^(1<<p);
}
LL Solve(vector<int> v,int p)
{
    if(!v.size()||p==-1)return 0;
    vector<int> d[2];int ret=0,rt;
    for(int i:v)d[(i>>p)&1].push_back(i);
    if(d[0].size()&&d[1].size())
    {
        ret=1<<(p+1);rt=tot=0;
        for(int i:d[0])insert(rt,i,30);
        for(int i:d[1])ret=min(ret,Query(rt,i,30));
    }
    return ret+Solve(d[0],p-1)+Solve(d[1],p-1);
}
int main()
{
    mem(first,-1);
    n=read();
    for(int i=1;i<n;i++)a=read(),b=read(),c=read(),addEdge(a,b,c);
    Dfs(0,0);
    vector<int> A;
    for(int i=1;i<=n;++i)A.push_back(v[i]);
    printf("%lld\n",Solve(A,30));
    return 0;
}
```

C.Easy

题意

选取两个长度都为K序列A和B□满足$\sum_{i = 1}^{K} a_i=N$且 $\sum_{i = 1}^{K} b_i = M$

此时会得到分数为$P = \prod_{i=1}^{K}min(a_i,b_i)$

求所有可能的序列的分数之和模998244353

题解

不妨设$n<m$

构造如下生成函数

$$(x + x^2 + \ldots+x^n)^k(y + y^2 + \ldots+y^m)^k$$

则 x^ny^m的系数为方案数

现在需要给每种方案一个P的系数

假如x^iy^j中i是较小的，那么如果同时抹去一对xy那么恰好有i中情况，也就是说x^iy^j能从i中$x^{i - t}y^{j - t}$乘上$(xy)^{t}$得来

那么构造

$$(x + x^2 + \ldots+x^n)^k(y + y^2 + \ldots+y^m)^k(1+xy+(xy)^2+\ldots+(xy)^{n - k})^{k}$$

$x^n y^m$系数即为所求

经过推导

最后推出式子为

$$\sum_{i=0}^{n-k}C_{k - i - 1}^{i}C_{k + n - k - i - 1}^{n - k - i}C_{k + m - k - i - 1}^{m - k - i}$$

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 1000005,maxm = 100005,INF = 0x3f3f3f3f,P = 998244353;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
int fac[maxn],inv[maxn],fv[maxn];
void init(){
    int E = 1000000;
    fac[0] = 1;
```

```
    for (int i = 1; i <= E; i++) fac[i] = 1ll * fac[i - 1] * i % P;
    inv[0] = inv[1] = 1;
    for (int i = 2; i <= E; i++) inv[i] = 1ll * (P - P / i) * inv[P % i] %
P;
    fv[0] = 1;
    for (int i = 1; i <= E; i++) fv[i] = 1ll * fv[i - 1] * inv[i] % P;
}
int C(int n,int m){
    return 1ll * fac[n] * fv[m] % P * fv[n - m] % P;
}
int main(){
    init();
    int T = read();
    while (T--){
        int n = read(),m = read(),k = read(),ans = 0;
        if (n > m) swap(n,m);
        for (int i = 0; i <= n - k; i++){
            ans = (ans + 1ll * C(k + i - 1,i) * C(n- i - 1,n - k - i) % P *
C(m - i - 1,m - k - i) % P) % P;
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

D.Drop Voicing

题意

有一个n个数字的环，每次可以选择一个位置插入到另一个位置，问至少操作几次可以使得环有序

题解

枚举环断开的位置，就成了链上的插入排序问题，计算出LIS后剩下的就是最少移动次数

```
#include<algorithm>
#include<iostream>
#include<cstdlib>
#include<cstring>
#include<cstdio>
#include<vector>
#include<queue>
#include<cmath>
#include<map>
#include<set>
#define LL long long int
```

```cpp
#define REP(i,n) for (int i = 1; i <= (n); i++)
#define Redge(u) for (int k = h[u],to; k; k = ed[k].nxt)
#define cls(s,v) memset(s,v,sizeof(s))
#define mp(a,b) make_pair<int,int>(a,b)
#define cp pair<int,int>
using namespace std;
const int maxn = 100005,maxm = 100005,INF = 0x3f3f3f3f;
inline int read(){
    int out = 0,flag = 1; char c = getchar();
    while (c < 48 || c > 57){if (c == '-') flag = 0; c = getchar();}
    while (c >= 48 && c <= 57){out = (out << 1) + (out << 3) + c - 48; c =
getchar();}
    return flag ? out : -out;
}
int n,p[maxn];
int f[maxn],len[maxn],A[maxn];
int getlis(){
    for (int i = 0; i <= n; i++) len[i] = INF;
    len[0] = 0;
    int ans = 0;
    REP(i,n){
        int L = 0,R = ans;
        while (L < R){
            int mid = (L + R + 1) >> 1;
            if (len[mid] < A[i]) L = mid;
            else R = mid - 1;
        }
        f[i] = L + 1;
        len[f[i]] = min(len[f[i]],A[i]);
        ans = max(ans,f[i]);
    }
    return ans;
}
int main(){
    int ans = INF;
    n = read();
    for (int i = 0; i < n; i++) p[i] = read();
    for (int i = 0; i < n; i++){
        for (int j = 1; j <= n; j++) A[j] = p[(i + j - 1) % n];
        ans = min(ans,n - getlis());
    }
    printf("%d\n",ans);
    return 0;
}
```

E.Bogo Sort

题意

给你个$1到n$的置换，问有多少个排列能经过若干次这样的置换变成顺序序列。

题解

单考虑一个环，肯定是这个环对着转的数量是len那么若干个环呢？那就求一下他们长度的lcm.

本次用的是高精度板子，下回碰到这种直接上python

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=100010;
int n,to[maxn],size[maxn],cnt;
bool vis[maxn];
struct data
{
    int l,v[maxn];
    data(){l=1;memset(v,0,sizeof(v));}
    data operator = (const int& t)
    {
        l=1;v[1]=t;
        while(v[l]>9) v[l+1]+=v[l]/10,v[l]%=10,l++;
        return *this;
    }
    data operator * (const int& t)
     {
        data c;c.l=l;
        for(int i=1;i<=l;i++)c.v[i]=v[i]*t;
        for(int i=1;i<c.l;i++)c.v[i+1]+=c.v[i]/10,c.v[i]%=10;
        while(c.v[c.l]>9)c.v[c.l+1]+=c.v[c.l]/10,c.v[c.l]%=10,c.l++;
        return c;
    }
    data operator *= (const int& t)
    {
        *this=*this*t;
        return *this;
    }
    data operator / (const int& t)const
```

```
    {
        data c;c.l=0;int f=0;
        for(int i=l;i;i--)
        {
            f=f*10+v[i];
            if(f>=t)c.l=max(c.l,i);
            c.v[i]=f/t;
            f%=t;
        }
        return c;
    }
    int operator % (const int& t)const
     {
        int f=0;
         for(int i=l;i;i--)f=(f*10+v[i])%t;
         return f;
     }
};
int gcd(int a,int b){return b==0 ? a : gcd(b,a%b);}
void print(data s){for(int i=min(n,s.l);i;i--)printf("%d",s.v[i]);}
int main()
{
    n=read();
    for(int i=1;i<=n;i++)to[i]=read();
    for(int i=1;i<=n;i++)
        if(!vis[i])
        {
            int now=i;
            size[++cnt]=1;
            vis[now]=1;
            while(!vis[to[now]])
                vis[to[now]]=1,now=to[now],size[cnt]++;
        }
    sort(size+1,size+cnt+1);
    cnt=unique(size+1,size+cnt+1)-size-1;
    data ans;
    ans=size[1];
    int GCD;
    for(int i=2;i<=cnt;i++)
    {
        int tmp=ans%size[i];
        GCD=gcd(tmp,size[i]);
        ans=ans*size[i];
        ans=ans/GCD;
    }
    print(ans);
    return 0;
}
```

F.DPS

题意

模拟

题解

大水题

```
```

H.Interval

题意

题解

```
```

I.Hard Math Problem

题意

你需要在一个无限大的平面网格上放置 G 个H 个E 三种物品，其中 H 的四周必须有至少一个 G 和 E 个问 H 的最大占比是多少

题解

假设 每个 H 的四周都只有一个 G 和 E 个那么一个 G 和 E 最多可以满足四个 H 个所以占比就是 4/6

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#define ll long long
using namespace std;
int read()
{
    int k=0,f=1;char c=getchar();
```

```
    for(;!isdigit(c);c=getchar()) if(c=='-') f=-1;
    for(;isdigit(c);c=getchar()) k=k*10+c-'0';return k*f;
}
int main()
{
    puts("0.666667");
    return 0;
}
```

K.Git Merge

题意

给你一份俩哥们不同风格混起来的代码，让你通过#ifdef branch1 #ifdefbranch2 #else #endif这五个编译命令融合进代码，使得总代码长度最短。

题解

首先给整个代码标号，第一个人写的部分标为1，第二个人写的部分标为2，公共部分为0，那么第一个人完整版就是01，第二个人是02，之后就是类似最长公共子序列的DP了来求融合代码总长度了。
设$dp[i][j][0/1/2]$表示当前匹配到了第一份代码的第i行，第二份代码的第j行，0表示当前已经放下了endif，1表示还在branch1中，2表示在branch2中，转移是$O(1)$的。DP完之后从最后一步往回搜来进行还原源代码。

```
#include<bits/stdc++.h>
using namespace std;
#define mem(a,b) memset(a,b,sizeof(a))
typedef long long LL;
typedef unsigned long long ULL;
typedef pair<int,int> PII;
#define X first
#define Y second
inline int read()
{
    int x=0,f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-')f=-1;c=getchar();}
    while(isdigit(c)){x=x*10+c-'0';c=getchar();}
    return x*f;
}
const int maxn=4010;
string a,s[maxn],prt[maxn];
int
n,Case,tot,id[maxn],list1[maxn],list2[maxn],l1,l2,dp[maxn][maxn][3],len;//[0/1/2]表示是否当前在分支中(是否放下了endif)
ULL h[maxn];
bool cmp(int a,int b){return h[list1[a]]==h[list2[b]];}
ULL Hash(string str)
{
```

```
    ULL res=0;
    for(int i=0;i<str.length();i++)res=res*133+str[i];
    return res;
}
void dfs(int i,int j,int k)
{
    if(i==0 && j==0)return;
    if(cmp(i,j) && k==0)
    {
        if(i && j && dp[i][j][0]==dp[i-1][j-1][0]+1)
        {
            prt[++len]=s[list1[i]];
            return dfs(i-1,j-1,k);
        }
        if(i && j && (dp[i][j][0]==dp[i-1][j-1][2]+2 ||
dp[i][j][0]==dp[i-1][j-1][1]+2))
        {
            prt[++len]=s[list1[i]];prt[++len]="#endif";
            return dp[i][j][0]==dp[i-1][j-1][2]+2 ? dfs(i-1,j-1,2) :
dfs(i-1,j-1,1);
        }
    }
    else if(k==0)
    {
        prt[++len]="#endif";
        return dp[i][j][0]==dp[i][j][1]+1 ? dfs(i,j,1) : dfs(i,j,2);
    }
    else if(k==1)
    {
        prt[++len]=s[list1[i]];
        if(i && dp[i][j][1]==dp[i-1][j][1]+1)return dfs(i-1,j,1);
        if(i && dp[i][j][1]==dp[i-1][j][0]+2){prt[++len]="#ifdef
branch1";return dfs(i-1,j,0);}
        return;
    }
    else
    {
        prt[++len]=s[list2[j]];
        if(j && dp[i][j][2]==dp[i][j-1][2]+1)return dfs(i,j-1,2);
        if(j && dp[i][j][2]==dp[i][j-1][1]+2){prt[++len]="#else";return
dfs(i,j-1,1);}
        if(j && dp[i][j][2]==dp[i][j-1][0]+2){prt[++len]="#ifdef
branch2";return dfs(i,j-1,0);}
        return;
    }
}
int main()
{
    while(getline(cin,a))
    {
```

```
        if(a[0]=='<')Case=1;
        else if(a[0]=='=')Case=2;
        else if(a[0]=='>')Case=0;
        else ++n,s[n]=a,id[n]=Case;
    }
    for(int i=1;i<=n;i++)h[i]=Hash(s[i]);
    for(int i=1;i<=n;i++)
    {
        if(id[i]!=2)list1[++l1]=i;
        if(id[i]!=1)list2[++l2]=i;
    }
    for(int i=0;i<=l1;i++)for(int
j=0;j<=l2;j++)dp[i][j][0]=dp[i][j][1]=dp[i][j][2]=maxn;
    dp[0][0][0]=0;
    for(int i=0;i<=l1;i++)
        for(int j=0;j<=l2;j++)
        {
            if(i && j &&
cmp(i,j))dp[i][j][0]=min(dp[i-1][j-1][0]+1,min(dp[i-1][j-1][1]+2,dp[i-1][j-1
][2]+2));
            if(i)dp[i][j][1]=min(dp[i-1][j][1]+1,dp[i-1][j][0]+2);
if(j)dp[i][j][2]=min(dp[i][j-1][2]+1,min(dp[i][j-1][1]+2,dp[i][j-1][0]+2));
            dp[i][j][0]=min(dp[i][j][0],min(dp[i][j][1]+1,dp[i][j][2]+1));
        }
    dfs(l1,l2,0);
    for(int i=len;i;i--)cout<<prt[i]<<endl;
    return 0;
}
```

训练实况

别问，本场白给 2点过了四道签到题 之后垃圾时间

训练总结

wxg: 这场题目偏难，而且讨论的时候因为自己不会最小异或生成树就没提，导致成绩惨淡。

hxm:这场比赛偏难，菜是原罪

fyh: