Huantwofat 之 ACM 模板库

目录

1 扩展欧几里得 2
2 巴什博弈模板 3
3 博弈论之 Grungy 数及 sg 函数 3
4 网络流 Dinic 算法 4
5 polya 定理模板 6
6 超级 gcd 模板 7
7 威佐夫博奕模板 7
8 LCA 之离线算法（tarjan） 8
9 卢卡斯定理模板（大数组合） 10
10 LCA 在线算法 11
11 强连通分量（tarjan 算法） 12
12 欧拉错排公式 14
13 三分模板 14
14 最小费用最大流模板 15
15 中国剩余定理模板 17
16 后缀数组（dc3） 18
17 数据输入加速 20
18 平面的划分 21
19 高斯消元（解方程） 21
20 高斯消元 开关类问题（异或方程） 23
21 四边形优化模板 24
22 斜率 dp 模板 25
23 分解大数质因数模板 27
24 强连通分量（桥） 29
25 强连通分量（割点） 30
26 网络流判圈 31
27、无源无汇上下限网络流 35
28 匈牙利算法 39
28 有源汇的上下界最小流 39
29 上下限网络流 43
30 枚举一个数二进制表示下的子集 44
31 上下限网络流大攻略 44
32 AC 自动机数组实现 44
33 树链剖分模板（改边） 46
34 树链剖分模板（改点） 49
35 第一类斯特林数 52
36 第二类斯特林数 52
37 KM 算法模板（二分图的最大权匹配） 53
38 mancher(O(n)回文串)算法 54
39 无向图最小割 store-wragner 算法 55
40 原根判断条件和个数 56
41 泛化背包 57
42 二维 RMQ 58

43 欧拉路模板 59
44 一堆石子成块减一的总操作数 61
45 主席树模板 61
46 二维线段树 63
47 莫队： 65
48 后缀自动机 66
49 阶梯博弈 67
50 素数定理 68
51 得分序列合法条件 68
52 SBT 69
53 splay 73
54 回文自动机 77
55 长度不小于 k 的公共子串的个数 79
56 树上莫队 80
57 Matrix-tree 定理 83
58 FFT 84
59 树上莫队带修改 86
60 原始对偶 90
61 Hopcroft-Carp 92
62 常用外挂 93

1 扩展欧几里得

```
int x,y;
int exgcd(int a,int b)
{
   if(b==0)
   {
      x=1;
      y=0;
      return a;
   }
   int p,t;
   p=exgcd(b,a%b);
   t=x;
   x=y;
   y=t-a/b*y;
   return p;
         }
```

2 巴什博弈模板

//只有一堆 n 个物品,两个人轮流从这堆物品中取物,规定每次至少取一个,最多取 m 个.最后取光者得胜.

//若(m+1) | n，则先手必败，否则先手必胜。

//显然,如果 n=m+1,那么由于一次最多只能取 m 个,所以,无论先取者拿走多少个,后取者都能够一次拿走剩余的物品,后者取胜.因此我们发现了如何取胜的法则：如果 n=(m+1)r+s,(r 为任意自然数,s≤m),那么先取者要拿走 s 个物品,如果后取者拿走 k(≤m)个,那么先取者再拿走 m+1-k 个,结果剩下(m+1)(r-1)个,以后保持这样的取法,那么先取者肯定获胜.总之,要保持给对手留下(m+1)的倍数,就能最后获胜.

3 博弈论之 Grungy 数及 sg 函数

```
#include<stdio.h>
#include<string.h>
#define N 10000
int grun[N],b[N];
bool ha[N];
void grundy(int n)
{
   int i,j;
   memset(grun,0,sizeof(grun));
   for(i=1;i<=N;i++)
   {
      memset(ha,0,sizeof(ha));
      for(j=0;j<n;j++)
      {
         if(i>=b[j])
         {
            ha[grun[i-b[j]]]=1;
         }
      }
      for(j=0;j<=N;j++)
      {
         if(!ha[j])
            break;
      }
      grun[i]=j;
   }
}
//1.可选步数为 1~m 的连续整数，直接取模即可，SG(x) = x % (m+1);

//2.可选步数为任意步，SG(x) = x;

//3.可选步数为一系列不连续的数，用 GetSG()
```

4 网络流 Dinic 算法

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define inf 100000000
using namespace std;
struct pi
{
   int to;
   int cost;
   int rev;
}pp;
vector<pi >g[505];
queue<int>q;
int s,t;
int line[505],leve[505];
int map[505][505];
int min(int a,int b)
{
   int p;
   p=a;
   if(b<a)
      p=b;
   return p;
}
void add(int a,int b,int cos)
{
   pp.to=b;
   pp.cost=cos;
   pp.rev=(int)g[b].size();
   g[a].push_back(pp);
   pp.to=a;
   pp.cost=0;
   pp.rev=(int)g[a].size()-1;
   g[b].push_back(pp);
   return ;
}
void bfs(void)
{
   q.push(s);
   int p,f,i;
   while(!q.empty())
   {
      p=q.front();
      q.pop();
      f=(int)g[p].size();
      for(i=0;i<f;i++)
```

```
        {
            pi &e=g[p][i];
            if(e.cost>0&&line[e.to]<0)
            {
                line[e.to]=line[p]+1;
                q.push(e.to);
            }
        }
    }
    return ;
}
int dfs(int v,int f)
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
long long  dinic()
{
    int f;
    long long flow=0;
    while(1)
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
        {
            flow+=f;
        }
    }
```

```
}
int main()
{
   int i;
   for(i=1;i<=s;i++)
      g[i].clear();
         }
```

5 polya 定理模板

```
//设 G 是 p 个对象的一个置换群，用 m 种颜色涂染 p 个对象，则不同染色方案为：
//l=(sigma(m^(c(g[i]))))/|G|;c[g[i]]为 c[i]循环节个数。
//其中 G={g1 ,…gs}  c(gi )为置换 gi 的循环节数(i=1…s)
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<math.h>
using namespace std;
int gcd(int a,int b)
{
   while(b^=a^=b^=a%=b)
      ;
   return a;
}
//为环时的 polay
void polya(int n,int m)
{
   int s=0,i,j,p;
   for(i=1;i<=n;i++)
   {
      s+=(int)pow((double)m,(double)gcd(n,i));
   }
   if(n&1)
      s+=n*(int)pow((double)m,(double)(n+1)/2);
   else
   {
      p=(int)pow((double)m,(double)n/2);
      s+=n/2*p+n/2*p*m;
   }
   s=s/n/2;
   return ;
}
//1. 对于旋转，有 c(gi) = gcd(n,i)，i 为转动几颗珠子。
//2. 对于翻转，当 n 为奇数时，c(gi) = n/2+1；当 n 为偶数时，有 n/2 个的循环节数为 n/2+1，有
n/2 个的循环节数为 n/2。
//如果旋转 4 下，及正方形的旋转//l=(m^(n^2)+2*m^([(n^2+3)/4])+m^([(n^2+1)/2]))
//为正方形时
int polya2(int n,int m)
{
```

```
    return
((int)pow((double)m,n*n)+2*(int)pow((double)m,(n*n+3)/4)+(int)pow((double)m,(n*n+1)/2))/4;
        }
```

6 超级 gcd 模板

```
int gcd(int a, int b)
{
        while ( b ^= a ^= b ^= a %= b );
        return a;
}
```

7 威佐夫博奕模板

```
//有两堆各若干个物品,两个人轮流从某一堆或同时从两堆中取同样多的物品,规定每次至少取一个,多者不限,最后取光者得胜.这种规则下游戏是颇为复杂的。我们用（a[k]，b[k]）（a[k] ≤ b[k] ,k=0，1，2，...,n)表示两堆物品的数量并称其为局势。如果甲面对（0，0），那么甲已经输了，这种局势我们称为奇异局势。
//奇异局势下先手必败，非奇异局势下先手必胜。
//这种情况下是颇为复杂的.我们用(ak,bk)(ak ≤bk ,k=0,1,2,...,n)表示两堆物品的数量并称其为局势,如果甲面对(0,0),那么甲已经输了,这种局势我们称为奇异局势.前几个奇异局势是：(0,0)、(1,2)、(3,5)、(4,7)、(6,10)、(8,13)、(9,15)、(11,18)、(12,20).
//可以看出,a0=b0=0,ak 是未在前面出现过的最小自然数,而 bk= ak + k,奇异局势有如下三条性质：
//1、任何自然数都包含在一个且仅有一个奇异局势中.
//由于 ak 是未在前面出现过的最小自然数,所以有 ak > ak-1 ,而 bk= ak + k > ak-1 + k-1 = bk-1 > ak-1 .所以性质 1.成立.
//2、任意操作都可将奇异局势变为非奇异局势.
//事实上,若只改变奇异局势(ak,bk)的某一个分量,那么另一个分量不可能在其他奇异局势中,所以必然是非奇异局势.如果使(ak,bk)的两个分量同时减少,则由于其差不变,且不可能是其他奇异局势的差,因此也是非奇异局势.
//3、采用适当的方法,可以将非奇异局势变为奇异局势.
//假设面对的局势是(a,b),若 b = a,则同时从两堆中取走 a 个物体,就变为了奇异局势(0,0)；如果 a = ak ,b > bk,那么,取走 b - bk 个物体,即变为奇异局势；如果 a = ak , b < bk ,则同时从两堆中拿走 ak - ab - ak 个物体,变为奇异局势( ab - ak , ab - ak+ b - ak)；如果 a > ak ,b= ak + k,则从一堆中拿走多余的数量 a - ak 即可；如果 a < ak ,b= ak + k,分两种情况,第一种,a=aj (j < k),从第二堆里面拿走 b - bj 即可；第二种,a=bj (j < k),从第二堆里面拿走 b - aj 即可.
//从如上性质可知,两个人如果都采用正确操作,那么面对非奇异局势,先拿者必胜；反之,则后拿者取胜.
//那么任给一个局势(a,b),怎样判断它是不是奇异局势呢？我们有如下公式：

//ak =[k(1+√5)/2](下取整), bk= ak + k (k∈N)
//奇妙的是其中出现了有关黄金分割数的式子：(1+√5)/2 =1.618...,若两堆物品个数分别为 x,y(x<y)，则 k=y-x，再判断 x 是否等于[(y-x)*( √5+1)/2] 即可得知是否是奇异局势。
```

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<math.h>
#define eps 1e-10
using namespace std;
const double jin=(1.0+sqrt(5.0))/2;
int main()
{
   int n,p,k,f;
   while(scanf("%d%d",&n,&p)!=EOF)
   {
      if(n==p)
      {
         printf("1\n");
         continue;
      }
      if(n>p)
         swap(n,p);
      k=p-n;
      f=(int)floor(k*jin+eps);
      if(f==n)
         printf("1\n");
      else
         printf("0\n");
   }
   return 0;
        }
```

8 LCA 之离线算法（tarjan）

```
#include<stdio.h>
#include<algorithm>
#include<vector>
#define maxn 1000000
using namespace std;
int fa[maxn],lca[maxn];//lca[maxn]记录某条边的祖先值，而非节点
struct pi
{
   int to;
   int cost;
   int num;
}pp;
vector<pi>g[maxn];
vector<pi>gg[maxn];//输入待处理的值
int vis[maxn];
int find(int a)
{
   if(fa[a]==a)
      return a;
```

```
    return fa[a]=find(fa[a]);
}
void add(int a,int b,int cost)
{
    pp.to=b;
    pp.cost=cost;
    g[a].push_back(pp);
    pp.to=a;
    pp.cost=cost;
    g[b].push_back(pp);
}
void add1(int a,int b,int tot)
{
    pp.to=b;
    pp.num=tot;
    gg[a].push_back(pp);
    pp.to=a;
    pp.num=tot;
    gg[b].push_back(pp);
}
void LCA(int u)
{
    int k,i,p;
    k=gg[u].size();
    fa[u]=u;
    for(i=0;i<k;i++)
    {
        pp=gg[u][i];
        if(vis[pp.to])
        {
            lca[pp.num]=find(pp.to);//运用回溯，当正好回溯到一点时他的儿子节点的祖先为他，一
层一层的往上回溯。
        }
    }
    vis[u]=1;
    k=g[u].size();
    for(i=0;i<k;i++)
    {
        pp=g[u][i];
        if(!vis[pp.to])
        {
            p=pp.to;
            LCA(p);
            fa[p]=u;
        }
    }
    return ;
        }
```

9 卢卡斯定理模板（大数组合）

```
#include<stdio.h>
#include<string.h>
#include<math.h>
#include<iostream>
#include<algorithm>
using namespace std;
__int64 p;
__int64 jieji[200005];
__int64 ppow(__int64 n,__int64 m)
{

    __int64 s=1,k=n%p;
    while(m>0)
    {
        if(m&1)
        {
            s=(s*k)%p;
        }
        k=(k*k)%p;
        m>>=1;
    }
    return s;
}
__int64  zuhe(__int64 n,__int64 m)
{
    if(m==0)
        return 1;
    __int64 s=1;
    __int64 q,f;
    while(n>0&&m>0)
    {
        q=n%p;
        f=m%p;
        if(q<f)
            return 0;
        s=((s*jieji[q])%p*ppow((jieji[f]*jieji[q-f])%p,p-2))%p;
        n=n/p;
        m=m/p;
    }
    return s;
}
int main()
{
    __int64 n,m,t,k;
    int i;
    cin>>t;
    while(t--)
    {
        scanf("%I64d%I64d%I64d",&n,&m,&p);
```

```
        jieji[0]=1;
        jieji[1]=1;
        for(i=2;i<=p;i++)
            jieji[i]=(jieji[i-1]*i)%p;
        k=zuhe(n+m,n);
        printf("%I64d\n",k);
    }
    return 0;
}
```

10 LCA 在线算法

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#define max 400005
using namespace std;
int deap[max],vis[max],dis[max],kk[32][max];
int maxlog;
struct pi
{
    int to;
    int cost;
}pp;
vector<pi>g[max];
void init(int v,int p,int d)
{
    vis[v]=1;
    deap[v]=d;
    int i,k;
     kk[0][v]=p;
    for(i=1;i<maxlog;i++)
    {
        if(kk[i-1][v]<0)
            kk[i][v]=-1;
        else
        {
            kk[i][v]=kk[i-1][kk[i-1][v]];
        }
    }
    k=g[v].size();
    for(i=0;i<k;i++)
    {
        if(!vis[g[v][i].to])
        {
            dis[g[v][i].to]=dis[v]+g[v][i].cost;
            init(g[v][i].to,v,d+1);
        }
    }
```

```
    return ;
}
int find(int a,int b)
{
    if(deap[a]>deap[b])
        swap(a,b);
    int i,f;
    f=deap[b]-deap[a];
    for(i=0;i<maxlog;i++)
    {
        if((f>>i)&1)
            b=kk[i][b];
    }
    if(b==a)
        return a;
    for(i=maxlog-1;i>=0;i--)
    {
        if(kk[i][a]!=kk[i][b])
        {
            a=kk[i][a];
            b=kk[i][b];
        }
    }
    return kk[0][a];
}
int solve(int a,int b)
{
    int f;
    f=find(a,b);
    return dis[a]+dis[b]-2*dis[f];
          }
```

11 强连通分量（tarjan 算法）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#define maxn 1105
using namespace std;
int low[maxn],dnf[maxn],que[maxn],tear,head[maxn],c[maxn],mark[maxn];
bool map[maxn][maxn];
vector<int>g[maxn];
void init(int n){
    for(int i=1;i<=n;i++) g[i].clear();
}
int cal,tot;
void tarjin(int p){
    dnf[p]=low[p]=cal++;
    int i,k;
    k=g[p].size();
```

```
    que[tear++]=p;
    for(i=0;i<k;i++){
        int v=g[p][i];
        if(!dnf[v]){
            tarjin(v);
            low[p]=min(low[p],low[v]);//如果儿子就比较 low
        }
        else if(!mark[v]){
            low[p]=min(low[p],dnf[v]);//与栈里面的点进行比较，如果 mark[i]有值就代表已经出栈
了。
        }
    }
        if(low[p]==dnf[p]){
        while(tear>0){
            mark[que[tear-1]]=tot;
            if(que[tear-1]==p){
                tear--;
                break;
            }
            tear--;
        }
        tot++;
    }
}
void solve(int n){
    int i;
    tear=0;
    cal=1;
    tot=1;
    memset(dnf,0,sizeof(dnf));
    memset(c,0,sizeof(c));
    memset(mark,0,sizeof(mark));
    for(i=1;i<=n;i++){
        tear=0;
        if(!dnf[i]) tarjin(i);
    }
    for(i=1;i<=n;i++){
        c[mark[i]]++;
    }
 }
int main()
{
    int i,j,n,m,p,t;
    scanf("%d",&t);
    while(t--){
        scanf("%d%d",&n,&m);
        init(n);
        for(i=0;i<m;i++){
            scanf("%d%d",&p,&j);
            g[p].push_back(j);
        }
        solve(n);
    }
```

```
    }
```

12 欧拉错排公式

```
void init()
{
   s[0]=0; s[1]=0; s[2]=1;
   int i;
   for (i=3;i<=100;i++)
      s[i]=(i-1)*(s[i-1]+s[i-2])%mod;
        }
```

13 三分模板

```
//三分求极值法
//   二分法早就失去了他的意义了。不过还是可以用三分法来实现的，就是二分中再来二分。比如我们定义了 L 和 R，m = (L + R) / 2，mm = (mid + R) / 2; 如果 mid 靠近极值点，则 R = mm；否则就是 mm 靠近极值点，则 L = m;这样的话，极值还是可以求的
#include<stdio.h>
#include<iostream>
#include<algorithm>
using namespace std;
const double EPS = 1e-10;
double calc(double n)
{
   return;
}
double solve(double L, double R)
{
   double M, RM;
   while (L + EPS < R)
   {
      M = (L + R) / 2;
      RM = (M + R) / 2;
      if (calc(M) < calc(RM)) //计算最小值
         R = RM;
      else
         L = M;
   }
   return R;
}

//整数三分
int cal(){
   return 1;
}
void solve(){
```

```
    int l,r,m1,m2;
    l=1;
    r=10000;
    int ans;
    while(l<=r){
        m1=l+(r-l)/3;
        m2=r-(r-l)/3;
        int t=cal();
        int tt=cal();
        if(t<tt){//最大值
            l=m1+1;
            ans=tt;
        }
        else{
            r=m2-1;
            ans=t;
        }
    }
}
```

14 最小费用最大流模板

```
#include<stdio.h>
#include<string.h>
#include<algorithm>
#include<iostream>
#include<vector>
#include<queue>
#include<queue>
#define inf 0x3f
using namespace std;
struct pi
{
    int to;
    int cap;
    int cost;
    int rev;
};
vector<pi>g[205];
int dis[205];
int pre[205],pree[205];
int sink,source;
int vis[205];
int a[205];
char c[205];
void add(int a,int b,int cap,int cost)
{
    pi pp;
    pp.to=b;
    pp.cap=cap;
    pp.cost=cost;
```

```
    pp.rev=(int)g[b].size();
    g[a].push_back(pp);
    pp.to=a;
    pp.cap=0;
    pp.cost=-cost;
    pp.rev=(int)g[a].size()-1;
    g[b].push_back(pp);
    return ;
}
int spfa(void)
{
    int i,p,k;
    memset(dis,0x3f ,sizeof(dis));
    memset(vis,0,sizeof(vis));
    memset(pre,-1,sizeof(pre));
    memset(pree,-1,sizeof(pree));
    dis[source]=0;
    vis[source]=1;
    queue<int>q;
    pi pp;
    q.push(source);
    while(!q.empty())
    {
        p=q.front();
        k=(int)g[p].size();
        q.pop();
        vis[p]=0;
        if(p==sink)
            continue;
        for(i=0;i<k;i++)
        {
            pp=g[p][i];
            if(pp.cap>0&&dis[pp.to]>dis[p]+pp.cost)
            {
                dis[pp.to]=dis[p]+pp.cost;
                pre[pp.to]=p;
                pree[pp.to]=i;
                if(!vis[pp.to])
                {
                    q.push(pp.to);
                    vis[pp.to]=1;
                }
            }
        }
    }
    return pre[sink]!=-1;
}
int min(int a,int b)
{
    int p;
    p=a;
    if(b<a)
        p=b;
```

```
    return p;
}
int minflow(void)
{
    int i,f;
    int res=0;
    while(spfa())
    {
        f=inf;
        for(i=sink;i!=source;i=pre[i])
        {
            f=min(f,g[pre[i]][pree[i]].cap);
        }
        res+=f*dis[sink];
        for(i=sink;i!=source;i=pre[i])
        {
            pi &e=g[pre[i]][pree[i]];
            e.cap-=f;
            g[i][e.rev].cap+=f;
        }
    }
    return res;
}
```

15 中国剩余定理模板

```
#include<stdio.h>
#include<math.h>
#include<algorithm>
using namespace std;
int x, y,q;
int exgcd(int a,int b)
{
    int p;
    if(b==0)
    {
        x=1;
        y=0;
        return a;
    }
    q=exgcd(b,a%b);
    p=x;
    x=-y;
    y=-p-(a/b)*y;
    return q;
}
int main()
{
    int p,e,i,d,count,q,t,f,k,N=0;
    while(1)
    {
```

```
        N++;
        scanf("%d%d%d%d",&p,&e,&i,&d);
        if(p==-1&&e==-1&&i==-1&&d==-1)
            break;
        if(p==0)
            p=23;
        if(e==0)
            e=28;
        if(i==0)
            i=33;
        q=23*28*33;
        count=0;
        t=28*33;
        f=23*33;
        k=23*28;
        exgcd(t,23);
        if(x<0)
            x=x%23+23;
        count+=p*x*t;
        exgcd(f,28);
        if(x<0)
            x=x%28+28;
        count+=f*x*e;
        exgcd(k,33);
        if(x<0)
            x=x%33+33;
        count+=k*x*i-d;
        count=count%q;
        if(count<=0)
            count+=q;
        printf("Case %d: the next triple peak occurs in %d days.\n",N,count);
    }
    return 0;
        }
```

16 后缀数组（dc3）

```
/*
 一定要注意把几个串连起来的时候连接点千万别相同
 */
#include<cstdio>
#include<cstring>
#include<vector>
#include<algorithm>
#define maxn 200100
using namespace std;
int r[maxn];
int Rank[maxn],sa[maxn],height[maxn];
int wa[maxn],wb[maxn],wv[maxn],ws[maxn];
char a[maxn],b[maxn];
int cmp(int *r,int a,int b,int le)
```

```
{
        return r[a]==r[b]&&r[a+le]==r[b+le];
}
void da(int *r,int *sa,int n,int m)
{
        int i,j,p,*x=wa,*y=wb,*t;
        for ( i = 0; i < m; i++) ws[i]=0;
        for ( i = 0; i < n; i++) ws[ x[i] = r[i] ]++;
        for ( i = 1; i < m; i++) ws[i]+=ws[i-1];
        for ( i = n-1; i >= 0; i--) sa[--ws[x[i]]]=i;
        for ( j = 1,p=1; p <n ; j*=2,m=p)
        {
            for ( p = 0,i=n-j; i < n; i++) y[p++]=i;
            for ( i = 0; i < n; i++) if(sa[i]>=j) y[p++]=sa[i]-j;
            for ( i = 0; i < n; i++) wv[i]=x[y[i]];
            for ( i = 0; i < m; i++) ws[i]=0;
            for ( i = 0; i < n; i++) ws[wv[i]]++;
            for ( i = 1; i < m; i++) ws[i]+=ws[i-1];
            for ( i = n-1; i >= 0 ; i--) sa[--ws[wv[i]]]=y[i];
            for (t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
                  x[sa[i]] = cmp(y,sa[i-1],sa[i],j)?p-1:p++;
        }
        return ;
}
void calheight( int *r,int *sa,int n)
{
        int i,j,k=0;
        for ( i = 1; i <=n ; i++) Rank[sa[i]]=i;
        for(i=0;i<n;height[Rank[i++]]=k)
            for(k?k--:0,j=sa[Rank[i]-1];r[i+k]==r[j+k];k++);
        return ;
}
struct pi
{
   int min;
}pp[4*maxn];
void build(int le,int ri,int tot)
{
   if(le==ri)
   {
      pp[tot].min=height[le];
      return ;
   }
   int mid;
   mid=(le+ri)/2;
   build(le,mid,2*tot);
   build(mid+1,ri,2*tot+1);
   pp[tot].min=min(pp[2*tot].min,pp[2*tot+1].min);
   return ;
}
int query(int le,int ri,int tot,int ll,int rr)
{
   int p,q;
```

```
    p=100000000;
    q=100000000;
    if(le>ri)
        return 0;
    if(le<=ll&&ri>=rr)
    {
        return pp[tot].min;
    }
    int mid;
    mid=(ll+rr)/2;
    if(le<=mid)
    {
        p=query(le,ri,2*tot,ll,mid);
    }
    if(ri>mid)
    {
        q=query(le,ri,2*tot+1,mid+1,rr);
    }
    if(p==100000000&&q==100000000)
        return 0;
    return min(p,q);
}
int main()
{
    int i,n,k,f,le,ri,mid,p;
    while(scanf("%d%d",&n,&k)!=EOF)
    {
        for(i=0;i<n;i++)
        {
            scanf("%d",&f);
            r[i]=f+1;/*
                    一定要注意把几个串连起来的时候连接点千万别相同
                    */
        }
            r[n]=0;//为了使 rank 从 1 开始，防止 height[rank[i]-1]越界。
        da(r,sa,n+1,20002);
        calheight(r,sa,n);
        for(i=2;i<=n;i++)
            height[i];      //height[i]数组的定义 sa[i-1]和 sa[i]的最长公共前缀，而 rank 从 1 到 n,所以
height 数组从 2 到 n。
    }
    return 0;
          }
```

17 数据输入加速

```
template <class T>
inline void scan_d(T &ret) {
    char c; ret=0;
```

```
    while((c=getchar())<'0'||c>'9');
    while(c>='0'&&c<='9') ret=ret*10+(c-'0'),c=getchar();
}
```

18 平面的划分

```
/*已知经过同一个的 n 个平面，任意三个平面不经过同一条直线，若这 n 个平面将空间分成 f（n）个部分，则 f（3）= f(n)=
(1)、f(3)=8，见上图。
(2)、当 n>3 时，每增加一个面，这面就要与前面 n-1 个面都相交，因为过同一点，两平面如果有一个公共点就有一条公共直线，这样就会把前面平面划分的空间一分为二，f(n)-f(n-1)=2(n-1),然后累加得 f(n)=n^2-n+2*/
```

19 高斯消元（解方程）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
using namespace  std;
double b[205][205],x[205],a[25][25];
void guass(int equ,int val){
    int i,j,n,p ,k;
    double s2,q;
    p=0;
    for(i=0;i<equ&&p<val;i++,p++){
        n=i;
        for(j=i+1;j<equ;j++){
            if(fabs(b[j][p])>fabs(b[n][p])) n=j;
        }
        if(n!=i){
            for(j=p;j<val;j++){
                swap(b[i][j],b[n][j]);
            }
            swap(x[i],x[n]);
        }
        if(fabs(b[i][p])<1e-6){
            i--;
            continue;
        }
        for(j=i+1;j<equ;j++){
            if(fabs(b[j][p])<1e-9) continue;
            q=b[j][p]/b[i][p];
            for(k=p;k<val;k++){
                b[j][k]=q*b[i][k]-b[j][k];
            }
            x[j]=q*x[i]-x[j];
```

```
        }
    }
    for(i=equ-1;i>=0;i--){
        s2=x[i];
        for(j=val-1;j>i;j--) s2-=b[i][j]*x[j];
        x[i]=s2/b[i][i];
    }
}
int aabs(int p,int f){
    if(f>p) return f-p;
    return p-f;
}
int main()
{
    int i,j,n,m,p,k,f,v,N=0;
    while(1){
        scanf("%d%d%d",&m,&n,&p);
        if(m==0&&n==0&&p==0) break;
        if(N!=0) printf("\n");
        N++;
        for(i=0;i<n;i++){
            for(j=0;j<m;j++)
                scanf("%lf",&a[i][j]);
        }
        memset(b,0,sizeof(b));
        for(i=0;i<n;i++){
            for(j=0;j<m;j++){
                v=0;
                for(f=0;f<n;f++){
                    for(k=0;k<m;k++){
                        if(aabs(i,f)+aabs(j,k)<=p){
                            b[i*m+j][f*m+k]=1;
                            v++;
                        }
                    }
                }
                x[i*m+j]=v*a[i][j];
            }
        }
        guass(n*m,m*n);
        for(i=0;i<n;i++){
            for(j=0;j<m;j++){
                printf("%8.2lf",x[i*m+j]);
            }
            printf("\n");
        }
    }
        }
```

20 高斯消元 开关类问题（异或方程）

```
#include<cstdio>
#include<algorithm>
#include<cstring>
#define LL __int64
using namespace std;
int b[55][55],x[55],c[55][55];
int flag;//判断是否无解。
int guass(int equ,int val){
    int i,j,n,m,p,k,q;
    p=0;
    for(i=0;i<equ&&p<val;i++,p++){
        k=b[i][p];
        n=i;
        for(j=i+1;j<equ;j++){
            if(b[j][p]>k) {k=b[j][p];
                n=j;
            }
        }
        if(n!=i){
            for(j=p;j<val;j++){
                swap(b[i][j],b[n][j]);
            }
            swap(x[i],x[n]);
        }
        if(b[i][p]==0) {
            i--;
            continue;
        }
        for(j=i+1;j<equ;j++){
            if(b[j][p]==0) continue;
            for(m=p;m<val;m++){
                b[j][m]=b[i][m]^b[j][m];
            }
            x[j]=x[i]^x[j];
        }
    }
    q=i;
    m=0;
    for(i=0;i<equ;i++){
        p=0;
        for(j=0;j<val;j++){
            if(b[i][j]!=0){
                p=1;
                break;
            }
        }
        if(p==0&&x[i]!=0){
            flag=1;
            return 0;
```

```
        }
        if(p) m++;
    }
    return val-q;//返回不确定变元个数
          }
```

21 四边形优化模板

```
/*
对于 dp[i][j]=dp[i][k]+d[k][j]+w[i][j]的 dp 方程，如果满足 w[i][j]+w[i'][j']<=w[i'][j]+w[i][j']
(i'<=i<=j<=j')则 w[i][j]是凸的，也就是说，对于 dp[i][j]的决策 s[i][j],必然满足不等式 s[i][j-
1]<=s[i][j]<=s[i+1][j].所以求决策时只需要循环从 s[i][j-1]到 s[i+1][j]就行，然后求 s[i][j],注意循
环长度。
 区间 dp 一般用四边形优化
 */
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
int dp[1005][1005];
int s[1005][1005];
struct pi{
    int x;
    int y;
}pp[1005];
int main()
{
    int i,j,n,m,k;
    while(scanf("%d",&n)!=EOF){
        memset(dp,0x3f,sizeof(dp));
        memset(s,0,sizeof(s));
        for(i=1;i<=n;i++) {scanf("%d%d",&pp[i].x,&pp[i].y);
            s[i][i]=i;
            dp[i][i]=0;
        }
        for(i=1;i<=n-1;i++){//四边形优化一定要优化长度
            for(j=1;j+i<=n;j++){
                for(k=s[j][j+i-1];k<=s[j+1][j+i];k++){
                    m=pp[k].y-pp[j+i].y+pp[k+1].x-pp[j].x;
                    if(dp[j][j+i]>dp[j][k]+dp[k+1][j+i]+m){
                        dp[j][j+i]=dp[j][k]+dp[k+1][j+i]+m;
                        s[j][j+i]=k;
                    }
                }
            }
        }
        printf("%d\n",dp[1][n]);
    }
}
```

22 斜率 dp 模板

/*
我们假设 k<j<i。如果在 j 的时候决策要比在 k 的时候决策好，那么也是就是 dp[j]+M+(sum[i]-sum[j])^2<dp[k]+M+(sum[i]-sum[k])^2。(因为是最小花费嘛，所以优就是小于)

两边移项一下，得到：(dp[j]+num[j]^2-(dp[k]+num[k]^2))/(2*(num[j]-num[k]))<sum[i]。我们把 dp[j]-num[j]^2 看做是 yj，把 2*num[j]看成是 xj。

那么不就是 yj-yk/xj-xk<sum[i]么？ 左边是不是斜率的表示？

那么 yj-yk/xj-xk<sum[i]说明了什么呢？ 我们前面是不是假设 j 的决策比 k 的决策要好才得到这个表示的？ 如果是的话，那么就说明 g[j,k]=yj-jk/xj-xk<sum[i]代表这 j 的决策比 k 的决策要更优。

关键的来了：现在从左到右，还是设 k<j<i，如果 g[i,j]<g[j,k]，那么 j 点便永远不可能成为最优解，可以直接将它踢出我们的最优解集。为什么呢？

我们假设 g[i,j]<sum[i]，那么就是说 i 点要比 j 点优，排除 j 点。

如果 g[i,j]>=sum[i]，那么 j 点此时是比 i 点要更优，但是同时 g[j,k]>g[i,j]>sum[i]。这说明还有 k 点会比 j 点更优，同样排除 j 点。

排除多余的点，这便是一种优化！

接下来看看如何找最优解。

设 k<j<i。

由于我们排除了 g[i,j]<=g[j,k]的情况，所以整个有效点集呈现一种上凸性质，即 k j 的斜率要大于 j i 的斜率。

这样，从左到右，斜率之间就是单调递减的了。当我们的最优解取得在 j 点的时候，那么 k 点不可能再取得比 j 点更优的解了，于是 k 点也可以排除。换句话说，j 点之前的点全部不可能再比 j 点更优了，可以全部从解集中排除。

于是对于这题我们对于斜率优化做法可以总结如下：

1，用一个单调队列来维护解集。

2，假设队列中从头到尾已经有元素 a b c。那么当 d 要入队的时候，我们维护队列的上凸性质，即如果 g[d,c]<g[c,b]，那么就将 c 点删除。直到找到 g[d,x]>=g[x,y]为止，并将 d 点加入在该位置中。

3，求解时候，从队头开始，如果已有元素 a b c，当 i 点要求解时，如果 g[b,a]<sum[i]，那么说明 b 点比 a 点更优，a 点可以排除，于是 a 出队。最后 dp[i]=getDp(q[head])。

*/

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
using namespace std;
typedef int LL;
LL a[500005];
LL dp[500005];
int q[500005];
LL s[500005];
LL get1(int n,int m){
   return dp[n]-dp[m]+s[n]*s[n]-s[m]*s[m];
}
LL get2(int n,int m){
   return s[n]-s[m];
}
int main()
{
   int i,n,m,tear,rear;
   while(scanf("%d%d",&n,&m)!=EOF){
      s[0]=0;
      dp[0]=0;
      for(i=1;i<=n;i++){
         scanf("%d",&a[i]);
         s[i]=s[i-1]+a[i];
      }
      rear=0;
      tear=0;
      q[tear++]=0;
      for(i=1;i<=n;i++){
         while(rear+1<tear){
            if(get1(q[rear+1],q[rear])<=2*s[i]*get2(q[rear+1],q[rear])) rear++;
            else break;
         }
         dp[i]=dp[q[rear]]+(s[i]-s[q[rear]])*(s[i]-s[q[rear]])+m;
         while(tear>rear+1){
            if(get1(i,q[tear-1])*get2(q[tear-1],q[tear-2])<=get1(q[tear-1],q[tear-2])*get2(i,q[tear-1]))
tear--;
            else break;
         }
         q[tear++]=i;
      }
      printf("%d\n",dp[n]);
   }
}
```

23 分解大数质因数模板

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
#include<iostream>
#include<algorithm>
#define LL long long
using namespace std;
//************************************************************************
// Miller_Rabin 算法进行素数测试
//速度快，而且可以判断 <2^63 的数
//************************************************************************
const int S=20;//随机算法判定次数，S 越大，判错概率越小
//计算 (a*b)%c.  a,b 都是 long long 的数，直接相乘可能溢出的
//  a,b,c <2^63
LL mult_mod(LL a,LL b,LL c)
{
    a%=c;
    b%=c;
    LL ret=0;
    while(b)
    {
        if(b&1){ret+=a;ret%=c;}
        a<<=1;
        if(a>=c)a%=c;
        b>>=1;
    }
    return ret;
}
//计算  x^n %c
LL pow_mod(LL x,LL n,LL mod)//x^n%c
{
    if(n==1)return x%mod;
    x%=mod;
    LL tmp=x;
    LL ret=1;
    while(n)
    {
        if(n&1) ret=mult_mod(ret,tmp,mod);
        tmp=mult_mod(tmp,tmp,mod);
        n>>=1;
    }
    return ret;
}
//以 a 为基,n-1=x*2^t      a^(n-1)=1(mod n)  验证 n 是不是合数
//一定是合数返回 true,不一定返回 false
bool check(LL a,LL n,LL x,LL t)
```

```
{
    LL ret=pow_mod(a,x,n);
    LL last=ret;
    for(int i=1;i<=t;i++)
    {
        ret=mult_mod(ret,ret,n);
        if(ret==1&&last!=1&&last!=n-1) return true;//合数
        last=ret;
    }
    if(ret!=1) return true;
    return false;
}

// Miller_Rabin()算法素数判定
//是素数返回 true.(可能是伪素数，但概率极小)
//合数返回 false;

bool Miller_Rabin(LL n)
{
    if(n<2)return false;
    if(n==2)return true;
    if((n&1)==0) return false;//偶数
    LL x=n-1;
    LL t=0;
    while((x&1)==0){x>>=1;t++;}
    for(int i=0;i<S;i++)
    {
        LL a=rand()%(n-1)+1;//rand()需要 stdlib.h 头文件
        if(check(a,n,x,t))
            return false;//合数
    }
    return true;
}
//***************************************************
//pollard_rho 算法进行质因数分解
//***************************************************
LL factor[100];//质因数分解结果（刚返回时是无序的）
int tol;//质因数的个数。数组小标从 0 开始

LL gcd(LL a,LL b)
{
    if(a==0)return 1;//???????
    if(a<0) return gcd(-a,b);
    while(b)
    {
        LL t=a%b;
        a=b;
        b=t;
    }
    return a;
}
LL Pollard_rho(LL x,LL c)
```

```
{
    LL i=1,k=2;
    LL x0=rand()%x;
    LL y=x0;
    while(1)
    {
        i++;
        x0=(mult_mod(x0,x0,x)+c)%x;
        LL d=gcd(y-x0,x);
        if(d!=1&&d!=x) return d;
        if(y==x0) return x;
        if(i==k){y=x0;k+=k;}
    }
}
//对 n 进行素因子分解
void findfac(LL n)
{
    if(Miller_Rabin(n))//素数
    {
        factor[tol++]=n;
        return;
    }
    LL p=n;
    while(p>=n)p=Pollard_rho(p,rand()%(n-1)+1);
    findfac(p);
    findfac(n/p);
}
int main()
{
    long long n;
    int t;
    cin>>t;
    while(t--){
        cin>>n;
        tol=0;
    findfac(n);
    if(tol==1){
        printf("Prime\n");
        continue;
    }
    sort(factor,factor+tol);
        cout<<factor[0]<<endl;
    }
}
```

24 强连通分量（桥）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
```

```
#include<vector>
#define maxn 1005
using namespace std;
int vis[maxn],low[maxn],dnf[maxn],yes[maxn],fa[maxn];
vector<int > g[maxn];
int map[1005][1005];
void init(int n){
   for(int i=1;i<=n;i++) g[i].clear();
}
int zi[maxn],pp;
int cal;
void tarjin(int p,int pa,int n){
   int k,i;
   low[p]=dnf[p]=cal++;
   k=g[p].size();
   for(i=0;i<k;i++){
      int v=g[p][i];
      if(!dnf[v]){
         tarjin(v,p,n);
         low[p]=min(low[p],low[v]);
         if(low[v]>dnf[p]){//判断是否是桥
            pp++;//桥的个数
         }
      }
      else if(v!=pa){//一定要注意求桥时不能遍历父亲节点，因为判断条件是大于，否则变成了
环。
         low[p]=min(low[p],dnf[v]);
      }
   }
}
void solve(int n){
   int i,m;
   cal=1;
   for(i=1;i<=n;i++){
      if(!dnf[i])
         tarjin(i,-1,i);
   }
}
```

25 强连通分量（割点）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
#define maxn 1005
using namespace std;
int vis[maxn],low[maxn],dnf[maxn],yes[maxn],fa[maxn];
vector<int > g[maxn];
```

```
int map[1005][1005];
void init(int n){
   for(int i=1;i<=n;i++) g[i].clear();
}
int zi[maxn],pp;
int cal;
void tarjin(int p,int pa,int n){
   int k,i,q=0;
   low[p]=dnf[p]=cal++;
   k=g[p].size();
   for(i=0;i<k;i++){
      int v=g[p][i];
      if(!dnf[v]){
         tarjin(v,p,n);
         low[p]=min(low[p],low[v]);
         if(low[v]>=low[p]){
            yes[p]=1;//表示 p 是割点，可以包含 p 的环
         }
         q++;
      }
      else if(v!=pa){
         low[p]=min(low[p],dnf[v]);
      }
   }
   if(p==n) zi[p]=q;//根节点不加 1.
   else zi[p]=q+1;//有多个分子图，也就是连着几块联通块。
}
void solve(int n){
   int i,m;
   memset(zi,0,sizeof(zi));
   cal=1;
   for(i=1;i<=n;i++){
   if(!dnf[i])
   tarjin(i,-1,i);
   }
}
```

26 网络流判圈

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define inf 100000000
#define LL long long
#define maxn 820
using namespace std;
struct pi
```

```
{
   int to;
   int cost;
   int rev;
}pp;
vector<pi >g[maxn];
queue<int>q;
int c[450][450];
int s,t;
int line[maxn],leve[maxn];
bool vis[452][452];
int min(int a,int b)
{
   int p;
   p=a;
   if(b<a)
      p=b;
   return p;
}
void add(int a,int b,int cos)
{
   pp.to=b;
   pp.cost=cos;
   pp.rev=(int)g[b].size();
   g[a].push_back(pp);
   pp.to=a;
   pp.cost=0;
   pp.rev=(int)g[a].size()-1;
   g[b].push_back(pp);
   return ;
}
void bfs(void)
{
   q.push(s);
   int p,f,i;
   while(!q.empty())
   {
      p=q.front();
      q.pop();
      f=(int)g[p].size();
      for(i=0;i<f;i++)
      {
         pi &e=g[p][i];
         if(e.cost>0&&line[e.to]<0)
         {
            line[e.to]=line[p]+1;
            q.push(e.to);
         }
      }
   }
   return ;
}
int dfs(int v,int f)
```

```
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
long long  dinic()
{
    int f;
    long long flow=0;
    while(1)
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
        {
            flow+=f;
        }
    }
}
int main()
{
    int i,j,n,m,p,k,f,f1,f2;
    LL flow,s1,s2;
    while(scanf("%d%d%d",&n,&m,&k)!=EOF){
        for(i=0;i<=n+m+1;i++) g[i].clear();
        s1=0;
        s2=0;
        f=0;
        for(i=1;i<=n;i++){
            scanf("%d",&p);
```

```
            add(0,i,p);
            if(p>k*m||p<0){
                f=1;
            }
            s1+=p;
        }
        for(i=1;i<=m;i++){
            scanf("%d",&p);
            if(p>k*n||p<0){
                f=1;
            }
            add(n+i,n+m+1,p);
            s2+=p;
        }
        if(f){
            printf("Impossible\n");
            continue;
        }
        if(s1!=s2){
            printf("Impossible\n");
            continue;
        }
        for(i=1;i<=n;i++){
            for(j=1;j<=m;j++){
                add(i,n+j,k);
            }
        }
        s=0;
        t=n+m+1;
        flow=dinic();
        if(flow!=s1){
            printf("Impossible\n");
            continue;
        }
        for(i=1;i<=n;i++){
            p=(int)g[i].size();
            for(j=0;j<p;j++){
                pp=g[i][j];
                if(pp.to>n&&pp.to<=n+m)
                    c[i][pp.to-n]=pp.cost;//用残量网络判断是否含环
            }
        }
        memset(vis,0,sizeof(vis));//如果残量网络有两行全未达到极值说明图中流量不唯一，也就是
说明图中含环
        int flag=0;
        for(i=1;i<=n;i++){
            for(j=1;j<=m;j++){
                for(p=j+1;p<=m;p++){
                    f1=0,f2=0;
                    if(c[i][j]!=0&&c[i][p]!=k){
                        if(vis[p][j]){
                            flag=1;
                            break;
```

```
                }
                f1=1;
            }
            if(flag) break;
            if(c[i][j]!=k&&c[i][p]!=0){
                if(vis[j][p]){
                    flag=1;
                    break;
                }
                f2=1;
            }
            if(f1) vis[j][p]=1;
            if(f2) vis[p][j]=1;
        }
        if(flag) break;
    }
    if(flag) break;
}
if(flag){
    printf("Not Unique\n");
}
else{
    printf("Unique\n");
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            if(j==1){
                printf("%d",k-c[i][j]);
            }
            else{
                printf(" %d",k-c[i][j]);
            }
        }
        printf("\n");
    }
}
}
}
```

27、无源无汇上下限网络流

```
/*一种方法是 添加附加源汇 S,T  对于某点 u, 设 M(u)=sigma(B[i,u])-sigma(B[u,j]) ，
 则根据流量平衡条件有 M(u)同时等于 sigma(g[u,j])-sigma(g[i,u])
 若 M(u)<0，即 sigma(g[u,j]) < sigma(g[i,u]) 进入 u 的流量比从 u 出去的多，
 所以 u -> T 连容量为  -(sigma(B[i,u])-sigma(B[u,j]) ) 的边
 同理. M(u)>0 时，即 S->u 连容量为 sigma(B[i,u])-sigma(B[u,j])  的边.
 然后再 对于任意边(i,u)/(u,j) 连一条 C[u,v]-B[u,v]的边.
 这样 只需对新的网络求一遍最大流即可. 若出附加源点的边都满流即是存在可行流，反之不
然.
 满流的必要条件是显然的. 不满流不能保证加上 B[,]后流量平衡. 前面都白费了.
*/
```

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define LL int
#define inf 100000000
#define maxn 1500
using namespace std;
struct pi
{
   int to;
   int cost;
   int rev;
   int id;
}pp;
vector<pi >g[maxn];
queue<int>q;
int s,t;
int line[maxn],leve[maxn];
int min(int a,int b)
{
   int p;
   p=a;
   if(b<a)
      p=b;
   return p;
}
void add(int a,int b,int cos,int id)
{
   pp.to=b;
   pp.cost=cos;
   pp.id=id;
   pp.rev=(int)g[b].size();
   g[a].push_back(pp);
   pp.to=a;
   pp.cost=0;
   pp.id=-1;
   pp.rev=(int)g[a].size()-1;
   g[b].push_back(pp);
   return ;
}
void bfs(void)
{
   q.push(s);
   int p,f,i;
   while(!q.empty())
   {
      p=q.front();
      q.pop();
      f=(int)g[p].size();
      for(i=0;i<f;i++)
```

```
        {
            pi &e=g[p][i];
            if(e.cost>0&&line[e.to]<0)
            {
                line[e.to]=line[p]+1;
                q.push(e.to);
            }
        }
    }
    return ;
}
int dfs(int v,int f)
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
LL  dinic()
{
    int f;
    LL flow=0;
    while(1)
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
        {
            flow+=f;
        }
    }
```

```
}
int a[20005];
struct ppi{
    int x;
    int y;
    int co1;
    int co2;
}pp1[100005];
int main()
{
    int i,j,n,m,p,k,f,q;
    s=0;
    while(scanf("%d%d",&n,&m)!=EOF){
        t=n+1;
        for(i=0;i<=n+1;i++) g[i].clear();
        for(i=0;i<m;i++){
            scanf("%d%d%d%d",&pp1[i].x,&pp1[i].y,&pp1[i].co1,&pp1[i].co2);
            add(pp1[i].x,pp1[i].y,pp1[i].co2-pp1[i].co1,i);
            a[pp1[i].x]+=pp1[i].co1;
            a[pp1[i].y]-=pp1[i].co1;
        }
        for(i=1;i<=n;i++){
            if(a[i]>0) add(i,t,a[i],0);
            else add(s,i,-a[i],0);
        }//必要流量出流量比较多的与 T 连边，入的多的与 S 连边，跑一遍最大流。
        p=dinic();
        memset(a,0,sizeof(a));
        k=0;
        f=g[0].size();
        for(i=0;i<f;i++){//如果与源点连的边都满流则为可行流，否则不可行。
            if(g[0][i].cost!=0){
                k=1;
                break;
            }
        }
        if(k) printf("NO\n");
        else{
            for(i=1;i<=n;i++){
                k=g[i].size();
                for(j=0;j<k;j++){
                    if(g[i][j].id!=-1&&g[i][j].to!=s&&g[i][j].to!=t){
                        a[g[i][j].id]=pp1[g[i][j].id].co2-g[i][j].cost;//基础流量加上必要流量，注意网络里这
条边的流量就是除基础流量外的跑的流量。
                    }
                }
            }
            printf("YES\n");
            for(i=0;i<m;i++){
                printf("%d\n",a[i]);
            }
        }
    }
}
```

28 匈牙利算法

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#include<vector>
#define maxn 1500
using namespace std;
int match[maxn];
vector<int >g[maxn];
bool use[maxn];
bool dfs(int u){
   int p,i,j,n,k;
   use[u]=1;
   n=g[u].size();
   for(i=0;i<n;i++){
      k=g[u][i];
      p=match[k];
      if(p<0||!use[p]&&dfs(p)){
         match[u]=g[u][i];
         match[g[u][i]]=u;
         return 1;
      }
   }
   return 0;
}
int hungry(int n){
   int m=0,i,j;
   memset(match,-1,sizeof(match));
   for(i=0;i<n;i++){
      if(match[i]<0){
         memset(use,0,sizeof(use));
         if(dfs(i)){
            m++;
         }
      }
   }
   return m;
}
int main()
{
   int i,j,n,m,p,k;
}
```

28 有源汇的上下界最小流

```
#include<stdio.h>
#include<iostream>
```

```
#include<algorithm>
#include<vector>
#include<string.h>
#include<queue>
#define LL int
#define inf 100000000
#define maxn 150
using namespace std;
struct pi
{
   int to;
   int cost;
   int rev;
   int id;
}pp;
vector<pi >g[maxn];
queue<int>q;
int s,t;
int line[maxn],leve[maxn];
int min(int a,int b)
{
   int p;
   p=a;
   if(b<a)
      p=b;
   return p;
}
void add(int a,int b,int cos,int id)
{
   pp.to=b;
   pp.cost=cos;
   pp.rev=(int)g[b].size();
   pp.id=id;
   g[a].push_back(pp);
   pp.to=a;
   pp.cost=0;
   pp.rev=(int)g[a].size()-1;
   pp.id=-1;
   g[b].push_back(pp);
   return ;
}
void bfs(void)
{
   q.push(s);
   int p,f,i;
   while(!q.empty())
   {
      p=q.front();
      q.pop();
      f=(int)g[p].size();
      for(i=0;i<f;i++)
      {
         pi &e=g[p][i];
```

```
            if(e.cost>0&&line[e.to]<0)
            {
                line[e.to]=line[p]+1;
                q.push(e.to);
            }
        }
    }
    return ;
}
int dfs(int v,int f)
{
    int p;
    if(v==t)
        return f;
    for(int &i=leve[v];i<g[v].size();i++)
    {
        pi &e=g[v][i];
        if(line[v]<line[e.to]&&e.cost>0)
        {
            p=dfs(e.to,min(f,e.cost));
            if(p>0)
            {
                e.cost-=p;
                g[e.to][e.rev].cost+=p;
                return p;
            }
        }
    }
    return 0;
}
LL  dinic()
{
    int f;
    LL flow=0;
    while(1)
    {
        memset(line,-1,sizeof(line));
        memset(leve,0,sizeof(leve));
        line[s]=0;
        bfs();
        if(line[t]<0)
            return flow;
        f=dfs(s,inf);
        if(f==0)
            continue;
        flow+=f;
        while((f=dfs(s,inf))>0)
        {
            flow+=f;
        }
    }
}
struct ppi{
```

```
    int from;
    int to;
    int a;
    int b;
}pp1[100005];
char c[1005];
int a[105];
int b[100005];
int main()
{
    int i,j,n,m,p,k,N=0,le,ri,mid;
    while(scanf("%d%d",&n,&m)!=EOF){
        memset(a,0,sizeof(a));
        for(i=0;i<=n+1;i++) g[i].clear();
        for(i=0;i<m;i++){
            scanf("%d%d%d",&pp1[i].from,&pp1[i].to,&pp1[i].b);
            scanf("%d",&p);
            if(p==0){
                pp1[i].a=0;
            }
            else pp1[i].a=pp1[i].b;
            a[pp1[i].from]+=pp1[i].a;
            a[pp1[i].to]-=pp1[i].a;
        }
        s=0;
        t=n+1;
        le=0;
        ri=1000000000;
        while(le<=ri){//将 t-》s 连一条有上界的边，二分上界。
            mid=(le+ri)/2;
            for(i=0;i<=t;i++) g[i].clear();
            for(i=0;i<m;i++){
                add(pp1[i].from,pp1[i].to,pp1[i].b-pp1[i].a,i);
            }
            for(i=1;i<=n;i++){
                if(a[i]==0) continue;
                if(a[i]>0) add(i,t,a[i],-1);
                else add(s,i,-a[i],-1);
            }
            add(n,1,mid,-1);
            dinic();
            p=0;
            k=g[s].size();
            for(i=0;i<k;i++){
                if(g[s][i].cost!=0){
                    p=1;
                    break;
                }
            }
            if(p) le=mid+1;
            else ri=mid-1;
        }
        if(le>=1000000000){
```

```
            printf("Impossible\n");
        }
        else{
            printf("%d\n",le);
            for(i=0;i<=t;i++) g[i].clear();
            for(i=0;i<m;i++){
                add(pp1[i].from,pp1[i].to,pp1[i].b-pp1[i].a,i);
            }
            for(i=1;i<=n;i++){
                if(a[i]==0) continue;
                if(a[i]>0) add(i,t,a[i],-1);
                else add(s,i,-a[i],-1);
            }
            add(n,1,le,-1);
            dinic();
            for(i=1;i<=n;i++){
                k=g[i].size();
                for(j=0;j<k;j++){
                    if(g[i][j].id!=-1){
                        b[g[i][j].id]=pp1[g[i][j].id].b-g[i][j].cost;
                    }
                }
            }
            for(i=0;i<m;i++){
                if(i==0) printf("%d",b[i]);
                else printf(" %d",b[i]);
            }
            printf("\n");
        }
    }
}
```

29 上下限网络流

```
/*
1、无源无汇可行流。

由流量守恒 sigma(g[u,i])+sigma(b[u,i])=sigma(g[i,v])+sigma(b[i,v]).

其中 b[u,i]是流量下界，g[u,i]<=c[u,i]-b[u,i]。c[u,i]是流量上界，最后得到

sigma(g[u,i])=sigma(g[i,v])+p。如果 p 大于 0，就添一条 i 到 t 流量为 p 的边，其中，t 是超级源点。

如果小于 0，就添一条 s 到 i 的流量为-p 的边。跑一遍 dinic，如果与 s 相连的边有一条边不满流就不是可行流。

跑完最大流之后每天边的流量等于流量上界减去残余流量。

2.有源汇最大最小流。
```

一、最大流：

如果对于网路中流量为 a，则连一条 t->s 流量下界为 a 的边变成无源无汇网络，则这个网络一定存在可行流。

所以可以二分 a，也就是说，二分 t-》s 流量下界，判断是否是可行流即可。每条边流量即为无源无汇网络每条边流量；

二、最小流：同理最大流，二分 t->s 的流量上界，然后用可行流判断即可。
*/

30 枚举一个数二进制表示下的子集

```
#include<cstdio>
int main()
{
   int i,j,s;
   for(i=s;i;i=(i-1)&s);//枚举 s 子集
}
```

31 上下限网络流大攻略

1、无源无汇可行流。
由流量守恒 sigma(g[u,i])+sigma(b[u,i])=sigma(g[i,v])+sigma(b[i,v]).
其中 b[u,i]是流量下界，g[u,i]<=c[u,i]-b[u,i]。c[u,i]是流量上界，最后得到 sigma(g[u,i])=sigma(g[i,v])+p。如果 p 大于 0，就添一条 i 到 t 流量为 p 的边，其中，t 是超级源点。
如果小于 0，就添一条 s 到 i 的流量为-p 的边。跑一遍 dinic，如果与 s 相连的边有一条边不满流就不是可行流。
跑完最大流之后每天边的流量等于流量上界减去残余流量。
2.有源汇最大最小流。
一、最大流：
如果对于网路中流量为 a，则连一条 t->s 流量下界为 a 的边变成无源无汇网络，则这个网络一定存在可行流。
所以可以二分 a，也就是说，二分 t-》s 流量下界，判断是否是可行流即可。每条边流量即为无源无汇网络每条边流量；
二、最小流：同理最大流，二分 t->s 的流量上界，然后用可行流判断即可。

32 AC 自动机数组实现

```
#include<cstdio>
```

```
#include<cstring>
#include<string>
#include<algorithm>
#include<queue>
#include<iostream>
#define sig 26
#define maxn 260050
using namespace std;
int ch[maxn][26],tot;
int fail[maxn];
char c[10005][52];
char d[maxn];
int val[maxn];
queue<int>q;
void get_trie(int n){
    tot=0;
    int i,j,m,p;
    memset(val,0,sizeof(val));
    memset(fail,0,sizeof(fail));
    memset(ch,0,sizeof(ch));
    for(i=0;i<n;i++){
        m=(int)strlen(c[i]);
        p=0;
        for(j=0;j<m;j++){
            if(ch[p][c[i][j]-'a']) p=ch[p][c[i][j]-'a'];
            else{
                ch[p][c[i][j]-'a']=++tot;
                p=tot;
            }
        }
        val[p]++;
    }
}
void get_fail()
{
    queue<int>Q;
    int root=0;
    fail[root] = root;
    for(int i = 0;i < 26;i++)
        if(ch[root][i] == 0)
            ch[root][i] = root;
        else
        {
            fail[ch[root][i]] = root;
            Q.push(ch[root][i]);
        }
    while( !Q.empty() )
    {
        int now = Q.front();
        Q.pop();
        for(int i = 0;i < 26;i++)
            if(ch[now][i] == 0)
                ch[now][i] = ch[fail[now]][i];
```

```
            else
            {
                fail[ch[now][i]]=ch[fail[now]][i];
                Q.push(ch[now][i]);
            }
    }
}
int find(char *T){
    int i,j,n,cnt,temp;
    cnt=0;
    n=(int)strlen(T);
    j=0;
    for(i=0;i<n;i++){
        while(j&&!ch[j][T[i]-'a']) j=fail[j];
        j=ch[j][T[i]-'a'];
        temp=j;
        while(temp&&val[temp]!=-1){
            cnt+=val[temp];
            val[temp]=-1;
            temp=fail[temp];
        }
    }
    return cnt;
}
int main()
{
    int i,n,t;
    cin>>t;
    while(t--){
        scanf("%d",&n);
        for(i=0;i<n;i++) scanf("%s",c[i]);
        get_trie(n);
        get_fail();
        scanf("%s",d);
        printf("%d\n",find(d));
            }
}
```

33 树链剖分模板（改边）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#include<vector>
#define maxn 100005
using namespace std;
int top[maxn],fa[maxn],son[maxn],size[maxn],id[maxn],num,dep[maxn],tot;
struct pppi{
    int to;
    int next;
```

```
}pp2[2*maxn];
int head[maxn];
vector<int >g[maxn];
void add(int u,int v){
    pp2[tot].to=v;
    pp2[tot].next=head[u];
    head[u]=tot++;
    pp2[tot].to=u;
    pp2[tot].next=head[v];
    head[v]=tot++;
}
struct ppi{
    int from;
    int to;
    int cost;
}pp1[maxn];
struct pi{
    int le,ri;
    int sum;
}pp[4*maxn];
void build(int tot,int l,int r){
    pp[tot].sum=0;
    pp[tot].le=l;
    pp[tot].ri=r;
    if(l==r) return ;
    build(2*tot,l,(l+r)/2);
    build(2*tot+1,(l+r)/2+1,r);
}
void merg(int tot,int l,int k){
    if(pp[tot].ri==pp[tot].le){
        pp[tot].sum=k;
        return ;
    }
    int mid;
    mid=(pp[tot].le+pp[tot].ri)/2;
    if(l<=mid) merg(2*tot,l,k);
    else merg(2*tot+1,l,k);
    pp[tot].sum=pp[2*tot].sum+pp[2*tot+1].sum;
}
int query(int tot,int l,int r){
    if(pp[tot].le>=l&&pp[tot].ri<=r){
        return pp[tot].sum;
    }
    int s=0;
    int mid=(pp[tot].le+pp[tot].ri)/2;
    if(l<=mid) s+=query(2*tot,l,r);
    if(r>mid) s+=query(2*tot+1,l,r);
    return s;
}
void dfs1(int u,int pa,int d){
    dep[u]=d;
    size[u]=1;
    son[u]=0;
```

```
    fa[u]=pa;
    int k,v;
    k=head[u];
    while(k!=-1){
        v=pp2[k].to;
        if(v==pa){
            k=pp2[k].next;
            continue;
        }
        dfs1(v,u,d+1);
        size[u]+=size[v];
        if(size[son[u]]<size[v]) son[u]=v;
        k=pp2[k].next;
    }
}
void dfs2(int u,int pa,int tp){
    int k,v;
    id[u]=num++;
    top[u]=tp;
    if(son[u]) dfs2(son[u],u,tp);
    k=head[u];
    while(k!=-1){
        v=pp2[k].to;
        if(v==pa||v==son[u]){
            k=pp2[k].next;
            continue;
        }
        dfs2(v,u,v);
        k=pp2[k].next;
    }
}
int get(int u,int v){
    int s=0,to1,to2;
    to1=top[u];
    to2=top[v];
    while(to1!=to2){
        if(dep[to1]<dep[to2]){
            swap(to1,to2);
            swap(u,v);
        }
        s+=query(1,id[to1],id[u]);
        u=fa[to1];
        to1=top[u];
    }
    if(u==v) return s;
    if(dep[u]>dep[v]) swap(u,v);
    s+=query(1,id[u]+1,id[v]);
    return s;
}
int main()
{
    int i,n,m,p,k,s;
    scanf("%d%d%d",&n,&m,&s);
```

```
    build(1,1,n-1);
    memset(head,-1,sizeof(head));
    tot=0;
    for(i=1;i<=n-1;i++){
        scanf("%d%d%d",&pp1[i].from,&pp1[i].to,&pp1[i].cost);
        add(pp1[i].from,pp1[i].to);
    }
    num=0;
    dfs1(1,1,1);
    dfs2(1,1,1);
    for(i=1;i<n;i++){
        if(fa[pp1[i].from]==pp1[i].to){
            merg(1,id[pp1[i].from],pp1[i].cost);
        }
        else merg(1,id[pp1[i].to],pp1[i].cost);
    }
    for(i=0;i<m;i++){
        scanf("%d",&p);
        if(p==0){
            scanf("%d",&k);
            printf("%d\n",get(s,k));
            s=k;
        }
        else{
            scanf("%d%d",&p,&k);
            if(fa[pp1[p].from]==pp1[p].to) merg(1,id[pp1[p].from],k);
            else  merg(1,id[pp1[p].to],k);
        }
    }
}
```

34 树链剖分模板（改点）

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#define maxn 30005
using namespace std;
int top[maxn],fa[maxn],size[maxn],son[maxn],id[maxn],dep[maxn],head[maxn],a[maxn],num,tot;
void init(void){
    memset(head,-1,sizeof(head));
    num=0;
    tot=0;
}
struct ppi{
    int to;
    int next;
}pp1[maxn<<1];
struct pi{
    int le;
```

```
    int ri;
    int sum;
}pp[maxn<<2];
void add(int u,int v){
    pp1[tot].to=v;
    pp1[tot].next=head[u];
    head[u]=tot++;
    pp1[tot].to=u;
    pp1[tot].next=head[v];
    head[v]=tot++;
}
void build(int tot,int l,int r){
    pp[tot].le=l;
    pp[tot].ri=r;
    pp[tot].sum=0;
    if(l==r) return ;
    build(2*tot,l,(l+r)/2);
    build(2*tot+1,(l+r)/2+1,r);
}
void merg(int tot,int x,int p){
    if(pp[tot].le==pp[tot].ri){
        pp[tot].sum=p;
        return ;
    }
    int mid=(pp[tot].le+pp[tot].ri)/2;
    if(x<=mid) merg(2*tot,x,p);
    else merg(2*tot+1,x,p);
    pp[tot].sum=pp[2*tot].sum+pp[2*tot+1].sum;
}
void dfs1(int u,int pa,int d){
    dep[u]=d;
    fa[u]=pa;
    size[u]=1;
    son[u]=0;
    int k,v;
    k=head[u];
    while(k!=-1){
        v=pp1[k].to;
        if(v!=pa){
            dfs1(v,u,d+1);
            size[u]+=size[v];
            if(size[son[u]]<size[v]) son[u]=v;
        }
        k=pp1[k].next;
    }
}
void dfs2(int u,int pa,int tp){
    top[u]=tp;
    id[u]=++num;
    if(son[u]) dfs2(son[u],u,tp);
    int k,v;
    k=head[u];
    while(k!=-1){
```

```
        v=pp1[k].to;
        if(v!=pa&&v!=son[u]){
            dfs2(v,u,v);
        }
        k=pp1[k].next;
    }
}
int query(int tot,int l,int r){
    if(pp[tot].le>=l&&pp[tot].ri<=r) return pp[tot].sum;
    int s=0;
    int mid=(pp[tot].le+pp[tot].ri)/2;
    if(l<=mid) s+=query(2*tot,l,r);
    if(r>mid) s+=query(2*tot+1,l,r);
    return s;
}
int get(int u,int v){
    int to1,to2,s=0;
    to1=top[u];
    to2=top[v];
    while(to1!=to2){
        if(dep[to1]<dep[to2]){
            swap(to1,to2);
            swap(u,v);
        }
        s+=query(1,id[to1],id[u]);
        u=fa[to1];
        to1=top[u];
    }
    if(dep[u]>dep[v]) swap(u,v);
    s+=query(1,id[u],id[v]);
    return s;
}
int main()
{
    int N=0,i,t,k,n,m,p;
    cin>>t;
    while(t--){
        printf("Case %d:\n",++N);
        scanf("%d",&n);
        init();
        for(i=1;i<=n;i++) scanf("%d",&a[i]);
        for(i=1;i<n;i++){
            scanf("%d%d",&p,&k);
            p++;
            k++;
            add(p,k);
        }
        build(1,1,n);
        dfs1(1,1,1);
        dfs2(1,1,1);
        for(i=1;i<=n;i++){
            merg(1,id[i],a[i]);
        }
```

```
        scanf("%d",&m);
        for(i=0;i<m;i++){
            scanf("%d",&p);
            if(p==0){
                scanf("%d%d",&p,&k);
                p++;
                k++;
                printf("%d\n",get(p,k));
            }
            else{
                scanf("%d%d",&p,&k);
                p++;
                merg(1,id[p],k);
            }
        }
    }
}
```

35 第一类斯特林数

```
/*
 s(p,k)的一个的组合学解释是：将 p 个物体排成 k 个非空循环排列（圆排列）的方法数。

 s(p,k)的递推公式： s(p,k)=(p-1)*s(p-1,k)+s(p-1,k-1) ,1<=k<=p-1
 边界条件：s(p,0)=0 ,p>=1  s(p,p)=1  ,p>=0

 递推关系的说明：
 考虑第 p 个物品，p 可以单独构成一个非空循环排列，这样前 p-1 种物品构成 k-1 个非空循环排列，方法数为 s(p-1,k-1)；
 也可以前 p-1 种物品构成 k 个非空循环排列，而第 p 个物品插入第 i 个物品的左边，这有(p-1)*s(p-1,k)种方法。
*/
```

36 第二类斯特林数

```
/*
 斯特林数出现在许多组合枚举问题中. 对第一类斯特林数 StirlingS1[n,m]， 给出恰包含 m 个圈的 n 个元素 的排列数目. 斯特林数满足母函数关系 . 注意某些 的定义与 Mathematica 中的不同，差别在于因子 . 第二类斯特林数 StirlingS2[n,m]给出把 n 个可区分小球分配到 m 个不可区分的的盒子，且盒子没有空盒子的方法的数量. 它们满足关系 . 划分函数 PartitionsP[n]给出把整数 n 写为正整数的和，不考虑顺序的方法的数目. PartitionsQ[n]给出把整数 n 写为正整数的和，并且和中的整数是互不相同的 写法的数目
 设 S(p,k)是斯特林数
 S(p,k)的一个组合学解释是：将 p 个物体划分成 k 个非空的不可辨别的（可以理解为盒子没有编号）集合的方法数。
```

S(p,k)的递推公式是：
S(p,k) = k*S(p-1,k) + S(p-1,k-1) ,1<= k <=p-1
边界条件：
S(p,p) = 1 ,p>=1
S(p,0) = 0 ,p>=1
递推关系的说明：考虑第 p 个物品，p 可以单独构成一个非空集合，此时前 p-1 个物品构成 k-1 个非空的不可辨别的集合，方法数为 S(p-1,k-1)；也可以前 p-1 种物品构成 k 个非空的不可辨别的集合，第 p 个物品放入任意一个中，这样有 k*S(p-1,k)种方法。
*/

37 KM 算法模板（二分图的最大权匹配）

```
#include <stdio.h>
#include <algorithm>
#include <string.h>
#include <iostream>
using namespace std;
/*  KM 算法
 *   复杂度 O(nx*nx*ny)
 *  求最大权匹配
 *   若求最小权匹配，可将权值取相反数，结果取相反数
 *  点的编号从 0 开始
 */
const int N = 310;
const int INF = 0x3f3f3f3f;
int nx,ny;//两边的点数
int g[N][N];//二分图描述
int match[N],lx[N],ly[N];//y 中各点匹配状态，x,y 中的点标号
int slack[N];
bool visx[N],visy[N];

bool DFS(int x)
{
   visx[x] = true;
   for(int y = 0; y < ny; y++)
   {
      if(visy[y])continue;
      int tmp = lx[x] + ly[y] - g[x][y];
      if(tmp == 0)
      {
         visy[y] = true;
         if(match[y] == -1 || DFS(match[y]))
         {
            match[y] = x;
            return true;
         }
      }
      else if(slack[y] > tmp)
         slack[y] = tmp;
```

```
    }
    return false;
}
int KM()
{
    memset(match,-1,sizeof(match));
    memset(ly,0,sizeof(ly));
    for(int i = 0;i < nx;i++)
    {
        lx[i] = -INF;
        for(int j = 0;j < ny;j++)
            if(g[i][j] > lx[i])
                lx[i] = g[i][j];
    }
    for(int x = 0;x < nx;x++)
    {
        for(int i = 0;i < ny;i++)
            slack[i] = INF;
        while(true)
        {
            memset(visx,false,sizeof(visx));
            memset(visy,false,sizeof(visy));
            if(DFS(x))break;
            int d = INF;
            for(int i = 0;i < ny;i++)
                if(!visy[i] && d > slack[i])
                    d = slack[i];
            for(int i = 0;i < nx;i++)
                if(visx[i])
                    lx[i] -= d;
            for(int i = 0;i < ny;i++)
            {
                if(visy[i])ly[i] += d;
                else slack[i] -= d;
            }
        }
    }
    int res = 0;
    for(int i = 0;i < ny;i++)
        if(match[i] != -1)
            res += g[match[i]][i];
    return res;
}
```

38 mancher(O(n)回文串)算法

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
using namespace std;
```

```
char d[200000],c[300000];
int p[300000],r[300005];//半径长度//加入以 i 为中心左边到 a 都是回文串那 r[i]=i-a+1;
void mancher(int n){
    int i,j,id,mx;
    r[0]=1;
    mx=0;
    id=0;
    for(i=1;i<=2*n;i++){
        if(i>=mx) r[i]=1;
        else {
            r[i]=min(r[id-(i-id)],mx-i);
        }
        while(i-r[i]>=0&&i+r[i]<=2*n&&c[i-r[i]]==c[i+r[i]]) r[i]++;
        if(mx<r[i]+i){
            mx=r[i]+i;
            id=i;
        }
    }
}
int main()
{
    int i,j,n,p,id,re;
    while(scanf("%s",d)!=EOF){
        n=strlen(d);
        c[0]='#';
        for(i=1;i<=n;i++){
            c[2*i-1]=d[i-1];
            c[2*i]='#';
        }
        mancher(n);
        p=1;
        for(i=1;i<=2*n;i++){
            p=max(p,r[i]-1);
        }
        printf("%d\n",p);
    }
}
```

39 无向图最小割 store-wragner 算法

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
using namespace std;
const int maxn=550;
typedef long long LL;
int map[maxn][maxn];
int w[maxn];
bool vis[maxn],use[maxn];
int m1,s,t;
```

```
void store(int n){
   memset(vis,0,sizeof(vis));
   memset(w,0,sizeof(w));
   int i,j,mm,f=0;
   for(i=0;i<n;i++){
      mm=-100000;
      for(j=0;j<n;j++){
         if(!vis[j]&&!use[j]){
            if(mm<w[j]){
               mm=w[j];
               f=j;
            }
         }
      }
      if(t==f){
         m1=w[t];
         return ;
      }
      s=t;
      t=f;
      vis[f]=1;
      for(j=0;j<n;j++){
         if(!vis[j]&&!use[j]){
            w[j]+=map[f][j];
         }
      }
   }
   m1=w[t];
}
int  solve(int n){
   int ans=1000000000;
   int i,j;
   memset(use,0,sizeof(use));
   for(i=0;i<n-1;i++){
      s=-1;
      t=-1;
      store(n);
      if(ans>m1) ans=m1;
      use[t]=1;
      for(j=0;j<n;j++){
         map[s][j]+=map[t][j];
         map[j][s]=map[s][j];
      }
   }
   return ans;
}
```

40 原根判断条件和个数

```
/*
如果（a,m)=1,a 是模 m 的原根，那么 m 的必要条件是 m=1、2、4、p、2*p、p^n。
```

如果 m 存在原根那么模 m 的原根个数等于 φ(φ(m))
*/

41 泛化背包

```
/*
 在背包容量为 V 的背包问题中，泛化物品是一个定义域为 0..V 中的整数的函数 h，当分配给它的费用为 v 时，能得到的价值就是 h(v)。
 如果面对两个泛化物品 h 和 l，要用给定的费用从这两个泛化物品中得到最大的价值，怎么求呢？事实上，对于一个给定的费用 v，只需枚举将这个费用如何分配给两个泛化物品就可以了。同样的，对于 0..V 的每一个整数 v，可以求得费用 v 分配到 h 和 l 中的最大价值 f(v)。
即：f(v) = maxf{h(k) + l(v - k)} 0 <= k <= v
*/
/*
 对于要递归的子节点，直接把父节点的 dp 值赋给它
 更新子节点。
 用上面所说的更新泛化物品的方法对于父节点用完成递归的子节点进行更新。
 for (int k=lim-c[j];k>=0;k--) dp[j][k]=dp[now][k];
 DP(j,lim-c[j]);
 for (int k=c[j];k<=lim;k++)
 dp[now][k]=Max(dp[now][k],dp[j][k-c[j]]+v[j]);

 注意必须选择这个节点才能选下面的!
 注意下面的 dp 的取值范围!
*/
/*
void dfs(int u,int k,int pa,int gg){
    int j,p;
    p=head[u];
    for(;p!=-1;p=pp1[p].next){
        int q;
        q=pp1[p].to;
        if(q!=pa){
            if(gg>=pp[q].a){
                if(!vis[q])
                    for(j=0;j<=gg-pp[q].a;j++) dp[q][j]=dp[u][j];
                dfs(q,k^1,u,gg-pp[q].a);
                if(!vis[q]){
                    for(j=pp[q].a;j<=gg;j++) dp[u][j]=max(dp[u][j],dp[q][j-pp[q].a]+pp[q].b);
                }
                else{
                    for(j=gg;j>=pp[q].a;j--) dp[u][j]=max(dp[u][j],dp[u][j-pp[q].a]+pp[q].b);
                }
            }
        }
    }
}
*/
```

42 二维 RMQ

```
#include <cstdio>
#include <cstring>
#include <string>
#include <cstdlib>
#include <iostream>
#include <cmath>
#define N 405
using namespace std;
typedef long long LL;
int dp[N][N][9][9],dp1[N][N][9][9]; ;
int n , m ;
void init() {
   for(int i = 1 ; i <= n ; ++i) {
      for(int j = 1 ; j <= m ;++j) {
         scanf("%d" ,&dp[i][j][0][0]) ;
         dp1[i][j][0][0]=dp[i][j][0][0];
      }
   }
   for(int r = 0 ; (1<<r) <= n ; ++r) {
      for(int c = 0 ; (1<<c) <= m ; ++c) {
         if(r == 0 && c == 0) continue ;
         for(int i = 1 ; i+(1<<r)-1 <= n ; ++i) {
            for(int j = 1 ; j+(1<<c)-1 <= m ; ++j) {
               if(r) {
                  dp[i][j][r][c] = max(dp[i][j][r-1][c] , dp[i+(1<<(r-1))][j][r-1][c]) ;
               }else {
                  dp[i][j][r][c] = max(dp[i][j][r][c-1] , dp[i][j+(1<<(c-1))][r][c-1]) ;//先算当 r 为 0 的时
候
               }
            }
         }
      }
   }
   for(int r = 0 ; (1<<r) <= n ; ++r) {
      for(int c = 0 ; (1<<c) <= m ; ++c) {
         if(r == 0 && c == 0) continue ;
         for(int i = 1 ; i+(1<<r)-1 <= n ; ++i) {
            for(int j = 1 ; j+(1<<c)-1 <= m ; ++j) {
               if(r) {
                  dp1[i][j][r][c] = min(dp1[i][j][r-1][c] , dp1[i+(1<<(r-1))][j][r-1][c]) ;
               }else {
                  dp1[i][j][r][c] = min(dp1[i][j][r][c-1] , dp1[i][j+(1<<(c-1))][r][c-1]) ;//先算当 r 为 0 的
时候
               }
            }
         }
      }
   }
}
```

```
int query(int x1 , int y1 , int x2 , int y2) {
    int kx = (int)(log(x2 - x1 + 1.0)/log(2.0)) ;
    int ky = (int)(log(y2 - y1 + 1.0)/log(2.0)) ;//log2 居然 CE
    int m1 = dp[x1][y1][kx][ky] ;
    int m2 = dp[x2-(1<<kx)+1][y1][kx][ky] ;
    int m3 = dp[x1][y2-(1<<ky)+1][kx][ky] ;
    int m4 = dp[x2-(1<<kx)+1][y2-(1<<ky)+1][kx][ky] ;
    int tmp = max(m1 , max(m2 , max(m3 , m4))) ;//分为 4 个部分
    return tmp;
}
int query1(int x1 , int y1 , int x2 , int y2) {
    int kx = (int)(log(x2 - x1 + 1.0)/log(2.0)) ;
    int ky = (int)(log(y2 - y1 + 1.0)/log(2.0)) ;//log2 居然 CE
    int m1 = dp1[x1][y1][kx][ky] ;
    int m2 = dp1[x2-(1<<kx)+1][y1][kx][ky] ;
    int m3 = dp1[x1][y2-(1<<ky)+1][kx][ky] ;
    int m4 = dp1[x2-(1<<kx)+1][y2-(1<<ky)+1][kx][ky] ;
    int tmp = min(m1 , min(m2 , min(m3 , m4))) ;//分为 4 个部分
    return tmp;
}
```

43 欧拉路模板

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#include<vector>
using namespace std;
vector<int >g[10005];
int vis[100005];
int in[10005],out[10005],use[10005];
char c[6];
int a[200005],cnt;
int change(char c){
    if(c>='a'&&c<='z') return c-'a';
    if(c>='A'&&c<='Z') return 26+c-'A';
    return 52+c-'0';
}
void add(int a,int b){
    g[a].push_back(b);
}
int it[10005];
void dfs(int u,int ee){
    int p;
    p=(int)g[u].size();
    while(it[u]<p){
        int m=it[u];
        it[u]++;
```

```
        use[g[u][m]]=1;
        dfs(g[u][m],0);
    }
    a[cnt++]=u;//放外面的好处是对于不能走的点也可以记录，而且一定要后序（因为后序能保证不能走的点一定是终点，而先序却不行）
}
int main()
{
    int i,m;
    int x=0,y=0;
    m=0;
    for(i=0;i<=10000;i++){
        if(in[i]!=out[i]){
            if(in[i]==out[i]+1){
                x++;
            }
            else if(out[i]==in[i]+1){
                y++;
            }
            else{
                m=1;
            }
        }
    }
    if(!((x==0&&y==0)||(x==1&&y==1))) m=1;//
    if(m) printf("NO\n");
    else{
        if(x==0){
            for(i=0;i<=10000;i++){
                if(in[i]!=0){
                    use[i]=1;
                    dfs(i,0);
                    break;
                }
            }
        }
        else{
            for(i=0;i<=10000;i++){
                if(out[i]==in[i]+1){
                    use[i]=1;
                    dfs(i,0);
                    break;
                }
            }
        }
        for(i=0;i<=10000;i++){
            if(!(in[i]==0&&out[i]==0)){
                if(!use[i]){
                    m=1;
                    break;
                }
            }
        }
```

```
        if(m) printf("NO\n");
        else{
            printf("YES\n");
        }
    }
}
```

44 一堆石子成块减一的总操作数

```
/*
假如石子堆和为 s 且最多选 m 个石子堆减一，那么一共需要操作 max（max(a[i]),s/m）次
*/
```

45 主席树模板

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
using namespace std;
const int maxn=100005;
struct pi{
    int sum;
    int lson;
    int rson;
}pp[maxn*18];
int root[maxn],tot;
void build(int cnt,int l,int r){
    pp[cnt].sum=0;
    if(l==r) return ;
    pp[cnt].lson=tot+1;
    tot++;
    build(tot,l,(l+r)/2);
    pp[cnt].rson=tot+1;
    tot++;
    build(tot,(l+r)/2+1,r);
}
void merg(int qq,int cnt,int n,int p,int k){
    int le,ri,mid;
    le=1;
    ri=n;
    while(le<=ri){
        mid=(le+ri)/2;
        pp[cnt]=pp[qq];
        pp[cnt].sum+=k;
        if (le==ri) break;
        if(p<=mid){
            pp[cnt].lson=tot+1;
```

```
            tot++;
            ri=mid;
            cnt=tot;
            qq=pp[qq].lson;
        }
        else{
            pp[cnt].rson=tot+1;
            tot++;
            le=mid+1;
            cnt=tot;
            qq=pp[qq].rson;
        }
    }
}
int query(int cnt,int le,int ri,int l,int r){
    int s=0;
    int mid;
    if(le>=l&&ri<=r){
        return pp[cnt].sum;
    }
    mid=(le+ri)/2;
    if(l<=mid)
        s+=query(pp[cnt].lson,le,mid,l,r);
    if(r>mid) s+=query(pp[cnt].rson,mid+1,ri,l,r);
    return s;
}
int a[maxn],b[maxn];
int get(int l,int r,int n,int k){
    int le,ri,mid;
    le=1;
    ri=n;
    while(le<=ri){
        mid=(le+ri)/2;
        int q=query(root[r],1,n,1,mid)-query(root[l-1],1,n,1,mid);
        if(q<k) le=mid+1;
        else ri=mid-1;
    }
    return le;
}
int main()
{
    int i,n,m;
    while(cin>>n>>m){
        root[0]=0;
        tot=0;
        for(i=1;i<=n;i++){
            scanf("%d",&a[i]);
            b[i]=a[i];
        }
        sort(b+1,b+1+n);
        for(i=1;i<=n;i++){
            a[i]=(int)(lower_bound(b+1,b+1+n,a[i])-b);
        }
```

```
        for(i=1;i<=n;i++){
            root[i]=tot+1;
            tot++;
            merg(root[i-1],root[i],n,a[i],1);
        }
        for(i=0;i<m;i++){
            int a,bb,c;
            scanf("%d%d%d",&a,&bb,&c);
            printf("%d\n",b[get(a,bb,n,c)]);
        }
    }
}
```

46 二维线段树

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
using namespace std;
const int maxn=805;
struct pi{
    int max;
    int min;
}pp[maxn<<2][maxn<<2];
void init(){
    int i,j;
    for(i=0;i<maxn<<2;i++){
        for(j=0;j<maxn<<2;j++){
            pp[i][j].max=0;
            pp[i][j].min=1000000000;
        }
    }
}
void merg1(int tot,int l,int r,int y,int k,int to1,int la){
    if(l==r){
        if(la==1) pp[to1][tot].max=pp[to1][tot].min=k;
        else{
            pp[to1][tot].max=max(pp[2*to1][tot].max,pp[2*to1+1][tot].max);
            pp[to1][tot].min=min(pp[2*to1][tot].min,pp[2*to1+1][tot].min);
        }
    }
    else{
        int mid=(l+r)/2;
        if(y<=mid) merg1(2*tot,l,mid,y,k,to1,la);
        else merg1(2*tot+1,mid+1,r,y,k,to1,la);
        pp[to1][tot].max=max(pp[to1][2*tot].max,pp[to1][2*tot+1].max);
        pp[to1][tot].min=min(pp[to1][2*tot].min,pp[to1][2*tot+1].min);
    }
}
int n;
```

```
void merg2(int tot,int l,int r,int x,int y,int k){
    if(l==r){
        merg1(1,1,n,y,k,tot,1);
    }
    else{
        int mid=(l+r)/2;
        if(x<=mid) merg2(2*tot,l,mid,x,y,k);
        else merg2(2*tot+1,mid+1,r,x,y,k);
        merg1(1,1,n,y,k,tot,0);
    }
}
int query1(int tot,int l,int r,int x,int y,int to1,int k){
    if(l>=x&&r<=y){
        if(k==0) return pp[to1][tot].max;
        return pp[to1][tot].min;
    }
    int s;
    if(k==0) s=0;
    else s=1000000000;
    int mid=(l+r)/2;
    if(k==0){
        if(x<=mid) s=max(s,query1(2*tot,l,mid,x,y,to1,k));
        if(y>mid) s=max(s,query1(2*tot+1,mid+1,r,x,y,to1,k));
    }
    else{
        if(x<=mid) s=min(s,query1(2*tot,l,mid,x,y,to1,k));
        if(y>mid) s=min(s,query1(2*tot+1,mid+1,r,x,y,to1,k));
    }
    return s;
}
int query2(int tot,int l,int r,int x1,int y1,int x,int y,int k){
    if(l>=x1&&r<=y1){
        return query1(1, 1, n, x, y, tot, k);
    }
    int s,mid;
    mid=(l+r)/2;
    if(k==0){
        s=0;
        if(x1<=mid){
            s=max(s,query2(2*tot,l,mid,x1,y1,x,y,k));
        }
        if(y1>mid)s=max(s,query2(2*tot+1,mid+1,r,x1,y1,x,y,k));
    }
    else{
        s=1000000000;
        if(x1<=mid){
            s=min(s,query2(2*tot,l,mid,x1,y1,x,y,k));
        }
        if(y1>mid)s=min(s,query2(2*tot+1,mid+1,r,x1,y1,x,y,k));
    }
    return s;
}
int main()
```

```
{
    int i,j,m,t,N=0;
    cin>>t;
    while(t--){
        cin>>n;
        init();
        for(i=1;i<=n;i++){
            for(j=1;j<=n;j++){
                scanf("%d",&m);
                merg2(1,1,n,i,j,m);
            }
        }
        cin>>m;
        printf("Case #%d:\n",++N);
        for(i=0;i<m;i++){
            int a,b,l;
            int x,y,x1,y1,x2,y2;
            scanf("%d%d%d",&a,&b,&l);
            x1=a-(l-1)/2;
            if(x1<1) x1=1;
            y1=a+(l-1)/2;
            if(y1>n) y1=n;
            x2=b-(l-1)/2;
            if(x2<1) x2=1;
            y2=b+(l-1)/2;
            if(y2>n) y2=n;
            x=query2(1,1,n,x1,y1,x2,y2,0);
            y=query2(1,1,n,x1,y1,x2,y2,1);
            printf("%d\n",(x+y)/2);
            merg2(1,1,n,a,b,(x+y)/2);
        }
    }
}
```

47 莫队：

```
/*
 把询问[l,r]抽象成一个点(l,r)，题目就转化为求 n 个点的最小曼哈顿哈密尔顿路；由于这是个 NP 问题，所以我们希望找到一个稍微劣一点又不是很劣的但是能快速求得方案。莫涛大神在他的论文里使用了二维曼哈顿距离最小生成树
 二维曼哈顿距离最小生成树可以用区域划分法+树状数组/线段树维护区间极值在 nlogn 的时间复杂度内完成构图，并用 Kruskal 在 n log n 时间内求得，但是代码比较繁琐。
 这里介绍一个优美的替代品——分块
 将 n 个数分成 sqrt(n)块
 按区间排序，以左端点所在块内为第一关键字，右端点为第二关键字，进行排序
 也就是以( pos [l],r )排序
 搞得过程是这样的：
 一、i 与 i+1 在同一块内，r 单调递增，所以 r 是 O(n)的。由于有 n^0.5 块,所以这一部分时间复杂度是 n^1.5。
```

二、i 与 i+1 跨越一块，r 最多变化 n，由于有 n^0.5 块，所以这一部分时间复杂度是 n^1.5
三、i 与 i+1 在同一块内时变化不超过 n^0.5，跨越一块也不会超过 2*n^0.5，不妨看作是 n^0.5。由于有 n 个数，所以时间复杂度是 n^1.5
于是就变成了 O(n^1.5)了
*/

48 后缀自动机

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
using namespace std;
typedef long long LL;
inline LL C2(LL x){
    return (x * (x - 1)) >> 1;
}
const int ALB = 26;
const int MAXN =1000005;
int arr[MAXN];
char str[MAXN];
struct NODE{
    NODE *par, *go[ALB];
    int val;
    int cnt;
    int size;
    NODE(int val):par(NULL),val(val),size(1){
        cnt=-1;
        fill_n(go, ALB, (NODE*)NULL);
    }
    NODE(){};
}node[MAXN<<1], *last,*root,*no;
int ncnt;
void Init(){
    ncnt = 1;
    root=&node[0];
    *root=NODE(0);
    last=root;
}
void Append(int x){
    NODE *p = last, *np = &node[ncnt ++];
    *np = NODE(p->val + 1);
    for(; p && !p->go[x]; p = p->par)p->go[x] = np;
    if(p){
        NODE *q = p->go[x];
```

```
        if(p->val + 1 == q->val){
            np->par = q;
        }
        else{
            NODE *nq = &node[ncnt ++];
            *nq = *q;
            nq->size = 0;
            nq->val = p->val + 1;
            q->par = nq;
            np->par = nq;
            for(; p && p->go[x] == q; p = p->par)
                p->go[x] = nq;
        }
    }
    else{
        np->par = root;
    }
    last = np;
}
void Deal(int N){
    int i;
    NODE *cur, *p;
    static int c[MAXN];
    memset(c,0,sizeof(c));
    static NODE* top[MAXN<<1];
    for(i = 0; i < ncnt; i ++)
        c[node[i].val] ++;
    for(i = 1; i <= N; i ++)
        c[i] += c[i - 1];
    for(i = 0; i < ncnt; i ++)
        top[-- c[node[i].val]] = &node[i];
    for(i = ncnt - 1; i > 0; i --){
        cur = top[i];
        p = cur->par;
        p->size += cur->size;
    }
    }
```

49 阶梯博弈

/*
如这就是一个阶梯博弈的初始状态 2 1 3 2 4 ... 只能把后面的点往前面放...如何来分析这个问题呢...其实阶梯博弈经过转换可以变为 Nim..把所有奇数阶梯看成 N 堆石子..做 nim..把石子从奇数堆移动到偶数堆可以理解为拿走石子..就相当于几个奇数堆的石子在做 Nim..(如所给样例..2^3^4=5 不为零所以先手必败)为什么可以这样来转化？

假设我们是先手...所给的阶梯石子状态的奇数堆做 Nim 先手能必胜...我就按照能赢的步骤将奇数堆的石子移动到偶数堆...如果对手也是移动奇数堆..我们继续移动奇数堆..如果对手将偶数堆的石子移动到了奇数堆..那么我们紧接着将对手所移动的这么多石子从那个奇数堆移动到下面的偶数堆...两次操作后...相当于偶数堆的石子向下移动了几个..而奇数堆依然是原来的样子...即为必胜的状态...就算后手一直在移动偶数堆的石子到奇数堆..我们就一直跟着他将石子继续往下移..保持奇数堆不变...如此做下去..我可以跟着后手把偶数堆的石子移动到 0..然后你就不能移动这些石子了...所以整个过程..将偶数堆移动到奇数堆不会影响奇数堆做 Nim 博弈的过程..整个过程可以抽象为奇数堆的 Nim 博弈...

其他的情况...先手必输的...类似推理...只要判断奇数堆做 Nim 博弈的情况即可...

为什么是只对奇数堆做 Nim 就可以...而不是偶数堆呢？...因为如果是对偶数堆做 Nim...对手移动奇数堆的石子到偶数堆..我们跟着移动这些石子到下一个奇数堆...那么最后是对手把这些石子移动到了 0..我们不能继续跟着移动...就只能去破坏原有的 Nim 而导致胜负关系的不确定...所以只要对奇数堆做 Nim 判断即可知道胜负情况...*/

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
   int t,i,j,n,m,N=0;
   cin>>t;
   while(t--){
      scanf("%d",&n);
      m=0;
      for(i=1;i<=n;i++){
         scanf("%d",&j);
         if((i%3==2||(i%3==0&&i%2==0))) m=m^j;
      }
      printf("Case %d: ",++N);
      if(m) printf("Alice\n");
      else printf("Bob\n");
   }
}
```

50 素数定理

```
/*
10^9 方内的连续素数之间距离不超过 300
*/
```

51 得分序列合法条件

```
/*
对于一个序列得分 0<=s1<=s2<=s3...<=sn<=n
```

一定满足：
1、s1+..+si>=C(i,2)
2、s2+...+sn=C(n,2)
*/

52 SBT

```
#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <cmath>
#include <algorithm>
#include <set>
#include <map>
#include <vector>
#include <queue>
#include <ctime>
using namespace std;
#define LL long long
const int N = 10005;
const int INF=0x7FFFFFFF;

struct SBT
{
   int key,left,right,size;
} tree[N];

int root,top;

void left_rot(int &x)
{
   int y = tree[x].right;
   tree[x].right = tree[y].left;
   tree[y].left = x;
   tree[y].size = tree[x].size;//转上去的节点数量为先前此处节点的 size
   tree[x].size = tree[tree[x].left].size + tree[tree[x].right].size + 1;
   x = y;
}

void right_rot(int &x)
{
   int y = tree[x].left;
   tree[x].left = tree[y].right;
   tree[y].right = x;
   tree[y].size = tree[x].size;
   tree[x].size = tree[tree[x].left].size + tree[tree[x].right].size + 1;
   x = y;
}
```

```
void maintain(int &x,bool flag)
{
   if(flag == false)//左边
   {
      if(tree[tree[tree[x].left].left].size > tree[tree[x].right].size)//左孩子的左子树大于右孩子
         right_rot(x);
      else if(tree[tree[tree[x].left].right].size > tree[tree[x].right].size)//右孩子的右子树大于右孩子
      {
         left_rot(tree[x].left);
         right_rot(x);
      }
      else return;
   }
   else //右边
   {
      if(tree[tree[tree[x].right].right].size > tree[tree[x].left].size)//右孩子的右子树大于左孩子
         left_rot(x);
      else if(tree[tree[tree[x].right].left].size > tree[tree[x].left].size)//右孩子的左子树大于左孩子
      {
         right_rot(tree[x].right);
         left_rot(x);
      }
      else return;
   }
   maintain(tree[x].left,false);
   maintain(tree[x].right,true);
   maintain(x,true);
   maintain(x,false);
}

/*
 *insert 没有合并相同的元素，如果出现相同的元素则把它放到右子树上，这样能保证求第 k 小
数的时候对相同元素也能正确
 */
void insert(int &x,int key)
{
   if(x == 0)
   {
      x = ++top;
      tree[x].left = tree[x].right = 0;
      tree[x].size = 1;
      tree[x].key = key;
   }
   else
   {
      tree[x].size ++;
      if(key < tree[x].key) insert(tree[x].left,key);
      else  insert(tree[x].right,key);//相同元素插入到右子树中
      maintain(x, key >= tree[x].key);//每次插入把平衡操作压入栈中
   }
}
```

```
int del(int &p,int w)
{
   if (tree[p].key==w || (tree[p].left==0 && w<tree[p].key) || (tree[p].right==0 && w>tree[p].key))
   {
      int delnum=tree[p].key;
      if (tree[p].left==0 || tree[p].right==0) p=tree[p].left+tree[p].right;
      else tree[p].key=del(tree[p].left,INF);
      return delnum;
   }
   if (w<tree[p].key) return del(tree[p].left,w);
   else return del(tree[p].right,w);
}

int  remove(int &x,int key)
{
   int d_key;
   //if(!x) return 0;
   tree[x].size --;
   if((key == tree[x].key)||(key < tree[x].key && tree[x].left == 0) ||
     (key>tree[x].key && tree[x].right == 0))
   {
      d_key = tree[x].key;
      if(tree[x].left && tree[x].right)
      {
         tree[x].key = remove(tree[x].left,tree[x].key+1);
      }
      else
      {
         x = tree[x].left + tree[x].right;
      }
   }
   else if(key > tree[x].key)
      d_key = remove(tree[x].right,key);
   else if(key < tree[x].key)
      d_key = remove(tree[x].left,key);
   return d_key;
}

int getmin()
{
   int x;
   for(x = root ; tree[x].left; x = tree[x].left);
   return tree[x].key;
}

int getmax()
{
   int x;
   for(x = root ; tree[x].right; x = tree[x].right);
   return tree[x].key;
}

int select(int &x,int k)//求第 k 小数
```

```
{
   int r = tree[tree[x].left].size + 1;
   if(r == k) return tree[x].key;
   else if(r < k) return select(tree[x].right,k - r);
   else return select(tree[x].left,k);
}

int rank(int &x,int key)//求 key 排第几
{
   if(key < tree[x].key)
      return rank(tree[x].left,key);
   else if(key > tree[x].key)
      return rank(tree[x].right,key) + tree[tree[x].left].size + 1;
   return tree[tree[x].left].size + 1;
}

int pred(int &x,int y,int key)//前驱 小于
{
   if(x == 0) return y;
   if(tree[x].key < key)
      return pred(tree[x].right,x,key);
   else return pred(tree[x].left,y,key);
}

int succ(int &x,int y,int key)//后继 大于
{
   if(x == 0) return y;
   if(tree[x].key > key)
      return succ(tree[x].left,x,key);
   else return succ(tree[x].right,y,key);
}

void inorder(int &x)
{
   if(x==0) return;
   else
   {
      inorder(tree[x].left);
      cout<<x<<" "<<tree[x].key<<" "<<" "<<tree[x].size<<" "<<tree[tree[x].left].key<<"
"<<tree[tree[x].right].key<<endl;
      inorder(tree[x].right);
   }
}

int main()
{
   root = top = 0;
   char ch;
   int x,tmp;
   while(scanf("%c %d",&ch,&x))
   {
      switch(ch)
      {
```

```
            case 'I':
                insert(root,x);
                break;
            case 'D':
                remove(root,x);
                break;
            case 'K':
                tmp=select(root,x);
                printf("%d\n",tmp);
                break;
            case 'C':
                printf("%d\n",rank(root,x));
                break;
            case 'P':
                tmp = pred(root,0,x);
                printf("%d\n",tree[tmp].key);
                break;
            case 'S':
                tmp = succ(root,0,x);
                printf("%d\n",tree[tmp].key);
                break;
            case 'L':
                inorder(root);
                break;
        }

    }
    return 0;
}
```

53 splay

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
using namespace std;
const int maxn=300005;
int ch[maxn][2],size[maxn],fa[maxn],flag[maxn];
void push(int tot){
    if(flag[tot]){
        swap(ch[tot][0],ch[tot][1]);
        if(ch[tot][0])
            flag[ch[tot][0]]^=1;
        if(ch[tot][1])
            flag[ch[tot][1]]^=1;
        flag[tot]=0;
    }
}
int root;
void update(int tot){
```

```
    size[tot]=1+size[ch[tot][0]]+size[ch[tot][1]];
}
int build(int l,int r,int ff){
    if(l>r) return 0;
    int mid=(l+r)/2;
    fa[mid]=ff;
    ch[mid][0]=build(l,mid-1,mid);
    ch[mid][1]=build(mid+1,r,mid);
    update(mid);
    return mid;
}
void roat(int p,int k){
    int f=fa[p];
    if(ch[p][k^1]){
        fa[ch[p][k^1]]=f;
    }
    ch[f][k]=ch[p][k^1];
    ch[p][k^1]=f;
    if(root==f){
        root=p;
    }
    else{
        int k=f==ch[fa[f]][1];
        ch[fa[f]][k]=p;
    }
    fa[p]=fa[f];
    fa[f]=p;
    update(f);
    update(p);
}
void splay(int p,int q){
    while(fa[p]!=q){
        push(fa[fa[p]]);
        push(fa[p]);
        push(p);
        if(fa[fa[p]]==q){
            roat(p,p==ch[fa[p]][1]);
            return ;
        }
        int kind=fa[p]==ch[fa[fa[p]]][1];
        if(p==ch[fa[p]][kind]){
            roat(p,kind);
            roat(p,kind);
        }
        else{
            roat(p,kind^1);
            roat(p,kind);
        }
    }
}
int get(int p,int k){
    push(p);
    if(size[ch[p][0]]+1==k) return p;
```

```
    if(size[ch[p][0]]<k) return get(ch[p][1],k-size[ch[p][0]]-1);
    return get(ch[p][0],k);
}
void flip(int l,int r,int n){
    if(l==1&&r==n){
        flag[root]^=1;
        return ;
    }
    if(l==1){
        r=get(root,r+1);
        splay(r,0);
        flag[ch[root][0]]^=1;
    }
    else if(r==n){
        l=get(root,l-1);
        splay(l,0);
        flag[ch[root][1]]^=1;
    }
    else{
        l=get(root,l-1);
        r=get(root,r+1);
        splay(l,0);
        splay(r,l);
        flag[ch[r][0]]^=1;
    }
}
void getle(int p){
    while(1){
        push(p);
        if(ch[p][0]==0) return ;
        push(ch[p][0]);
        roat(ch[p][0],0);
        p=fa[p];
    }
}
void insert(int l,int r,int x,int n){
    if(l==1&&r==n) return ;
    int w;
    int l1=l,r1=r;
    if(l==1){
        r=get(root,r+1);
        splay(r,0);
        w=ch[r][0];
        ch[r][0]=0;
        update(r);
    }
    else if(r==n){
        l=get(root,l-1);
        splay(l,0);
        w=ch[l][1];
        ch[l][1]=0;
        update(l);
    }
```

```
    else{
        l=get(root,l-1);
        r=get(root,r+1);
        splay(l,0);
        splay(r,l);
        w=ch[r][0];
        ch[r][0]=0;
        update(r);
        update(l);
    }
    if(x==0){
        getle(root);
        ch[root][0]=w;
        fa[w]=root;
        update(root);
    }
    else{
        int ww=get(root,x);
        splay(ww,0);
        if(x==n-(r1-l1+1)){
            ch[root][1]=w;
            fa[w]=root;
            update(root);
        }
        else{
            getle(ch[root][1]);
            ch[ch[root][1]][0]=w;
            fa[w]=ch[root][1];
            update(ch[root][1]);
            update(root);
        }
    }
}
char c[4];
int a[maxn];
int main()
{
    int i,n,m;
    while(1){
        scanf("%d%d",&n,&m);
        if(n==-1&&m==-1) break;
        memset(ch,0,sizeof(ch));
        memset(flag,0,sizeof(flag));
        root=0;
        fa[0]=size[0]=0;
        root=build(1,n,0);
        for(i=0;i<m;i++){
            scanf("%s",c);
            if(c[0]=='F'){
                int a,b;
                scanf("%d%d",&a,&b);
                flip(a,b,n);
            }
```

```
            else{
                int a,b,c;
                scanf("%d%d%d",&a,&b,&c);
                insert(a,b,c,n);
            }
        }
        for(i=1;i<=n;i++){
            splay(i,0) ;
            a[size[ch[root][0]]+1]=i;
        }
        for(i=1;i<=n;i++){
            if(i!=1) printf(" ");
            printf("%d",a[i]);
        }
        printf("\n");
    }
}
```

54 回文自动机

```
#include<cstdio>
#include<cstring>
#include<iostream>
#include<algorithm>
using namespace std;
const int maxn=300005;
int len[maxn];
int ch[maxn][26];
int fail[maxn];
long long cnt[maxn];
int sum[maxn];
int tot;
int s[maxn];
int p1;
int last;
char c[maxn];
int get_fail(int p){
    if(s[p1-len[p]-1]==s[p1]){
        return p;
    }
    return get_fail(fail[p]);
}
int ss;
void insert(int p){
    s[++p1]=c[p];
    int pp=get_fail(last);
    if(ch[pp][c[p]]==0){
        ch[pp][c[p]]=++tot;
        ss++;
        len[tot]=len[pp]+2;
```

```
        int now=tot;
        if(pp==1){
            fail[now]=0;
        }
        else{
            int pp1=ch[get_fail(fail[pp])][c[p]];
            fail[now]=pp1;
        }
        sum[now]=sum[fail[now]]+1;
    }
    last=ch[pp][c[p]];
    cnt[last]++;
}
void init(){
    memset(ch,0,sizeof(ch));
    memset(fail,0,sizeof(fail));
    memset(cnt,0,sizeof(cnt));
    len[0]=0;
    len[1]=-1;
    fail[0]=1;
    sum[0]=sum[1]=0;
    tot=1;
    last=0;
    s[0]=-1;
    p1=0;
}
void count(){
    for(int i=tot;i>=2;i--){
        cnt[fail[i]]+=cnt[i];
    }

}
int main()
{
    int i,n;
    scanf("%s",c);
    n=strlen(c);
    init();
    for(i=0;i<n;i++) c[i]=c[i]-'a';
    for(i=0;i<n;i++){
        insert(i);
    }
    long long s=0;
    count();
    for(int i=2;i<=tot;i++){
        s=max(s,(long long)cnt[i]*len[i]);
    }
    cout<<s<<endl;
}
```

55 长度不小于 k 的公共子串的个数

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<cmath>
#define N 100005
#define LL long long
#define maxn 200005
using namespace std;
//以下为倍增算法求后缀数组
int wa[maxn],wb[maxn],wv[maxn],Ws[maxn];
int cmp(int *r,int a,int b,int l)
{return r[a]==r[b]&&r[a+l]==r[b+l];}
void da(const char *r,int *sa,int n,int m){
   int i,j,p,*x=wa,*y=wb,*t;
   for(i=0;i<m;i++) Ws[i]=0;
   for(i=0;i<n;i++) Ws[x[i]=r[i]]++;
   for(i=1;i<m;i++) Ws[i]+=Ws[i-1];
   for(i=n-1;i>=0;i--) sa[--Ws[x[i]]]=i;
   for(j=1,p=1;p<n;j*=2,m=p){
      for(p=0,i=n-j;i<n;i++) y[p++]=i;
      for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
      for(i=0;i<n;i++) wv[i]=x[y[i]];
      for(i=0;i<m;i++) Ws[i]=0;
      for(i=0;i<n;i++) Ws[wv[i]]++;
      for(i=1;i<m;i++) Ws[i]+=Ws[i-1];
      for(i=n-1;i>=0;i--) sa[--Ws[wv[i]]]=y[i];
      for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
         x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
   }
   return;
}
int sa[maxn],Rank[maxn],height[maxn];
//求 height 数组
void calheight(const char *r,int *sa,int n){
   int i,j,k=0;
   for(i=1;i<=n;i++) Rank[sa[i]]=i;
   for(i=0;i<n;height[Rank[i++]]=k)
      for(k?k--:0,j=sa[Rank[i]-1];r[i+k]==r[j+k];k++);
   return;
}
char str[maxn],ch[maxn];
int k;
int s[maxn][2];
LL tot,top;
int main(){
   while(scanf("%d",&k)!=EOF&&k){
      int l1,l2;
      scanf("%s%s",str,ch);
```

```
        l1=strlen(str);l2=strlen(ch);
        str[l1]='@';
        for(int i=l1+1;i<=l1+l2;i++)
            str[i]=ch[i-l1-1];
        int n=l1+l2+1;
        str[n]='\0';
        da(str,sa,n+1,130);
        calheight(str,sa,n);
        tot=top=0;
        LL sum=0;
        for(int i=1;i<=n;i++){
            if(height[i]<k) top=tot=0;
            else{
                int cnt=0;
                if(sa[i-1]<l1) cnt++,tot+=height[i]-k+1;
                while(top>0&&height[i]<=s[top-1][0]){
                    top--;
                    tot-=s[top][1]*(s[top][0]-height[i]);
                    cnt+=s[top][1];
                }
                s[top][0]=height[i];s[top++][1]=cnt;
                if(sa[i]>l1) sum+=tot;
            }
        }
        tot=top=0;
        for(int i=1;i<=n;i++){
            if(height[i]<k) top=tot=0;
            else{
                int cnt=0;
                if(sa[i-1]>l1) cnt++,tot+=height[i]-k+1;
                while(top>0&&height[i]<=s[top-1][0]){
                    top--;
                    tot-=s[top][1]*(s[top][0]-height[i]);
                    cnt+=s[top][1];
                }
                s[top][0]=height[i];s[top++][1]=cnt;
                if(sa[i]<l1) sum+=tot;
            }
        }
        printf("%I64d\n",sum);
    }
    return 0;
}
```

56 树上莫队

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<cmath>
```

```
#include<stack>
#include<string.h>
#define maxn 100005
using namespace std;
typedef long long LL;
int deap[maxn],vis[maxn],dis[maxn],kk[20][maxn];
int belong[maxn];
int size[maxn];
int maxlog;
vector<int>g[maxn];
int blo,block;
stack<int>st;
void init(int n){
   maxlog=18;
   blo=pow(n,2.0/3)/2;
   block=0;
   memset(vis,0,sizeof(vis));
   for(int i=1;i<=n;i++) g[i].clear();
}
void dfs(int v,int p,int d)
{
   size[v]=0;
   vis[v]=1;
   deap[v]=d;
   int i,k;
   kk[0][v]=p;
   for(i=1;i<maxlog;i++){
      if(kk[i-1][v]<0)
         kk[i][v]=-1;
      else{
         kk[i][v]=kk[i-1][kk[i-1][v]];
      }
   }
   k=(int )g[v].size();
   for(i=0;i<k;i++){
      if(!vis[g[v][i]]){
         dis[g[v][i]]=dis[v]+1;
         dfs(g[v][i],v,d+1);
         size[v]+=size[g[v][i]];
         if(size[v]>=blo){
            ++block;
            while(!st.empty()){
               int p=st.top();
               st.pop();
               belong[p]=block;
            }
            size[v]=0;
         }
      }
   }
   st.push(v);
   size[v]++;
   return ;
```

```
}
int find(int a,int b)
{
    if(deap[a]>deap[b])
        swap(a,b);
    int i,f;
    f=deap[b]-deap[a];
    for(i=0;i<maxlog;i++)
    {
        if((f>>i)&1)
            b=kk[i][b];
    }
    if(b==a)
        return a;
    for(i=maxlog-1;i>=0;i--)
    {
        if(kk[i][a]!=kk[i][b])
        {
            a=kk[i][a];
            b=kk[i][b];
        }
    }
    return kk[0][a];
}
struct pi{
    int x,y;
    int a,b;
    int id;
}pp[maxn];
int cmp(pi a,pi b){
    if(belong[a.x]!=belong[b.x]) return belong[a.x]<belong[b.x];
    return belong[a.y]<belong[b.y];
}
int ans[maxn];
int c[maxn];
int a[maxn];
int an;
void venxor(int u){
    if(vis[u]){
        a[c[u]]--;
        if(a[c[u]]==0) an--;
        vis[u]=0;
    }
    else{
        if(a[c[u]]==0) an++;
        a[c[u]]++;
        vis[u]=1;
    }
}
void mov(int u,int v){
    if(deap[u]<deap[v]) swap(u,v);
    while(deap[u]>deap[v]){
        venxor(u);
```

```
        u=kk[0][u];
    }
    while(u!=v){
        venxor(u);
        venxor(v);
        u=kk[0][u];
        v=kk[0][v];
    }
}
```

57 Matrix-tree 定理

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<cmath>
#include<algorithm>
#include<vector>
#define xx first
#define yy second
#define mp(a,b) make_pair(a,b)
using namespace std;
const int N=17;
typedef  pair<int ,int>p1;
typedef long long LL;
int degree[N];
LL C[N][N];
const int mod=1000000007;
vector<p1>g[N];
int det(LL a[][N],int n)//生成树计数:Matrix-Tree 定理,度数矩阵减邻接矩阵
{
    LL ret=1;
    for(int i=1; i<n; i++)
    {
        for(int j=i+1; j<n; j++)
            while(a[j][i])
            {
                LL t=a[i][i]/a[j][i];
                for(int k=i; k<n; k++)
                    a[i][k]=(a[i][k]-a[j][k]*t)%mod;
                for(int k=i; k<n; k++)
                    swap(a[i][k],a[j][k]);
                ret=-ret;
            }
        if(a[i][i]==0)
            return 0;
        ret=ret*a[i][i];
        ret%=mod;
    }
    /*
     if(ret<0)
```

```
    ret=-ret;
    */
   return ret;
}
vector<int>gg;
int dp[1<<17];
int main()
{
   int n;
   cin>>n;
   for(int i=0;i<n-1;i++){
      int p;
      scanf("%d",&p);
      for(int j=0;j<p;j++){
         int x,y;
         scanf("%d%d",&x,&y);
         x--;
         y--;
         g[i].push_back(mp(x,y));
      }
   }
   int ans=0;
   int p=1<<(n-1);
   for(int i=1;i<p;i++){
      memset(C,0,sizeof(C));
      int tot=0;
      for(int j=0;j<(n-1);j++){
         if(i&(1<<j)){
            tot++;
            int p=g[j].size();
            for(int f=0;f<p;f++){
               C[g[j][f].xx][g[j][f].xx]++;
               C[g[j][f].yy][g[j][f].yy]++;
               C[g[j][f].xx][g[j][f].yy]--;
               C[g[j][f].yy][g[j][f].xx]--;
            }
         }
      }
      dp[i]=det(C,n);
      if((n-1-tot)%2==0) ans+=dp[i];
      else ans-=dp[i];
      ans%=mod;
   }
   ans=(ans+mod)%mod;
   cout<<ans<<endl;
   return 0;
}
```

58 FFT

```
#include<cstdio>
```

```
#include<cmath>
#include<cstring>
#include<iostream>
#include<algorithm>
using namespace std;
typedef double db;
const int MAXN=320005;
const double pi = acos(-1.0);
struct P {
    double r , i ;
    P () {}
    P ( double r , double i ) : r ( r ) , i ( i ) {}
    P operator + ( const P& p ) const {
        return P ( r + p.r , i + p.i ) ;
    }
    P operator - ( const P& p ) const {
        return P ( r - p.r , i - p.i ) ;
    }
    P operator * ( const P& p ) const {
        return P ( r * p.r - i * p.i , r * p.i + i * p.r ) ;
    }
} ;
void DFT ( P y[] , int n , int rev ) {
    for ( int i = 1 , j , k , t ; i < n ; ++ i ) {
        for ( j = 0 , t = i , k = n >> 1 ; k ; k >>= 1 , t >>= 1 ) {
            j = j << 1 | t & 1 ;
        }
        if ( i < j ) swap ( y[i] , y[j] ) ;
    }
    for ( int s = 2 , ds = 1 ; s <= n ; ds = s , s <<= 1 ) {
        P wn ( cos ( rev * 2 * pi / s ) , sin ( rev * 2 * pi / s ) ) ;
        for ( int k = 0 ; k < n ; k += s ) {
            P w ( 1 , 0 ) , t ;
            for ( int i = k ; i < k + ds ; ++ i ) {
                y[i + ds] = y[i] - ( t = w * y[i + ds] ) ;
                y[i] = y[i] + t ;
                w = w * wn ;
            }
        }
    }
}

void FFT ( P x1[] , P x2[] , int n ) {
    DFT ( x1 , n , 1 ) ;
    DFT ( x2 , n , 1 ) ;
    for ( int i = 0 ; i < n ; ++ i ) {
        x1[i] = x1[i] * x2[i] ;
```

```
    }
    DFT ( x1 , n , -1 ) ;
    for ( int i = 0 ; i < n ; ++ i ) {
        x1[i].r /= n ;
    }
}
```

59 树上莫队带修改

```
#include<stdio.h>
#include<iostream>
#include<algorithm>
#include<vector>
#include<cmath>
#include<stack>
#include<string.h>
#define maxn 100005
using namespace std;
typedef long long LL;
int deap[maxn],vis[maxn],dis[maxn],kk[20][maxn];
int belong[maxn];
int size[maxn];
int maxlog;
vector<int>g[maxn];
int blo,block;
stack<int>st;
void init(int n){
   maxlog=18;
   blo=pow(n,2.0/3)/2;
   block=0;
   memset(vis,0,sizeof(vis));
   for(int i=1;i<=n;i++) g[i].clear();
}
void dfs(int v,int p,int d)
{
   size[v]=0;
   vis[v]=1;
   deap[v]=d;
   int i,k;
   kk[0][v]=p;
   for(i=1;i<maxlog;i++){
      if(kk[i-1][v]<0)
         kk[i][v]=-1;
      else{
         kk[i][v]=kk[i-1][kk[i-1][v]];
      }
   }
   k=(int )g[v].size();
   for(i=0;i<k;i++){
```

```
        if(!vis[g[v][i]]){
            dis[g[v][i]]=dis[v]+1;
            dfs(g[v][i],v,d+1);
            size[v]+=size[g[v][i]];
            if(size[v]>=blo){
                ++block;
                while(!st.empty()){
                    int p=st.top();
                    st.pop();
                    belong[p]=block;
                }
                size[v]=0;
            }
        }
    }
    st.push(v);
    size[v]++;
    return ;
}
int find(int a,int b)
{
    if(deap[a]>deap[b])
        swap(a,b);
    int i,f;
    f=deap[b]-deap[a];
    for(i=0;i<maxlog;i++)
    {
        if((f>>i)&1)
            b=kk[i][b];
    }
    if(b==a)
        return a;
    for(i=maxlog-1;i>=0;i--)
    {
        if(kk[i][a]!=kk[i][b])
        {
            a=kk[i][a];
            b=kk[i][b];
        }
    }
    return kk[0][a];
}
struct pi{
    int x,y;
    int id;
    int no;
}pp[maxn];
int cmp(pi a,pi b){
    if(belong[a.x]==belong[b.x]&&belong[a.y]==belong[b.y]) return a.no<b.no;
    if(belong[a.x]!=belong[b.x]) return belong[a.x]<belong[b.x];
    return belong[a.y]<belong[b.y];
}
LL ans[maxn];
```

```
int a[maxn];
int c[maxn];
int v[maxn];
int pre[maxn];
int w[maxn];
LL an;
void venxor(int u){
    if(vis[u]){
        vis[u]=0;
        an-=((LL)v[c[u]]*w[a[c[u]]]);
        a[c[u]]--;
    }
    else{
        vis[u]=1;
        a[c[u]]++;
        an+=((LL)v[c[u]]*w[a[c[u]]]);
    }
}
void chan(int x,int y){
    if(vis[x]){
        venxor(x);
        c[x]=y;
        venxor(x);
    }
    else{
        c[x]=y;
    }
}
void mov(int u,int v){
    if(deap[u]<deap[v]) swap(u,v);
    while(deap[u]>deap[v]){
        venxor(u);
        u=kk[0][u];
    }
    while(u!=v){
        venxor(u);
        venxor(v);
        u=kk[0][u];
        v=kk[0][v];
    }
}
struct change{
    int x;
    int p;
    int pre;
}pp1[maxn];
int main()
{
    int n,m,q;
    cin>>n>>m>>q;
    init(n);
    for(int i=1;i<=m;i++){
        scanf("%d",&v[i]);
```

```
}
for(int i=1;i<=n;i++) scanf("%d",&w[i]);
for(int i=1;i<n;i++){
    int x,y;
    scanf("%d%d",&x,&y);
    g[x].push_back(y);
    g[y].push_back(x);
}
for(int i=1;i<=n;i++) scanf("%d",&c[i]);
dfs(1,-1,0);
while(!st.empty()){
    int p=st.top();
    st.pop();
    belong[p]=block;
}
memset(vis,0,sizeof(vis));
int x=0,y=0;
for(int i=1;i<=n;i++) pre[i]=c[i];
for(int i=0;i<q;i++){
    int p,a,b;
    scanf("%d%d%d",&p,&a,&b);
    if(p==0){
        ++y;
        pp1[y].x=a;
        pp1[y].p=b;
        pp1[y].pre=pre[a];
        pre[a]=b;
    }
    else{
        x++;
        pp[x].x=a;
        pp[x].y=b;
        pp[x].id=x;
        pp[x].no=y;
    }
}
sort(pp+1,pp+1+x,cmp);
for(int i=1;i<=x;i++){
    if(i==1){
        mov(pp[i].x,pp[i].y);
        for(int j=1;j<=pp[i].no;j++){
            chan(pp1[j].x,pp1[j].p);
        }
    }
    else{
        mov(pp[i-1].x,pp[i].x);
        mov(pp[i-1].y,pp[i].y);
        if(pp[i].no!=pp[i-1].no){
            if(pp[i].no>pp[i-1].no){
                for(int j=pp[i-1].no+1;j<=pp[i].no;j++){
                    chan(pp1[j].x,pp1[j].p);
                }
            }
```

```
            else{
                for(int j=pp[i-1].no;j>pp[i].no;j--){
                    chan(pp1[j].x,pp1[j].pre);
                }
            }
        }
    }
    int p=find(pp[i].x,pp[i].y);
    venxor(p);
    ans[pp[i].id]=an;
    venxor(p);
  }
  for(int i=1;i<=x;i++){
    printf("%lld\n",ans[i]);
  }
}
```

60 原始对偶

```
#include <cstdio>
#include <cstring>
#include <deque>
#include <algorithm>
using namespace std;

const int V=440, E=V*2, maxint=0x3F3F3F3F;

struct etype
{
  int t, c, u;
  etype *next, *pair;
  etype() {}
  etype(int T, int C, int U, etype* N): t(T), c(C), u(U), next(N) {}
  void* operator new(unsigned, void* p){return p;}
} *e[V], Te[E+E], *Pe;

int S, T, n, piS, cost;
bool v[V];

void addedge(int s, int t, int c, int u)
{
  e[s] = new(Pe++) etype(t, +c, u, e[s]);
  e[t] = new(Pe++) etype(s, -c, 0, e[t]);
  e[s]->pair = e[t];
  e[t]->pair = e[s];
}

int aug(int no, int m)
{
  if (no == T) return cost += piS * m, m;
  v[no] = true;
```

```
    int l = m;
    for (etype *i = e[no]; i; i = i->next)
        if (i->u && !i->c && !v[i->t])
        {
            int d = aug(i->t, l < i->u ? l : i->u);
            i->u -= d, i->pair->u += d, l -= d;
            if (!l) return m;
        }
    return m - l;
}

bool modlabel()
{
    static int d[V]; memset(d, 0x3F, sizeof(d)); d[T] = 0;
    static deque<int> Q; Q.push_back(T);
    while(Q.size())
    {
        int dt, no = Q.front(); Q.pop_front();
        for(etype *i = e[no]; i; i = i->next)
            if(i->pair->u && (dt = d[no] - i->c) < d[i->t])
                (d[i->t] = dt) <= d[Q.size() ? Q.front() : 0]
                ? Q.push_front(i->t) : Q.push_back(i->t);
    }
    for(int i = 0; i < n; ++i)
        for(etype *j = e[i]; j; j = j->next)
            j->c += d[j->t] - d[i];
    piS += d[S];
    return d[S] < maxint;
}

int ab[V], *pab[V], w[V];

struct lt
{
    bool operator()(int* p1,int* p2) {return *p1 < *p2;}
};

int main()
{
    int t;
    scanf("%d",&t);
    while(t--)
    {
        memset(e,0,sizeof(e));
        Pe = Te;
        static int m, k;
        scanf("%d %d", &m, &k);
        int abz = 0;
        for(int i = 0; i < m; ++i)
        {
            scanf("%d", pab[abz] = &ab[abz]), abz++;
            scanf("%d", pab[abz] = &ab[abz]), abz++;
            scanf("%d", &w[i]);
```

```
    }
    sort(&pab[0], &pab[abz], lt());
    int c=0xDEADBEEF; n=0;
    for(int i = 0; i < abz; ++i)
    {
      if(c != *pab[i]) c = *pab[i], ++n;
      *pab[i] = n;
    }
    ++n, S = 0, T = n++;
    for(int i = 0; i < T; ++i) addedge(i, i+1, 0, k);
    for(int i = 0; i < m; ++i) addedge(ab[i+i], ab[i+i+1], -w[i], 1);
    piS = cost = 0;
    while(modlabel())
      do memset(v, 0, sizeof(v));
    while(aug(S, maxint));
    printf("%d\n", -cost);
  }
  return 0;
}
```

61 Hopcroft-Carp

```
const int MAXN=3000;
const int INF=1<<28;
int g[MAXN][MAXN],Mx[MAXN],My[MAXN],Nx,Ny;
int dx[MAXN],dy[MAXN],dis;
bool vst[MAXN];
bool searchP()
{
    queue<int>Q;
    dis=INF;
    memset(dx,-1,sizeof(dx));
    memset(dy,-1,sizeof(dy));
    for(int i=0;i<Nx;i++)
        if(Mx[i]==-1)
        {
            Q.push(i);
            dx[i]=0;
        }
    while(!Q.empty())
    {
        int u=Q.front();
        Q.pop();
        if(dx[u]>dis)  break;
        for(int v=0;v<Ny;v++)
            if(g[u][v]&&dy[v]==-1)
            {
```

```
                dy[v]=dx[u]+1;
                if(My[v]==-1)  dis=dy[v];
                else
                {
                    dx[My[v]]=dy[v]+1;
                    Q.push(My[v]);
                }
            }
    }
    return dis!=INF;
}
bool DFS(int u)
{
    for(int v=0;v<Ny;v++)
        if(!vst[v]&&g[u][v]&&dy[v]==dx[u]+1)
        {
            vst[v]=1;
            if(My[v]!=-1&&dy[v]==dis) continue;
            if(My[v]==-1||DFS(My[v]))
            {
                My[v]=u;
                Mx[u]=v;
                return 1;
            }
        }
    return 0;
}
int MaxMatch()
{
    int res=0;
    memset(Mx,-1,sizeof(Mx));
    memset(My,-1,sizeof(My));
    while(searchP())
    {
        memset(vst,0,sizeof(vst));
        for(int i=0;i<Nx;i++)
            if(Mx[i]==-1&&DFS(i))  res++;
    }
    return res;
}
```

62 常用外挂

```
#include<cstdio>
#include<cstring>
```

```
#include<algorithm>
#include<cctype>
using namespace std;

//C++¿©'ª
#pragma comment(linker, "/STACK:1024000000,1024000000")

//ð¡»Î π "
inline char getc(){
        static const int BUFLEN = 1 << 15;
        static char B[BUFLEN], *S = B, *T = B;
        if(S==T) S=B, T=S+fread(B, 1, BUFLEN, stdin);
        return (S==T) ? 0 : *(S ++);
}
int ReadInt(){
        char ch; int aa=0;
        do ch = getc(); while(!isdigit(ch));
        do aa = aa*10+ch-'0', ch = getc();
        while(isdigit(ch));
        return aa;
}

//O(N)≈≈ - Ú+¿Î…¢ª Ø
int Distinct(ul arr[], const int N){
        int i, n2;
        static int cnt[0x10000], tab1[MAXN], tab2[MAXN];
        static ul arr2[MAXN];

        //µ/ "ª ¬÷
        for(i = 0; i <= 0xFFFF; i ++) cnt[i] = 0;
        for(i = 0; i < N; i ++) cnt[arr[i] & 0xFFFF] ++;
        for(i = 0; i < 0xFFFF; i ++) cnt[i + 1] += cnt[i];
        for(i = N - 1; i >= 0; i --){
            tab1[i] = -- cnt[arr[i] & 0xFFFF];
            arr2[tab1[i]] = arr[i];
        }
        //µ/ð¸¬÷
        for(i = 0; i <= 0xFFFF; i ++) cnt[i] = 0;
        for(i = 0; i < N; i ++) cnt[arr2[i] >> 16] ++;
        for(i = 0; i < 0xFFFF; i ++) cnt[i + 1] += cnt[i];
        for(i = N - 1; i >= 0; i --){
            tab2[i] = -- cnt[arr2[i] >> 16];
            arr[tab2[i]] = arr2[i];
        }
        //÷µ¿Î…¢ª Ø
        arr2[0] = n2 = 1;
        for(i = 1; i < N; i ++){
```

```
            if(arr[i] != arr[i - 1]) ++ n2;
            arr2[i] = n2;
        }
        //∏¥ ‘≠  ″◊È
        for(i = 0; i < N; i ++) arr[i] = arr2[tab2[tab1[i]]];
        return n2;
}

//¥Û  ″≥À∑®»°  ƒ£
//»Á π˚  «64Œª “™–¥imulq,idivq ∫Õr?x
#define MO 1000000007
#ifdef _MSC_VER
inline int ml(int a,int b){
        int ret;
        _asm{
            mov eax, a;
            mov ebx, b;
            mov ecx, MO;
            imul ebx;
            idiv ecx;
            mov ret, edx;
        }
        return ret;
}
#else
inline int ml(int a,int b){
        int ret;
        __asm__ __volatile__
("\timull %%ebx\n\tidivl %%ecx\n" :"=d"(ret):"a"(a),"b"(b),"c"(MO));
        return ret;
}
#endif
```