# Template V1.23

Judge

September 23, 2016

## 目录

**1 DLX** **1**
1.1 精确覆盖 DLX . . . . . 1
1.2 重复覆盖 DLX . . . . . 1

**2 Gauss** **2**
2.1 高斯 double . . . . . 2
2.2 高斯消元 . . . . . 2
2.3 高斯消元 mod . . . . . 3
2.4 高斯消元 xor . . . . . 4

**3 NTT** **5**
3.1 FFT . . . . . 5
3.2 NTT . . . . . 5
3.3 分治 +fft . . . . . 6

**4 math** **7**
4.1 CRT . . . . . 7
4.2 LL 黑科技 . . . . . 7
4.3 about prime . . . . . 7
4.4 ext_gcd . . . . . 8
4.5 inv(gcd) . . . . . 8
4.6 phi . . . . . 8

**5 spfa** **8**
5.1 Spfa 最短路 . . . . . 8

**6 划分树** **9**
6.1 划分树比 k 小数量 . . . . . 9
6.2 划分树第 k 大 . . . . . 9

**7 匈牙利** **10**
7.1 二分匹配 . . . . . 10

**8 矩阵** **10**
8.1 矩阵 . . . . . 10

**9 组合** **11**
9.1 组合数求 mod . . . . . 11

**10 网络流** **11**
10.1 MinCost_bhb . . . . . 11
10.2 dinic_bhb . . . . . 11

# 1 DLX

## 1.1 精确覆盖 DLX

```
using namespace std;
#define N 1111111
#define M 8111111
const int MaxM = 1111;
const int MaxN = 1111;
const int maxnode = MaxN * MaxM;
int flag;
struct DLX
{
    int n,m,size;
    int U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],Col[maxnode];
    int H[MaxN],S[MaxM];
    int ansd;
    void init(int _n,int _m)
    {
        n = _n;
        m = _m;
        for(int i = 0;i <= m;i++)
        {
            S[i] = 0;
            U[i] = D[i] = i;
            L[i] = i-1;
            R[i] = i+1;
        }
        R[m] = 0; L[0] = m;
        size = m;
        for(int i = 1;i <= n;i++)H[i] = -1;
    }
    void Link(int r,int c)
    {
        ++S[Col[++size]=c];
        Row[size] = r;
        D[size] = D[c];
        U[D[c]] = size;
        U[size] = c;
        D[c] = size;
        if(H[r] < 0)H[r] = L[size] = R[size] = size;
        else
        {
            R[size] = R[H[r]];
            L[R[H[r]]] = size;
            L[size] = H[r];
            R[H[r]] = size;
        }
    }
    void remove(int c)
    {
        L[R[c]] = L[c]; R[L[c]] = R[c];
        for(int i = D[c];i != c;i = D[i])
            for(int j = R[i];j != i;j = R[j])
            {
                U[D[j]] = U[j];
                D[U[j]] = D[j];
                --S[Col[j]];
            }
    }
    void resume(int c)
    {
        for(int i = U[c];i != c;i = U[i])
            for(int j = L[i];j != i;j = L[j])
                ++S[Col[U[D[j]]=D[U[j]]=j]];
        L[R[c]] = R[L[c]] = c;
    }
    int ans[MaxN];
    void Dance(int d)
    {
        if (flag) return;
        if(R[0] == 0)
        {
            flag=1;
            printf("%d ",d);
            rep(i,0,d)
            printf("%d%c",ans[i],i==d-1?'\n':' ');
            return;
        }
        int c = R[0];
        for(int i = R[0];i != 0;i = R[i])
            if(S[i] < S[c])
                c = i;
        remove(c);
        for(int i = D[c];i != c;i = D[i])
        {
            for(int j = R[i];j != i;j = R[j]) remove(Col[j]);
            ans[d]=Row[i];
            Dance(d+1);
            for(int j = L[i];j != i;j = L[j]) resume(Col[j]);
        }
        resume(c);
    }
}dlx;
int main()
{
  int n,m;
  while (~scanf("%d%d",&n,&m))
  {
    dlx.init(n,m);
    rep(i,0,n)
    {
      int k,x;
      scan(&k);
      rep(j,0,k)
      {
        scan(&x);
        dlx.Link(i+1,x);
      }
    }
    flag=0;
    dlx.Dance(0);
    if (!flag) puts("NO");
  }
    return 0;
}
/*
6 5
1 2
2 3
3 4
4 5
3 5
2 4 5 6
*/
```

## 1.2 重复覆盖 DLX

```
const int MaxM = 15*15+10;
const int MaxN = 15*15+10;
const int maxnode = MaxN * MaxM;
const int INF = 0x3f3f3f3f;
struct DLX
{
    int n,m,size;
    int U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],Col[maxnode];
    int H[MaxN],S[MaxM];
    int ansd;
    void init(int _n,int _m)
    {
        n = _n;
        m = _m;
        for(int i = 0;i <= m;i++)
        {
            S[i] = 0;
            U[i] = D[i] = i;
            L[i] = i-1;
            R[i] = i+1;
        }
        R[m] = 0; L[0] = m;
        size = m;
        for(int i = 1;i <= n;i++)H[i] = -1;
    }
    void Link(int r,int c)
    {
        ++S[Col[++size]=c];
        Row[size] = r;
        D[size] = D[c];
        U[D[c]] = size;
        U[size] = c;
        D[c] = size;
```

```
        if(H[r] < 0)H[r] = L[size] = R[size] = size;
        else
        {
            R[size] = R[H[r]];
            L[R[H[r]]] = size;
            L[size] = H[r];
            R[H[r]] = size;
        }
    }
    void remove(int c)
    {
        for(int i = D[c];i != c;i = D[i])
            L[R[i]] = L[i], R[L[i]] = R[i];
    }
    void resume(int c)
    {
        for(int i = U[c];i != c;i = U[i])
            L[R[i]] = R[L[i]] = i;
    }
    bool v[MaxM];
    int f()
    {
        int ret = 0;
        for(int c = R[0]; c != 0;c = R[c])v[c] = true;
        for(int c = R[0]; c != 0;c = R[c])
            if(v[c])
            {
                ret++;
                v[c] = false;
                for(int i = D[c];i != c;i = D[i])
                    for(int j = R[i];j != i;j = R[j])
                        v[Col[j]] = false;
            }
        return ret;
    }
    void Dance(int d)
    {
        if(d + f() >= ansd)return;
        if(R[0] == 0)
        {
            if(d < ansd)ansd = d;
            return;
        }
        int c = R[0];
        for(int i = R[0];i != 0;i = R[i])
            if(S[i] < S[c])
                c = i;
        for(int i = D[c];i != c;i = D[i])
        {
            remove(i);
            for(int j = R[i];j != i;j = R[j])remove(j);
            Dance(d+1);
            for(int j = L[i];j != i;j = L[j])resume(j);
            resume(i);
        }
    }
};
```

# 2 Gauss

## 2.1 高斯 double

```
///高斯消元模板
const double eps = 1e-10;    ///精度
double Aug[N][N]; ///增广矩阵
bool free_x[N];          ///判断是否是不确定的变元
double x[N];             ///解集

int sign(double x){ return (x>eps) - (x<-eps);}
int Gauss(int n,int m)//n 变元数量，m 方程数目
{
    memset(x,0,sizeof(x));
    memset(free_x,1,sizeof(free_x));
    int row,col,max_r;
    for(row=0,col=0; row<m&&col<n; row++,col++)
    {
        max_r = row;
        for(int i = row+1; i < m; i++)
        {///找到当前列所有行中的最大值(做除法时减小误差)
            if(sign(fabs(Aug[i][col]) - fabs(Aug[max_r][
            col])) > 0)
                max_r = i;
        }
        if(max_r != row){///将该行与当前行交换
            for(int j = row; j < n+1; j++)
                swap(Aug[max_r][j],Aug[row][j]);
        }
        if(sign(Aug[row][col])==0)
        {///当前列row行以下全为0(包括row行)
            row--;
            continue;
        }
        for(int i = row+1; i < m; i++)
        {
            if(sign(Aug[i][col])==0)
                continue;
            double ta = Aug[i][col]/Aug[row][col];
            for(int j = col; j < n+1; j++)
                Aug[i][j] -= Aug[row][j]*ta;
        }
    }
    ///无解或者多个解的情况
    for(int i = row; i < m; i++)
    {
        if(sign(Aug[i][col]))
            return -1;///col=n存在0...0,a的情况,无解
    }
    if(row < n)
    {
        for(int i = row-1; i >=0; i--)
        {
            int free_num = 0;   ///自由变元的个数
            int free_index;     ///自由变元的序号
            for(int j = 0; j < n; j++)
            {
                if(sign(Aug[i][j])!=0 && free_x[j])
                    free_num++,free_index=j;
            }
            if(free_num > 1) continue;  ///该行中的不确定
            的变元的个数超过1个,无法求解,它们仍然为不确定
            的变元
            ///只有一个不确定的变元free_index,可以求解出该
            变元,且该变元是确定的
            double tmp = Aug[i][n];
            for(int j = 0; j < n; j++)
            {
                if(sign(Aug[i][j])!=0 && j!=free_index)
                    tmp -= Aug[i][j]*x[j];
            }
            x[free_index] = tmp/Aug[i][free_index];
            free_x[free_index] = false;
        }
        return n-row;///存在0...0,0的情况,有多个解,自由变
        元个数为n-row个
    }
    ///无解或者多个解的情况
    for(int i = n-1; i >= 0; i--)
    {
        double tmp = Aug[i][n];
        for(int j = i+1; j < n; j++)
            if(sign(Aug[i][j])!=0)
                tmp -= Aug[i][j]*x[j];
        x[i] = tmp/Aug[i][i];
    }
    return 0;///有且仅有一个解,严格的上三角矩阵(n==m)
}
```

## 2.2 高斯消元

```
const int MAXN=50;
namespace gaosi
{
    int a[MAXN][MAXN];//增广矩阵
    int x[MAXN];//解集
    bool free_x[MAXN];//标记是否是不确定的变元

    /*
    void Debug(void)
    {
        int i, j;
        for (i = 0; i < equ; i++)
        {
            for (j = 0; j < var + 1; j++)
            {
                cout << a[i][j] << " ";
```

```
            }
            cout << endl;
        }
        cout << endl;
    }
    */
void init(){memset(a, 0, sizeof(a));};
    inline int gcd(int a,int b)
    {
        int t;
        while(b!=0)
        {
            t=b;
            b=a%b;
            a=t;
        }
        return a;
    }
    inline int lcm(int a,int b)
    {
        return a/gcd(a,b)*b;//先除后乘防溢出
    }

    // 高斯消元法解方程组(Gauss-Jordan elimination).(-2表示有浮点数解，但无整数解，
    //-1表示无解，0表示唯一解，大于0表示无穷解，并返回自由变元的个数)
    //有equ个方程，var个变元。增广矩阵行数为equ,分别为0到equ-1,列数为var+1,分别为0到var.
    int Gauss(int equ,int var)
    {
        int i,j,k;
        int max_r;// 当前这列绝对值最大的行.
        int col;//当前处理的列
        int ta,tb;
        int LCM;
        int temp;
        int free_x_num;
        int free_index;

        for(int i=0;i<=var;i++)
        {
            x[i]=0;
            free_x[i]=true;
        }

        //转换为阶梯阵.
        col=0; // 当前处理的列
        for(k = 0;k < equ && col < var;k++,col++)
        {// 枚举当前处理的行.
    // 找到该col列元素绝对值最大的那行与第k行交换.(为了在除法时减小误差)
            max_r=k;
            for(i=k+1;i<equ;i++)
            {
                if(abs(a[i][col])>abs(a[max_r][col]))
                max_r=i;
            }
            if(max_r!=k)
            {// 与第k行交换.
                for(j=k;j<var+1;j++) swap(a[k][j],a[max_r][j]);
            }
            if(a[k][col]==0)
            {// 说明该col列第k行以下全是0了，则处理当前行的下一列.
                k--;
                continue;
            }
            for(i=k+1;i<equ;i++)
            {// 枚举要删去的行.
                if(a[i][col]!=0)
                {
                    LCM = lcm(abs(a[i][col]),abs(a[k][col]));
                    ta = LCM/abs(a[i][col]);
                    tb = LCM/abs(a[k][col]);
                    if(a[i][col]*a[k][col]<0)tb=-tb;//异号的情况是相加
                    for(j=col;j<var+1;j++)
                    {
                        a[i][j] = a[i][j]*ta-a[k][j]*tb;
                    }
                }
            }
        }

      //  Debug();

        // 1. 无解的情况: 化简的增广阵中存在(0, 0, ..., a)这样的行(a != 0).
        for (i = k; i < equ; i++)
        { // 对于无穷解来说，如果要判断哪些是自由变元，那么初等行变换中的交换就会影响，则要记录交换.
            if (a[i][col] != 0) return -1;
        }
        // 2. 无穷解的情况: 在var * (var + 1)的增广阵中出现(0, 0, ..., 0)这样的行，即说明没有形成严格的上三角阵.
        // 且出现的行数即为自由变元的个数.
        if (k < var)
        {
            // 首先，自由变元有var - k个，即不确定的变元至少有var - k个.
            for (i = k - 1; i >= 0; i--)
            {
                // 第i行一定不会是(0, 0, ..., 0)的情况，因为这样的行是在第k行到第equ行.
                // 同样，第i行一定不会是(0, 0, ..., a), a != 0的情况，这样的无解的.
                free_x_num = 0; // 用于判断该行中的不确定的变元的个数，如果超过1个，则无法求解，它们仍然为不确定的变元.
                for (j = 0; j < var; j++)
                {
                    if (a[i][j] != 0 && free_x[j])
                    free_x_num++, free_index = j;
                }
                if (free_x_num > 1) continue; // 无法求解出确定的变元.
                // 说明就只有一个不确定的变元free_index，那么可以求解出该变元，且该变元是确定的.
                temp = a[i][var];
                for (j = 0; j < var; j++)
                {
                    if (a[i][j] != 0 && j != free_index)
                    temp -= a[i][j] * x[j];
                }
                x[free_index] = temp / a[i][free_index];
                // 求出该变元.
                free_x[free_index] = 0; // 该变元是确定的.
            }
            return var - k; // 自由变元有var - k个.
        }
        // 3. 唯一解的情况: 在var * (var + 1)的增广阵中形成严格的上三角阵.
        // 计算出Xn-1, Xn-2 ... X0.
        for (i = var - 1; i >= 0; i--)
        {
            temp = a[i][var];
            for (j = i + 1; j < var; j++)
            {
                if (a[i][j] != 0) temp -= a[i][j] * x[j];
            }
            if (temp % a[i][i] != 0) return -2; // 说明有浮点数解，但无整数解.
            x[i] = temp / a[i][i];
        }
        return 0;
    }
}
```

## 2.3 高斯消元 mod

```
const int MOD= 7;
const int MAXN=400;
int a[MAXN][MAXN];//增广矩阵
int x[MAXN];//解集
bool free_x[MAXN];//标记是否是不确定的变元

inline int gcd(int a,int b)
{
    int t;
    while(b!=0)
    {
        t=b;
```

```
        b=a%b;
        a=t;
    }
    return a;
}
inline int lcm(int a,int b)
{
    return a/gcd(a,b)*b;//先除后乘防溢出
}

// 高斯消元法解方程组(Gauss-Jordan elimination).(-2表示有浮点数解，但无整数解，
//-1表示无解，0表示唯一解，大于0表示无穷解，并返回自由变元的个数)
//有equ个方程，var个变元。增广矩阵行数为equ,分别为0到equ-1,列数为var+1,分别为0到var.
int Gauss(int equ,int var)
{
    int i,j,k;
    int max_r;// 当前这列绝对值最大的行.
    int col;//当前处理的列
    int ta,tb;
    int LCM;
    int temp;
    int free_x_num;
    int free_index;

    for(int i=0;i<=var;i++)
    {
        x[i]=0;
        free_x[i]=true;
    }
    col=0;
    for(k = 0;k < equ && col < var;k++,col++)
    {
        max_r=k;
        for(i=k+1;i<equ;i++)
          if(abs(a[i][col])>abs(a[max_r][col])) max_r=i;
        if(max_r!=k)
          for(j=k;j<var+1;j++) swap(a[k][j],a[max_r][j]);

        if(a[k][col]==0)
        {
            k--;
            continue;
        }
        for(i=k+1;i<equ;i++)
        {
            if(a[i][col]!=0)
            {
                LCM = lcm(abs(a[i][col]),abs(a[k][col]));
                ta = LCM/abs(a[i][col]);
                tb = LCM/abs(a[k][col]);
                if(a[i][col]*a[k][col]<0)tb=-tb;//异号的情况是相加
                for(j=col;j<var+1;j++)
                {
                    a[i][j] = ((a[i][j]*ta-a[k][j]*tb)%MOD+MOD)%MOD;
                }
            }
        }
    }


    for (i = k; i < equ; i++)
    {
        if ( a[i][col]  != 0) return -1;//无解
    }
    if (k < var)
    {
        // 首先，自由变元有var - k个，即不确定的变元至少有var - k个.
        for (i = k - 1; i >= 0; i--)
        {
            // 第i行一定不会是(0, 0, ..., 0)的情况，因为这样的行是在第k行到第equ行.
            // 同样，第i行一定不会是(0, 0, ..., a), a != 0 的情况，这样的无解的.
            free_x_num = 0; // 用于判断该行中的不确定的变元的个数，如果超过1个，则无法求解，它们仍然为不确定的变元.
            for (j = 0; j < var; j++)
            {
                if (a[i][j] != 0 && free_x[j]) free_x_num++, free_index = j;
            }
            if (free_x_num > 1) continue; // 无法求解出确定的变元.
            // 说明就只有一个不确定的变元free_index，那么可以求解出该变元，且该变元是确定的.
            temp = a[i][var];
            for (j = 0; j < var; j++)
            {
                if (a[i][j] != 0 && j != free_index) temp -= a[i][j] * x[j]%MOD;
                temp=(temp%MOD+MOD)%MOD;
            }
            x[free_index] = (temp / a[i][free_index])%MOD; // 求出该变元.
            free_x[free_index] = 0; // 该变元是确定的.
        }
        return var - k; // 自由变元有var - k个.
    }
    for (i = var - 1; i >= 0; i--)
    {
        temp = a[i][var];
        for (j = i + 1; j < var; j++)
        {
            if (a[i][j] != 0) temp -= a[i][j] * x[j];
            temp=(temp%MOD+MOD)%MOD;
        }
        while (temp % a[i][i] != 0) temp+=MOD;
        x[i] =( temp / a[i][i])%MOD ;
    }
    return 0;
}
```

## 2.4 高斯消元 xor

```
int equ,var;
int a[110][110];
int x[110];
int free_x[110];
int free_num;

//返回值为-1表示无解，为0是唯一解，否则返回自由变元个数
int Gauss()
{
    int max_r, col, k;
    free_num = 0;
    for(k = 0, col = 0; k < equ && col < var; k++, col++)
    {
        max_r = k;
        for(int i = k+1 ; i < equ; i++)
            if(abs(a[i][col]) > abs(a[max_r][col]))
                max_r = i;
        if(a[max_r][col] == 0)
        {
            k--;
            free_x[free_num++] = col; //自由变元
            continue;
        }
        if(max_r != k)
        {
            for(int j = col; j < var+1; j++)
                swap(a[k][j],a[max_r][j]);
        }
        for(int i = k+1; i < equ;i++)
            if(a[i][col] != 0)
                for(int j = col; j < var+1;j++)
                    a[i][j] ^= a[k][j];
    }
    for(int i = k;i < equ;i++)
        if(a[i][col] != 0)
            return -1;
    if(k < var)
    {//多解求最小1个数
            int ans = INF;
            int t=var-k;
            int tot = (1<<t);
            for(int i = 0;i < tot;i++)
            {
                int cnt = 0;
                for(int j = 0;j < t;j++)
                {
```

```
                    if(i&(1<<j))
                    {
                        x[free_x[j]] = 1;
                        cnt++;
                    }
                    else x[free_x[j]] = 0;
                }
                for(int j = var-t-1;j >= 0;j--)
                {
                    int idx;
                    for(idx = j;idx < var;idx++)
                        if(a[j][idx])
                            break;
                    x[idx] = a[j][var];
                    for(int l = idx+1;l < var;l++)
                        if(a[j][l])
                            x[idx] ^= x[l];
                    cnt += x[idx];
                }
                ans = min(ans,cnt);
            }
            printf("%d\n",ans);

        return var-k;
    }
    for(int i = var-1; i >= 0;i--)
    {
        x[i] = a[i][var];
        for(int j = i+1; j < var;j++)
            x[i] ^= (a[i][j] && x[j]);
    }
    return 0;
}
```

# 3 NTT

## 3.1 FFT

```
#include <bits/stdc++.h>
#define maxn 400200
#define mod 313
#define PI acos(-1.0)//acosl(-1.0)
using namespace std;
typedef double LD;//long double
typedef long long LL;
//typedef complex<LD> cpx;
struct cpx
{
  LD x, y;
  cpx(){}
  cpx(LD x, LD y):x(x),y(y){}
};
cpx operator +(cpx a, cpx b)
{
  return cpx(a.x+b.x, a.y+b.y);
}
cpx operator -(cpx a, cpx b)
{
  return cpx(a.x-b.x, a.y-b.y);
}
cpx operator *(cpx a, cpx b)
{
  return cpx(a.x*b.x-a.y*b.y, a.x*b.y+a.y*b.x);
}

int rev(int x, int n)
{
    int tmp=0;
    for (int i=n>>1;i;i>>=1,x>>=1)
        tmp=tmp<<1|x&1;
    return tmp;
}

void fft(cpx *a, int n, int flag)
{
  for (int i=0,j=i;i<n;i++,j=rev(i, n))
    if (i<j) swap(a[i], a[j]);
  for (int k=1;k<n;k<<=1)
  {
    cpx wn(cos(PI/k), flag*sin(PI/k));
    //cpx wn(cosl(PI/i), flag*sinl(PI/i));
    cpx w(1, 0);
    for (int i=0;i<k;i++,w=w*wn)
      for (int j=i;j<n;j+=(k<<1))
      {
        cpx x=a[j], y=w*a[j|k];
        a[j]=x+y;
        a[j|k]=x-y;
      }
  }
  if (flag==-1)
    for (int i=0;i<n;i++)
      a[i].x/=n, a[i].y/=n;
}

cpx A[maxn], B[maxn];
int a[maxn], b[maxn], c[maxn];
int n;
void roll(int *a, int *b, int *c, int n, int m)
{
  int num=1;
  while (num<n+m) num<<=1;//move out if slow
  for (int i=0;i<num;i++) A[i]=(i<n)?cpx(a[i],0):cpx(0,0);
  for (int i=0;i<num;i++) B[i]=(i<m)?cpx(b[i],0):cpx(0,0);
  fft(A, num, 1);
  fft(B, num, 1);
  for (int i=0;i<num;i++) A[i]=A[i]*B[i];
  fft(A, num, -1);
  for (int i=0;i<num;i++) c[i]=(LL)(A[i].x+0.5)%mod;
}
```

## 3.2 NTT

```
LL quick_mod(LL a, LL b, LL m)
{
    LL ans = 1;
    a %= m;
    while(b)
    {
        if(b & 1) ans = ans * a % m;
        b >>= 1;
        a = a * a % m;
    }
    return ans;
}
namespace ntt{
    const int N = (1 << 19)+20;
    const int P = 998244353;
    const int G = 3;
    const int NUM = 25;

    LL  wn[NUM];
    LL  a[N], b[N];
    LL mul(LL x,LL y)
    {
        return (x*y-(LL)(x/(long double)P*y+1e-3)*P+P)%P;
    }

    void GetWn()
    {
        for(int i = 0; i < NUM; i++)
        {
            int t = 1 << i;
            wn[i] = quick_mod(G, (P - 1) / t, P);
        }
    }


    void Rader(LL a[], int len)
    {
        int j = len >> 1;
        for(int i = 1; i < len - 1; i++)
        {
            if(i < j) swap(a[i], a[j]);
            int k = len >> 1;
            while(j >= k)
            {
                j -= k;
                k >>= 1;
            }
            if(j < k) j += k;
        }
    }

    void NTT(LL a[], int len, int on)
```

```
    {
        Rader(a, len);
        int id = 0;
        for(int h = 2; h <= len; h <<= 1)
        {
            id++;
            for(int j = 0; j < len; j += h)
            {
                LL w = 1;
                for(int k = j; k < j + h / 2; k++)
                {
                    LL u = a[k] % P;
                    LL t = w * a[k + h / 2] % P;
                    a[k] = (u + t) % P;
                    a[k + h / 2] = (u - t + P) % P;
                    w = w * wn[id] % P;
                }
            }
        }
        if(on == -1)
        {
            for(int i = 1; i < len / 2; i++)
                swap(a[i], a[len - i]);
            LL inv = quick_mod(len, P - 2, P);
            for(int i = 0; i < len; i++)
                a[i] = a[i] * inv % P;
        }
    }

    void Conv(LL a[], LL b[], int n)
    {
        NTT(a, n, 1);
        NTT(b, n, 1);
        for(int i = 0; i < n; i++)
            a[i] = a[i] * b[i] % P;
        NTT(a, n, -1);
    }
    void workNTT(LL a[], LL b[],int L1,int L2,int &len)
    {
        GetWn();
        len = 1;
        while(len < 2 * L1 || len <2 * L2) len <<= 1;
        rep(i,L1,len) a[i] = 0;
        rep(i,L2,len) b[i] = 0;
        Conv(a, b, len);
    }
    //////////////////////找关于P的G(原根)
    /////////////////
    void deal()
    {
        int a[100]={0};
        int n=P-1;
        int x=sqrt(n)+1;
        a[0]=1;
        rep(i,2,x)
        {
            while (n%i==0) a[a[0]]=i,n/=i;
            if (a[a[0]]) a[0]++;
        }
        if (n!=1) a[a[0]++]=n;
        n=P;
        int t=1;
        while (t++)
        {
            int flag=0;
            rep(i,1,a[0])
            if (quick_mod(t,(n-1)/a[i],n)==1) flag=1;
            if (!flag) break;
        }
        cout<<t<<endl;
    }
    //
    ////////////////////////////////////////////////////////////////////////////

}
```

## 3.3 分治 +fft

```
#include <bits/stdc++.h>
#define INF 0x3f3f3f3f
#define EPS 1e-8
#define PI acos(-1.0)
#define LL long long
#define ld double
#define ULL unsigned long long
#define rep(i,a,b) for(int i=a;i<b;i++)
#define PII pair<int,int>
#define PLL pair<LL,LL>
#define MP make_pair
#define sf(x) scanf("%d",&x)
#define sqr(x) ((x)*(x))
template <class T>
inline void rd(T &x) { char c = getchar(); x = 0;while(!
isdigit(c)) c = getchar();
while(isdigit(c)) { x = x * 10 + c - '0'; c = getchar();
}}
#define IN freopen("in.txt","r",stdin);
#define OUT freopen("out.txt","w",stdout);
using namespace std;
const int N=1e5+10;
const int M=313;
struct cpx
{
    ld r,i;
    cpx(){}
    cpx(ld r ,ld i):r(r),i(i) {}
    cpx operator + (const cpx& t) const
    {
        return cpx(r+t.r,i+t.i) ;
    }
    cpx operator - (const cpx& t) const
    {
        return cpx(r-t.r,i-t.i);
    }
    cpx operator * (const cpx& t) const
    {
        return cpx(r*t.r-i*t.i,r*t.i+i*t.r);
    }
};
const double pi=acos(-1.0);
int rev(int x, int n)
{
    int tmp=0;
    for (int i=1;i<n;i<<=1)
    {
        tmp<<=1;
        if (x & i) tmp |=1;
    }
    return tmp;
}

void fft(cpx *a, int n, int flag)// 1 -1
{
    for (int i=0;i<n;i++)
        if (i<rev(i, n))
            swap(a[rev(i, n)], a[i]);
    for (int i=1;i<n;i<<=1)
    {
        cpx wn(cos(pi/i), flag*sin(pi/i));
        for (int j=0;j<n;j+=(i<<1))
        {
            cpx w(1, 0);
            for (int k=0;k<i;k++, w=w*wn)
            {
                cpx x=a[j+k], y=w*a[j+k+i];
                a[j+k]=x+y;
                a[j+k+i]=x-y;
            }
        }
    }
    if (flag==-1) for (int i=0;i<n;i++) a[i].r/=n;
}
cpx x[1<<20],y[1<<20];
int a[N];
LL f[N];
void cdq(int l,int r)
{
  //cout<<l<<" "<<r<<endl;
  if (l==r)
  {
    (f[l]+=a[l])%=M;
    return;
  }
  int mid=l+r>>1;
  cdq(l,mid);
  int L=1;
```

```
  while (L<=r-l+1) L<<=1;

  rep(i,0,L)
  {
    if (l+i<=mid) x[i]=cpx(f[i+l],0);
    else x[i]=cpx(0,0);
    if (i+1+l<=r) y[i]=cpx(a[i+1]%M,0);
    else y[i]=cpx(0,0);
  }

  fft(x,L,1);
  fft(y,L,1);
  rep(i,0,L)
    x[i]=x[i]*y[i];
  fft(x,L,-1);
  rep(i,mid+1,r+1)
  {
    f[i]+=(LL)(x[i-l-1].r+0.5);
    f[i]%=M;
  }
  cdq(mid+1,r);
}
int main()
{

  int n;
  while (~sf(n)&&n)
  {
    memset(f,0,sizeof(LL)*(n+2));
    rep(i,0,n)
    rd(a[i+1]);
    cdq(1,n);
    cout<<f[n]<<endl;
    }
    return 0;
}
/*
2
8 6
*/
```

# 4 math

## 4.1 CRT

```
vector<LL> linear_mod_equation(LL a, LL b, LL n)
//线性方程求解
//ax = b (mod n)
{
    LL x, y, d;
    vector<LL> sol;
    sol.clear();
    EXT_GCD(a, n, d, x, y);
    if( b%d ) d = 0;
    else
    {
        sol.push_back(x * (b/d) % n);
        for (int i = 1; i < d; i++)
            sol.push_back((sol[i-1] + n/d + n) % n);
    }
    return sol;
}
LL mega_mod(int n)
//解 n 个一元线性同于方程组
//x ≡ r (mod a)
//求x
{
    LL a1, a2, r1, r2, d, c, x, y, x0,s;
    bool flag = true;
    scanf("%lld%lld", &a1, &r1);
    for(int i = 1; i < n; i++)
    {
        scanf("%lld%lld", &a2, &r2);
        if(!flag) continue;
        c = r2 - r1;
        EXT_GCD(a1, a2, d, x, y);
        if(c%d!=0)
        {
            flag = false;
            continue;
        }
        x0 = x*c/d;
        s = a2/d;
        x0 = (x0%s+s)%s;
        r1=r1+x0*a1;
        a1=a1*a2/d;
    }
    if(flag) return r1;
    else return -1LL;
}

LL CRT(LL *a, LL *m, int n)
//中国剩余定理
//x ≡ a[i] (mod m[i])
//m[i] is coprime
{
    LL MM = 1, Mi, x0, y0, d, ret = 0;
    for(int i = 0; i < n; i++)
        MM *= m[i];
    for(int i = 0; i < n; i++)
    {
        Mi = MM/m[i];
        EXT_GCD(Mi, m[i], d, x0, y0);
        ret = (ret+Mi*x0*a[i]) % MM;
    }
    if(ret < 0)
        ret += MM;
    return ret;
}
```

## 4.2 LL 黑科技

```
LL mult( LL A, LL B, LL Mo )
{
    LL temp = ( ( LL ) ( ( db ) A*B/Mo+1e-6 ) * Mo );
    return A*B - temp;
}
```

## 4.3 about prime

```
const int maxn = 3010020;
bool isprime[maxn];
LL prime[maxn];
int doprime(LL n)
//prime[] 储存质数。1-based index;
{
    int nprime = 0;
    memset(isprime, true, sizeof(isprime));
    isprime[1] = false;
    for(LL i = 2; i <= n; i++)
    {
        if(isprime[i])
        {
            prime[++nprime] = i;
            for(LL j = i*i; j <= N; j+=i)
                isprime[j] = false;
        }
    }
    return nprime;
}
LL slow_mul(LL a, LL b, LL p)
{
    // cout << a << " " << b << endl;
    LL ret = 0;
    while(b) {
        if(b & 1) ret = (ret + a) % p;
        a = (a + a) % p;
        b >>= 1;
    }
    return ret % p;
}

LL pow_mod(LL a, LL b, LL p)
//快速幂
{
    LL ret = 1;
    while(b) {
        if(b & 1) ret = (ret*a)%p;
        a = (a*a)%p;
        b >>= 1;
    }
    return ret%p;
}
```

```

//判断Mp = 2^p-1 是否为梅森素数
bool lucas_lehmer(int p)
{
    if(p == 2) return true;
    LL m = (1LL<<p)-1LL, tmp = 4LL;
    for(int i = 0; i < p-2; i++)
    {
        tmp = (slow_mul(tmp, tmp, m) - 2 + m) % m;
    }
    if(tmp == 0LL) return true;
    return false;
}

LL witness(LL a,LL b,LL c)
{
    if(b==0)return 1;
    LL x,y,t=0;
    while((b&1)==0)
        b>>=1,t++;
    y=x=pow_mod(a,b,c);
    //二次探测
    while(t--)
    {
        y=slow_mul(x,x,c);
        if(y==1 && x!=1 && x!=c-1)
            return false;
        x=y;
    }
    return y==1;
}
bool miller_rabin(LL n)
//..质数为true, 非质数为false..
{
    if(n==2)return true;
    if(n<2 || (n&1)==0)return false;
    for(int i=0;i<3;i++)
        if(witness(rand()%(n-2)+2,n-1,n)!=1)
            return false;
    return true;
}
LL ANS = INF;
LL pollard_rho(LL n,LL c)
//..随机返回一个 n 的约数..
{
    if(n%2==0)return 2;
    LL i=1,k=2,x=rand()%n,y=x,d;
    while(1){
        i++;
        x=(slow_mul(x,x,n)+c)%n;
        d=gcd(y-x,n);
        if(d==n)return n;
        if(d!=n && d>1)return d;
        if(i==k) y=x,k<<=1;
    }
}
void calc(LL n,LL c=240)
//寻找最小的约数..
{
    if(n==1)return;
    if(miller_rabin(n)){
        ANS=min(ANS,n);
        return;
    }
    LL k=n;
    while(k==n)k=pollard_rho(n,c--);
    calc(k,c),calc(n/k,c);
}
```

## 4.4 ext_gcd

```
LL ext_gcd(LL a, LL b, LL& x, LL& y)
// a >= 0, b > 0
{
    LL x1=0LL, y1=1LL, x0=1LL, y0=0LL;
    LL r = (a%b + b) % b;
    LL q = (a-r) / b;
    x = 0LL,y = 1LL;
    while(r)
    {
        x=x0-q*x1;y=y0-q*y1;
        x0=x1;y0=y1;
        x1=x;y1=y;
        a=b;b=r;
        r=a%b;
        q=(a-r)/b;
    }
    return b;
}
```

## 4.5 inv(gcd)

```
void EXT_GCD(LL a, LL b, LL &d, LL &x, LL &y)
//a , b 任意
{
    if(!b) {d = a, x = 1, y = 0;}
    else {EXT_GCD(b, a % b, d, y, x), y -= x * (a / b);}
}

//递归求逆元
//p, x 互质
LL inv(LL a, LL c)
// 用扩展欧几里得求逆元
// 要求 a, c 互质
// 如果没有逆元返回 -1
{
  LL d, x, y;
  EXT_GCD(a, c, d, x, y);
  return d == 1 ? (x + c) % c : -1;
}
```

## 4.6 phi

```
//欧拉函数
LL calphi(LL n)
{
    LL res = n;
    for(LL i = 2; i*i <= n; i++)
  if(n%i==0)
    {
        res -= res/i;
        while(n%i==0) n/=i;
    }
    if(n > 1)
        res -= res/n;
    return res;
}

//欧拉函数预处理
int phi[maxn];
void getpthi(int n)
{
    memset(phi, 0, sizeof(phi));
    phi[1] = 1;
    for(int i = 2; i <= n; i++)if(!phi[i])
    {
        for(int j = i; j <= n; j+=i)
        {
            if(!phi[j])
                phi[j] = j;
            phi[j] = phi[j]/i*(i-1);
        }
    }
}
```

# 5 spfa

## 5.1 Spfa 最短路

```
#include <iostream>
#include <cmath>
#include <algorithm>
#include <cstdio>
#include <queue>
#include <cstring>
#define N 1006
using namespace std;

struct node{
    int x, y, w, next;
}e[100006];
int a[N], d[N], v[N], head[N],
    ans, tot, n, m, x, y, w;
queue <int> Q;
```

```
inline void addEdge(int x,int y,int w)
{
    tot++; e[tot].x = x; e[tot].y = y; e[tot].w = w;
    e[tot].next = head[x]; head[x] = tot;
}
void spfa(int S)
{
    int x, k;
    memset(d, 0x3f, sizeof(d));
    memset(v, 0, sizeof(v));
    Q.push(S); v[S] = 1;
    d[S] = 0;
    while(!Q.empty())
    {
        x = Q.front(); Q.pop();
        v[x] = 0; k = head[x];
        while(k != -1)
        {
            if(d[x]+e[k].w < d[e[k].y])
            {
                d[e[k].y] = d[x] + e[k].w;
                if(!v[e[k].y])
                {
                    Q.push(e[k].y);
                    v[e[k].y] = 1;
                }
            }
            k = e[k].next;
        }
    }
}
inline void init()
{
    memset(head, -1, sizeof(head));
}
int main()
{
    freopen("test.in", "r", stdin);
    while(scanf("%d%d", &n, &m) != EOF)
    {
        init();
        for(int i = 0; i < m; i++){
            scanf("%d%d%d", &x, &y, &w);
            addEdge(x, y, w);
            addEdge(y, x, w);
        }
        int st, ed;
        scanf("%d%d", &st, &ed);
        spfa(st);
        int ans = d[ed];
        if(ans >= 0x3f3f3f3f) ans = -1;
        printf("%d\n", ans);
    }
}
```

# 6 划分树

## 6.1 划分树比 k 小数量

```
#define N 111111
#define M 22
int sorted[N];
int lef[22][N],f[22][N];
int n;
void build(int L=1,int R=n,int p=0)
{
    if (L==R) return;
    int mid=L+R>>1;
    int same=mid-L+1;
    int X=sorted[mid];
    rep(i,L,R+1)
        if (f[p][i]<X) same--;
    int lp=L,rp=mid+1;
    rep(i,L,R+1)
    {
        if (f[p][i]<X) f[p+1][lp++]=f[p][i];
        else if (f[p][i]==X&&same) f[p+1][lp++]=f[p][i],
        same--;
        else f[p+1][rp++]=f[p][i];
        lef[p][i]=lef[p][L-1]+lp-L;
    }
    build(L,mid,p+1);
    build(mid+1,R,p+1);
}
int qry(int l,int r,int k,int L=1,int R=n,int dep=0)
{
    if (l>r) return 0;
    if (l==r) return f[dep][l]<=k;
    int cnt=lef[dep][r]-lef[dep][l-1];
    int mid=L+R>>1;
    if (sorted[mid]<=k)
    {
        int nr=r+lef[dep][R]-lef[dep][r];
        int nl=nr-(r-l-cnt);
        return cnt+qry(nl,nr,k,mid+1,R,dep+1);
    }else{
        int nl=L+lef[dep][l-1]-lef[dep][L-1];
        int nr=cnt+nl-1;
        return qry(nl,nr,k,L,mid,dep+1);
    }
}
int main()
{
    int T,m,Case=0;
    scanf("%d",&T);
    while (T--)
    {
        memset(lef,0,sizeof(lef));
        scanf("%d%d",&n,&m);
        rep(i,1,n+1)
        {
            scanf("%d",&f[0][i]);
            sorted[i]=f[0][i];
        }
        sort(sorted+1,sorted+1+n);
        build();
        printf("Case %d:\n",++Case);
        rep(i,0,m)
        {
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            x++,y++;
            printf("%d\n",qry(x,y,z));
        }
    }
}
/*
1
10 20
1 4 2 3 5 6 7 8 9 0
1 3 2
1
10 1
1 5 2 3 6 4 7 3 0 0
3 9 2
*/
```

## 6.2 划分树第 k 大

```
template <class T>
inline void rd(T &x) { char c = getchar(); x = 0;while(!
isdigit(c)) c = getchar();
while(isdigit(c)) { x = x * 10 + c - '0'; c = getchar();
}}
using namespace std;
#define N 111111
#define M 22
int f[M][N];
int lef[M][N];
int sorted[N];
int n;
void build(int l=0,int r=n-1,int dep=0)
{
  if (l==r) return;
  int mid=(l+r)>>1;
  int s_same=0;
  int X=sorted[mid];
  int pl=l,pr=mid+1;
  rep(i,l,r+1)
    if (f[dep][i]<X) s_same++;
  s_same=mid-l+1-s_same;
  rep(i,l,r+1)
  {
    if (f[dep][i]<X)
    {
```

```
      f[dep+1][pl++]=f[dep][i];
      lef[dep][i]=(i-l?lef[dep][i-1]:0)+1;
    }else if (f[dep][i]==X&&s_same)
    {
      f[dep+1][pl++]=f[dep][i];
      s_same--;
      lef[dep][i]=(i-l?lef[dep][i-1]:0)+1;
    }else
    {
      f[dep+1][pr++]=f[dep][i];
        lef[dep][i]=(i-l?lef[dep][i-1]:0);
    }
  }
  build(l,mid,dep+1);
  build(mid+1,r,dep+1);
}
int qry(int l,int r,int k,int L=0,int R=n-1,int dep=0)
{
  int mid=(L+R)>>1;

  if (l==r) return f[dep][l];
  int cnt=lef[dep][r]-(l-L?lef[dep][l-1]:0);
  if (k>cnt)
  {
    int nl,nr;
    nl=mid+1+l-L-(l-L?lef[dep][l-1]:0);
    nr=mid+1+r-L-lef[dep][r];
    return qry(nl,nr,k-cnt,mid+1,R,dep+1);
  }else
  {
    int nl,nr;
    nl=L+(l-L?lef[dep][l-1]:0);
    nr=L+lef[dep][r]-1;
    return qry(nl,nr,k,L,mid,dep+1);
  }
}
int main()
{
      int m;
      rd(n);rd(m);
      memset(f,0,sizeof(f));
      rep(i,0,n)
      {
        rd(sorted[i]);
        f[0][i]=sorted[i];
      }
      sort(sorted,sorted+n);
      build();
      rep(i,0,m)
      {
        int s,t,k;
        rd(s);rd(t);rd(k);
        s--,t--;
        printf("%d\n",qry(s,t,k));
      }
    return 0;
}
/*
10 3
1 5 2 7 5 4 3 8 7 7
2 5 3
4 4 1
1 7 3
*/
```

# 7 匈牙利

## 7.1 二分匹配

```
using namespace std;
struct nod
{
  int y,id;
}a[N];
int n;
bool cmp(nod a,nod b)
{
  if (a.y!=b.y) return a.y>b.y;
  return a.id<b.id;
}
int b[N][N];
int linker[N],ma[N];
bool use[N];
bool dfs(int u)
{
  rep(i,1,n+1)
  if (b[u][i]&&!use[i])
  {
    use[i]=true;
    if (linker[i]==-1||dfs(linker[i]))
    {
      linker[i]=u;
      ma[u]=i;
      return true;
    }
  }
  return false;
}
void hungary()
{
  int u;
  memset(linker,-1,sizeof(linker));
  rep(i,0,n)
  {
    memset(use,0,sizeof(use));
    if (!dfs(a[i].id)) ma[a[i].id]=0;
  }
  rep(i,1,n+1)
  printf("%d%c",ma[i],i==n?'\n':' ');
}
int main()
{
  scan(&n);
  rep(i,0,n)
  {
    scan(&a[i].y);
    a[i].id=i+1;
  }
  sort(a,a+n,cmp);
  rep(i,1,n+1)
  {
    int m;
    scan(&m);
    rep(j,0,m)
    {
      int x;
      scan(&x);
      b[i][x]=1;
    }
  }
  hungary();
  return 0;
}
```

# 8 矩阵

## 8.1 矩阵

```
#define N 2
#define M 1000000007
LL fis[N];
struct mart
{
    LL a[N][N];
    mart(int x){
        memset(a,0,sizeof(a));
        if (x==1)
        rep(i,0,N) a[i][i]=1;
        if (x==2)
        {
            a[0][0]=lala;a[0][1]=0;
            a[1][0]=1;a[1][1]=1;
        }
    };
    mart operator *(mart &b)
    {
        mart c(0);
        rep(i,0,N)
            rep(j,0,N)
            if (a[i][j])
                rep(k,0,N)
                c.a[i][k]=(c.a[i][k]+a[i][j]*b.a[j][k])%M;
        return c;
    }
```

```
    void show()
    {
        puts("_________________");
        rep(i,0,N)
        {
            rep(j,0,N)
            printf("%d ",a[i][j]);
            puts("");
        }
        puts("_________________");
    }
};
LL pow(LL k)
{
    if (k<0) return 1;
    mart ret(1),a(2);
    //a.show();
    while (k)
    {
        if (k&1) ret=ret*a;
        a=a*a;k>>=1;
    }
    // ret.show();
    LL ans=0;
    rep(i,0,N)
    ans=(ans+fis[i]*ret.a[i][0]%M)%M;
    return ans;
}
```

# 9 组合

## 9.1 组合数求 mod

```
LL exp_mod(LL a, LL b, LL p) {
    LL res = 1;
    while(b != 0) {
        if(b&1) res = (res * a) % p;
        a = (a*a) % p;
        b >>= 1;
    }
    return res;
}

LL Comb(LL a, LL b, LL p) {
    if(a < b)   return 0;
    if(a == b)  return 1;
    if(b > a - b)   b = a - b;

    LL ans = 1, ca = 1, cb = 1;
    for(LL i = 0; i < b; ++i) {
        ca = (ca * (a - i))%p;
        cb = (cb * (b - i))%p;
    }
    ans = (ca*exp_mod(cb, p - 2, p)) % p;
    return ans;
}

LL Lucas(int n, int m, int p) {
     LL ans = 1;

     while(n&&m&&ans) {
        ans = (ans*Comb(n%p, m%p, p)) % p;
        n /= p;
        m /= p;
     }
     return ans;
}
```

# 10 网络流

## 10.1 MinCost_bhb

```
struct edge{
  int x, ne, c, f, w;
};

struct MinCostFlow{
  edge e[M];
  int S, T, pos, quantity, cost;
  int head[N], dis[N], pre[N], at[N];
  queue <int> q;
  bool used[N];

  void adde(int u, int v, int c, int w){
    //printf("Add edge : %d -> %d c : %d w : %d\n", u, v, c, w);
    e[++pos] = (edge){v, head[u], c, 0, w};
    head[u] = pos;
    e[++pos] = (edge){u, head[v], c, c, -w};
    head[v] = pos;
  }

  bool spfa(){
    memset(dis, 0x3f, sizeof(dis));
    memset(used, 0, sizeof(used));
    used[S] = true;
    while (!q.empty())
      q.pop();
    q.push(S);
    dis[S] = 0;
    while (!q.empty()){
      int x = q.front();
      //printf("Now : %d\n", x);
      for (int i = head[x]; i; i = e[i].ne){
        int y = e[i].x;
        if (e[i].c > e[i].f && dis[x] + e[i].w < dis[y]){
          dis[y] = dis[x] + e[i].w;
          at[y] = i;
          pre[y] = x;
          if (!used[y]){
            used[y] = true;
            q.push(y);
          }
        }
      }
      used[x] = false;
      q.pop();
    }
    //printf("Spfa : %d\n", dis[T]);
    return dis[T] != INF;
  }

  void update(){
    int cut = INF;
    for (int i = T; i != S; i = pre[i]){
      cut = min(cut, e[at[i]].c - e[at[i]].f);
    }
    //printf("Cut %d's path : %d -> ", T);
    for (int i = T; i != S; i = pre[i]){
      e[at[i]].f += cut;
      e[at[i] ^ 1].f -= cut;
      //printf(" -> %d", pre[i]);
    }
    quantity += cut;
    cost += cut * dis[T];
    //puts("________");
  }

  void init(int s, int t){
    S = s;
    T = t;
    pos = 1;
    quantity = cost = 0;
    memset(head, 0, sizeof(head));
  }

  PII work(){
    //puts("Starting");
    while (spfa())
      update();
    //printf("%d %d\n", quantity, cost);
    return make_pair(quantity, cost);
  }
}flow;
```

## 10.2 dinic_bhb

```
struct edge
{
  int x, ne, c, f;
};

struct NetFlow
{
```

```
  edge e[M];
  int head[N], h[N], dis[N], q[N], stack[N];
  bool used[N];
  int pos, stop, top, S, T;
  LL flow;

  void init(int s, int t)
  {
    pos = 1;
    flow = top = 0;
    S = s;
    T = t;
    memset(head, 0, sizeof(head));
  }

  void adde(int u, int v, int c)
  {
    e[++pos] = (edge){v, head[u], c, 0};
    head[u] = pos;
    e[++pos] = (edge){u, head[v], c, c};
    head[v] = pos;
  }

  bool number()
  {
    memset(dis, 0, sizeof(dis));
    memset(used, 0, sizeof(used));
    int p1, p2, x;
    used[q[p1 = p2 = 1] = S] = true;
    while (p1 <= p2)
    {
      x = q[p1++];
      for (int i = head[x]; i; i = e[i].ne)
        if (e[i].c > e[i].f && !used[e[i].x])
        {
          used[q[++p2] = e[i].x] = true;
          dis[e[i].x] = dis[x] + 1;
        }
    }
    if (!used[T])
      return false;
    memcpy(h, head, sizeof(head));
    return true;
  }

  bool dinic(int x)
  {
    if (x == T)
    {
      int cut = INF;
      for (int i = 1; i <= top; ++i)
        cut = min(cut, e[stack[i]].c - e[stack[i]].f);
      for (int i = 1; i <= top; ++i)
      {
        e[stack[i]].f += cut;
        e[stack[i] ^ 1].f -= cut;
        if (e[stack[i]].c == e[stack[i]].f)
          stop = i;
      }
      flow += cut;
      return true;
    }
    for (int &i = h[x]; i; i = e[i].ne)
      if (e[i].c > e[i].f && dis[x] == dis[e[i].x] - 1)
      {
        stack[++top] = i;
        if (dinic(e[i].x) && stop != top)
        {
          --top;
          return true;
        }
        --top;
      }
    return false;
  }

  LL maxflow()
  {
    while (number())
      dinic(S);
    return flow;
  }
}net;
```