Решение:
Сега ще докажа, че отговорът на задачата е равен на броя на пътищата от началния до крайния връх. Нека си припомним как се пресмята този брой:
Сортираме топологично върховете като началния връх е първи в получената редица. Сега нека въведем масива **A[]. A[l]** ще показва колко различни пътя има от началния връх до връх номер **l**. Нека топологичното сортиране е в масива **T[]**. Инициализираме **A[T[1]]=1**. Алгоритъма почва от **2** и върви до **N** по масива **T[]**.За всеки елемент **T[i]** смята **A[T[i]].** А то е равно на сумата на всички **T[j],** за които съществува ребро **j->i**.
Сега ще покажа, че отговорът на задачата **<=** броя на всички пътища. Нека приемем, че не е. Тогава от принципа на Дирихле ще следва, че има две коли с еднакъв път – а тогава значи или че те не са се изпреварили и не са в обратен ред или че са се изпреварили на някоя улица, което е забранено => не е възможно 2 коли да имат еднакви пътища => отговорът на задачата **<=** броя на всички пътища.
Сега ще покажа начин за придвижване на толкова коли. Нека до даден връх **k1** са се придвижили първите **A[k1]** коли в обратен ред и са продължили малко по някое друго ребро излизащо от **k1**. Нека това ребро е такова че веднъж продължили по него колите не могат да се върнат върхове **k1,k2,.....kl**. Нека това е така за върхове **k1,k2....kl** (само че до **A[k2]** са се придвижили следващите **A[k2]** коли, до **k3** следващите **A[k3]** коли и т.н.). Това е възможно да стане тъй като колите продължили напред от **k1** не пречат на колите пристигащи в **k2...kl**. И нека от тези върхове има ребро към връх **B** и всички коли са избрали да продължат по него. Тогава нека първо влязат в **B** и продължат по някое ребро колите от **kl**, после от **k(l-1)** и т.н. Така се получи че колите излязоха от **B** в обратен ред а броят на колите излезли от **B** е равен на сумата от **A[k1]+..+..+A[kl]**, но пък тази сума е равна на **A[B]**. Следователно за всеки връх **B** могат **A[B]** коли да дойдат от началния и да преминат през **B** в обратен ред.
Отговорът на задачата се намира в **A[T[N]]**. При извършване на всички сметки е необходимо да се пресмята по модул, тъй като числата могат да станат много големи.

```
#include<stdio.h>
#include<vector>
using namespace std;
#define pb push_back
#define sz size()
#define in stdin;//fopen("c:\\tp\\tests\\cars.in","r")
FILE*fn;

#define mx 512
#define base 1000000000
#define lb 9
#define type long long
struct L
{int ss[mx];
L(){for(int q1=0;q1<mx;q1++)ss[q1]=0;}
L(int a){for(int q1=mx-1,c1=a;q1>=0;q1--){ss[q1]=c1%base;c1/=base;}};
};
int dif(const L&a,const L&b)
  {int q1;
  for(q1=0;q1<mx;q1++)if(a.ss[q1]!=b.ss[q1])break;
  if(q1==mx)return 0;
```

```
  return a.ss[q1]-b.ss[q1];}
int operator<(const L&a,const L&b){int c1=dif(a,b);return c1<0;}
int operator>(const L&a,const L&b){int c1=dif(a,b);return c1>0;}
int operator==(const L&a,const L&b){int c1=dif(a,b);return c1==0;}
int operator!=(const L&a,const L&b){int c1=dif(a,b);return c1!=0;}
int operator<=(const L&a,const L&b){int c1=dif(a,b);return c1<=0;}
int operator>=(const L&a,const L&b){int c1=dif(a,b);return c1>=0;}
L operator+(const L&a,const L&b)
  {L c;
  for(int q1=mx-1,c1=0;q1>=0;q1--)
    {c1+=a.ss[q1]+b.ss[q1];
    c.ss[q1]=c1%base;
    c1/=base;}
  return c;}
L operator-(const L&a,const L&b)
  {L c;
  if(b>a)return b-a;
  for(int q1=mx-1,c1=0,c2=0;q1>=0;q1--)
    {c1=a.ss[q1]-b.ss[q1]-c2;
    c2=0;
    if(c1<0){c1+=base;c2=1;}
    c.ss[q1]=c1;}
  return c;}
L operator*(const L&a,int b)
  {L c;
  type c1=0;
  for(int q1=mx-1;q1>=0;q1--)
    {c1+=(type)a.ss[q1]*(type)b;
    c.ss[q1]=(int)(c1%(type)base);
    c1/=(type)base;}
  return c;}
L operator/(const L&a,int b)
  {L c;
  for(int q1=0,c1=0;q1<mx;q1++)
    {c1+=a.ss[q1];
    c.ss[q1]=c1/b;
    c1=(c1%b)*base;}
  return c;}
int operator%(const L&a,int b)
  {type c1=0;
  for(int q1=0;q1<mx;q1++)
    {c1+=a.ss[q1];
    c1=(c1%(type)b)*(type)base;}
  return (int)(c1/(type)base);}
L operator*(const L&a,const L&b)
  {L c;
  int q3;
  type c1=0;
  for(int q1=mx-1;q1>=0;q1--)for(int q2=mx-1;q2>=0;q2--)if(q1+q2-mx+1>=0)
    {c1=(type)a.ss[q1]*(type)b.ss[q2];
    for(q3=q1+q2-mx+1;q3>=0&&c1!=0;q3--)
      {c1+=c.ss[q3];
```

```
      c.ss[q3]=c1%(type)base;
      c1/=base;}}
  return c;}
L operator^(L a,L b)
  {L d=1,c;
  while(a%2==0&&b%2==0)
    {a=a/2;
    b=b/2;
    d=d*2;}
  while(a>0)
    {if(a%2==0)a=a/2;
    else if(b%2==0)b=b/2;
    else
      {c=(a-b)/2;
      if(a<b)b=c;else a=c;}}
  return d*b;}
L read(FILE*fn)
  {int q1,c1,c2,q2,c3;
  char d[mx*lb];
  L c;
  fscanf(fn,"%s",d);
  for(q1=0;d[q1];q1++);
  c1=q1;
  for(q1=c1-1,c2=0,c3=mx-1;q1>=0;q1-=lb,c2=0)
    {for(q2=lb-1;q2>=0;q2--)if(q1-q2>=0)c2=c2*10+d[q1-q2]-48;
    c.ss[c3--]=c2;}
  return c;}
int write(L a)
  {int q1;
  for(q1=0;q1<mx&&a.ss[q1]==0;q1++);
  if(q1==mx)q1--;
  printf("%d",a.ss[q1]);
  for(q1++;q1<mx;q1++)printf("%0*d",lb,a.ss[q1]);
  printf("\n");
  return 0;}

vector < vector < int > > nei;
vector < vector < int > > back;
vector < L > res;

int c,m,q1,q2,c1,c2,c3;

vector < int > top;
vector < bool > v;

int dfs(int a)
  {int q1;
  v[a]=1;
  for(q1=0;q1<nei[a].sz;q1++)if(!v[nei[a][q1]])dfs(nei[a][q1]);
  top.pb(a);
  return 0;}
int main()
```

```
{fn=in;
fscanf(fn,"%d",&c,&m);
nei.resize(c);
back=nei;
v.resize(c);
res.resize(c);
for(q1=0;q1<c;q1++)
  {fscanf(fn,"%d",&c1);
  for(q2=0;q2<c1;q2++)
    {fscanf(fn,"%d",&c2);
    c2--;
    nei[q1].pb(c2);
    back[c2].pb(q1);}
  }
fclose(fn);
for(q1=0;q1<c;q1++)if(back[q1].sz==0)break;
dfs(q1);
res[top[c-1]]=1;
for(q1=c-2;q1>=0;q1--)
  for(q2=0;q2<back[top[q1]].sz;q2++)
    res[top[q1]]=res[top[q1]]+res[back[top[q1]][q2]];
write(res[top[0]]);
return 0;}
```

Задача А3. ИЗРАЗ

Киро и приятелите му играят странна игра. Кодират математически израз с 0 и 1 в строго определен формат. Всеки от играчите има право да промени израза с максимум **K** замени. Печели този от тях, който след като се пресметне променения израз получи, в резултат на пресмятането на израза, максимално число. Изразът е дефиниран с правилата:

<**израз**>:=<цифра><знак><цифра><знак>...<цифра>

<**цифра**>:=0,1,2,3,4,5,6,7,8,9 <**знак**>:=+,-

Записването на цифрите и знаците в израза с 0 и 1 става по следния начин (удебеленият шрифт е за да личат по ясно нулите и единиците) :

0	1	2	3	4	5	6	7	8	9	+	–
111	00**1**	**111**	**111**	**101**	**111**	**111**	**111**	**111**	**111**	000	000
10**1**	00**1**	00**1**	00**1**	**1**0**1**	**1**00	**1**00	00**1**	**1**0**1**	**1**0**1**	0**1**0	000
10**1**	00**1**	**111**	**111**	**111**	**111**	**111**	00**1**	**111**	**111**	**111**	**111**
10**1**	00**1**	**1**00	00**1**	00**1**	00**1**	**1**0**1**	00**1**	**1**0**1**	00**1**	0**1**0	000
111	00**1**	**111**	**111**	00**1**	**111**	**111**	00**1**	**111**	**111**	000	000

Под промяна на израз с **K** замени се разбира инвертирането (замяна на 1 с 0 и обратно) на **K** цифри, така че общият брой на единиците в новия израз да е равен на броя на единиците в оригиналния израз. При промяна на израз, знак може да се трансформира само в знак и цифра може да се трансформира само в цифра, т.е. не може цифра да стане знак и обратното. В израз участва минимум една цифра и максимум 50. Пример за промяна на израз с 4 замени (с удебелен шрифт са промените):

Изразът преди промяната	Изразът след промяната
111000**111**	111000**00**1
100000000**1**	10**1**000001
11111100**1** <=> 5-7 = -2	111111001 <=> 8-1 = 7