Találka

Ádám és Éva szeretne találkozni. Éva az E városban, Ádám pedig az A városban van. Vonattal kívánnak utazni, és ismerik a teljes menetrendet. A menetrend n várost tartalmaz, és azt, hogy mely városok között van vonatjárat. Minden vonat adott i-edik városból indul és adott j-edik városba közlekedik és közben nem áll meg egyetlen közbülső állomáson sem. Olyan városban akarnak találkozni, ahova Éva a lehető legkevesebb átszállással tud utazni, de Ádám is el tud oda menni vonattal.

Feladat

Ijunk olyan programot, amely megad Éva számára egy legkevesebb átszállásos útvonalat olyan városba, ahova Ádám is el tud jutni vonattal!

Bemenet

A standard bemenet első sora négy egész számot tartalmaz, a városok n számát ($1 \leq n \leq 20000$), a járatok m számát ($1 \leq m \leq 1000000$), Éva E tartózkodási helyét és Ádám A tartózkodási helyét ($1 \leq E \neq A \leq n$). A városokat az $1, \ldots, n$ számokkal azonosítjuk.

Kimenet

A standard kimenet első sora azon város R sorszámát tartalmazza, ahova éva a legkevesebb átszállással el tud utazni. A második sot tartalmazza Éva útvonalát, a harmadik pedig Ádám útvonalát! Több megoldás esetén bármelyik megadható. Ha nincs a feltételnek megfelelő város, akkor az első és egyetlen sor a 0 számot tartalmazza!

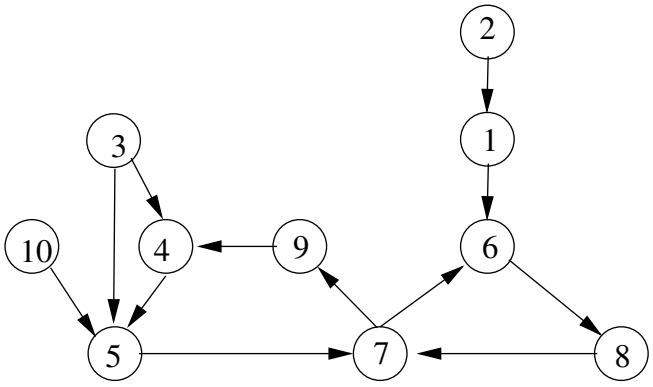
Példa

Bemenet

```
10 12 2 3
2 1
1 6
7 6
6 8
8 7
7 9
9 4
5 7
10 5
3 5
3 4
4 5
```

Kimenet

```
6
2 1 6
3 5 7 6
```



Korlátok

Időlimit: 0.1 mp.
Memórilimit: 32 MiB
Pontozás: a tesztesetek 40%-ában $n < 1000$

Megoldás

Tekintsük azt az irányított $G = (V, E)$ gráfot, amelynek pontjai a városok azaz $V = \{1, \ldots, n\}$, és (u, v) akkor és csak akkor él a gráfban, ha az u városból van közvetlen járat a v városba.
Jelölje $TavE(p)$ a p pont távolságát Éva E tartózkodási helyétől. $TavE(p)$ legyen ∞ ha nincs út E-ből p-be. Hasonlóan, $TavA(p)$ jelölje a p pont távolságát Ádám A tartózkodási helyétől. Ekkor a keresett R találkahely az alábbi algoritmussal számítható.

```
R:=0
minTav:=Végtelen
ciklus p:=1-töl n-ig
    ha TavE[p]<minTav és TavA[p]!=Végtelen akkor
        minTav:=TavE[p]
        R:=p;
    elágazás vége
 ciklus vége
```

$TavE$ és $TavA$ kiszámítását az alábbi SzeltBejar algoritmus adja. Ahhoz, hogy meg is tudjunk adni egy legrövedebb utat E-ből R-be, az algoritmus minden p pontra megadja azt az $ApaE[p]$ pontot, amely az E-ből p-be vezető legrövidebb úton a p-t megelőző pont (0, ha $p = E$).
Az algoritmus alapja a következő. Jelölje $T(k)$ a p pottól k távolságra lévő pontok halmazát ($k = 0, 1 \ldots, n-1$). Belátjuk, hogy bármely q-ra $q \in T(k)$ akkor és csak akkor, ha az algoritmus végén $Tav[q] = k$.
$k = 0$-ra igaz az állítás, mert $T(0) = \{p\}$. Tegyük fel, hogy minden $l < k$-ra igaz az állítás és legyen $q \in T(k)$. Ekkor van olyan u pont, hogy $u \to q$ él a gráfban és $u \in T(k-1)$. Az indukciós feltevésünk szerint az algoritmus helyesen számítja u távolságát, tehát $Tav[u] = k - 1$ lesz. Tehát amikor az u pontot kivesszi az S sorból, akkor $Tav[u] = k - 1$ és Tav[q]=Végtelen, tehát végrehajtódik a $Tav[q] := Tav[u] + 1$ értékadás. A fordított irányú tartalmazás hasonlóan látható be.

```
eljárás SzeltBejar(G,n,p,Tav,Apa)
//Be: G,n,p
//Ki: Tav,Apa
    ciklus i:=1-töl n-ig Tav[i]=Végtelen
    Tav[p]:=0; Apa[p]:=0;
    Sorba(S,p)
    ciklus amíg NemÜres(S)
        u:=Sorbol(S)
        ciklus minden v pontra ahol (u,v) G-ben
            ha Tav[v]=Végtelen akkor
                Tav[v[:=Tav[v]+1
                Apa[v]:=u;
                Sorba(S,v);
            elágazás vége
        ciklus vége
    ciklus vége
eljárás vége
```

Megvalósítás C++ nyelven

```
1  #include <iostream>
2  #include <vector>
3  #include <queue>
4  #define maxN 20001
5  using namespace std;
6  typedef vector<int> Graf[];
7  vector<int> G[maxN];
8  int n, E,A,R;
9  const int Inf=maxN+1;
10 void Beolvas(){
11 //Globális: G,GT,n,E,A,R
```

```
12     int m,p,q;
13     cin>>n>>m>>E>>A;
14     for (int i=0;i<m;i++){
15        cin>>p>>q;
16        G[p].push_back(q);
17     }
18  }
19  void SzeltBejar(Graf G, int p, int Tav[], int Apa[]){
20     queue<int> S;
21     for(int i=1;i<=n;i++) Tav[i]=Inf;
22     Tav[p]=0; Apa[p]=0;
23     S.push(p);
24     int u;
25     while (!S.empty()){
26        u=S.front(); S.pop();
27        for(int v:G[u])
28           if(Tav[v]==Inf){
29              Tav[v]=Tav[u]+1;
30              Apa[v]=u;
31              S.push(v);
32           }
33     }
34  }
35  int main(){
36     Beolvas();
37     int TavE[n+1]; int ApaE[n+1];
38     int TavA[n+1]; int ApaA[n+1];
39     SzeltBejar(G, A, TavA, ApaA);
40     SzeltBejar(G, E, TavE, ApaE);
41     int minTav=Inf; R=0;
42     for(int p=1;p<=n;p++)
43     if(TavE[p]<tav && TavA[p]<Inf){
44        minTav=TavE[p];
45        R=p;
46     }
47     cout<<R<<endl;
48     int p=R;
49     int Ut[n]; int hol=0;
50     while(p>0){
51        Ut[hol++]=p;
52        p=ApaE[p];
53     }
54     for(int i=hol-1;i>=0;i--)
55        cout<<Ut[i]<<"␣";
56     cout<<endl;
57     p=R; hol=0;
58     while(p>0){
59        Ut[hol++]=p;
60        p=ApaA[p];
61     }
62     for(int i=hol-1;i>=0;i--)
63        cout<<Ut[i]<<"␣";
64     cout<<endl;
65   return 0;
66   }
```