Hálózatok összekapcsolása

k szolgáltató üzemeltet internet szolgáltatást. Összesen n szolgáltató központ van, mindegyik valamelyik szolgáltatóhoz tartozik. Minden szolgáltató kiépítette saját hálózatát úgy, hogy bármely két központja között lehet adatátvitelt megvalósítani a hálozaton keresztül. A k különálló hálózatot össze akarják kapcsolni úgy, hogy bármely két központ között lehessen átvitel (nem feltétlenül közvetlen), függetlenül attól, hogy melyik hálózathoz tartoznak. Az összekapcsolást úgy akarják megvalósítani, hogy bizonyos központok között új adatátviteli vonalat építenek ki. A megvalósításra pályázatot írtak ki, amire m pályázat érkezett. Minden pályázat három adatot tartalmaz: A, B, C ami azt jelenti, hogy a pályázó az A és a B központ központ között C költséggel építeni ki vonalat.

Feladat

Ijunk olyan programot, amely kiszámítja, hogy mely pályázatokat kell elfogadni ahhoz, hogy a legkisebb összköltséggel összekapcsolják a hálózatokat!

Bemenet

A `standard bemenet` első sora három egész számot tartalmaz, a csomópontok n számát ($1 \leq n \leq 10\,000$), a hálózatok k számát ($1 \leq k \leq n$) és az ajánlatok m számát ($1 \leq m \leq 500\,000$). A központokat az $1, \ldots, n$ számokkal azonosítjuk. A következő k sor mindegyike egy-egy hálózathoz tartozó központok sorszámait tartalmazza egy-egy szóközzel elválasztva. Minden felsorolást egy 0 szám zár. Az ezt követő m sor mindegyike egy ajánlatot tartalmaz, három egész számot egy-egy szóközzel elválasztva: `A B C`, ami azt jelenzi, hogy a pályázó az `A` és a `B` különböző hálózathoz tartozó két központ között `C` költséggel építeni ki vonalat ($1 \leq C \leq 1\,000$).

Kimenet

A `standard kimenet` első sora két egész számot tartalmazzon, a legkisebb P összköltséget, amellyel a hálózatok összekapcsolhatók és ennek megvalósításához elfogadandó pályázatok l számát. A második sor a kiválasztott l pályázat sorszámait tartalmazza egy-egy szóközzel elválasztva, tetszőleges sorrendben. Több megoldás esetén bármelyik megadható.

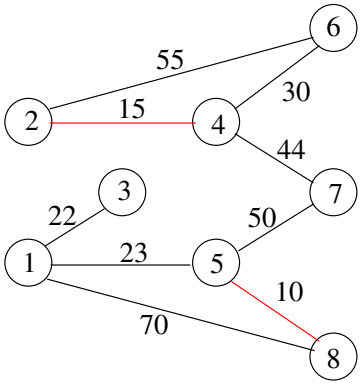
Példa

Bemenet

```
8 3 9
1 2 0
3 4 5 0
6 7 8 0
2 6 55
2 4 15
1 3 22
1 5 23
1 8 70
5 8 10
5 7 50
7 4 44
4 6 30
```

Kimenet

```
25 2
2 6
```



Korlátok

Időlimit: 0.5 mp.
Memórilimit: 32 MiB
Pontozás: a tesztesetek 40%-ában $n < 1000$

Megoldás

Tekintsük azt a $G = (V, E, suly)$ irányítatlan súlyozott gráfot, ahol az élek $suly : E \to N$ súlya a két pont között kiépítendő vonal költsége. Tekintsük úgy, hogy a súly érték 0 minden olyan `u v` párra, ahol `u` és `v` ugyanazon hálózathoz tartozik. A megvalósításban azonban nem kell felvenni ezeket a 0 súlyú éleket, hanem az egy hálózathoz tartozó pontokat összevonjuk egy részhalmazzá a beolvasáskor. Tehát a Beolvas után ha `u` és `v` ugyanazon hálózathoz tartozik, akkor `Holvan(u)=Holvan(v)`.
A megvalósításhoz az UnioHolvan műveleteket használjuk. Az értékhalmaz kezdetben az egyelemű $\{1\}, \ldots, \{n\}$ részhalmazok (a gráf pontjai: $1, \ldots, n$).
`Holvan(x)` művelet egy olyan `Nx` értéket ad eredményül, amely azt a részhalmazt reprezentálja, amelynek `x` eleme. Tehát ha `x` és `y` ugyanazon részhalmaz elemei, akkor `Holvan(x)=Holvan(x)`. Az `Unio(Nx,Ny)` egyesíti azt a két diszjunk részhalmazt, amelyeknek `Nx` és `Ny` a reprezentánsa. A gráf éleit a G tömb tartalmazza.

```
Beolvas
rendezzük az éleket suly szerint növekvöre
fael:=0 //a minimális feszítöfa éleinek száma
okolts:=0
ciklus i:=1-töl m-ig
    Nx:=Holvan(G[i].p)
    Nx:=Holvan(G[i].q)
    ha Nx!=Ny akkor
        Unio(Nx,Ny)
        Fa[fael]:=i
        fael:=fale+1
        okolts:=okolts+G[i].suly
    elágazás vége
ciklus vége
KiÍr(okolts,' ',fael)
KiÍr(Fa)
```

Megvalósítás C++ nyelven

```
1  #include <iostream>
2  #include <algorithm>
3  #define MaxN 10000
4  #define MaxM 500000
5  using namespace std;
6
7  struct El{
8    int p,q,az;
9    long suly;
10 };
11 int n,k,m;
12 El G[MaxM];
13 int Apa[MaxN];
14
15 void UnioHolvan(int n);
16 int Holvan(int x);
17 int Unio(int Nx, int Ny);
18
19 void Beolvas(){
20    int x,y,Nx,Ny;
21    cin>>n>>k>>m;
22    UnioHolvan(n);
23    for(int i=1;i<=k;i++){
24       cin>>x;
25       Nx=Holvan(x);
26       while(true){
27          cin>>y;
28          if(y==0) break;
29          Ny=Holvan(y);
30          Nx=Unio(Nx,Ny);
31       }
32    }
33    for(int i=0;i<m;i++){
34       cin>>G[i].p>>G[i].q>>G[i].suly;
35       G[i].az=i+1;
36    }
37 }
38
39 bool rend_rel(const El a, const El b) {
40    return a.suly < b.suly;
41 }
42
43 int main(){
44    int fael=0 ;
45    int Np,Nq;
46    Beolvas();
47    sort(begin(G),begin(G)+m,rend_rel );
48
49    int okolts=0;
50    int Mego[n];
51    for (int i=0; i<m; i++){
52       Np = Holvan(G[i].p);
53       Nq = Holvan(G[i].q);
```

```
54        if (Np!=Nq){
55           Mego[fael++]=G[i].az;
56           Unio(Np,Nq);
57           okolts+=G[i].suly;
58        }
59     }
60     cout<<okolts<<"␣"<<fael<<endl;
61     for(int i=0;i<fael;i++)
62        cout<<Mego[i]<<"␣";
63     cout<<endl;
64  }//main
65  void UnioHolvan(int n){
66    for (int x = 1; x <= n; ++x)
67      Apa[x] = -1;
68  }
69  int Holvan(int x){
70    int Nx = x;
71    while (Apa[Nx] > 0)
72      Nx = Apa[Nx];
73    int y = x;
74  //úttömörités
75    while (x != Nx){
76      y = Apa[x];
77      Apa[x] = Nx;
78      x = y;
79    }
80    return Nx;
81  }
82  int Unio(int Nx, int Ny){
83    if (Nx == Ny)
84      return Nx;
85    if (Apa[Nx] > Apa[Ny]){//egyesítés a ngyobbikhoz
86      int z = Nx;
87      Nx = Ny;
88      Ny = z;
89    }
90    Apa[Nx] += Apa[Ny];
91    Apa[Ny] = Nx;
92    return Nx;
93  }
```