Függvény adattípus

Horváth Gyula
horvath@inf.elte.hu

7.1. Függvény (map, SortedDictionary)

A Függvény adattípus kezelhető lenne a Halmaz adattípussal is, mivel minden f függvény, amelynek értelmezési tartománya az A halmaz és értékkészlete a B halmaz az $(a, f(a))$ párok halmaza. Tehát f az $A \times B$ direktszorzat olyan részhalmaza, amelyben nincs két olyan elem, amelyek első tagjai egyenlők és bármely $a \in A$-hoz van olyan $b \in B$, hogy $(a,b) \in f$. Az utóbbi feltétel azt jelenti, hogy f az A értelmezési tartomány minden elemére értelmezett. A gyakorlatban inkább parciális függvényekre van szükség, ahol az értelmezési tartomány nem minden eleméhez van függvényérték hozzárendelve, tehát ekkor a feltétel:

$$f : A \to B = R \subseteq A \times B : (\forall x \in A)(\forall y_1, y_2 \in B)((x,y_1) \in f \wedge (x,y_2) \in f \Rightarrow y_1 = y_2)$$

A továbbiakban függvényen, ha csak külön nem hangsúlyozzuk, parciális függvényt értünk.
A Függvény külön adattípusként való megvalósítását elsősorban hatékonysági szempontok indokolják.
A fentiek alapján nem meglepő, hogy a Függvény adattípus műveletei nagyon hasonlóak a Halmaz adattípus műveleteihez. Hagyományosan az értelmezési tartomány típusát Kulcs (Key) típusnak, az értékkészlet típusát Érték (Value) típusnak is nevezik.
Mindkét nyelvben az F függvénynek az x helyen felvett értékére az $F[x]$ tömbszerű hivatkozást is használhatjuk. C++ esetén az értéke $F.end()$, ha a függvény az x argumentumra nincs értéke. C# megszakítást generál, ha F az x argumentumra nincs értelmezve. A $F.TryGetValue(x)$ lekérdezést lehet használni, ha nem biztos, hogy értelmezett a függvény az x helyen.

Értékhalmaz: $Fuggveny(Kulcs, \text{Érték}) = \{\ F : Kulcs \to \text{Érték}\}$, $Kulcs$-on értelmezett a $<$ lineáris rendezési reláció.
Jelölés: $F.kulcs = \{x \in Kulcs : (\exists y \in \text{Érték})((x,y) \in F)\}$
Jelölés: $F[x] = y : (x,y) \in F$
Műveletek argumentumai: F : Függvény, $x : Kulcs$, y : Érték, $it : Iterator$

Előfeltétel	Művelet	Utófeltétel
$x \notin F.kulcs$	Bővít(F,x,y)	$F = Pre(F) \cup \{(x,y)\}$
$x \in F.kulcs$	Bővít(F,x,y)	$= it \wedge it.kulcs = x \wedge F = Pre(F)$
$x \in F.kulcs$	Töröl(F,x)	$F = Pre(F) - \{(x,y)\}$
$x \in F.kulcs$	Keres(F,x)	$= it : it.kulcs = x$
$x \in F.kulcs$	Módosít(F,x,y)	$F[x] = y$
$F = F$	ÜresE(F)	$= (F = \emptyset) \wedge F = Pre(F)$
$F = F$	Elemszám(F)	$= \lvert F \rvert \wedge F = Pre(F)$
$F \neq \emptyset$	Első(F)	$= it \wedge it.kulcs = \min(F.kulcs) \wedge Pre(F) = F$
$F \neq \emptyset$	Utolsó(F)	$= it \wedge it = \max(F.kulcs), \wedge Pre(F) = F$
$F \neq \emptyset$	Alsókorlát(F,x)	$= it \wedge it.kulcs = \min\{\overline{x} \in F.kulcs : x \leq \overline{x}\} \wedge Pre(F) = F$
$F \neq \emptyset$	Felsőkorlát(F,x)	$= it \wedge it.kulcs = \min\{\overline{x} \in F.kulcs : x < \overline{x}\} \wedge Pre(F) = F$
$F = F$	Üresít(F)	$F = \emptyset$

7.2. A Függvény C++ és C# műveletei

C++ esetén egy it Függvény iterátor $pair < Kulcs, \text{Érték} >$ típusú adatra való hivatkozás, az első tagra az $(*it).first$ a másodikra pedig a $(*it).second$ mezőkiválasztással hivatkozhatunk.

Művelet	**C++**	**C#**
Típusdef	$map < Kulcs, \text{Érték} >$	$SortedDictionary < Kulcs, \text{Érték} >$
Bővít(F,x,y)	$F.insert(x,y)$	$F.Add(x,y)$
Töröl(F,x)	$F.erase(x)$	$F.Remove(x)$
Érték(F,x)	$F[x]$	$F[x]$, $F.TryGetValue(x,\ out\ \text{Érték}y)$
Keres(F,x)	$F.find(x)$	$F.ContainsKey(x)$
Keres(F,y)	$-$	$F.ContainsValue(y)$
Módosít(F,x,y)	$F[x] = y$	$F[x] = y$
Módosít(F,x,y)	$(*it).second = y$	
ÜresE(F)	$F.empty()$	$F.Count == 0$
Elemszám(F)	$F.size()$	$F.Count$
Első(F)	$F.begin()$	$xy = F.Keys.GetEnumerator();$
		$xy.MoveNext();\ xy.Current$
Utolsó(F)	$--F.end()$	$-$
Alsókorlát(F,x)	$F.lower\_bound(x)$	$-$
Felsőkorlát(F,x)	$F.upper\_bound(x)$	$-$
Üresít(F)	$F.clear()$	$F.Clear$

7.3. Feladat: Leggyakoribb mérési érték

Egy kísérletben nagyszámú mérést végeztek, minden mérés eredménye egy természetes szám. A kutatók a mérési sorozatot elemzik. Szeretnék megadni azt a mérési értéket, amely a legtöbbször szerepel a mérési sorozatban.

Bemenet

A standard bemenet első sora a mérések n ($1 \leq n \leq 100000$) számát tartalmazza. A második sor tartalmazza az n mérési eredményt, ei ($1 \leq ei \leq 1\,000\,000\,000$).

Kimenet

A standard kimenet első sorába azt a mérési értéket kell írni, amely a legtöbbször szerepel a mérési sorozatban! Ha több ilyen lenne, akkor a legkisebbet kell kiírni.

Példa bemenet és kimenet

bemenet	**kimenet**
10 12 2 13 2 20 6 10 4 5 2	2 3

Korlátok

Időlimit: 0.2

Memória limit: 32 MiB

7.4. Megoldás

Vegyünk egy F függvényt, amely a mérési értékek gyakoriságát adja. Tehát $F[x]$ értéke az x mérési érték eddigi előfordulási száma. Az e_i mérési érték beolvasása után növeljük számlálóját; $F[e_i]++$ és ha igy nagyobb lett az eddigi maximumnál, akkor jegyezzük fel. Két megvalósítást mutatunk, az első tömbszerűen használja az F függvényt, a második pedig iterátort használ. Az utóbbi megoldás gyorsabb algoritmust eredményez.

7.4.1. C++ megvalósítás, tömbszerű megoldás

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int main (){
4     map<long, int> M;
5     map<long,int>::iterator it;
6     long n,x,maxadat,maxhany=0;
7     cin>>n;
8     for(int i=0;i<n;i++){
9        cin>>x;
10       it=M.find(x);                  //volt már x?
11       if(it==M.end()){               //x új adat
12          M[x]=1;
13       }else{                         //x már volt
14          int mx=M[x]+1;
15          M[x]=mx;                    //növeljük a számlálóját
16          if(mx>maxhany || mx==maxhany && x<maxadat){
17             maxadat=x;  maxhany=mx;//x az új leggyakoribb
18          }
19       }
20    }
21    cout<<maxadat<<" "<<maxhany<<endl;
22   return 0;
23 }
```

7.4.2. C++ megvalósítás iterátorral

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int main (){
4     map<long, int> M;
5     map<long,int>::iterator it;
6     long n,x,maxhany=1,maxadat,mx;
7     cin>>n;
8     for(int i=0;i<n;i++){
9        cin>>x;
10       it=M.find(x);                  //volt már x?
11       if(it==M.end()){               //x új adat
12          M.insert(pair<long,int>(x,1));
13       }else{
14          mx=(*it).second++;          //növeljük x számlálóját
15          if(mx>maxhany || mx==hany && x<maxadat){
16             maxhany=mx;                 //x az új leggyakoribb
17             maxadat=x;
18          }
19       }
20    }
21   cout<<maxadat<<" "<<maxhany<<endl;
22     return 0;
23 }
```

7.4.3. C# megvalósítás

```
1  using System; using System.Collections.Generic;
2  class Program{
3     public static void Main(){
4        int n;
5        n=Convert.ToInt32(Console.ReadLine());
6        var F = new SortedDictionary<long, int>();
7        long x,maxadat=0,maxhany=0;
8        string[] sor=Console.ReadLine().Split();
9        for(int i=0;i<n;i++){
10          x=Convert.ToInt64(sor[i]);
11          if(F.ContainsKey(x)){
12             int fx=F[x]+1;
13             F[x]=fx;
14             if(fx>maxhany || fx==maxhany && x<maxadat){
15                maxadat=x; maxhany=fx;
16             }
17          }else{
18             F.Add(x,1);
19          }
20       }
21       Console.WriteLine(maxadat+" "+maxhany);
22    }
23 }
```

7.5. Feladat: Hírlánc

Egy társaság hírláncot képezett. Ez azt jelenti, hogy a társaság minden tagja megadta, hogy ha hírt kap (vagy kivülről, vagy a társaság egy tagjától) akkor azt kinek továbbítja. Ezért ha valaki elindít egy hírt, akkor lehet, hogy visszakapja azt, de ilyenkor nem továbbítja azt.
Írjunk olyan programot, amely meghatározza a társaság azon tagjait, akik nem kapják vissza az általuk elindított hírt!

Bemenet

A standard bemenet első sora a társaság tagjainak n ($1 \leq n \leq 100\,000$) számát tartalmazza. A társaság tagjait személyi számukkal azonosítják, ami egy legfeljebb kilenc jegyű decimális szám. A további n sor mindegyike két egész számot tartalmaz, $x\ y$ ($1 < x, y < 1\,000\,000\,000$), ami azt jelenti, hogy az x tag az y tagnak továbbítja a hírt. A bemenetre teljesül, hogy mindenki, aki valakitől kap hírt, ő is továbbít valakinek.

Kimenet

A standard kimenet első sorába azon tagok m számát kell írni, akik nem kapják vissza az általuk elindított hírt! A második sor tartalmazza ezen tagok azonosítóit tetszőleges sorrendben.

Példa bemenet és kimenet

bemenet	kimenet
10 2 1 1 3 3 4 4 5 5 1 8 5 6 2 7 2 10 9 9 10	4 6 2 7 8

Korlátok

Időlimit: 0.2

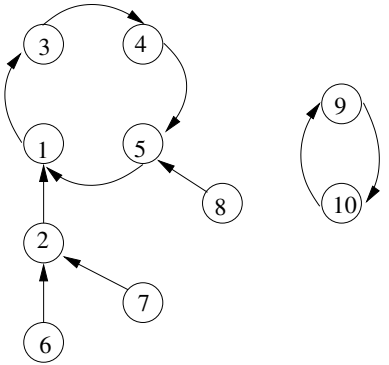
Memória limit: 32 MiB

7.6. Megoldás

Jelölje T a társaság tagjainak azonosítóit tartalmazó halmazt. A probléma bemenete megadható egy

$$Kuld : T \to T$$

függvénnyel, ahol minden $x \in T$ tagra $Kuld(x)$ az tag, akinek x átadja a hírt.



1. ábra. A példa ábrája

Tekintsük a $Kuld$ függvény ábráját, amelyen a körök a társaság tagjait, az irányított vonalak pedig az üzenet átadást reprezentálják. Az ábra alapján megadható, hogy kik nem kapják vissza a hírt; azok, akik nincsenek benne körben.

Tehát, ha meg tudjuk határozni, hogy kik vannak körben, akkor választ tudunk adni a feladat kérdésére is.

Színezzük a T halmaz elemit: Fehér, Szürke, Fekete és Piros színekkel. Kezdetben mindegyik színe

legyen Fehér.
Egy tetszőleges Fehér pontból indulva, Fehér pontokon át haladva a Kuld függvény szerint mindaddig, amíg nem Fehér pontba jutunk, és az érintett pontokat színezzük Szürkére.
Ha Szürke pontba jutottunk, akkor az körben van, ezért színezzük be a körben lévő pontokat Pirosra, a többit érintett pontot pedig Feketére.

7.6.1. C++ megvalósítás

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  enum Szinek {Feher, Szurke, Fekete, Piros};
4  int main (){
5     map<long, long> Kuld;
6     map<long, Szinek> Szin;
7     vector<long> T;
8     vector<long> Megoldas;
9     long n,x,y;
10    cin>>n;
11    for(int i=0;i<n;i++){//beolvasás
12       cin>>x>>y;
13       Kuld[x]=y;          //a Kuld függvény előállítása
14       T.push_back(x);     //x társasági tag
15       Szin[x]=Feher;
16    }
```

```
17      for(long x:T) if(Szin[x]==Feher){//x->...->xu lánc képzése
18          long xu=x;
19          while(Szin[xu]==Feher){
20              Szin[xu]=Szurke;
21              xu=Kuld[xu];
22          }
23          if(Szin[xu]==Szurke){//körben van xu
24              Szin[xu]=Piros;
25              long y=Kuld[xu];
26              while(y!=xu){
27                  Szin[y]=Piros;
28                  y=Kuld[y];
29              }
30          }
31          while(x!=xu){//a Szurke pontok Feketere festése
32              Szin[x]=Fekete;
33              x=Kuld[x];
34          }
35      }
36      for(long x:T) if(Szin[x]!=Piros)
37          Megoldas.push_back(x);
38      cout<<Megoldas.size()<<endl;        //eredmény kiíratása
39      for(long x:Megoldas)
40          cout<<x<<" "; cout<<endl;
41  }
```

7.6.2. C# megvalósítás

```
1  using System; using System.Collections.Generic;
2  class Program{
3     public enum Szinek{Feher, Szurke, Fekete, Piros};
4     public static void Main(){
5        var Kuld = new SortedDictionary<long, Long>();
6        var Szin = new SortedDictionary<long, Szinek>();
7        var T=new List<long>();
8        var Megoldas=new List<long>();
9        int n;
10       n=Convert.ToInt32(Console.ReadLine());
11       string[] sor;
12       long x,y;
13       for(int i=0;i<n;i++){
14          sor=Console.ReadLine().Split();
15          x=Convert.ToInt64(sor[0]);
16          y=Convert.ToInt64(sor[1]);
17          Kuld.Add(x,y));//a Kuld függvény előállítása
18          T.Add(x);
19          Szin[x]=Feher;
20       }
```

```
21          foreach(var x in T) if(Szin[x]==Feher){          //x->...->xu lánc k
22              long xu=x;
23              while(Szin[xu]==Feher){
24                Szin[xu]=Szurke;
25                xu=Kuld[xu];
26              }
27              if(Szin[xu]==Szurke){//körben van xu
28                Szin[xu]=Piros;
29                long y=Kuld[xu];
30                while(y!=xu){
31                    Szin[y]=Piros;
32                    y=Kuld[y];
33                 }
34              }
35              while(x!=xu){//a Szurke pontok Feketere festése
36                Szin[x]=Fekete;
37                x=Kuld[x];
38              }
39          }
40          Console.WriteLine(Megoldas.Count);      //eredmény kiíratása
41          foreach(var m in Megoldas)
42             Console.Write(m+"␣");
43          Console.WriteLine();
44      }
45  }
```