Vektor és Verem adattípus

Horváth Gyula
horvath@inf.elte.hu

2. Vektor (dinamikus tömb) adattípus

Értékhalmaz: $Vektor = \{\langle a_1, \ldots, a_n \rangle : a_i \in E, i = 1, \ldots, n, n \geq 0\}$
Műveletek argumentumai: $V : Vektor, x : E$

Előfeltétel	Művelet	Utófeltétel
	Üresit(V)	$V = \langle\rangle$
$V = \langle a_0, \ldots, a_{n-1} \rangle$	Végére(V,x)	$V = \langle a_0, \ldots, a_{n-1}, x \rangle$
$V = \langle a_0, \ldots, a_{n-1} \rangle \wedge 0 \leq i < n$	$Elem(V,i)$	$= a_i \wedge V = Pre(V)$
$V = \langle a_0, \ldots, a_{n-1} \rangle \wedge 0 \leq i < n$	$x = V[i]$	$x = a_i \wedge V = Pre(V)$
$V = \langle a_0, \ldots, a_{n-1} \rangle, \wedge 0 \leq i < n$	Módosit(V,i,x)	$V = \langle a_1, \ldots, a_{i-1}, x, a_{i-1}, \ldots, a_{n-1} \rangle$
$V = \langle a_0, \ldots, a_{n-1} \rangle, \wedge 0 \leq i < n$	$V[i] = x$	$V = \langle a_1, \ldots, a_{i-1}, x, a_{i-1}, \ldots, a_{n-1} \rangle$
$V = \langle a_0, \ldots, a_{n-1} \rangle$	Elemszám	$= n \wedge V = Pre(V)$
$V = \langle a_0, \ldots, a_{n-1} \rangle \wedge n > 0$	Utolsó(V,x)	$x = a_{n-1} \wedge V = Pre(V)$
$V = \langle a_0, \ldots, a_{n-1} \rangle \wedge n > 0$	Töröl(V)	$V = \langle a_0, \ldots, a_{n-2} \rangle$

2.1. A Vekor C++ és C# műveletei

Művelet	C++	C#
Típusdef	$vector<E>$	$List<E>$
Üresit(V)	$V.celar()$	$V.Clear()$
Végére(V,x)	$V.push\_back(x)$	$V.Add(x)$
$Elem(V,i)$	$V.at(i)$	
$V[i]$	$V[i]$	$V[i]$
Módosit(V,i,x)	$V[i]=x$	$V[i]=x$
Elemszám	$V.size()$	$V.Count$
Utolsó(V,x)	$V.back()$	
Töröl(V)	$V.pop\_back()$	

2.2. Feladat: sokat levelezők

Egy nagyvállalat nyilvántartja, hogy alkalmazottjai közül ki-kinek küldött levelet az adott hónapban. Az igazgató azt kéri, hogy adják meg azokat a dolgozókat, akik legalább K levelet küldtek.

Bemenet

A standard bemenet első sora három egész számot tartalmaz, a vállalat dolgozóinak n számát ($1 < n \leq 10\,000$), a hónapban küldött levelek m számát ($1 < m \leq 1\,000\,000$) és a k értékét ($1 \leq k \leq n$). A vállalat dolgozóit az $1, \ldots, n$ számokkal azonosítjuk.

Kimenet

A standard kimenet első sorába azon alkalmazottak s számát kell írni, akik legalább k levelet küldtek a hónapban. A következő s sor egy-egy ilyen alkalmazottat, és az általa küldött levelek címzettjeit tartalmazza. A sorban az első szám a leveleket küldő alkalmazott sorszáma legyen, ezt kövesse (egy-egy szóközzel elválasztva) azon alkalmazottak sorszámai (tetszőleges sorrendben), akiknek levelet küldött.

Példa bemenet és kimenet

bemenet

```
7 15 3
1 2
1 4
4 1
4 3
4 6
3 1
3 2
3 4
3 5
6 2
6 5
2 7
5 2
5 4
5 6
```

kimenet

```
3
3 1 2 4 5
4 1 3 6
5 2 4 6
```

Korlátok

Időlimit: 0.2

Memória limit: 32 MiB

2.3. Megoldás

minden x-re ($1 \leq n$) tekintsük a $Kapcsolat(x)$ halmazt

$$Kapcsolat(x) = \{y : x \text{ küldött levelet } y-nak\}$$

$$megoldas = \{x : |Kapcsolat(x)| \geq k\}$$

Minden x-re a $Kapcsolat(x)$ halmaz előállítható a beolvasáskor. A $Kapcsolat(x)$ halmaz ábrázolható Vektorral, mert csak hozzá kell venni egy új elemet, majd ha megoldás (elemszáma¿=k) akkor ki kell iratni, továbbá a kiíratásban az elemek sorrendje tetszőleges. Hasonlóan a $megoldas$ halmaz is ábrázolható Vektorral.

2.3.1. C++ megvalósítás

```
1  #include <iostream>
2  #include <vector>
3  #define maxN 10001
4
5  using namespace std;
6      vector<int> Kuld[maxN];
7      int n,k;
8  void Beolvas(){
9      int m, x,y;
10     cin>>n>>m>>k;
11     for(int i=1;i<=m;i++){
12         cin>>x>>y;
13         Kuld[x].push_back(y);
14     }
15 }
```

```
16 int main(){
17     vector<int> megoldas;
18     Beolvas();
19     for(int x=1;x<=n;x++){
20         if(Kuld[x].size()>=k)
21             megoldas.push_back(x);
22     }
23     cout<<megoldas.size()<<endl;
24     for(int x:megoldas){
25         cout<<x;
26         for(int y:Kuld[x])
27             cout<<" "<<y;
28         cout<<endl;
29     }
30
31     return 0;
32 }
```

2.3.2. C# megvalósítás

```
1  using System;
2  using System.Collections.Generic;
3
4  class Program{
5
6     static void Main(){
7        const int maxN=10000;
8        List<int>[] Kuld=new List<int>[maxN];
9        int n,m,k;
10       string[] sor=Console.ReadLine().Split();
11       n=Convert.ToInt32(sor[0]);
12       m=Convert.ToInt32(sor[1]);
13       k=Convert.ToInt32(sor[2]);
14       for(int i=1;i<=n;i++){
15          sor=Console.ReadLine().Split();
16          int x=Convert.ToInt32(sor[0]);
17          int y=Convert.ToInt32(sor[1]);
18          Kuld[x].Add(y);
19       }
```

```
20          List<int> Megoldas=new List<int>();
21          for(int x=1;x<=n;x++){
22              if(Kuld[x].Count>=k)
23                  Megoldas.Add(x);
24          }
25          Console.WriteLine(Megoldas.Count);
26          foreach(int x in Megoldas){
27              Console.WriteLine(x);
28              foreach(int y in Kuld[x])
29                Console.WriteLine(" "+Megoldas.Count);
30              Console.WriteLine();
31          }
32      }
33  }
```

3. Verem adattípus

3.1. Verem

Értékhalmaz: $Verem = \{\langle a_1, \ldots, a_n \rangle : a_i \in E, i = 1, \ldots, n, n \geq 0\}$
Műveletek argumentumai: $V : Verem, x : E$

Előfeltétel	Művelet	Utófeltétel
$V = \langle a_1, \ldots, a_n \rangle$	$VeremBe(V, x)$	$V = \langle a_1, \ldots, a_n, x \rangle$
$V = \langle a_1, \ldots, a_n \rangle \wedge n > 0$	VeremBől(V, x)	$V = \langle a_1, \ldots, a_{n-1} \rangle \wedge x = a_n$
$V = V$	ÜresE(V)	$= (Pre(V) = \langle\rangle) \wedge V = Pre(V)$
$V = \langle a_1, \ldots, a_n \rangle$	Elemszám(V)	$= n \wedge V = Pre(V)$
$V = \langle a_1, \ldots, a_n \rangle \wedge n > 0$	$Teteje(V)$	$= a_n \wedge V = Pre(V)$
$V = \langle a_1, \ldots, a_n \rangle \wedge n > 0$	Töröl(V)	$V = \langle a_1, \ldots, a_{n-1} \rangle$

3.2. A Verem C++ és C# műveletei

Művelet	C++	C#
Típusdef	$stack < E >$	$Stack < E >$
$VeremBe(V,x)$	$V.push(x)$	$V.Push(x)$
VereBől(V,x)	$x = V.top(); V.pop()$	$x = V.Pop()$
ÜresE(V)	$V.empty()$	$V.Count == 0$
Elemszám	$V.size()$	$V.Count$
$Teteje(V)$	$V.top()$	$V.Peek()$
Töröl(V)	$V.pop()$	$V.Pop()$

3.3. Feladat: Utcaseprő

Egy város minden utcája egyirányú, de bármely kereszteződésből bármely másikba el lehet jutni. Teljesül még az úthálózatra, hogy minden kereszteződésbe ugyanannyi utca fut be, mint amennyi onnan indul.
Adjunk meg egy olyan útvonalat az utcaseprő gépnek, amely minden utcán pontosan egyszer halad végég és végül visszaér oda, ahonnan indult.

Bemenet

A standard bemenet első a kereszteződések n számát ($1 < n \leq 1000$) és az utcák m számát ($1 < m \leq 10000$) tartalmazza. A kereszteződéseket az $1, \ldots, n$ számokkal azonosítjuk. A további m sor mindegyike egy utcát ad meg a két végpontjával p és q, ami azt jelenti, hogy az utca p-ből q felé egyirányú.

Kimenet

A standard kimenet első sorába kell kiírni az útvonalat, felsorolva a kereszteződéseket!
Több megoldás esetén bármelyik megadható.

Példa bemenet és kimenet

bemenet	kimenet
4 7 1 2 2 3 3 4 4 1 2 4 4 3 3 2	1 2 3 4 3 2 4 1

Korlátok

Időlimit: 0.2

Memória limit: 32 MiB

3.3.1. C++ megvalósítás

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int  main(){
4    int n,m,x,y;
5    cin>>n>>m;
6    stack<int> G[n+1];
7    for(int i=0;i<m; i++){
8      cin>>x>>y;
9      G[y].push(x);
10   }
11   stack<int> S;
12   S.push(1);
13   while(S.size()>0){
14     int p=S.top();
15     if(G[p].size()==0){
16       S.pop();
17       cout<<p<<"->";
18     }else{
19       int q=G[p].top(); G[p].pop();
20       S.push(q);
21     }
22   }
23   cout<<endl; }
```

3.3.2. C# megvalósítás

```
1  using System;
2  using System.Collections.Generic;
3
4  class Program{
5     public static void Main(){
6        int n,m;
7        string[] sor = Console.ReadLine().Split();
8        n=Convert.ToInt32(sor[0]);
9        m=Convert.ToInt32(sor[1]);
10       var G=new Stack<int>[n+1];
11       for(int i=1;i<=n;i++)
12          G[i]=new Stack<int>();
13       for(int i=0;i<m; i++){
14          sor = Console.ReadLine().Split();
15          int x=Convert.ToInt32(sor[0]);
16          int y=Convert.ToInt32(sor[1]);
17          G[y].Push(x);
18       }
```

```
19          var S=new Stack<int>();
20          S.Push(1);
21          while(S.Count>0){
22              int p=S.Peek();
23              if(G[p].Count==0){
24                  S.Pop();
25                  Console.Write(p+"->");
26              }else{
27                  int q=G[p].Pop();
28                  S.Push(q);
29              }
30          }
31          Console.WriteLine();
32      }
33  }
```