2. beadandó 2-es fázis

Neptun kód:	**AZXX1Z**	Név:	Soós Csaba
Beadás verziószáma:	1.		

Feladat

Programozási tételek összeépítése *

Új őrség küldése a Kínai Nagy Falra

A Kínai Nagy Falon N őrhelyet létesítettek. Közülük azonban csak M helyen van őrség. Két szomszédos őrhely közötti fal őrzött, ha legalább az egyik végén van őrség.

Készíts programot, amely megadja, hogy minimum hány helyre kell még őrséget küldeni, hogy minden fal őrzött legyen!

Bemenet

A *standard bemenet* első sorában az őrhelyek száma (1≤N≤100) és az őrségek száma (1≤M≤N) van, egy szóközzel elválasztva. A következő M sor az őrségek leírását tartalmazza, közülük az i-edik annak az őrhelynek a sorszáma, ahol az i-edik őrség van. Tudjuk, hogy minden helyen legfeljebb 1 őrség van.

Kimenet

A *standard kimenet* egyetlen sorába egyetlen egész számot kell írni: az új őrségek minimális számát, amivel elérhető, hogy minden fal őrzött legyen!

Példa

Bemenet	Kimenet
15 9 6 3 12 11 4 5 8 15 14	2

Korlátok

Időlimit: 0.1 mp.

Memórialimit: 32 MB

Specifikáció

Be: n∈N, m∈N, lista∈N[1..m]

Sa: orsegek∈N[1..n], orsegszamok∈N[1..m], kul∈N[1..m], hiany∈N[1..db], db∈N

Ki: uj∈N

2. beadandó 2-es fázis

Ef: (1<=n és n<=100) és (1<=m és m<=n)

Uf: ∀i∈[1..m]: (orsegek[lista[i]] = 1) és

orsegszamok = KIVÁLOGAT(i=1..n, orsegek[i] != 0, i).2 és

kul[1] = orsegszamok[1] - 1 és

∀i∈[2..m]: (kul[i] = orsegszamok[i] - orsegszamok[i-1] - 1) és

db = DARAB(i=1..m, kul[i]>1) és

hiany = KIVÁLOGAT(i=1..m, kul[i]>1, kul[i]).2 és

uj = SZUMMA(i=1..db, hiany[i]/2)

Sablon

Másolás sablon

i		y
e	→	f(e)
e+1	→	f(e+1)
...	→	...
u	→	f(u)

Feladat

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u]→H függvény. Rendeljük az [e..u] intervallum minden értékéhez az f függvény hozzá tartozó értékét!

Specifikáció

```
Be: e∈Z, u∈Z
Ki: y∈H[1..u-e+1]
Ef: -
Uf: ∀i∈[e..u]:(y[i-e+1]=f(i))
```

Rövidítve:

```
Uf: y=MÁSOL(i=e..u, f(i))
```

Algoritmus

```
i=e..u
  y[i-e+1]:=f(i)
```

Változó
i:Egész

Megszámolás
sablon

i	T(i)	érték
e	IGAZ	1
e+1	HAMIS	0
...	HAMIS	0
u	IGAZ	1
	db=	2

Feladat

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg, hogy az [e..u] intervallumon a T feltétel hányszor veszi fel az igaz értéket!

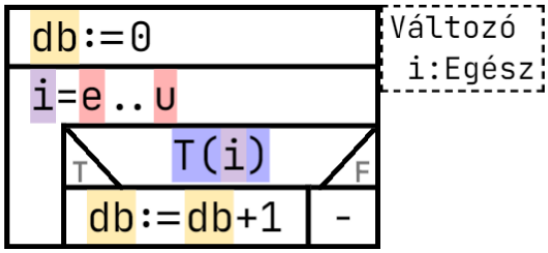
Specifikáció

```
Be: e∈Z, u∈Z
Ki: db∈N
Ef: -
Uf: db=SZUMMA(i=e..u, 1, T(i))
```

Rövidítve:

```
Uf: db=DARAB(i=e..u, T(i))
```

Algoritmus



Összegzés
sablon

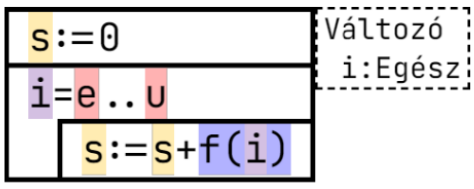
Feladat

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u]→H függvény. A H halmaz elemein értelmezett az összeadás művelet. Határozzuk meg az f függvény [e..u] intervallumon felvett értékeinek az összegét, azaz a $\sum_{i=e}^{u} f(i)$ kifejezés értékét! (e>u esetén ennek az értéke definíció szerint a nulla elem)

Specifikáció

```
Be: e∈Z, u∈Z
Ki: s∈H
Ef: -
Uf: s=SZUMMA(i=e..u, f(i))
```

Algoritmus



Kiválogatás sablon

i		T(i)		f(i)
e	→	HAMIS		
e+1	→	IGAZ	→ 1	f(e+1)
e+2	→	IGAZ	→ 2	f(e+2)
u	→	HAMIS		

Feladat

Adott az egész számok egy [e..u] intervalluma, egy ezen értelmezett T:[e..u]→Logikai feltétel és egy f:[e..u]→H függvény. Határozzuk meg az f függvény az [e..u] intervallum azon értékeinél felvett értékeit, amelyekre a T feltétel teljesül!

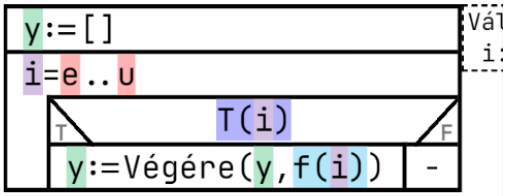
Specifikáció

```
Be: e∈Z, u∈Z
Ki: y∈H[1..]
Ef: -
Uf: ∀i∈[1..hossz(y)]:(
      ∃j∈[e..u]:T(j) és y[i]=f(j))
 és y⊆(f(e),f(e+1),...,f(u))
```

Rövidítve:

```
Uf: (,y)=KIVÁLOGAT(i=e..u,T(i),f(i))
```

Algoritmus



Visszavezetés

MÁSOL: ∀i∈[e..u]:(y[i-e+1]=f(i))

e..u ~ 1..m

y[i-e+1] ~ orsegek[lista[i]]

f(i) ~ 1

KIVÁLOGAT:

y ~ orsegszamok

e..u ~ 1..n

T(i) ~ orsegek[i] != 0

f(i) ~ i

MÁSOL: ∀i∈[e..u]:(y[i-e+1]=f(i))

e..u ~ 2..m

y[i-e+1] ~ kul[i]

f(i) ~ orsegszamok[i] - orsegszamok[i-1] - 1

2. beadandó 2-es fázis

DARAB:

db ~ db

e..u ~ 1..m

T(i) ~ kul[i] > 1

KIVÁLOGAT:

y ~ hiany

e..u ~ 1..m

T(i) ~ kul[i]>1

f(i) ~ kul[i]

SZUMMA:

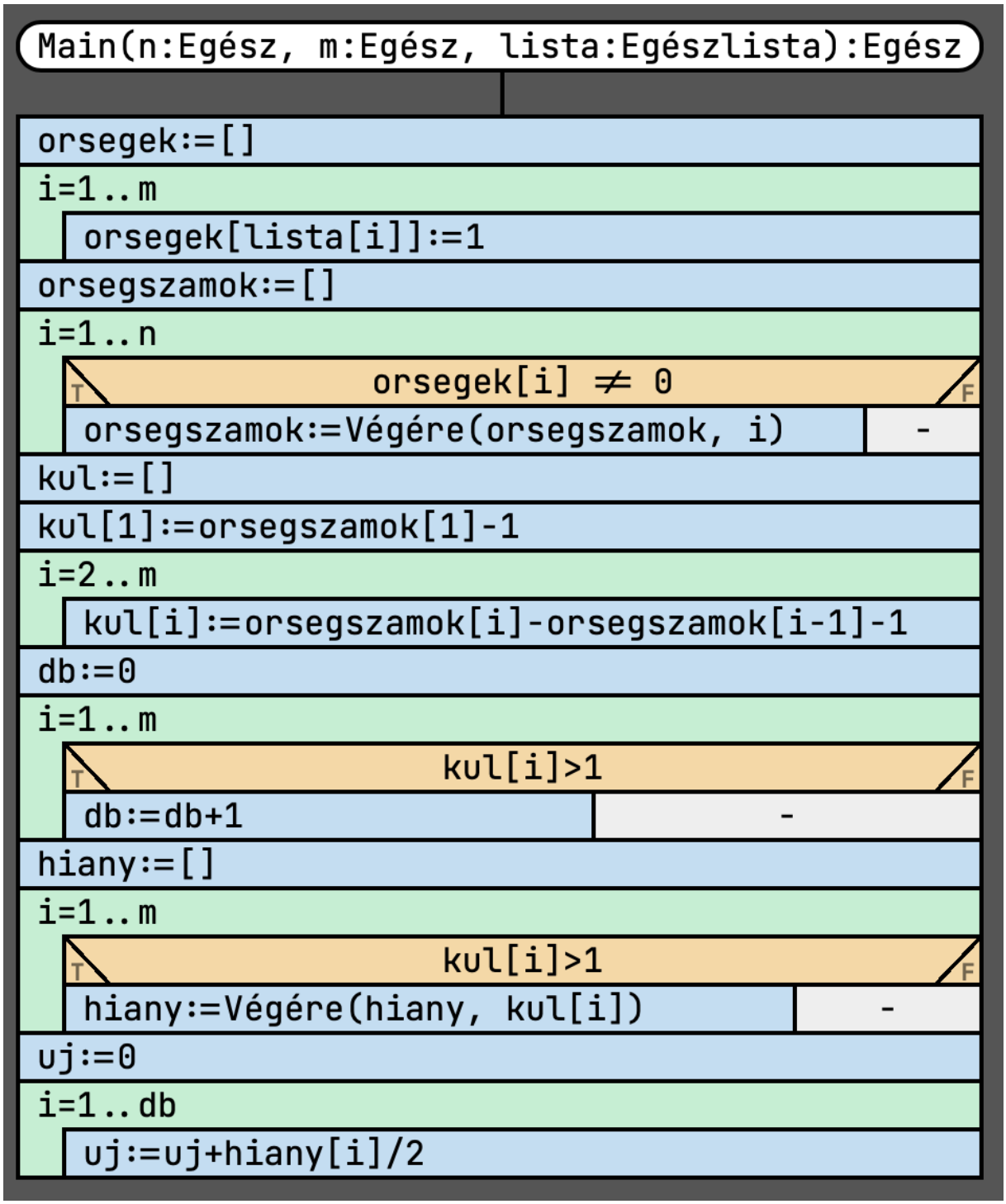
s ~ uj

e..u ~ 1..db

f(i) ~ hiany[i]/2

2. beadandó 2-es fázis

Algoritmus



Kód (C#)

```
using System;

namespace uj_orseg_kuldese_a_kinai_nagy_falra
{
    class Program
```

2. beadandó 2-es fázis

```
{
    static void Main(string[] args)
    {
        string[] sortomb = Console.ReadLine().Split(' ');
        int n = int.Parse(sortomb[0]);
        int m = int.Parse(sortomb[1]);

        int[] lista = new int[m];
        for (int i = 0; i < m; i++)
            lista[i] = (int.Parse(Console.ReadLine()));

        int[] orsegek = new int[n];
        for (int i = 0; i < m; i++)
            orsegek[lista[i]-1] = 1;

        int[] orsegszamok = new int[m];
        int j = 0;
        for (int i = 0; i < n; ++i){
            if (orsegek[i] != 0){
                orsegszamok[j] = i+1;
                ++j;
            }
        }

        int[] kul = new int[m+1];
        kul[0] = orsegszamok[0] - 1;
        for (int i = 1; i < m; i++)
            kul[i] = orsegszamok[i] - orsegszamok[i - 1] - 1;
        if (orsegszamok[orsegszamok.Length - 1] != n)
        {
            kul[kul.Length - 1] = n - orsegszamok[orsegszamok.Length-1];
```

2. beadandó 2-es fázis

```
        }

        int db = 0;
        for (int i = 0; i < m+1; i++){
          if (kul[i]>1){
            db = db + 1;
          }
        }

        int[] hiany = new int[db];
        j = 0;
        for (int i = 0; i < m+1; i++){
          if (kul[i]>1){
            hiany[j] = kul[i];
            ++j;
          }
        }

        int uj = 0;
        for (int i = 0; i < db; i++){
          uj = uj + hiany[i] / 2;
        }

        Console.WriteLine(uj);
      }
    }
}
/*
15 9
6
3
```

2. beadandó 2-es fázis

12

11

4

5

8

15

14

*/

2. beadandó 2-es fázis

Bíró pontszám és képernyőkép

2. beadandó 2-es fázis

Összpont: 100/

Teszt#	Pont	...Verdikt...	futási idő
1.1	3/3	Helyes	0.029 sec
2.1	3/3	Helyes	0.030 sec
3.1	3/3	Helyes	0.032 sec
4.1	3/3	Helyes	0.031 sec
5.1	3/3	Helyes	0.031 sec
6.1	3/3	Helyes	0.032 sec
7.1	3/3	Helyes	0.031 sec
8.1	3/3	Helyes	0.032 sec
9.1	4/4	Helyes	0.032 sec
10.1	4/4	Helyes	0.032 sec
11.1	4/4	Helyes	0.031 sec
12.1	4/4	Helyes	0.032 sec
13.1	4/4	Helyes	0.031 sec
14.1	4/4	Helyes	0.032 sec
15.1	4/4	Helyes	0.031 sec
16.1	4/4	Helyes	0.032 sec
17.1	4/4	Helyes	0.033 sec
18.1	4/4	Helyes	0.033 sec
19.1	4/4	Helyes	0.031 sec
20.1	4/4	Helyes	0.032 sec
21.1	4/4	Helyes	0.029 sec
22.1	4/4	Helyes	0.031 sec
23.1	4/4	Helyes	0.031 sec
24.1	4/4	Helyes	0.032 sec
25.1	4/4	Helyes	0.032 sec
26.1	4/4	Helyes	0.032 sec
27.1	4/4	Helyes	0.022 sec

Beadva: 2024-12-08 19:22:24.0

2. beadandó 2-es fázis

Saját tesztfájlok

1.

10 4

2

4

7

9

2.

8 3

1

4

7

Github repo:

https://github.com/csabisoos/elte/tree/main/1.%20felev/programozas%20gyak/beadando_2