ELTE | IK

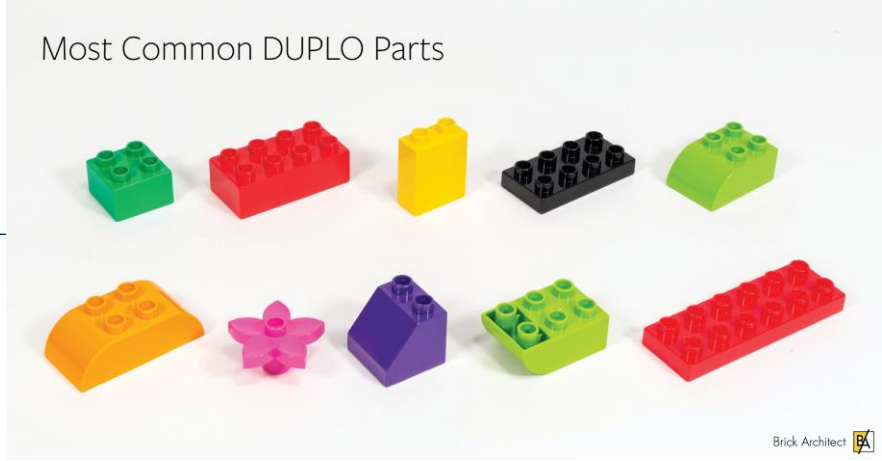
PROGRAMOZÁS

Mintamegoldás

Horváth Győző

Ismétlés

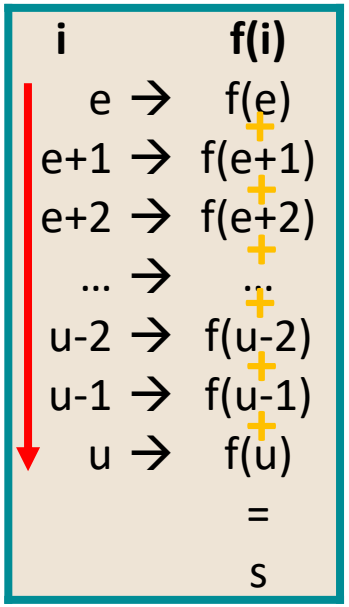
Programozási minták



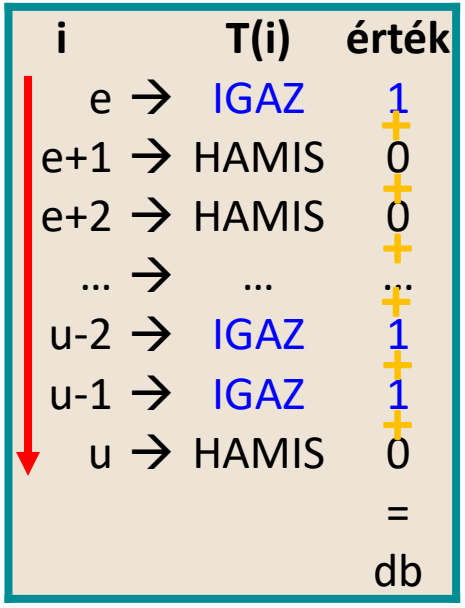
1. Összegzés
2. Megszámolás
3. Maximumkiválasztás
   a. Minimumkiválasztás
4. Feltételes maximumkeresés
5. Keresés
6. Eldöntés
   a. Mind eldöntés
7. Kiválasztás
8. Másolás
9. Kiválogatás

Programozási minták

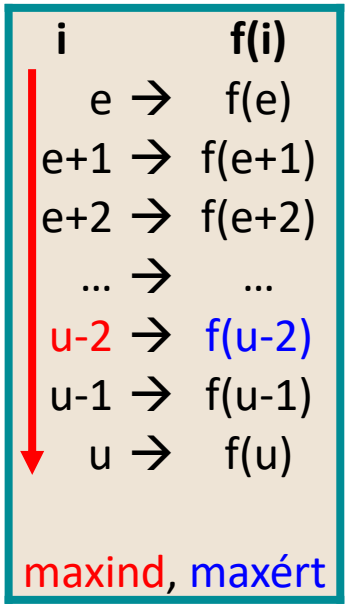
Összegzés



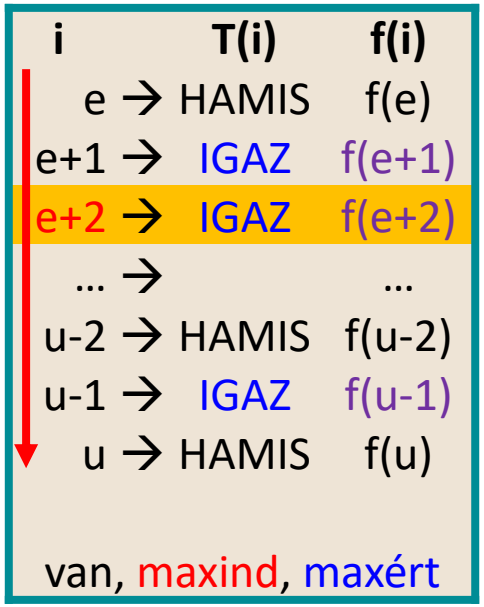
Megszámolás



Maximum kiválasztás

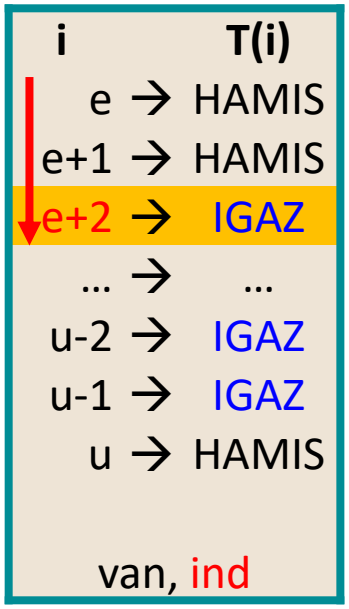


Feltételes maximumkeresés

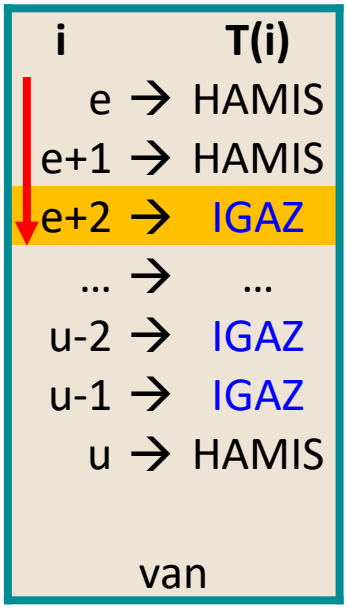


Programozási minták

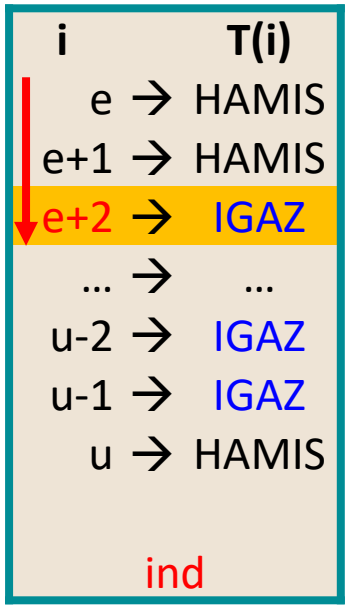
Keresés



Eldöntés

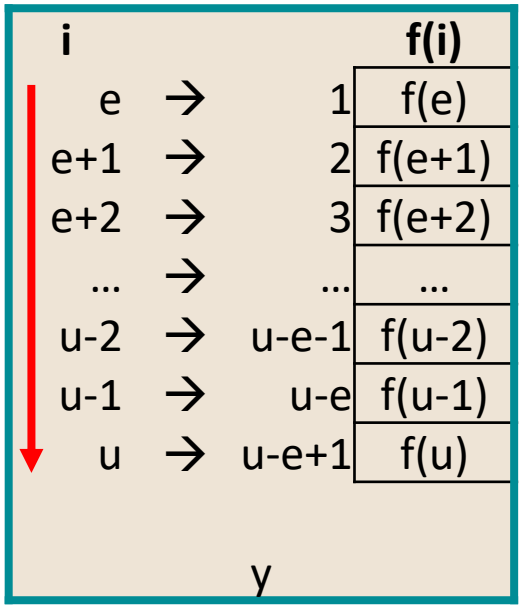


Kiválasztás

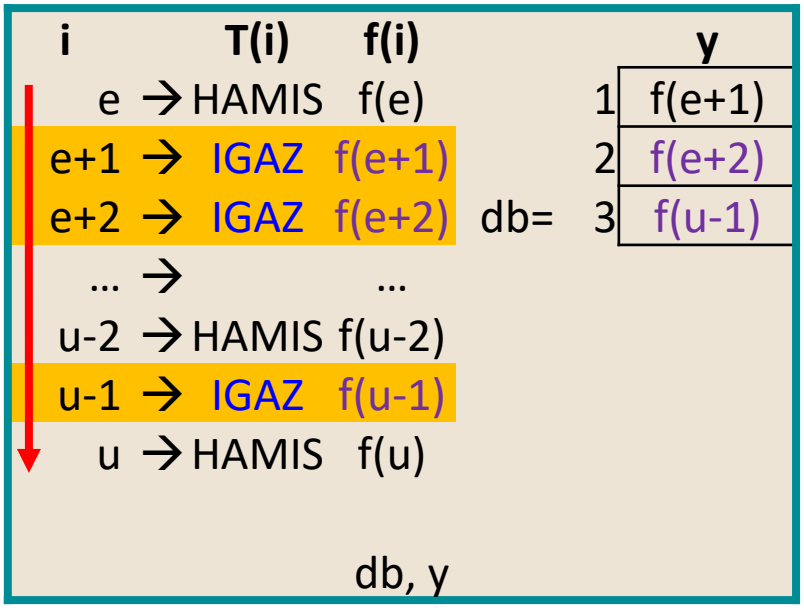


Programozási minták

Másolás



Kiválogatás



Feladatmegoldási minta

Feladatmegoldási minta
gyorsabb vonat az előzőnél

Feladat a Mesterről

Gyorsabb vonat az előzőnél

Ismerjük N vonat menetidejét Budapestről Siófokra.

Írj programot, amely megad egy vonatot, amely gyorsabb, mint az előző!

Bemenet

A *standard bemenet* első sorában a vonatok száma van (1≤N≤100). A következő N sor mindegyike egy-egy egész számot tartalmaz, az egyes vonatok menetidejét (1≤M≤1000).

Kimenet

A *standard kimenet* első sorába egy az előzőnél gyorsabb vonat sorszámát kell írni (ha több ilyen is van, akkor az elsőt)! Ha nincs ilyen vonat, akkor -1-et kell írni!

Példa

Bemenet	Kimenet
6 118 200 199 116 200 122	3

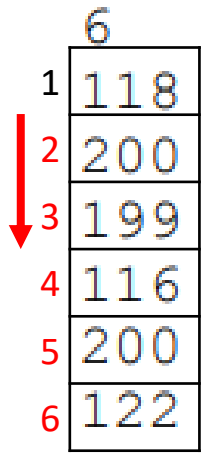
Feladatmegoldási minta
gyorsabb vonat az előzőnél

Biztosan van ilyen vonat? Ha igen, akkor melyik az?

→ keresés: adott tulajdonságú elem létezése és helye

Feladat:

Adj meg egy előzőnél gyorsabb vonatot!



Specifikáció:

```
Be: n∈N, midő∈N[1..n]
Ki: van∈L, melyik∈N
Ef: -
Uf: (van,melyik)=KERES(i=2..n, midő[i]<midő[i-1])
```

1. Mik az intervallum határai? (2..6)
2. Milyen tulajdonságot vizsgálunk az intervallum egyes pontján?
3. Milyen néven tároljuk a keresés eredményeit?

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

van, ind

Feladatmegoldási minta
gyorsabb vonat az előzőnél

Feladatsablon
(mintafeladat)

```
Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: (van,ind)=KERES(i=e..u,
                    T(i))
```

Előzőnél gyorsabb vonat
(konkrét feladat)

```
Be: n∈N, midő∈N[1..n]
Ki: van∈L, melyik∈N
Ef: -
Uf: (van,melyik)=KERES(i=2..n,
                       midő[i]<midő[i-1])
```

```
ind   ~ melyik
e..u  ~ 2..n
T(i)  ~ midő[i]<midő[i-1]
```

```
ind:=e
ind⩽u és nem T(ind)
    ind:=ind+1
van:=ind⩽u
```

```
melyik:=2
melyik⩽n és
nem midő[melyik]<midő[melyik-1]
    melyik:=melyik+1
van:=melyik⩽n
```

Feladatmegoldási minta
gyorsabb vonat az előzőnél

Kódolás alapsablonja

```
static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)



  // Beolvasás (specifikáció be)



  // Feldolgozás (algoritmus, stuki)



  // Kiírás (specifikáció ki)



}
```

Feladatmegoldási minta
gyorsabb vonat az előzőnél

Deklarálás

```
static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)
  int n;
  int[] mido;
  bool van;
  int melyik;
  // Beolvasás (specifikáció be)



  // Feldolgozás (algoritmus, stuki)



  // Kiírás (specifikáció ki)



}
```

Be: n∈N, midő∈N[1..n]
Ki: van∈L, melyik∈N

Feladatmegoldási minta
gyorsabb vonat az előzőnél

Beolvasás

```
static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)
  int n;
  int[] mido;
  bool van;
  int melyik;
  // Beolvasás (specifikáció be)
  Console.Write("n = ");
  int.TryParse(Console.ReadLine(), out n);
  mido = new int[n];
  for (int i = 1; i <= n; i++) {
    Console.Write("{0}. menetido = ", i);
    int.TryParse(Console.ReadLine(), out mido[i - 1]);
  }
  // Feldolgozás (algoritmus, stuki)



  // Kiírás (specifikáció ki)



}
```

Be: n∈N,

midő∈N[1..n]

Feladatmegoldási minta
gyorsabb vonat az előzőnél

Feldolgozás

```
static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)
  int n;
  int[] mido;
  bool van;
  int melyik;
  // Beolvasás (specifikáció be)
  Console.Write("n = ");
  int.TryParse(Console.ReadLine(), out n);
  mido = new int[n];
  for (int i = 1; i <= n; i++) {
    Console.Write("{0}. menetido = ", i);
    int.TryParse(Console.ReadLine(), out mido[i - 1]);
  }
  // Feldolgozás (algoritmus, stuki)
  melyik = 2;
  while (melyik <= n && !(mido[melyik - 1] < mido[melyik - 1 - 1])) {
    melyik = melyik + 1;
  }
  van = melyik <= n;
  // Kiírás (specifikáció ki)



}
```

```
melyik:=2
melyik⩽n és
nem midő[melyik]<midő[melyik-1]
    melyik:=melyik+1
van:=melyik⩽n
```

Feladatmegoldási minta
gyorsabb vonat az előzőnél

```
static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)
  int n;
  int[] mido;
  bool van;
  int melyik;
  // Beolvasás (specifikáció be)
  Console.Write("n = ");
  int.TryParse(Console.ReadLine(), out n);
  mido = new int[n];
  for (int i = 1; i <= n; i++) {
    Console.Write("{0}. menetido = ", i);
    int.TryParse(Console.ReadLine(), out mido[i - 1]);
  }
  // Feldolgozás (algoritmus, stuki)
  melyik = 2;
  while (melyik <= n && !(mido[melyik - 1] < mido[melyik - 1 - 1])) {
    melyik = melyik + 1;
  }
  van = melyik <= n;
  // Kiírás (specifikáció ki)
  if (van) {
    Console.WriteLine("Van, a(z) {0}. vonat gyorsabb az előzőnél.", melyik);
  }
  else {
    Console.WriteLine("Nincs gyorsabb vonat az előzőnél.");
  }
}
```

Kiírás

```
n = 6
1. menetido = 118
2. menetido = 200
3. menetido = 199
4. menetido = 116
5. menetido = 200
6. menetido = 122
Van, a(z) 3. vonat gyorsabb az előzőnél.
```

Ki: van∈L, melyik∈N

Feladatmegoldási minta
gyorsabb vonat az előzőnél

```
static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)
  int n;
  int[] mido;
  bool van;
  int melyik;
  // Beolvasás (specifikáció be)
  Console.Write("n = ");
  int.TryParse(Console.ReadLine(), out n);
  mido = new int[n];
  for (int i = 1; i <= n; i++) {
    Console.Write("{0}. menetido = ", i);
    int.TryParse(Console.ReadLine(), out mido[i - 1]);
  }
  // Feldolgozás (algoritmus, stuki)
  melyik = 2;
  while (melyik <= n && !(mido[melyik - 1] < mido[melyik - 1 - 1])) {
    melyik = melyik + 1;
  }
  van = melyik <= n;
  // Kiírás (specifikáció ki)
  if (van) {
    Console.WriteLine(melyik);
  }
  else {
    Console.WriteLine(-1);
  }
}
```

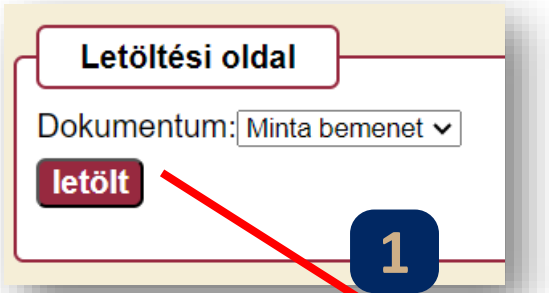
Kiírás módosítás

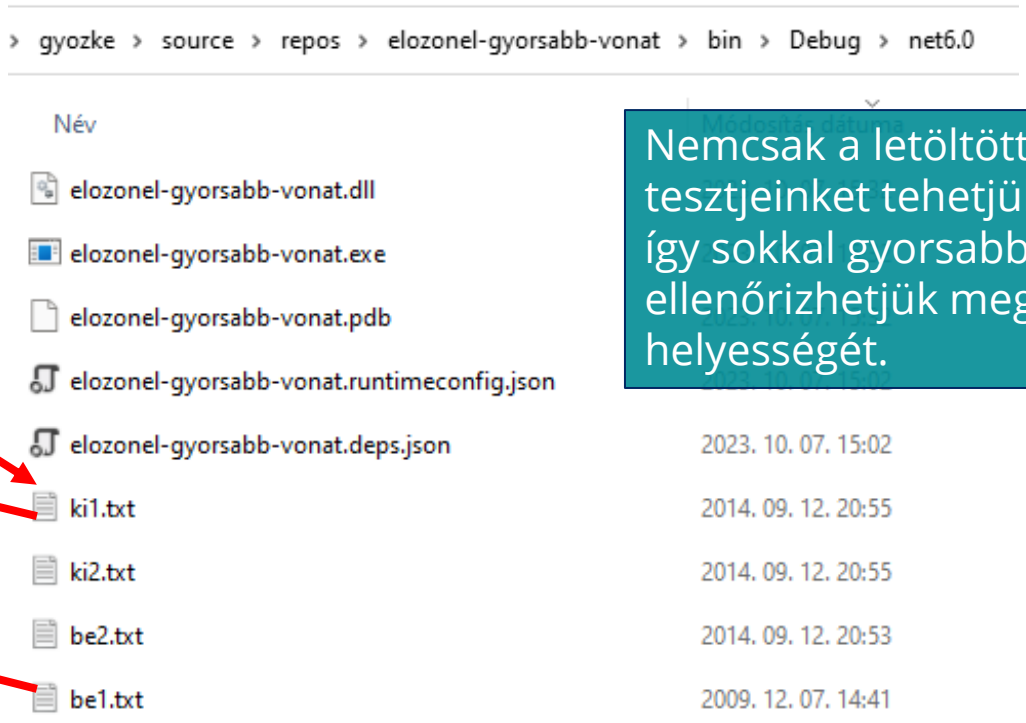
```
n = 6
1. menetido = 118
2. menetido = 200
3. menetido = 199
4. menetido = 116
5. menetido = 200
6. menetido = 122
3
```

Kimenet

A *standard kimenet* első sorába egy az előzőnél gyorsabb vonat sorszámát kell írni (ha több ilyen is van, akkor az elsőt)! Ha nincs ilyen vonat, akkor -1-et kell írni!

Feladatmegoldási minta
gyorsabb vonat az előzőnél





Nemcsak a letöltött, de saját tesztjeinket tehetjük fájlokba, így sokkal gyorsabban ellenőrizhetjük megoldásunk helyességét.

1

2

3

4

be1.txt

```
4
90
80
70
60
```

ki1.txt

```
2
```

```
c:\Users\gyozke\source\repos\elozonel-gyorsabb-
vonat\bin\Debug\net6.0>elozonel-gyorsabb-vonat.exe <be1.txt >ki1.txt
```

ki1.txt (saját)

```
n = 1. menetido = 2. menetido = 3. menetido =
4. menetido = 2
```

Feladatmegoldási minta
gyorsabb vonat az előzőnél

```
static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)
  int n;
  int[] mido;
  bool van;
  int melyik;
  // Beolvasás (specifikáció be)
  Console.Error.Write("n = ");
  int.TryParse(Console.ReadLine(), out n);
  mido = new int[n];
  for (int i = 1; i <= n; i++) {
    Console.Error.Write("{0}. menetido = ", i);
    int.TryParse(Console.ReadLine(), out mido[i - 1]);
  }
  // Feldolgozás (algoritmus, stuki)
  melyik = 2;
  while (melyik <= n && !(mido[melyik - 1] < mido[melyik - 1 - 1])) {
    melyik = melyik + 1;
  }
  // Kiírás (specifikáció ki)
  van = melyik <= n;
  if (van) {
    Console.WriteLine(melyik);
  }
  else {
    Console.WriteLine(-1);
  }
}
```

```
n = 6
1. menetido = 118
2. menetido = 200
3. menetido = 199
4. menetido = 116
5. menetido = 200
6. menetido = 122
3
```

```
c:\Users\gyozke\source\repos\elozonel-gyorsabb-vonat\bin\Debug\net6.0>elozonel-gyorsabb-vonat.exe <be1.txt >ki1.txt
```

ki1.txt
2

ki1.txt (saját)
2

Feladatmegoldási minta
gyorsabb vonat az előzőnél

```
using System;

static void Main(string[] args) {
  // Deklarálás (változók, specifikáció be,ki)
  int n;
  int[] mido;
  bool van;
  int melyik;
  // Beolvasás (specifikáció be)
  n = int.Parse(Console.ReadLine());
  mido = new int[n];
  for (int i = 0; i < n; i = i + 1) {
    mido[i] = int.Parse(Console.ReadLine());
  }
  // Feldolgozás (algoritmus)
  melyik = 2;
  while (melyik <= n && !(mido[melyik - 1] < mido[melyik - 1 - 1])) {
    melyik = melyik + 1;
  }
  van = melyik <= n;
  // Kiírás (specifikáció ki)
  if (van) {
    Console.WriteLine(melyik);
  }
  else {
    Console.WriteLine(-1);
  }
```

Utolsó beadás eredménye

Összpont: 100/100

Teszt#	Pont	...Üzenet...	Futási idő
1.1	3/3	Helyes	0.032 sec
2.1	3/3	Helyes	0.032 sec
3.1	3/3	Helyes	0.035 sec
4.1	3/3	Helyes	0.031 sec
5.1	3/3	Helyes	0.032 sec
6.1	3/3	Helyes	0.033 sec
7.1	3/3	Helyes	0.033 sec
8.1	3/3	Helyes	0.032 sec
9.1	4/4	Helyes	0.037 sec
10.1	4/4	Helyes	0.033 sec
11.1	4/4	Helyes	0.032 sec
12.1	4/4	Helyes	0.036 sec
13.1	4/4	Helyes	0.037 sec

Feladatmegoldási minta

Feladatmegoldási minta
legmagasabb tanuló

Ismerjük egy osztály tanulóinak magasságait. Mondd meg, hányadik diák a legmagasabb, és a magasságát is!

	mag	
	m	cm
1	1	52
2	1	77
3	1	23
4	1	65

hol =2,
maxmag=(m:1, cm:77)

Feladatmegoldási minta
legmagasabb tanuló

Feladat:

maximumkiválasztás

Ismerjük egy osztály tanulóinak magasságait. Mondd meg, hányadik diák a **leg**magasabb, és a magasságát is!

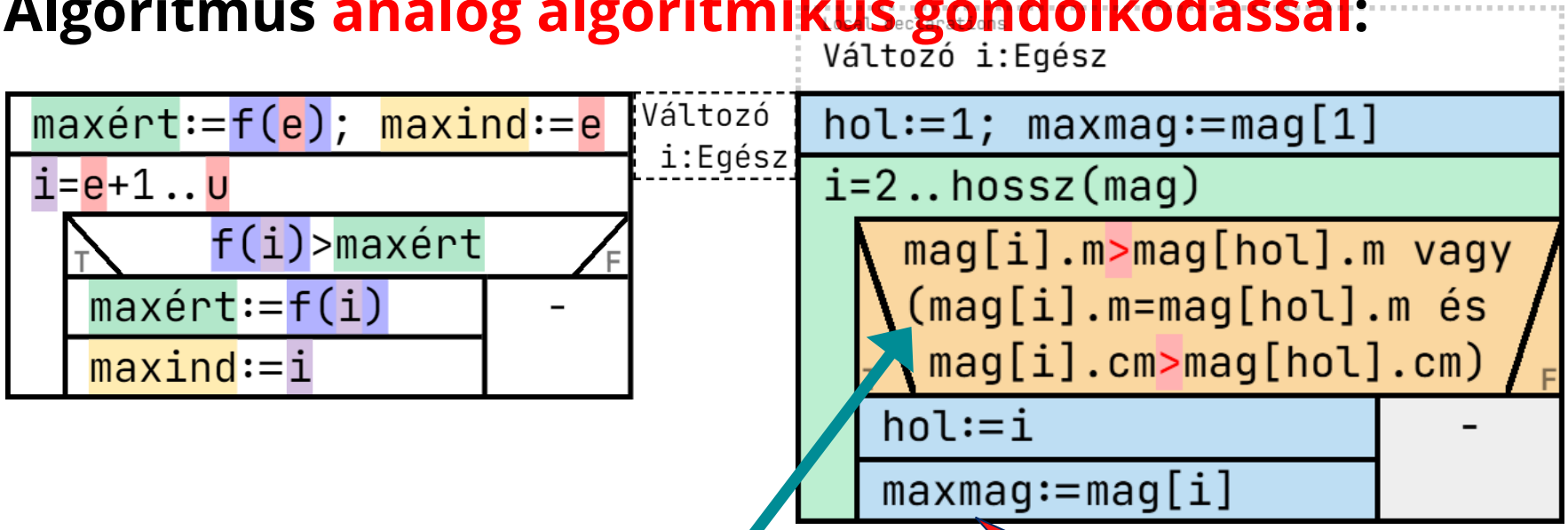
	mag	
	m	cm
1	1	52
2	1	77
3	1	23
4	1	65

Specifikáció:

```
Be: mag∈Mag[], Mag=(m:N x cm:N)
Ki: hol∈N, maxmag∈Mag
Ef: hossz(mag)>=1
Uf: hol∈[1..hossz(mag)] és
    ∀i∈[1..hossz(mag)]:(
     mag[hol].m > mag[i].m vagy
     (mag[hol].m = mag[i].m és mag[hol].cm >= mag[i].cm)) és
    maxmag=mag[hol]
```

Feladatmegoldási minta
legmagasabb tanuló

Algoritmus analóg algoritmikus gondolkodással:

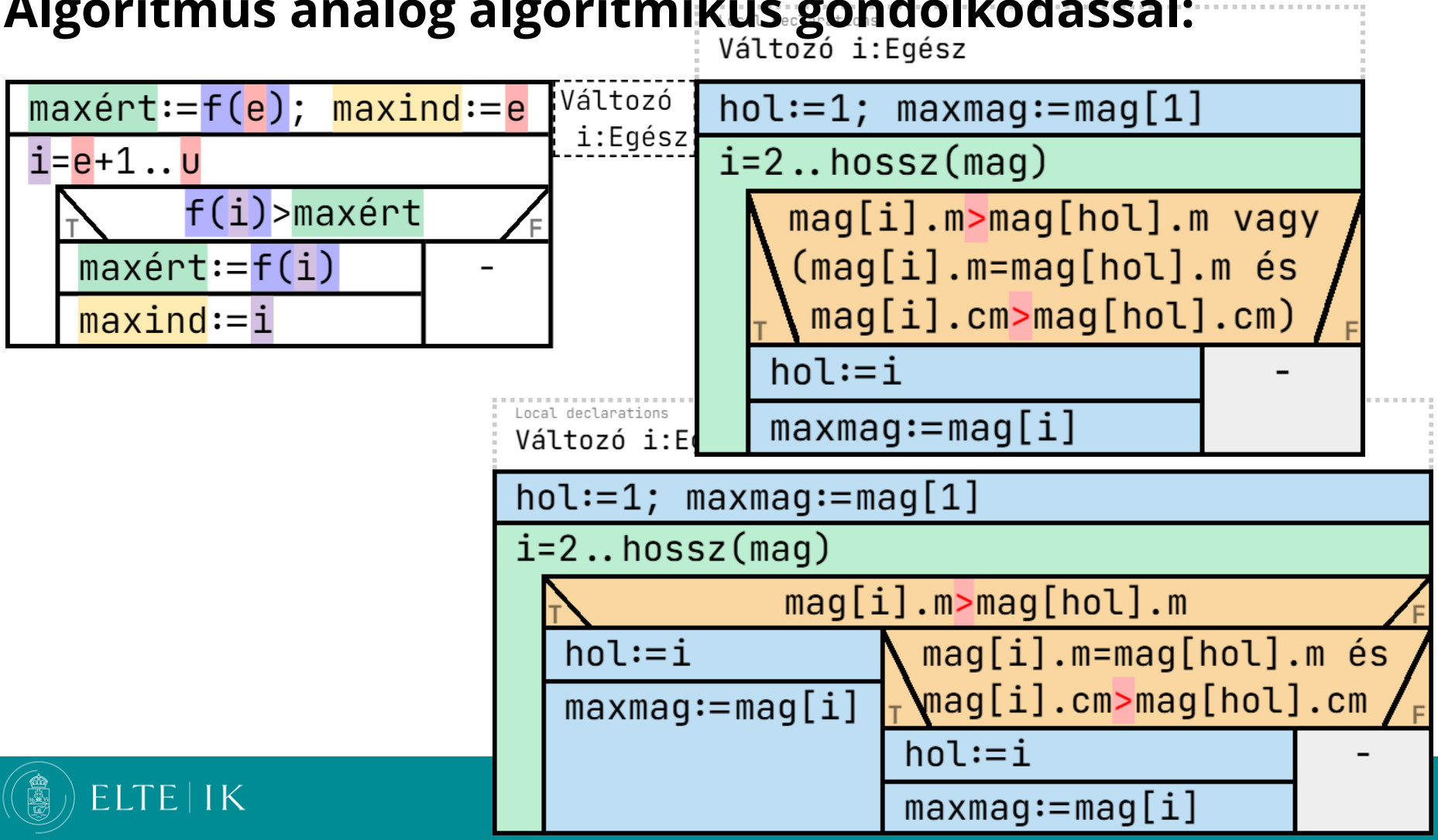


Be: mag∈Mag[], Mag=(m:N x cm:N)
Ki: hol∈N, maxmag∈Mag
Ef: hossz(mag)>=1
Uf: hol∈[1..hossz(mag)] és
∀i∈[1..hossz(mag)]:(
mag[hol].m > mag[i].m vagy
(mag[hol].m = mag[i].m és mag[hol].cm >= mag[i].cm)) és
maxmag=mag[hol]

Ez nem visszavezetés!

Feladatmegoldási minta
legmagasabb tanuló

Algoritmus analóg algoritmikus gondolkodással:



Feladatmegoldási minta
legmagasabb tanuló

Feladat:

maximumkiválasztás

Ismerjük egy osztály tanulóinak magasságait. Mondd meg, hányadik diák a legmagasabb, és a magasságát is!

Specifikáció:

```
Be: mag∈Mag[], Mag=(m:N x cm:N)
Ki: hol∈N, maxmag∈Mag
Ef: hossz(mag)>=1
Uf: (maxind,maxért)=MAX(i=e..u, f(i))
```

i		f(i)
e	→	f(e)
e+1	→	f(e+1)
e+2	→	f(e+2)
…	→	…
u-2	→	f(u-2)
u-1	→	f(u-1)
u	→	f(u)

maxind, maxért

	mag m	cm
1	1	52
2	1	77
3	1	23
4	1	65

1. Mik az intervallum határai?
2. Milyen értékeket veszünk az intervallum egyes pontján, amik közül a legnagyobb kikerül?
3. Milyen néven tároljuk az eredményeket?

A helyes f(i) megtalálása a cél!

Hogyan hasonlíthatunk össze rekordokat?

Hogyan rendelhetünk egyetlen értéket a rekordhoz, amiket összehasonlíthatunk?

Feladatmegoldási minta
legmagasabb tanuló

i		f(i)
e	→	f(e)
e+1	→	f(e+1)
e+2	→	f(e+2)
…	→	…
u-2	→	f(u-2)
u-1	→	f(u-1)
u	→	f(u)

maxind, maxért

Feladat:

maximumkiválasztás

Ismerjük egy osztály tanulóinak magasságait. Mondd meg, hányadik diák a legmagasabb, és a magasságát is!

Specifikáció:

```
Be: mag∈Mag[], Mag=(m:N x cm:N)
Sa: maxcm∈N
Ki: hol∈N, maxmag∈Mag
Ef: hossz(mag)>=1
Uf: (hol,maxcm)=MAX(i=1..hossz(mag),
        mag[i].m*100+mag[i].cm) és
    maxmag=mag[hol]
```

	mag m	cm	m*100+cm
1	1	52	152
2	1	77	177
3	1	23	123
4	1	65	165

Az f(i) a rekord adatkettesét képezi le egyetlen értékre, és ez kerül összehasonlításra.

Feladatmegoldási minta
legmagasabb tanuló

Feladatsablon

```
Be: e∈Z, u∈Z

Ki: maxind∈Z, maxért∈H
Ef: e<=u
Uf: (maxind,maxért)=
        MAX(i=e..u,f(i))
```

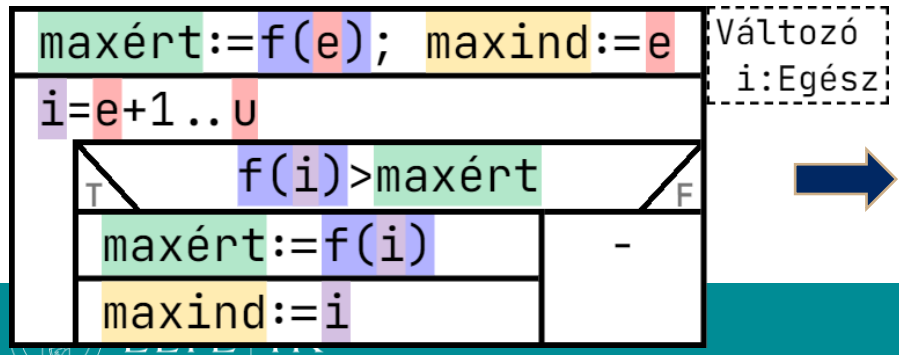
Legmagasabb tanuló

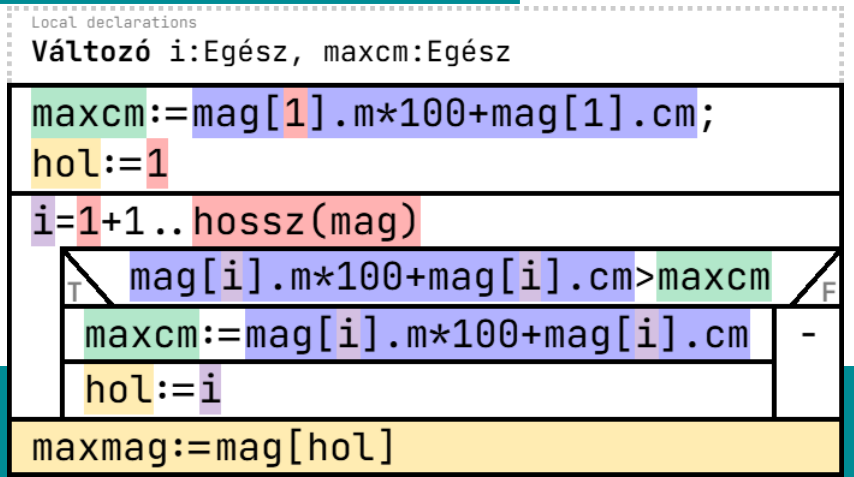
```
Be: mag∈Mag[], Mag=(m:N x cm:N)

Sa: maxcm∈N
Ki: hol∈N, maxmag∈Mag
Ef: hossz(mag)>=1
Uf: (hol,maxcm)=MAX(i=1..hossz(mag),
        mag[i].m*100+mag[i].cm) és
        maxmag=mag[hol]
```

Visszavezetés:

maxind, maxért	~	hol, maxcm
e..u	~	1..hossz(mag)
f(i)	~	mag[i].m*100+mag[i].cm

Algoritmus:





Feladatmegoldási minta
legmagasabb tanuló

i		f(i)
e	→	f(e)
e+1	→	f(e+1)
e+2	→	f(e+2)
…	→	…
u-2	→	f(u-2)
u-1	→	f(u-1)
u	→	f(u)

maxind, maxért

Feladat:

maximumkiválasztás

Ismerjük egy osztály tanulóinak magasságait. Mondd meg, hányadik diák a legmagasabb, és a magasságát is!

Specifikáció:

```
Be: mag∈Mag[], Mag=(m:N x cm:N)
Sa: maxcm∈N
Ki: hol∈N, maxmag∈Mag
Fv: f:N->N, f(i)=mag[i].m*100+mag[i].cm
Ef: hossz(mag)>=1
Uf: (hol,maxcm)=MAX(i=1..hossz(mag), f(i)) és
    maxmag=mag[hol]
```

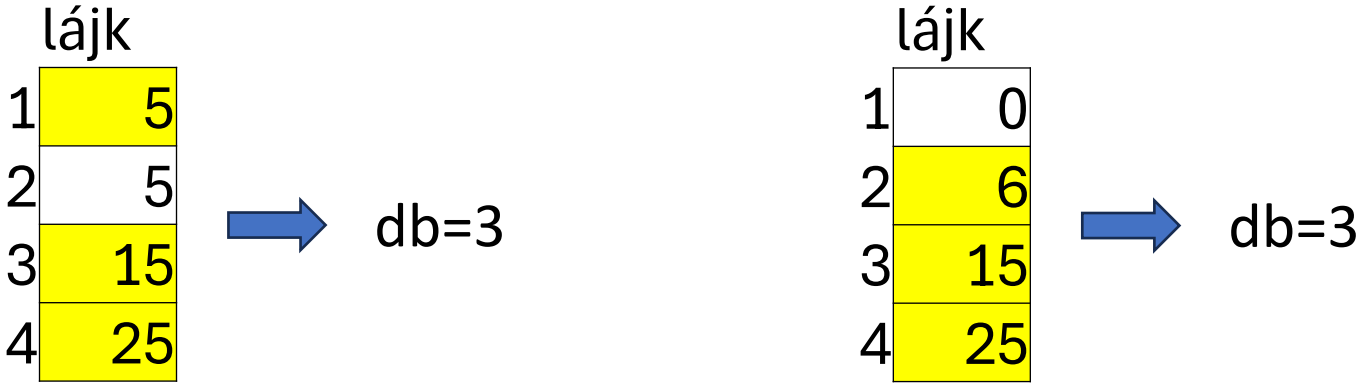
f(i)

	mag m	cm	m*100+cm
1	1	52	152
2	1	77	177
3	1	23	123
4	1	65	165

Feladatmegoldási minta

Feladatmegoldási minta
lájknövekedések száma

A Youtube minden napra megmondja egy videónkról, hogy hány lájkot kapott. Hány nap növekedett a lájkok száma az előző naphoz képest? (Az első nap előtt 0 lájkunk volt.)



Feladatmegoldási minta
lájknövekedések száma

i		T(i)	érték
e	→	IGAZ	1
e+1	→	HAMIS	0
e+2	→	HAMIS	0
…	→	…	…
u-2	→	IGAZ	1
u-1	→	IGAZ	1
u	→	HAMIS	0
			=
			db

Feladat:

megszámolás

Hány nap növekedett a lájkok száma az előző naphoz képest?

Specifikáció:

Be: lájkokszáma∈N[]
Ki: hány∈N
Ef: -
Uf: db=**DARAB**(i=e..u, T(i))

	lájk
1	5
2	5
3	15
4	25

→ hány=3

1. Mik az intervallum határai?
2. Milyen értékeket veszünk az intervallum egyes pontján, amik közül a legnagyobb kikerül?
3. Milyen néven tároljuk az eredményeket?

A helyes T(i) megtalálása a cél!

Feladatmegoldási minta
lájknövekedések száma

i		T(i)	érték
e	→	IGAZ	1
e+1	→	HAMIS	0
e+2	→	HAMIS	0
…	→	…	…
u-2	→	IGAZ	1
u-1	→	IGAZ	1
u	→	HAMIS	0
			=
			db

Feladat:

megszámolás

Hány nap növekedett a lájkok száma az előző naphoz képest?

Specifikáció:

```
Be: lájkokszáma∈N[]
Ki: hány∈N
Ef: -
Uf: hány=DARAB(i=1..hossz(lájkokszáma),
        (i=1 -> lájkokszáma[i]>0) és
        (i>1 -> lájkokszáma[i]>lájkokszáma[i-1]))
```

T(i)

	lájk
1	5
2	5
3	15
4	25

→ hány=3

Feladatmegoldási minta
lájknövekedések száma

Feladatsablon

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

Lájknövekedések száma

```
Be: lájkokszáma∈N[]
Ki: hány∈N
Ef: -
Uf: hány=DARAB(i=1..hossz(lájkokszáma),
      (i=1 -> lájkokszáma[i]>0) és
      (i>1 ->
        lájkokszáma[i]>lájkokszáma[i-1]))
```

Visszavezetés:

```
db      ~ hány
e..u    ~ 1..hossz(lájkokszáma)
T(i)    ~ (i=1 -> lájkokszáma[i]>0) és
          (i>1 -> lájkokszáma[i]>lájkokszáma[i-1])
```

Feladatmegoldási minta
lájknövekedések száma

Feladatsablon

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

Lájknövekedések száma

```
Be: lájkokszáma∈N[]
Ki: hány∈N
Ef: -
Uf: hány=DARAB(i=1..hossz(lájkokszáma),
      (i=1 és lájkokszáma[i]>0) vagy
      (i>1 és
        lájkokszáma[i]>lájkokszáma[i-1]))
```

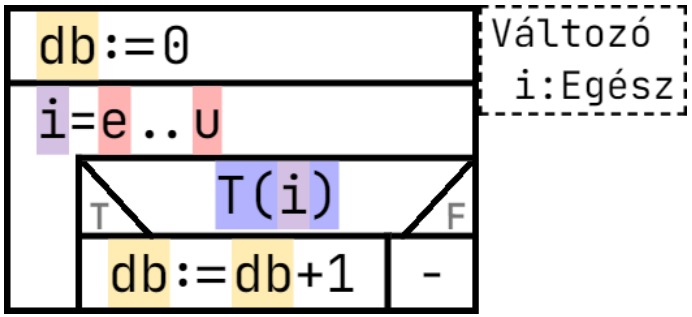
Visszavezetés:

```
db      ~ hány
e..u    ~ 1..hossz(lájkokszáma)
T(i)    ~ (i=1 és lájkokszáma[i]>0) vagy
          (i>1 és lájkokszáma[i]>lájkokszáma[i-1])
```

Feladatmegoldási minta
lájknövekedések száma

Algoritmus

```
db      ~ hány
e..u    ~ 1..hossz(lájkokszáma)
T(i)    ~ (i=1 és lájkokszáma[i]>0) vagy
          (i>1 és lájkokszáma[i]>lájkokszáma[i-1])
```




```
Local declarations
Változó i:Egész
hány:=0
i=1..hossz(lájkokszáma)
    (i=1 és lájkokszáma[i]>0) vagy
    (i>1 és lájkokszáma[i]>lájkokszáma[i-1])
T                                          F
    hány:=hány+1                           -
```

Feladatmegoldási minta
lájknövekedések száma – másképp

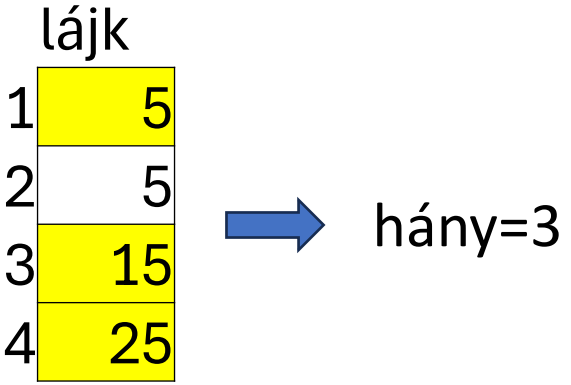
i		T(i)	érték
e	→	IGAZ	1
e+1	→	HAMIS	0
e+2	→	HAMIS	0
…	→	…	…
u-2	→	IGAZ	1
u-1	→	IGAZ	1
u	→	HAMIS	0
			=
			db

Feladat:

Hány nap növekedett a lájkok száma az előző naphoz képest?

Specifikáció:

```
Be: lájkokszáma∈N[]
Sa: db∈N
Ki: hány∈N
Ef: -
Uf: db=DARAB(i=2..hossz(lájkokszáma),
        lájkokszáma[i]>lájkokszáma[i-1]) és
    lájkokszáma[1]>0 -> hány=db+1 és
    nem(lájkokszáma[1]>0) -> hány=db
```



Az esetszétválasztás a mintán kívül van

Feladatmegoldási minta
lájknövekedések száma – másképp

Feladatsablon

```
Be: e∈Z, u∈Z

Ki: db∈N
Ef: -
Uf: db=DARAB(i=e..u, T(i))
```

Lájknövekedések száma

```
Be: lájkokszáma∈N[]
Sa: db∈N
Ki: hány∈N
Ef: -
Uf: db=DARAB(i=2..hossz(lájkokszáma),
        lájkokszáma[i]>lájkokszáma[i-1]))
  és lájkokszáma[1]>0       -> hány=db+1 és
     nem(lájkokszáma[1]>0) -> hány=db
```

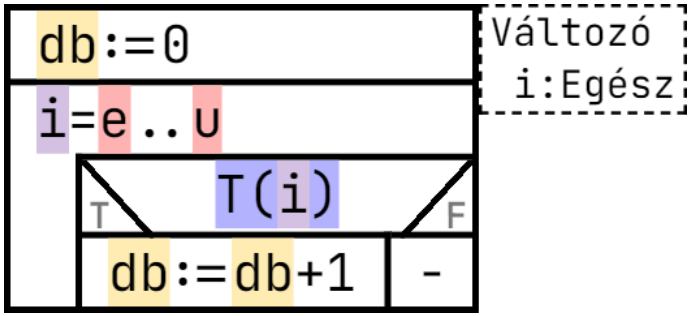
Visszavezetés:

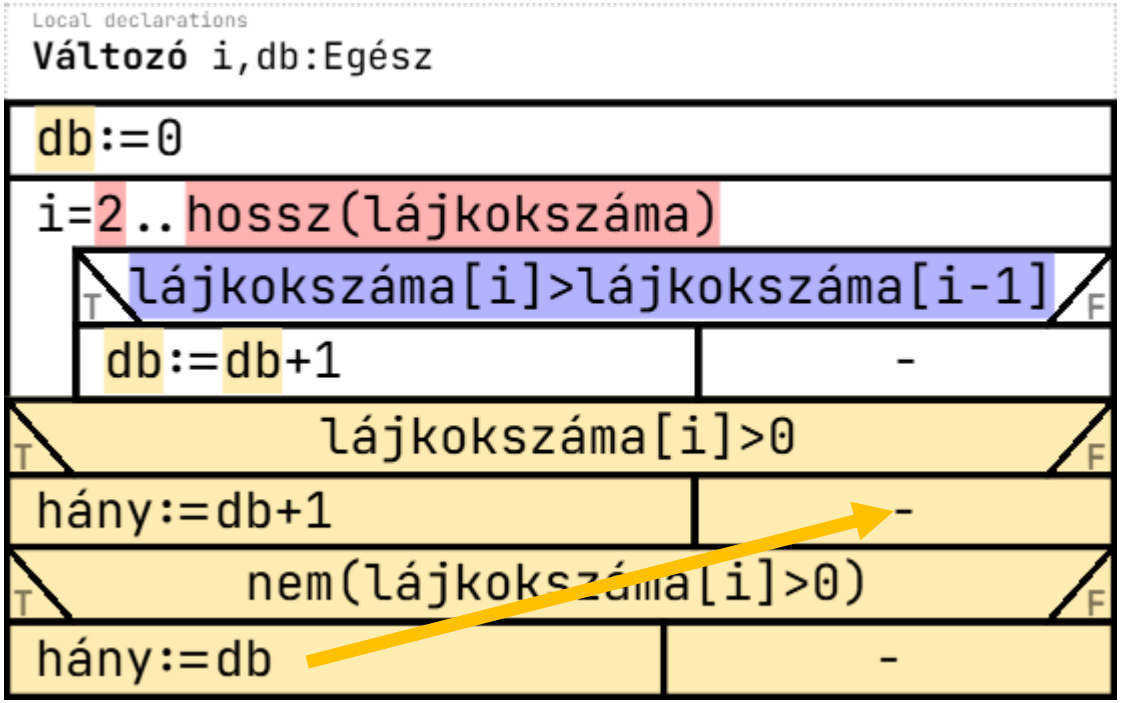
```
e..u    ~   2..hossz(lájkokszáma)
T(i)    ~   lájkokszáma[i]>lájkokszáma[i-1]
```

Feladatmegoldási minta
lájknövekedések száma – másképp

Algoritmus

```
e..u   ~   1..hossz(lájkokszáma)
T(i)   ~   lájkokszáma[i]>lájkokszáma[i-1]
```



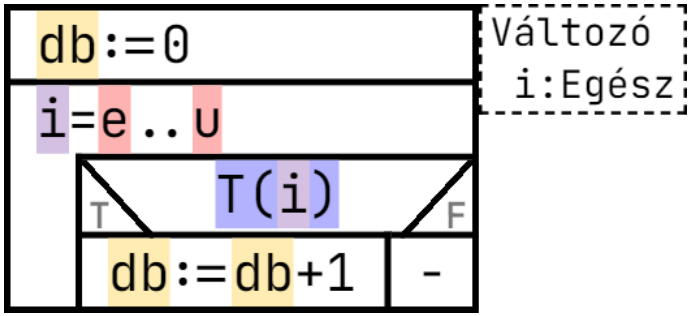


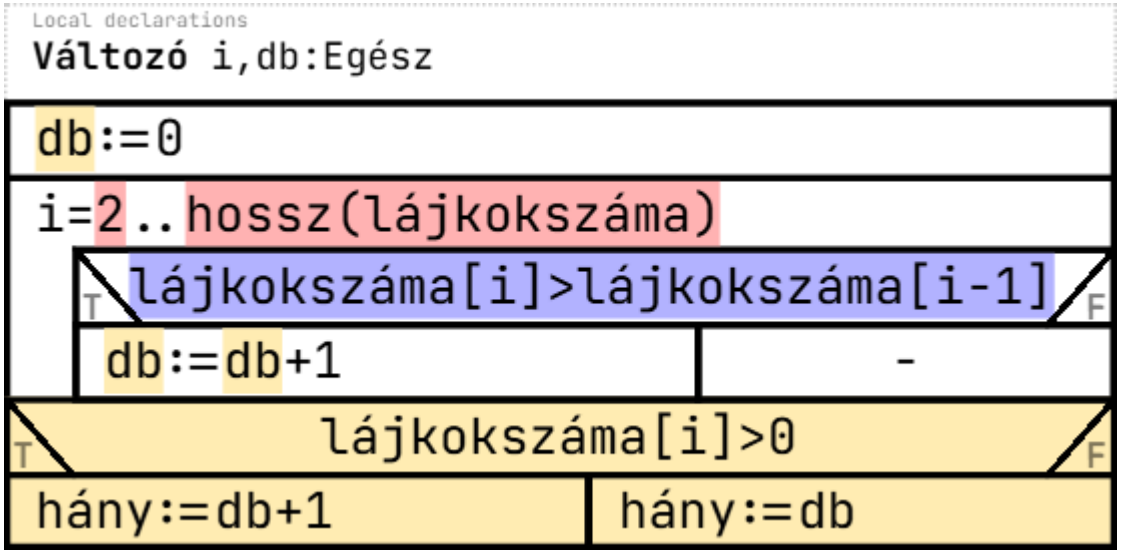

```
Uf: db=DARAB(i=2..hossz(lájkokszáma),
        lájkokszáma[i]>lájkokszáma[i-1]) és
    lájkokszáma[1]>0 -> hány=db+1 és
    nem(lájkokszáma[1]>0) -> hány=db
```

Feladatmegoldási minta
lájknövekedések száma – másképp

Algoritmus

```
e..u   ~   1..hossz(lájkokszáma)
T(i)   ~   lájkokszáma[i]>lájkokszáma[i-1]
```






```
Uf: db=DARAB(i=2..hossz(lájkokszáma),
        lájkokszáma[i]>lájkokszáma[i-1]) és
    lájkokszáma[1]>0 -> hány=db+1 és
    nem(lájkokszáma[1]>0) -> hány=db
```