

PROGRAMOZÁS Mintamegoldás mátrixra

Horváth Győző, Pluhár Zsuzsa

Ismétlés



Programozási minták

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







Feladat





Feladat:

A Roxfortban év végén n varázslótanoncról ismerjük az m tárgyból szerzett jegyét egy táblázatban. Dumbledore szeretné meghívni egy vajsörre azt a tanulót, akinek a legjobb lett az átlaga azok közül, akik csak 4-est és 5-öst szereztek.





		1	2	3	m=4
	1	5	5	5	3
	2	5	4	4	4
	3	5	3	2	4
	4	5	4	5	5
_	n=5	5	5	4	4

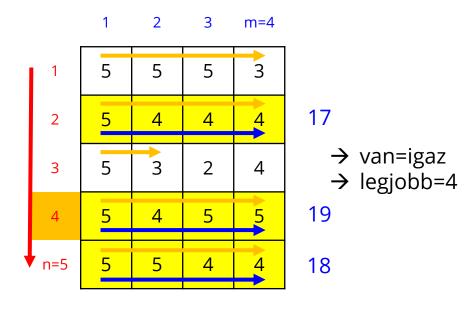


Feladat:

Egy egész számokat tartalmazó mátrixban melyik az a sor, amiben csak 4-es és 5-ös van, és az összege a legnagyobb?

Lépések:

- Minden sor összege kell.
 (→ összegzés)
- Ezek közül kell a legnagyobb.
 (→ maximum(kiválasztás))
- De csak, ha minden érték a sorban 4-es vagy 5-ös
 (→ mind eldöntés)
- 4. Feltételes maximumkeresésben összegzés



		1	2	3	m=4
	1	5	5	5	3
	2	5	4	4	4
	3	5	3	2	4
	4	5	4	5	5
_	n=5	5	5	4	4



→ van=igaz→ legjobb=4

18

Feladat:

Egy egész számokat tartalmazó mátrixban melyik az a sor, amiben csak 4-es és 5-ös van, és az összege a legnagyobb?

Specifikáció:

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $j \in \mathbb{N}[1...n, 1...m]$

Ki: van∈L, legjobb∈N

Fv: összeg:N->N,

összeg(diák)=SZUMMA(tantárgy=1..m,jegyek[diák,tantárgy])

Fv: jó:N->L,

jó(diák)=MIND(tantárgy=1..m,4<=jegyek[diák,tantárgy]<=5)</pre>

Ef: ∀sor∈[1..n]:(∀oszlop∈[1..m]:(1<=jegyek[sor,oszlop]<=5))</pre>

Uf: (van,legjobb,)=FELTMAX(diák=1..n,összeg(diák),jó(diák))



```
Mind eldöntés
      ~ tantárgy
e..u ~ 1..m
T(i) ~ 4<=jegyek[diák,tantárgy]<=5</pre>
```



Felt.max.ker.:

```
Be: e∈Z, u∈Z
```

Ki: van∈L, maxind∈Z, maxért∈H Ki: van∈L, legjobb∈N

Ef: -

Uf: (van, maxind, maxért)=

FELTMAX(i=e..u,f(i),T(i))

Összegzés sablon:

Be: e∈Z, u∈Z

Ki: s∈H

Ef: -

Uf: s=SZUMMA(i=e..u, f(i))

Mind eldöntés:

Be: e∈Z, u∈Z

Ki: mind∈L

Ef: -

Uf: mind=MIND(i=e..u,T(i)

Legjobb jó tanuló:

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $j \in \mathbb{N}[1...n, 1...m]$

Fv: összeg:N->N,

összeg(diák)=SZUMMA(tantárgy=1..m,

jegyek[diák,tantárgy])

Fv: jó:N->L,

jó(diák)=MIND(tantárgy=1..m,

4<=jegyek[diák,tantárgy]<=5)

Ef: $\forall sor \in [1..n]: (\forall oszlop \in [1..m]:$

(1<=jegyek[sor,oszlop]<=5))</pre>

Uf: (van,legjobb,)=FELTMAX(diák=1..n,

összeg(diák),jó(diák))

Feltételes max.keresés

maxind ~ legjobb

~ diák

e..u ~ 1..n

f(i) ~ összeg(diák

T(i) ~ jó(diák)

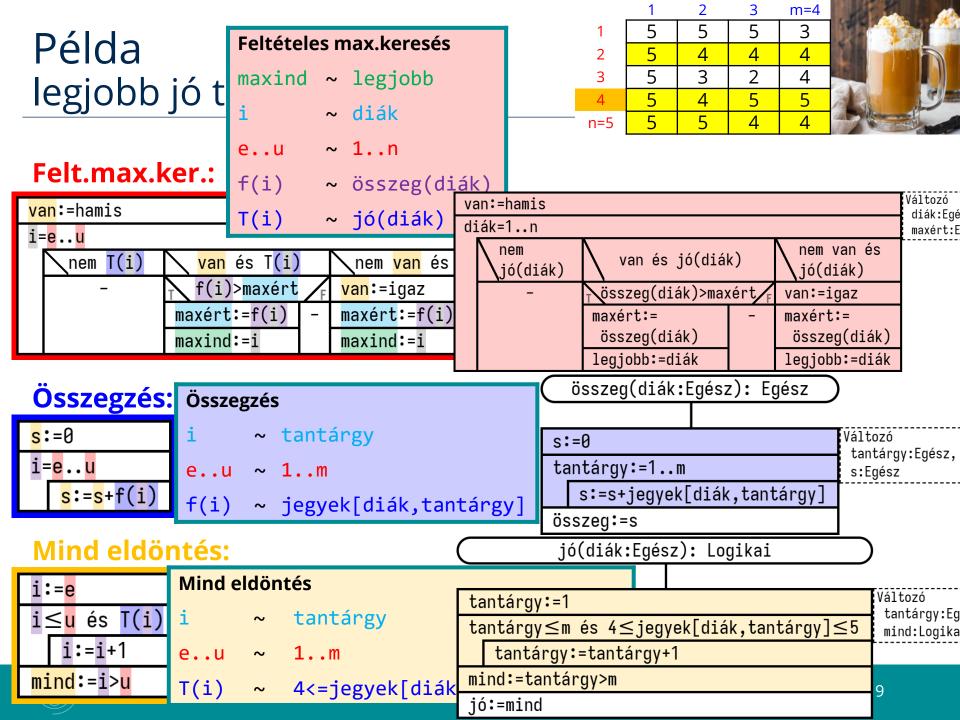
Összegzés

~ tantárgy

e..u ~ 1..m

f(i) ~ jegyek[diák,tantárgy]



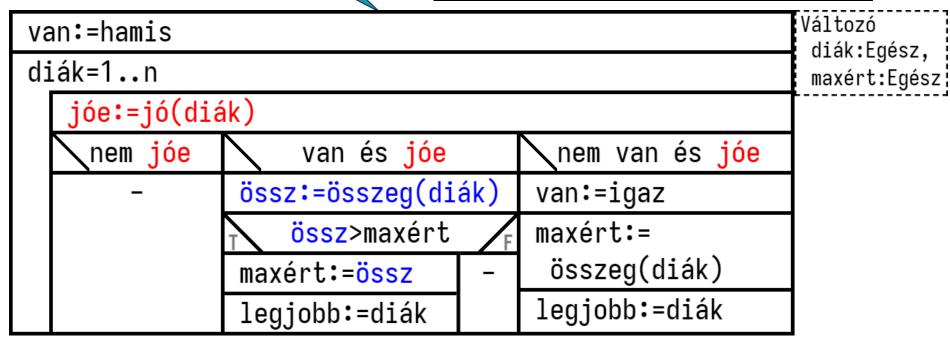


	1	2	3	m=4
1	5	5	5	3
2	5	4	4	4
3	5	3	2	4
4	5	4	5	5
n=5	5	5	4	4



Hatékonyítás: felesleges számolások elkerülése

Változó van:=hamis diák:Egész, diák=1..n maxért:Egész nem van és nem van és jó(diák) jó(diák) jó(diák) összeg(diák)>maxért van:=iqaz maxért:= maxért:= összeg(diák) összeg(diák) legjobb:=diák legjobb:=diák





Példa

```
static void Main(string[] args) {
                                                            4
 // deklarálás
                                                           n=5
 int n; int m; int[,] jegyek;
 bool van; int legjobb;
 beolvas(out n, out m, out jegyek);
 legjobb_jo_tanulo(n, m, jegyek, out van, out legjobb);
 kiir(van, legjobb);
static void beolvas(out int n, out int m, out int[,] jegyek) {
 Console.Write("Varazstanoncok szama = ");
  int.TryParse(Console.ReadLine(), out n);
 Console.Write("Jegyek szama = ");
 int.TryParse(Console.ReadLine(), out m);
 jegyek = new int[n, m];
 for (int i = 1; i <= n; i++) {
   for (int j = 1; j <= m; j++) {
      Console.Write("{0}. varazstanonc {1} jegye = ", i, j);
      int.TryParse(Console.ReadLine(), out jegyek[i - 1, j - 1]);
static void kiir(bool van, int legjobb) {
 if (van) {
   Console.WriteLine("A legjobb tanuló: {0}", legjobb);
  }
 else {
   Console.WriteLine("Nincs ilyen tanuló");
```



"Procedúrákra" bontva

5

4

5

```
    1
    2
    3
    m=4

    1
    5
    5
    5
    3

    2
    5
    4
    4
    4

    3
    5
    3
    2
    4

    4
    5
    4
    5
    5
```

```
static void legjobb_jo_tanulc(int n, int m, int[,] jegyek out bool van, out int legjobb) {
  int maxert = 0;
 legjobb = 0;
                                                                 "Procedúrákra" bontva
 van = false;
 for (int diak = 1; diak <= n; diak++) {</pre>
    bool joe = jo(diak, m, jegyek);
    if (van && joe) {
      int ossz = osszeg(diak, m, jegyek);
      if (ossz > maxert) {
        maxert = ossz;
                                          static bool jo(int diak, int m, int[,] jegyek) {
        legjobb = diak;
                                            int tantargy = 1;
                                            while (tantargy <= m &&</pre>
                                                4 <= jegyek[diak - 1, tantargy - 1] &&
    else if (!van && joe) {
                                                 jegyek[diak - 1, tantargy - 1] <= 5) {</pre>
      van = true;
                                              tantargy = tantargy + 1;
      maxert = osszeg(diak, m, jegyek);
      legjobb = diak;
                                            bool mind = tantargy > m;
                                            return mind;
                                          static int osszeg(int diak, int m, int[,] jegyek) {
                                            int s = 0;
                                            for (int tantargy = 1; tantargy <= m; tantargy++) {</pre>
                                              s = s + jegyek[diak - 1, tantargy - 1];
                                            return s;
```

	1	2	3	m=4
1	5	5	5	3
2	5	4	4	4
3	5	3	2	4
4	5	4	5	5
1=5	5	5	4	4



```
static void Main(string[] args) {
 // deklarálás
                                                          Függvényekre bontva
 int n; int m; int[,] jegyek;
 bool van; int legjobb;
 (n, m, jegyek) = beolvas();
 (van, legjobb) = legjobb jo tanulo(n, m, jegyek);
 kiir(van, legjobb);
static (int n, int m, int[,] jegyek) beolvas() {
 int n, m;
 int[,] jegyek;
 // ugyanaz, mint előzőleg
 return (n, m, jegyek);
static (bool van, int legjobb) legjobb_jo_tanulo(int n, int m, int[,] jegyek) {
 bool van;
 int legjobb;
 // ugyanaz, mint előzőleg
 return (van, legjobb);
```