



ELTE | IK

PROGRAMOZÁS

Mintamegoldás mátrixra

Horváth Győző, Pluhár Zsuzsa



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

Most Common DUPLO Parts



Brick Architect 



Feladat



Példa

legjobb jó tanuló



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.



Példa

legjobb jó tanuló

	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:

1. Minden **sor összege** kell.
(→ **összegzés**)
2. Ezek közül kell a **legnagyobb**.
(→ **maximum(kiválasztás)**)
3. 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	17
3	5	3	2	4	
4	5	4	5	5	19
n=5	5	5	4	4	18

→ van=igaz
→ legjobb=4

Példa

legjobb jó tanuló

	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**?

Specifikáció:

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

Ki: van $\in \mathbb{L}$, legjobb $\in \mathbb{N}$

Fv: összeg: $\mathbb{N} \rightarrow \mathbb{N}$,

$\text{összeg}(\text{diák}) = \text{SZUMMA}(\text{tantárgy} = 1..m, \text{jegyek}[\text{diák}, \text{tantárgy}])$

Fv: jó: $\mathbb{N} \rightarrow \mathbb{L}$,

$\text{jó}(\text{diák}) = \text{MIND}(\text{tantárgy} = 1..m, 4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5)$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{van}, \text{legjobb},) = \text{FELTMAX}(\text{diák} = 1..n, \text{összeg}(\text{diák}), \text{jó}(\text{diák}))$

	1	2	3	m=4	
1	5	5	5	3	
2	5	4	4	4	17
3	5	3	2	4	→ van=igaz → legjobb=4
4	5	4	5	5	19
n=5	5	5	4	4	18

Példa

legjobb jó tanuló

Mind eldöntés

$i \sim \text{tantárgy}$

$e..u \sim 1..m$

$T(i) \sim 4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5$



Felt.max.ker.:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $\text{van} \in \mathbb{L}, \text{maxind} \in \mathbb{Z}, \text{maxért} \in \mathbb{H}$

Ef: -

Uf: $(\text{van}, \text{maxind}, \text{maxért}) =$
 $\text{FELTMAX}(i = e..u, f(i), T(i))$

Összegzés sablon:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $s \in \mathbb{H}$

Ef: -

Uf: $s = \text{SZUMMA}(i = e..u, f(i))$

Mind eldöntés:

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $\text{mind} \in \mathbb{L}$

Ef: -

Uf: $\text{mind} = \text{MIND}(i = e..u, T(i))$

Legjobb jó tanuló:

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

Ki: $\text{van} \in \mathbb{L}, \text{legjobb} \in \mathbb{N}$

Fv: $\text{összeg}: \mathbb{N} \rightarrow \mathbb{N},$

$\text{összeg}(\text{diák}) = \text{SZUMMA}(\text{tantárgy} = 1..m,$
 $\text{jegyek}[\text{diák}, \text{tantárgy}])$

Fv: $\text{jó}: \mathbb{N} \rightarrow \mathbb{L},$

$\text{jó}(\text{diák}) = \text{MIND}(\text{tantárgy} = 1..m,$
 $4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5)$

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

$(1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{van}, \text{legjobb},) = \text{FELTMAX}(\text{diák} = 1..n,$
 $\text{összeg}(\text{diák}), \text{jó}(\text{diák}))$

Feltételes max.keresés

$\text{maxind} \sim \text{legjobb}$

$i \sim \text{diák}$

$e..u \sim 1..n$

$f(i) \sim \text{összeg}(\text{diák})$

$T(i) \sim \text{jó}(\text{diák})$

Összegzés

$i \sim \text{tantárgy}$

$e..u \sim 1..m$

$f(i) \sim \text{jegyek}[\text{diák}, \text{tantárgy}]$

Példa legjobb jó t

Felt.max.ker.:

Feltételes max.keresés

maxind ~ legjobb

i ~ diák

e..u ~ 1..n

f(i) ~ összeg(diák)

T(i) ~ jó(diák)

van:=hamis

i=e..u

	nem T(i)	van és T(i)	nem van és
	-	$f(i) > \text{maxért}$ maxért:=f(i) maxind:=i	van:=igaz maxért:=f(i) maxind:=i

	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:=hamis

diák=1..n

	nem jó(diák)	van és jó(diák)	nem van és jó(diák)
	-	$\text{összeg}(\text{diák}) > \text{maxért}$ maxért:=összeg(diák) legjobb:=diák	van:=igaz maxért:=összeg(diák) legjobb:=diák

Változó
diák:Egész
maxért:E

Összegzés: Összegzés

s:=0

i=e..u

s:=s+f(i)

i ~ tantárgy

e..u ~ 1..m

f(i) ~ jegyek[diák,tantárgy]

összeg(diák:Egész): Egész

s:=0

tantárgy:=1..m

s:=s+jegyek[diák,tantárgy]

összeg:=s

Változó
tantárgy:Egész,
s:Egész

Mind eldöntés:

i:=e

i ≤ u és T(i)

i:=i+1

mind:=i>u

Mind eldöntés

i ~ tantárgy

e..u ~ 1..m

T(i) ~ 4 ≤ jegyek[diák

jó(diák:Egész): Logikai

tantárgy:=1

tantárgy ≤ m és 4 ≤ jegyek[diák,tantárgy] ≤ 5

tantárgy:=tantárgy+1

mind:=tantárgy>m

jó:=mind

Változó
tantárgy:Egész
mind:Logika

Példa

legjobb jó tanuló

Hatékonyítás: felesleges számolások elkerülése

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

van:=hamis

diák=1..n

jóe:=jó(diák)

nem jóe	van és jóe		nem van és jóe
-	össz:=összeg(diák)		van:=igaz
	T össz>maxért		F maxért:=
	maxért:=össz		összeg(diák)
	legjobb:=diák		legjobb:=diák

Változó
diák:Egész,
maxért:Egész

Példa

```
static void Main(string[] args) {  
    // deklarálás  
    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");  
    }  
}
```

	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

„Procedúrákra” bontva



Példa

legjobb jó tanuló

	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_tanulo(int n, int m, int[,] jegyek, out bool van, out int legjobb) {
```

```
    int maxert = 0;
```

```
    legjobb = 0;
```

```
    van = false;
```

```
    for (int diak = 1; diak <= n; diak++) {
```

```
        bool joe = jo(diak, m, jegyek);
```

```
        if (van && joe) {
```

```
            int ossz = osszeg(diak, m, jegyek);
```

```
            if (ossz > maxert) {
```

```
                maxert = ossz;
```

```
                legjobb = diak;
```

```
            }
```

```
        }
```

```
        else if (!van && joe) {
```

```
            van = true;
```

```
            maxert = osszeg(diak, m, jegyek);
```

```
            legjobb = diak;
```

```
        }
```

```
    }
```

```
}
```

„Procedúrákra” bontva

```
static bool jo(int diak, int m, int[,] jegyek) {
```

```
    int tantargy = 1;
```

```
    while (tantargy <= m &&
```

```
        4 <= jegyek[diak - 1, tantargy - 1] &&
```

```
        jegyek[diak - 1, tantargy - 1] <= 5) {
```

```
        tantargy = tantargy + 1;
```

```
    }
```

```
    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++) {
```

```
        s = s + jegyek[diak - 1, tantargy - 1];
```

```
    }
```

```
    return s;
```

```
}
```



Példa

legjobb jó tanuló

	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



```
static void Main(string[] args) {  
    // deklaráció  
    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);  
}
```

Függvényekre bontva