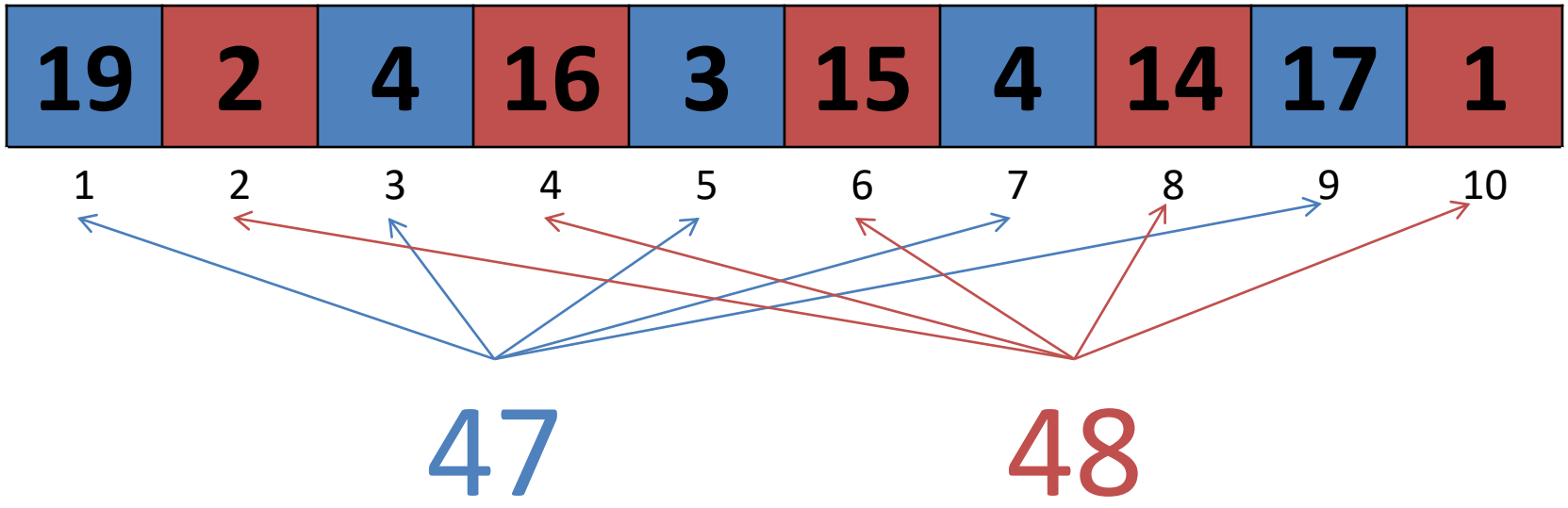
# Barátságos mérkőzések PÁRBAN

- Legyen egy n elemű természetes számsorozat (n páros). A játékosok felváltva választanak egy-egy elemet a számsor valamelyik végéről. Az nyer, aki a nagyobb összeget gyűjti össze.
  - Meg lehet-e verni a „tanárt”, amennyiben hagyod, hogy ő kezdjen?

| 19 | 2 | 4 | 16 | 3 | 15 | 4 | 14 | 17 | 1 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

# A kezdő minden lépésben, következetesen választhat páros/páratlan indexű elem között

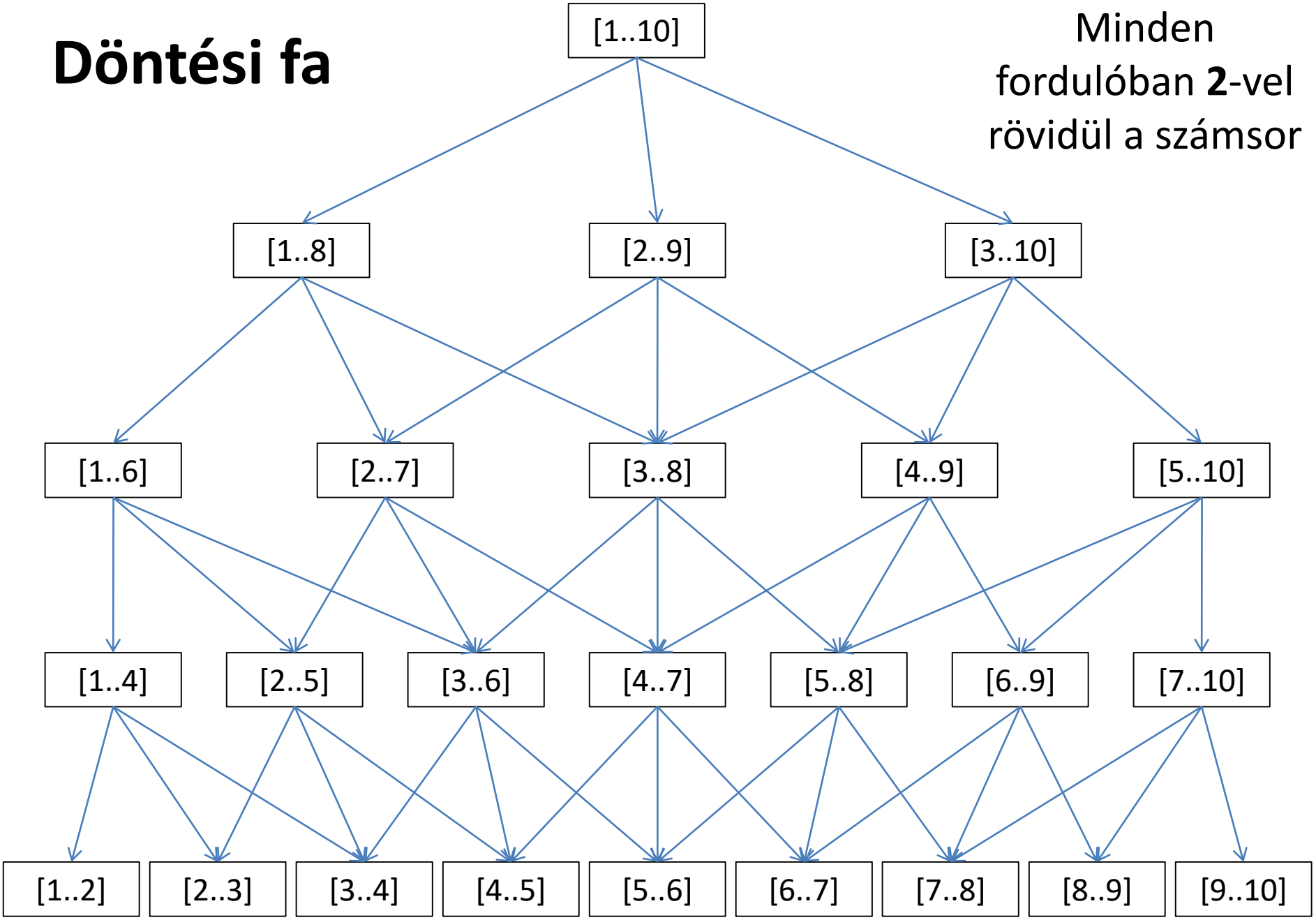


# Mennyi a maximális összeg, amit a kezdő *garantáltan* összeszedhet?

- Input: a[1..n]
- Cél: maximalizálni a kezdő *garantált* összegét
  - A barátod is mindig jól választ! Persze az általad, mint kezdő által, diktált keretek között.

# Döntési fa

Minden fordulóban **2**-vel rövidül a számsor

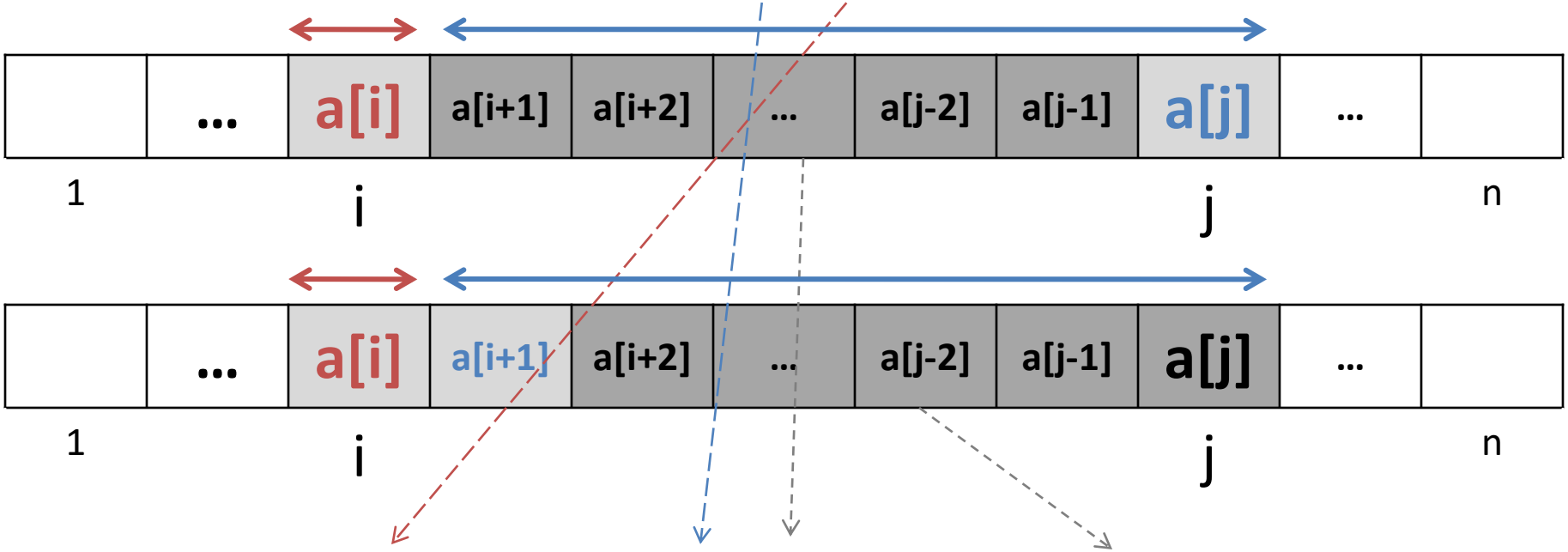


# Mennyi a maximális összeg, amit a kezdő *garantáltan* összeszedhet?

- Eredeti feladat:
  - mi a teljes tömbre (a[1..n]) vonatkozó optimum?
- Általános részfeladat:
  - mi az optimum az a[i..j], páros hosszú tömbszakaszra vonatkozóan, ha a kezdő van soron?

# Két út áll előttem, melyiket válasszam? a[i]-t vagy a[j]-t? Az előnyösebbet!

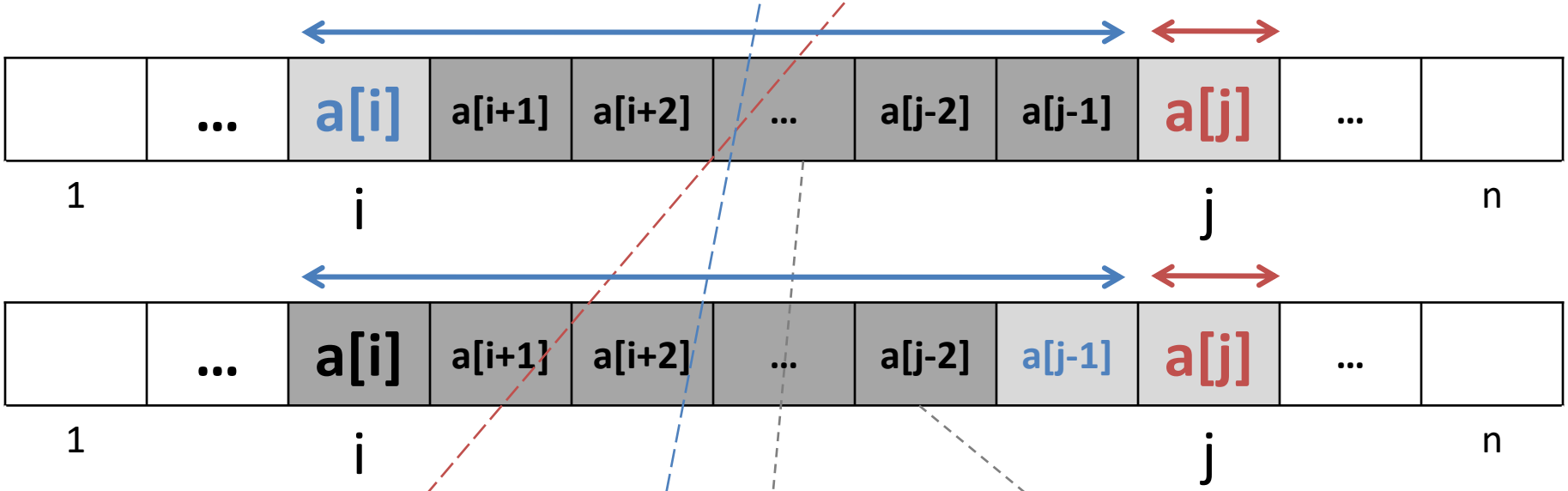
- Ha én a[i]-t választom, akkor
  - Két út áll előtted, melyiket választod? a[j] vagy a[i+1]? A számomra előnytelenebbet!




```
c[i][j] = max{a[i] + min{c[i+1][j-1], c[i+2][j]}, …}
```

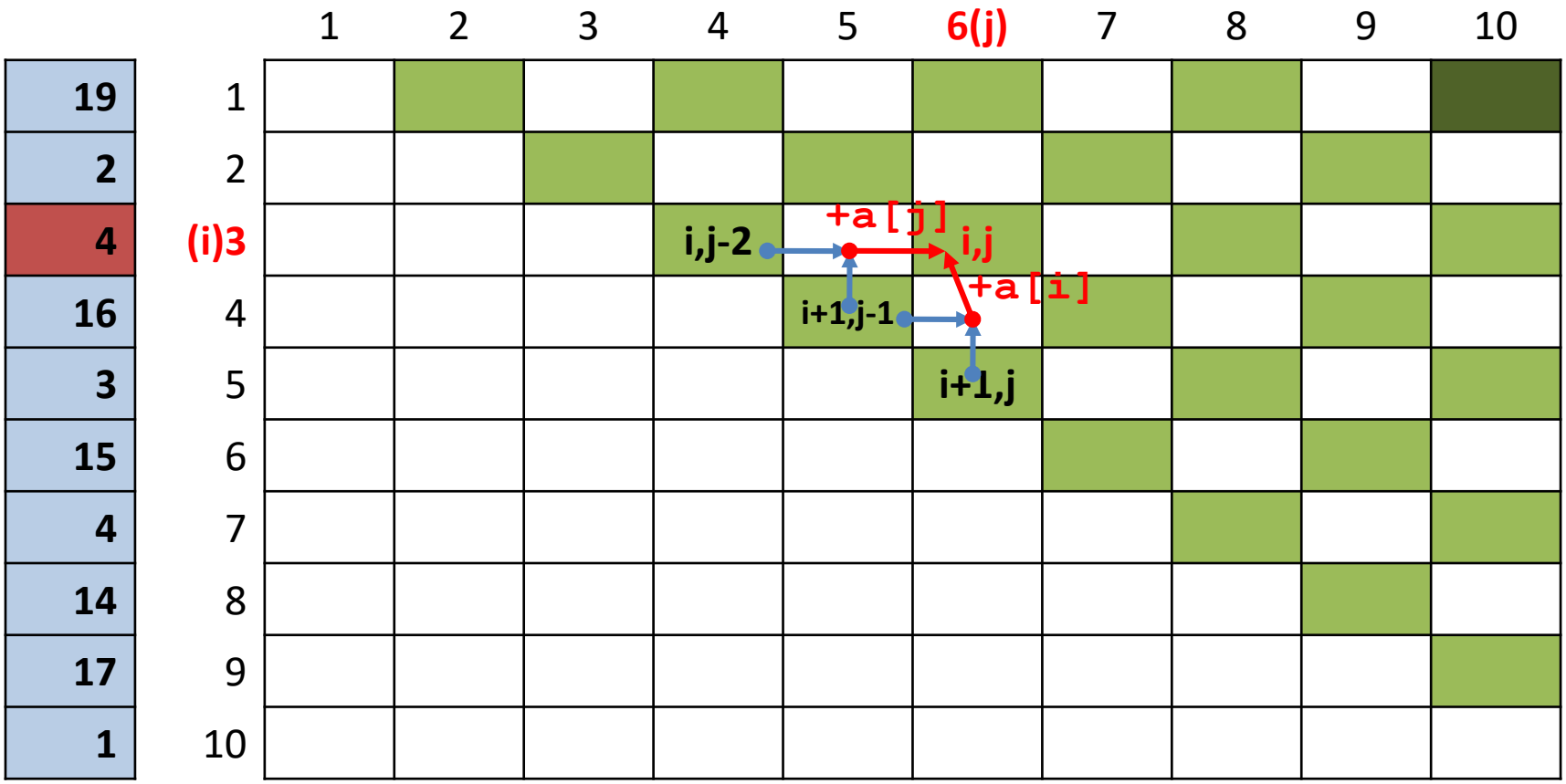
# Két út áll előttem, melyiket válasszam? a[i]-t vagy a[j]-t? Az előnyösebbet!

- Ha én a[j]-t választom, akkor
  - Két út áll előtted, melyiket választod? a[i] vagy a[j-1]? A számomra előnytelenebbet!




```
c[i][j] = max{a[i] + min{c[i+1][j-1], c[i+2][j]},
              a[j] + min{c[i+1][j-1], c[i][j-2]}}
```

| 19 | 2 | 4 | 16 | 3 | 15 | 4 | 14 | 17 | 1 |
|---|---|---|---|---|---|---|---|---|---|

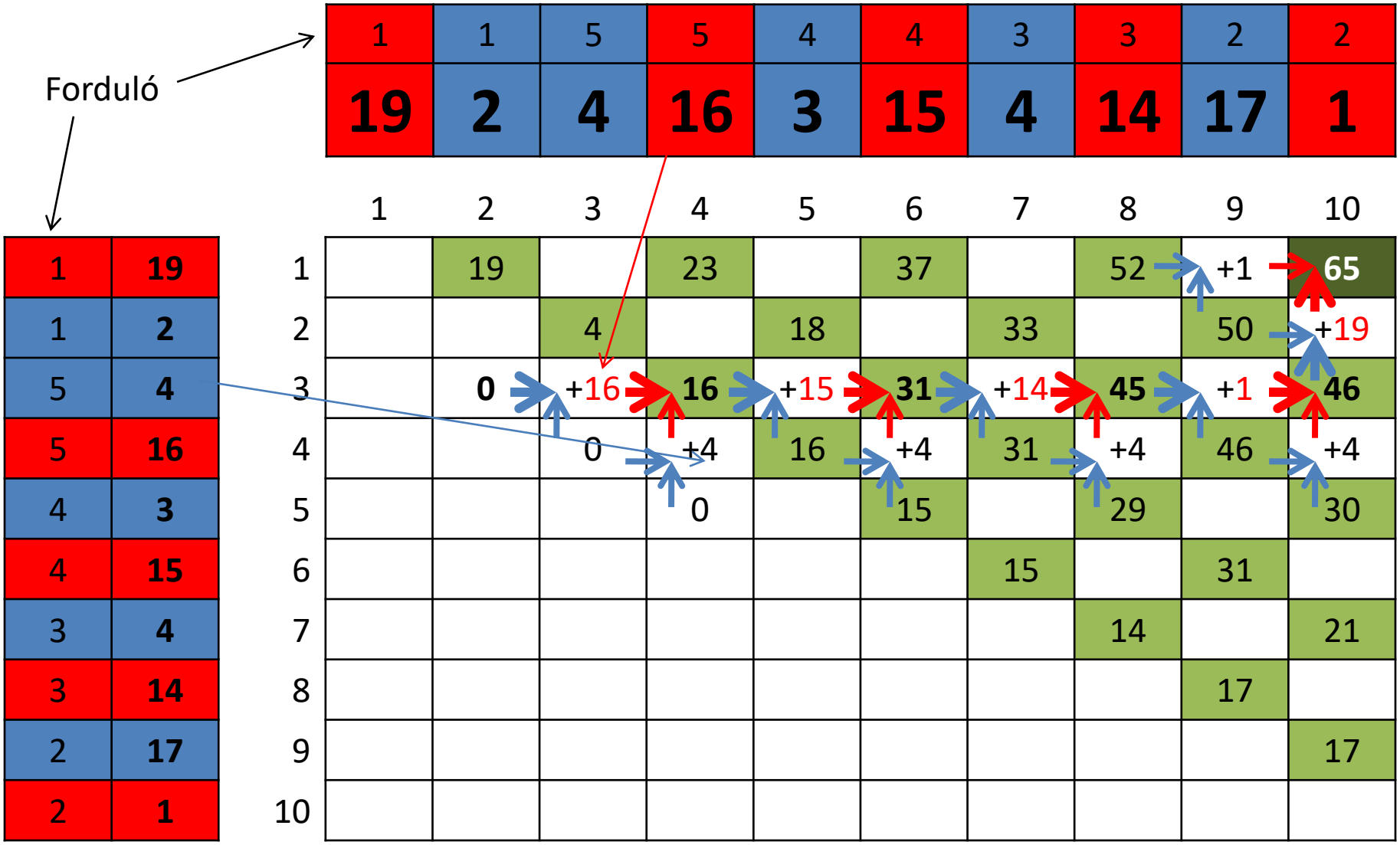



```
c[i][j] = max{a[i] + min{c[i+1][j-1], c[i+2][j]},
              a[j] + min{c[i+1][j-1], c[i][j-2]}}
```

| 19 | 2 | 4 | 16 | 3 | 15 | 4 | 14 | 17 | 1 |
|---|---|---|---|---|---|---|---|---|---|

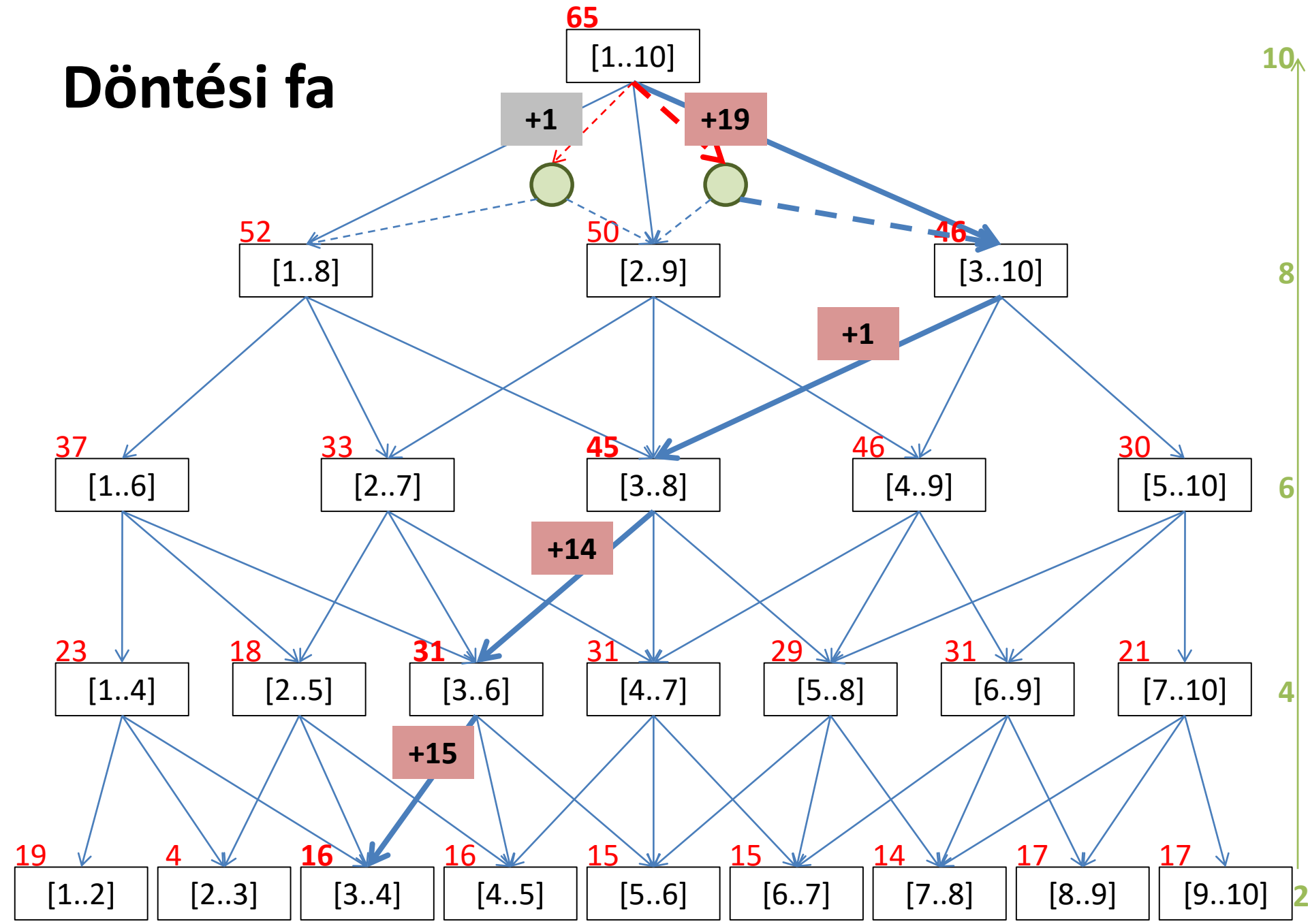
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 19 | 1 | | 19 | +16 | 23 | | 37 | | 52 | +1 | 65 |
| 2 | 2 | | | 4 | +19 | 18 | | 33 | | 50 | +19 |
| 4 | 3 | | | | 16 | | 31 | | 45 | | 46 |
| 16 | 4 | | | | | 16 | | 31 | | 46 | |
| 3 | 5 | | | | | | 15 | | 29 | | 30 |
| 15 | 6 | | | | | | | 15 | | 31 | |
| 4 | 7 | | | | | | | | 14 | | 21 |
| 14 | 8 | | | | | | | | | 17 | |
| 17 | 9 | | | | | | | | | | 17 |
| 1 | 10 | | | | | | | | | | |

```
c[i][j] = max{a[i] + min{c[i+1][j-1], c[i+2][j]},
              a[j] + min{c[i+1][j-1], c[i][j-2]}}
```




```
c[i][j] = max{a[i] + min{c[i+1][j-1], c[i+2][j]},
              a[j] + min{c[i+1][j-1], c[i][j-2]}}
```

# Döntési fa


65
[1..10]
+1
+19
52
[1..8]
50
[2..9]
46
[3..10]
+1
37
[1..6]
33
[2..7]
45
[3..8]
46
[4..9]
30
[5..10]
+14
23
[1..4]
18
[2..5]
31
[3..6]
31
[4..7]
29
[5..8]
31
[6..9]
21
[7..10]
+15
19
[1..2]
4
[2..3]
16
[3..4]
16
[4..5]
15
[5..6]
15
[6..7]
14
[7..8]
17
[8..9]
17
[9..10]
10
8
6
4
2

# Implementációs feladatok

- Implementált kétszemélyes játékként a „páros/páratlan” stratégiát, úgy hogy a gép kezd!.
- Implementáld a „garantáltan legjobb összeg” algoritmust az optimális összeg meghatározására!
  - Implementáld rekurzívan is!
  - Hogy oldanád meg memóriatakarékosan?
- Határozd meg az optimális választás-sorozatot is!
- Implementáld a „garantáltan legjobb összeg” algoritmust , mint kétszemélyes játékot (a gép kezd)

# Megoldás: optimum-érték (1)

```
. . .
for(  ;  ;  ){ //átlóról-átlóra
   for(  ;  ;  ){ //kurrens átló mentén
        c[i][j] = maxi( a[i] + mini(c[i+1][j-1], c[i+2][j]),
                        a[j] + mini(c[i+1][j-1], c[i][j-2]));
   }
}
. . .
```

# Megoldás: optimum-érték (2)

```
// a 0. átló csupa nullát tartalmaz
for( k = 1 ; k < n ; k += 2 ){ //átlónként
 for( i = 1, j = i + k ; j <= n ; ++i, ++j ){ // átlón
  c[i][j] = maxi(a[i] + mini(c[i+1][j-1],c[i+2][j]),
                 a[j] + mini(c[i+1][j-1],c[i][j-2]));
 }
}
printf("eredmeny: %i\n", c[1][n]);
```

# A gyorsaknak!

- Balanced Partition
  - You have a set of n integers each in the range 0 ... K. Partition these integers into two subsets such that you minimize |S1 - S2|, where S1 and S2 denote the sums of the elements in each of the two subsets.
- https://people.cs.clemson.edu/~bcdean/dp_practice/dp_4.swf

# Stratégia

- Egyszerűtől bonyolult fele haladva oldjuk meg a részfeladatokat (**?**)
- Részfeladatonként egy értéket tárolunk el (tömbben) (**?**)
  - optimális megoldást képviselő optimum értéket (**?**)
- Rekurzív képlet írja le, hogy a kurrens részfeladat:
  - optimuma miként építhető fel a közvetlen fiúrészfeladatok optimumaiból (optimális építkezés: *optimumokból optimálisan*) (**?**)

# Stratégia

1. Meghatározzuk a részfeladatok általános alakját.
2. Eldöntjük, hol fogjuk eltárolni az egyes részfeladatok optimális megoldásait jellemző optimum értékeket (vagy a megoldások számát).
3. Megkeressük azt a rekurzív képletet, amely matematikailag leírja, miként épül fel az általános részfeladat optimális megoldását jellemző optimum érték a részfeladatok optimum értékéből.
   - A kurrens részfeladat optimuma mely közvetlen fiú-részfeladatok optimumaiból építhető fel, és hogyan?
4. A rekurzív képlet alapján - az egyszerűtől haladva a bonyolult fele - feltöltjük a tömböt a részfeladatok optimum értékeivel.

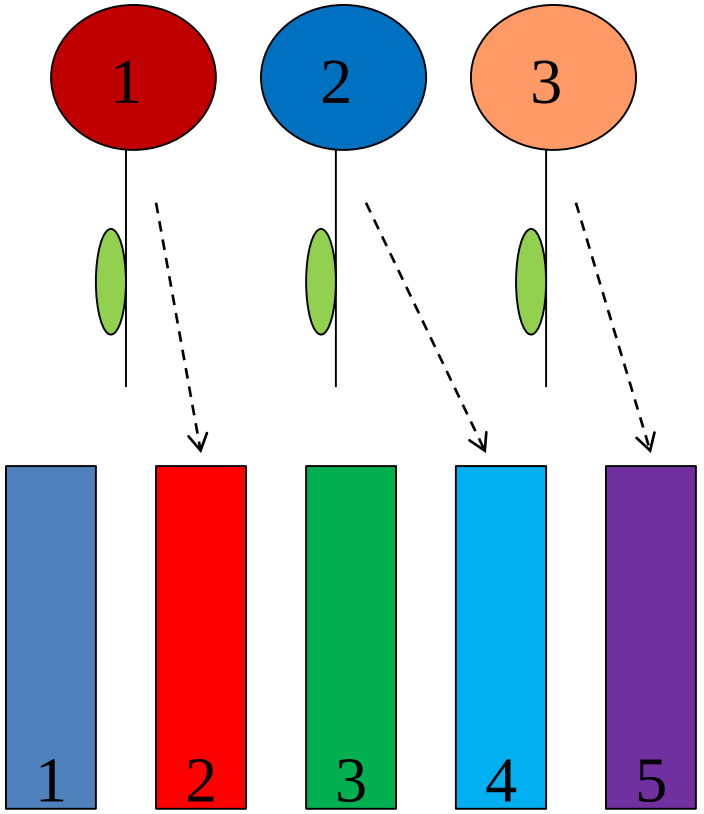
# *Virágüzlet-feladat*

- *Egy virágüzlet kirakatában van m váza (1, 2, …, m sorrendben) és ezekbe úgy kell elhelyezni az 1, 2, …, n virágokat (ebben a sorrendben; $n \leq m$), hogy az esztétikai összhatás maximális legyen. (Az e[1..n,1..m] tömb e[i,j] cellája azt tárolja, hogy az i virág a j vázában milyen esztétikai hatást kelt; az üresen maradt vázák esztétikai hatása nulla)*
  - *(Nemzetközi Informatika Olimpiász, Törökország, 1999)*

# *Virágüzlet-feladat*

- Példa 3 virágra és 5 vázára.
  - Az maximális esztétikai összhatás: 53.

| **e** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 7 | 23 | -5 | -24 | 16 |
| 2 | 5 | 21 | -4 | 10 | 23 |
| 3 | -21 | 5 | -4 | -20 | 20 |

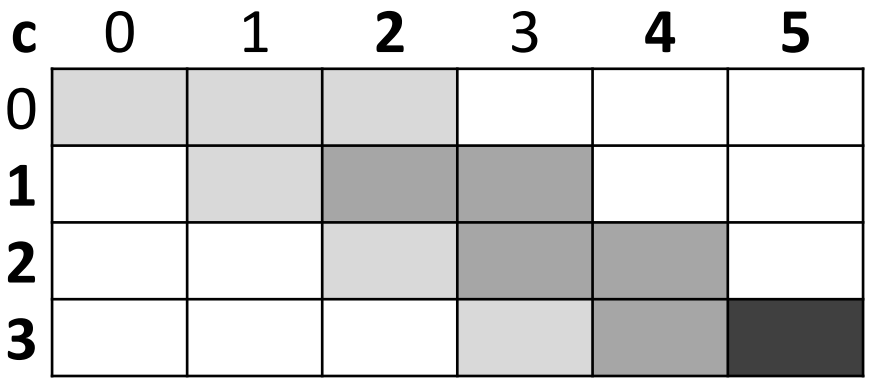


# Meghatározzuk a részfeladatok általános alakját

- *Általános alak*:
  - az 1..i virágok optimális elhelyezése az 1..j vázákba.
- *Optimum-érték*:
  - az optimális elhelyezés keltette esztétikai összhatás értéke.
- *Optimális megoldás*:
  - az optimális elhelyezés módja.
- *Triviális részfeladatok*:
  - i=0 (nulla virág elhelyezése bármennyi vázába);
  - i=j (ugyanannyi a virág, mint a váza).
- *Eredeti feladat*:
  - i=n, j=m.
- *Lentről felfele irány*:
  - i és j növekednek.

# Hol tároljuk a részfeladatok optimális megoldásait képviselő optimum értékeket?

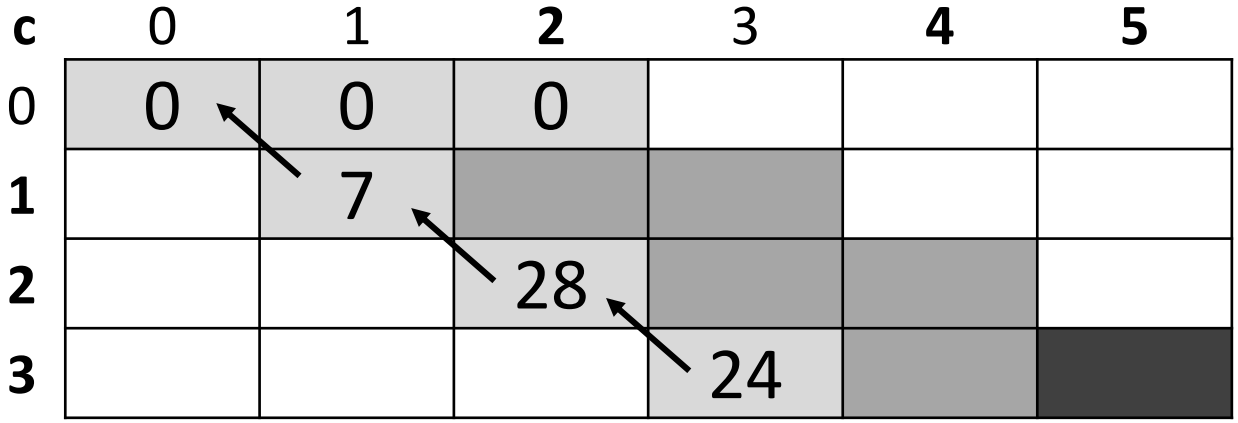
- *Optimum-értékek tömbje*:
  - c[0..n,0..m] 2-dimenziós tömb satírozott területe (i=1..n, j=i..m-n+i).
- *Triviális részfeladatokat képviselő cellák*:
  - c[0,j], j=0,m-n; c[i,i], i=0..n. (világos szürke)
- *Eredeti feladatot képviselő cella*:
  - c[n,m]. (sötét szürke)



# Meghatározunk egy általános rekurzív képletet

- *„Utolsó döntés”* az „(i,j) feladatot” illetően:
  - (1) az i. virág a j. vázába kerül, vagy
    - fiú-részfeladat: (i-1,j-1)
  - (2) a j. váza üresen marad.
    - fiú-részfeladat: (i,j-1)
- *A képlet optimalizálási ága*:
  - c[i,j] = max{c[i-1,j-1] + e[i,j]; c[i,j-1] + 0}
- *A képlet triviális ágai*:
  - c[0,j] = 0;
  - c[i,i] = c[i-1,i-1] + e[i,i]

# Megírjuk az iteratív algoritmust (1)




| e | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 7 | 23 | -5 | -24 | 16 |
| 2 | 5 | 21 | -4 | 10 | 23 |
| 3 | -21 | 5 | -4 | -20 | 20 |

| c | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | **0** | **0** | 0 | | | |
| 1 | | 7 | **23** | | | |
| 2 | | | 28 | | | |
| 3 | | | | 24 | | |

| c | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | **0** | **0** | 0 | | | |
| 1 | | 7 | **23** | **23** | | |
| 2 | | | 28 | 28 | **33** | |
| 3 | | | | 24 | 24 | **53** |

| e | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 7 | 23 | -5 | -24 | 16 |
| 2 | 5 | 21 | -4 | 10 | 23 |
| 3 | -21 | 5 | -4 | -20 | 20 |

# Megírjuk az iteratív algoritmust (2)

```
minden j ← 0 … m-n végezd //triviális eset-1
   c[0,j] ← 0
vége minden
minden i ← 1 … n végezd //triviális eset-2
   c[i,i] ← c[i-1,i-1] + e[i,i]
vége minden
minden i ← 1 … n végezd //apáról-apára
   minden j ← i+1 … m-n+i végezd
      ha c[i-1,j-1] + e[i,j] > c[i,j-1] akkor
         c[i,j] ← c[i-1,i-1] + e[i,j]
      különben
         c[i,j] ← c[i,i-1]
      vége ha
   vége minden
vége minden
```

# Kiolvassuk („fentről-lefele” irányba) az optimális döntéssorozatot