

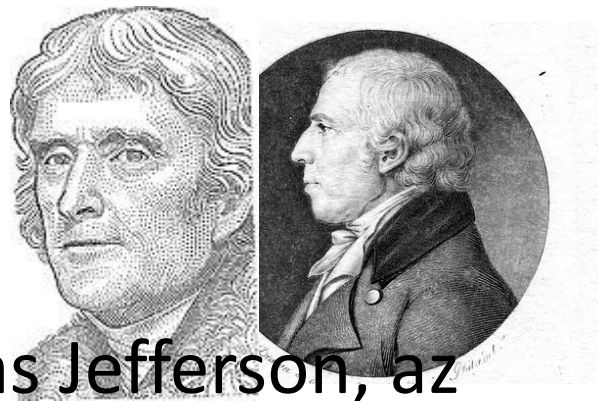
Dinamikus programozás – 1

10. ELŐADÁS

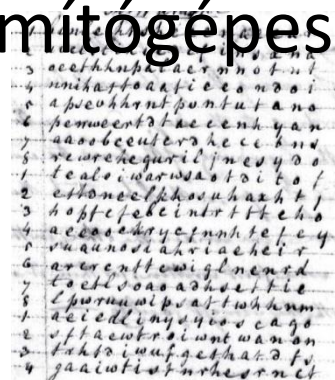
Sapientia-EMTE

2020-21

1801, Karácsony

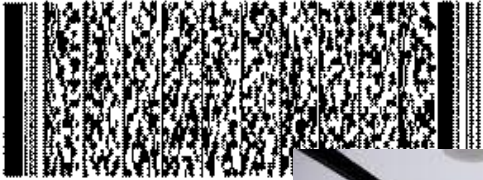


- 1801-ben Karácsonyra, Thomas Jefferson, az Amerikai Egyesült Államok akkori elnöke levelet kapott egyik matematikus barátjától, Robert Patterson-tól, aki egy általa tökéletesnek nevezett titkosítási rendszerről számolt be. Jefferson nyilván nem tudta feltörni a „tökéletes rendszert”, és az ezt követő 200 évben mások sem, viszont a közelmúltban egy Lawren Smithline nevű matematikusnak, számítógépes programok segítségével, sikerült.



Dinamikus programozás

1. binlelthseeneae
2. uelthieefinana
3. uelthhpalactnnot
4. unihattoaficee
5. apuohhntpntutano
6. pemoentataeentya
7. aaoeentersheci
8. rearegariljnesy
9. tealoicawastoi
10. etthneelpkshah
11. hoptefebcintth
12. aeecehrycfnph
13. ruauuoliahria
14. areentewiglon
15. toeloonasashett
16. lpuuawipacttwh
17. aciedlinyqiooca
18. vftacwotoluntwan
19. taktioaufgetha
20. gaaioitntnrhesn



A történet



- Richard Bellman
 - 1957: Bellman, „Dynamic Programming”
 - 1962: Bellman, Dreyfus, „Applied dynamic programming”
- Alkalmazási területek
 - Information theory
 - Control theory
 - Bioinformatics
 - Operations research
 - Computer science
 - Macro economy
 - Game theory
 - ...

Milyen feladatok oldhatók meg dinamikus programozással?

- Lebontásukból származó *különböző* részfeladatok száma a bemenet méretének polinom függvénye
 - a lebontásból származó exponenciálisan sok részfeladat közül számottevően sok azonos
- Optimalizálás alapelve
 - az optimális megoldás optimális részmegoldásokból épül fel
 - elegendő minden részfeladatnak csak az optimális megoldását meghatározni/tárolni

Egyszerűtől a bonyolult fele (1)



- Fibonacci számok generálása

The Fibonacci Sequence

1,1,2,3,5,8,13,21,34,55,89,144,233,377...

$$1+1=2$$

$$1+2=3$$

$$2+3=5$$

$$3+5=8$$

$$5+8=13$$

$$8+13=21$$

$$13+21=34$$

$$21+34=55$$

$$34+55=89$$

$$55+89=144$$

$$89+144=233$$

$$144+233=377$$

0	1	1	2	3	5	8	13	21	...
0	1	2	3	4	5	6	7	8	...



Egyszerűtől a bonyolult fele (2)



- Adott n féle pénzérme, melyek értékeit a $w[1..n]$ tömb tárolja, valamint egy S összeg. Fizessük ki az S összeget minimális pénzérme felhasználásával (bármelyikből bármennyi felhasználható).

– INPUT:

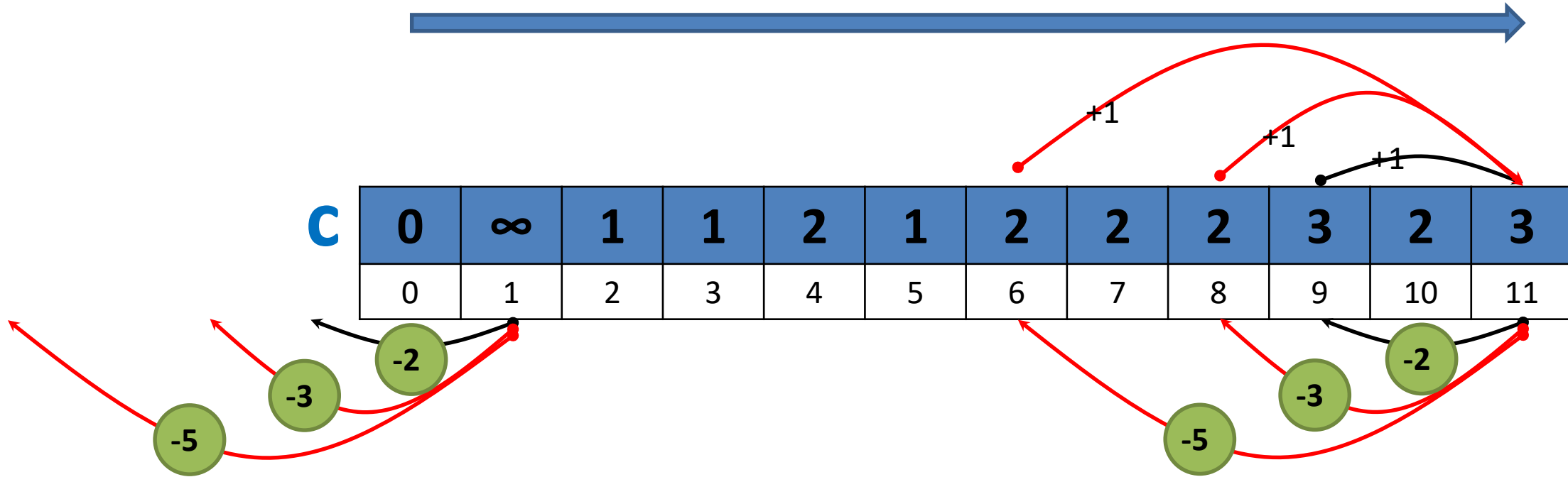
- $S=11$
- $n=3, w[1..3] = \{2,3,5\}$

– OUTPUT:

- $11 = 3+3+5$

Összeg	Min-érme-szám	Felhasznált érmék
0	0	-
1	∞	-
2	1	2
3	1	3
4	2	2, 2
5	1	5
6	2	3, 3
7	2	2, 5
8	2	3, 5
9	3	2, 2, 5
10	2	5, 5
11	3	3, 3, 5

$S=11, n=3, w[1..3] = \{2,3,5\}$



- A megoldást **felépítjük**, lépésről-lépésre:
 - Megoldjuk a feladatot rendre $i = 0, 1, \dots, S$ összegekre
 - Adott i -re a minimális pénzérme-számot **tárolja** $c[i]$
 - Kurrens i -re a $c[i]$ -t, a már rendelkezésre álló $c[j]$ -kből ($j < i$) számítjuk ki
 - Kell egy **rekurzív képlet**, amely ezt az építkezést matematikailag leírja
 - $i=0$ esetre a minimális pénzérme-szám 0 ($c[0] = 0$)
 - Az i összeg, mely j összegekből ($j < i$) építhető fel egy pénzérme hozzáadásával?
 - $c[i] = \min\{c[i-w[k]] + 1; \text{minden } k=1..n\text{-re, ahol } w[k] \leq i\}$

C

0	∞	1	1	2	1	2	2	2	3	2	3
0	1	2	3	4	5	6	7	8	9	10	11

$i-w[3]$ $i-w[2]$ $i-w[1]$ i

$$c[11] = 1 + \min \{ c[9], c[8], c[6] \}$$

```

c[0] = 0;
for ( i = 1 ; i <= S ; ++i ) {
    c[i] = VEGTELEN;
    for ( k = 1 ; k <= n ; ++k ) {
        if ( i-w[k] >= 0 && c[i-w[k]] < c[i] ) {
            c[i] = c[i-w[k]];
        }
    }
    ++c[i];
}
cout << c[S];

```

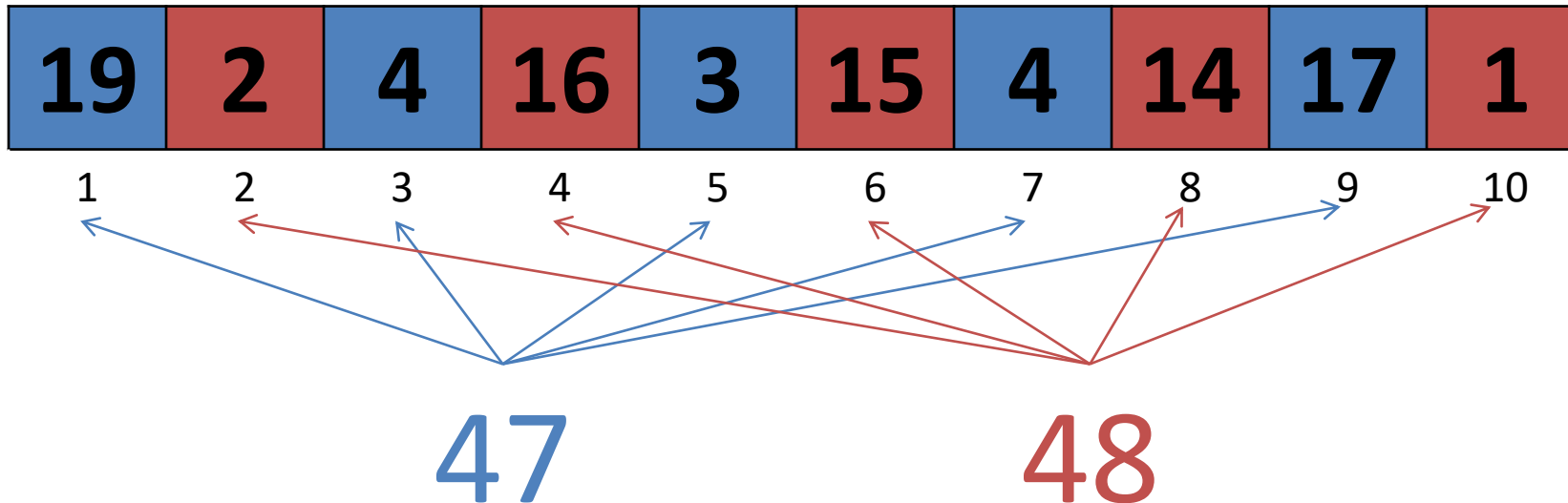
- $c[S]$ csak a minimális pénzérme-számot adja meg.
- Hogyan határoznád meg, hogy mely pénzérme-k biztosítják ezt az optimális értéket?
Például: $3 + 3 + 5 = 11$.
- És ha az is érdekel, hogy hányféleképpen lehet minimális számú pénzérme-vel kifizetni S -t? És melyek ezek?

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,
következétesen választhat
páros/páratlan indexű elem között



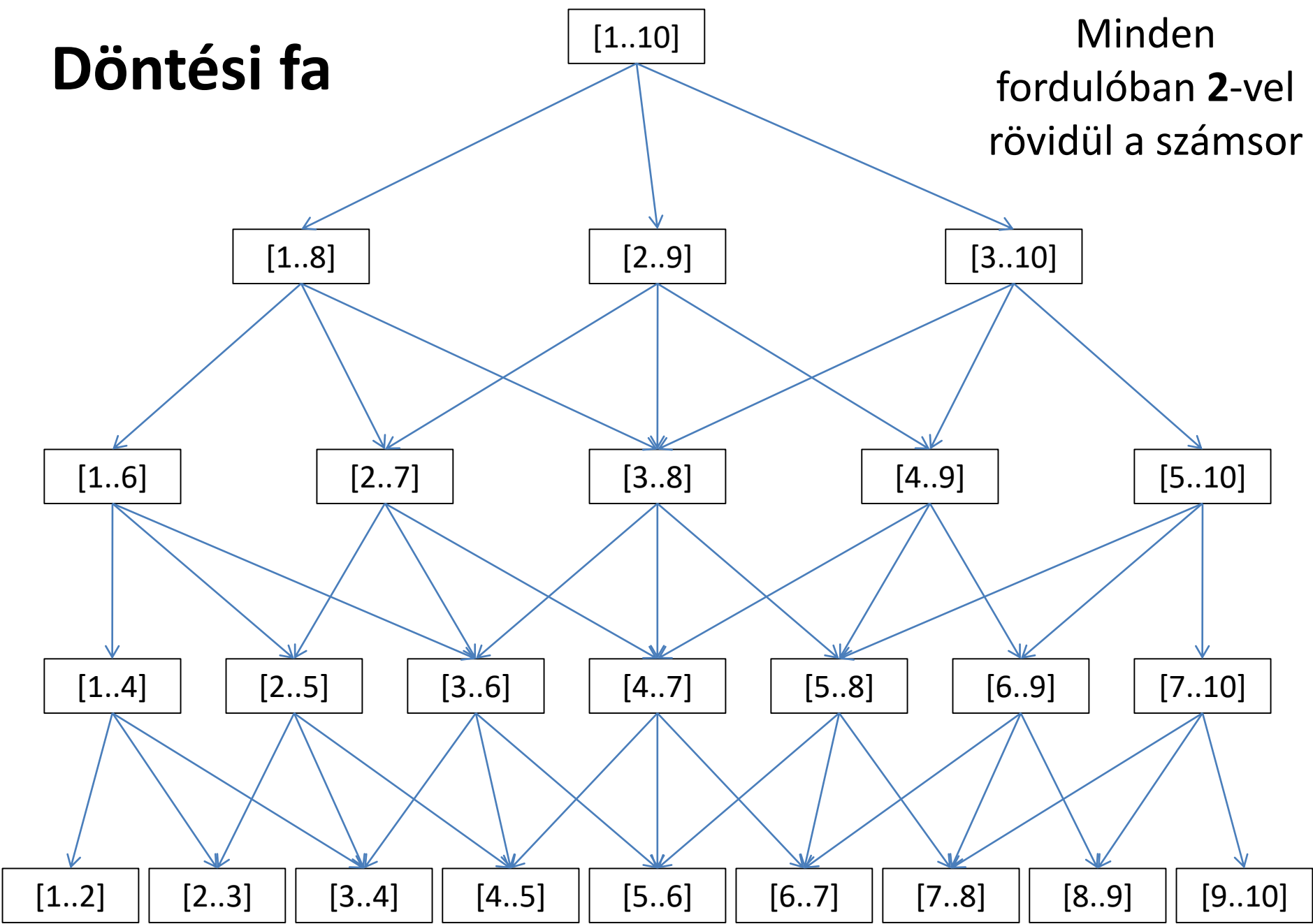
Mennyi a maximális összeg, amit a kezdő *garantáltan* összeszedhet?

- 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.



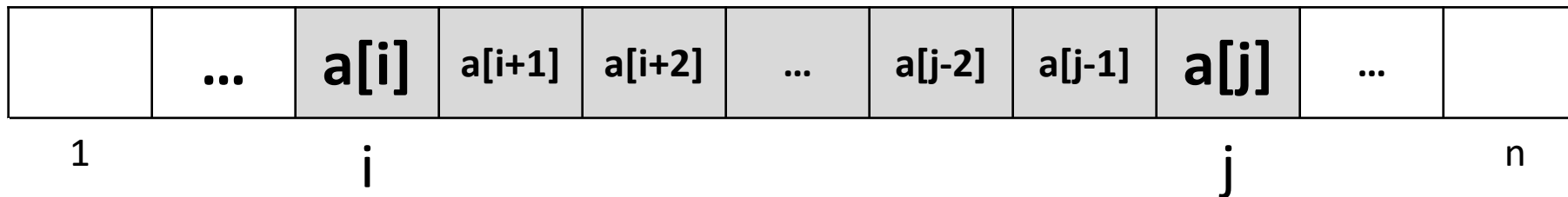
Döntési fa

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



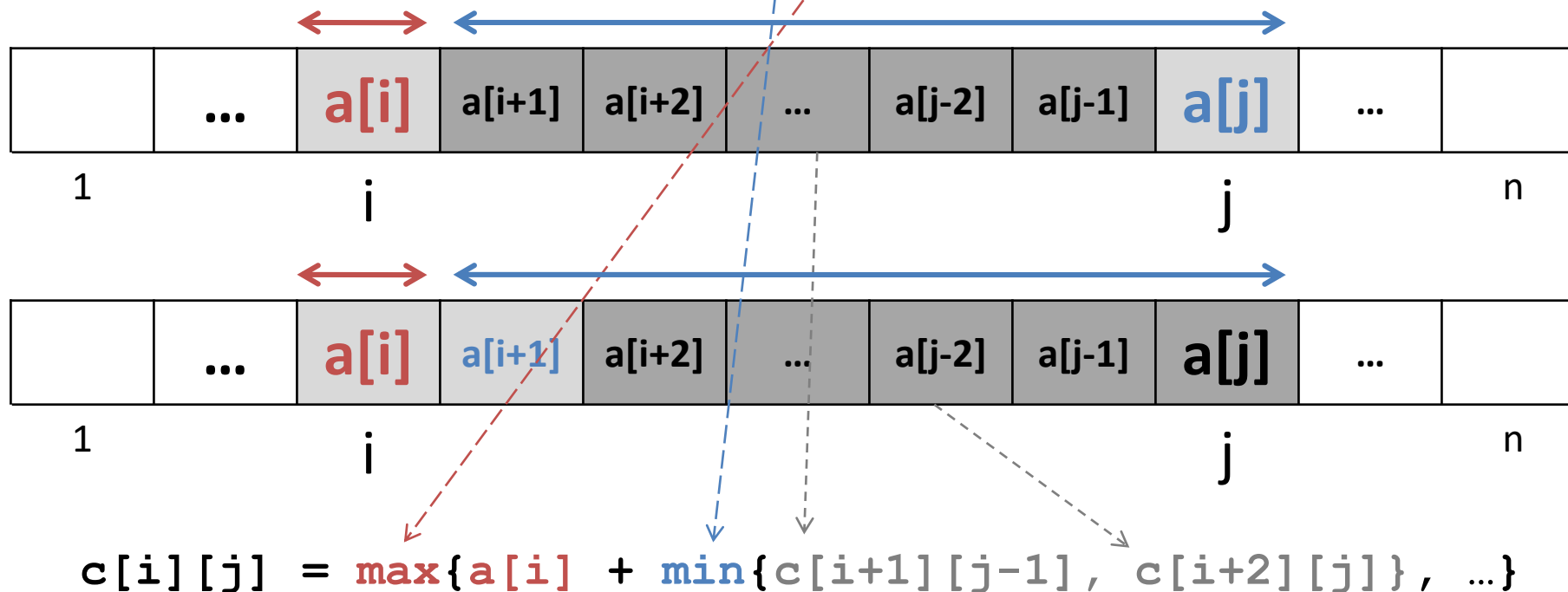
Mennyi a maximális összeg, amit a kezdő *garantáltan* összeszedhet?

- 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?



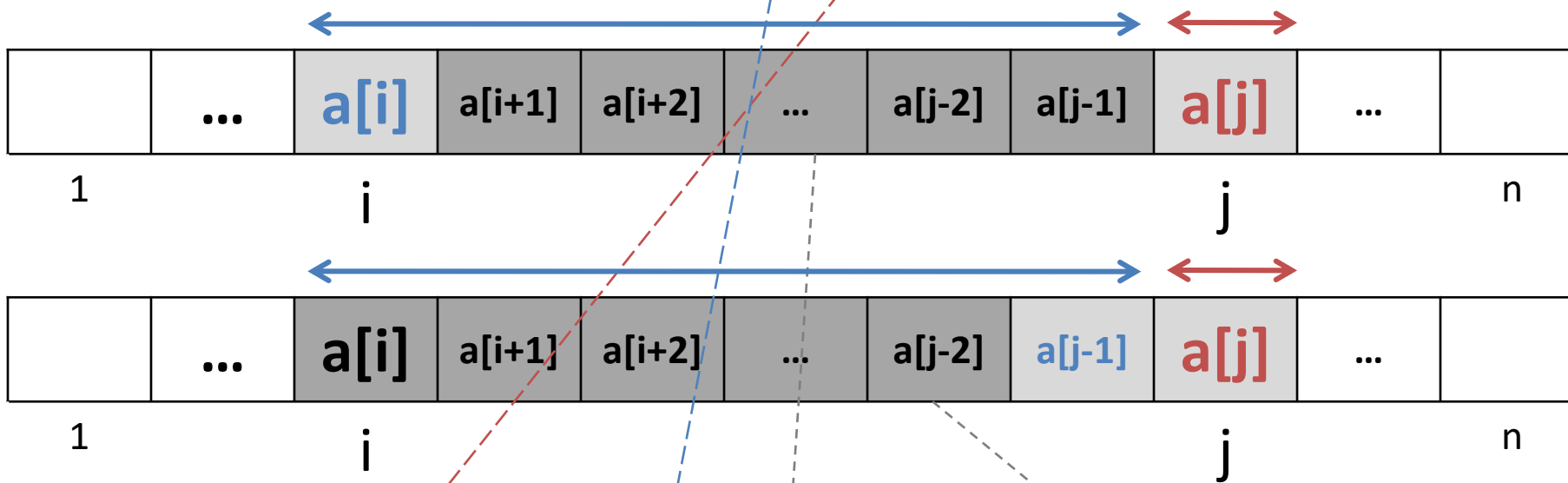
Két út áll előtttem, melyiket válasszam?
 $a[i]$ -t vagy $a[j]$ -t? Az **előnyösebbet**!

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



Két út áll előtttem, 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**!



$$c[i][j] = \max \left\{ \begin{array}{l} a[i] + \min\{c[i+1][j-1], c[i+2][j]\}, \\ a[j] + \min\{c[i+1][j-1], c[i][j-2]\} \end{array} \right\}$$

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

19
2
4
16
3
15
4
14
17
1

	1	2	3	4	5	6(j)	7	8	9	10
1										
2										
3(i)										
4										
5										
6										
7										
8										
9										
10										

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

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

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

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

Forduló

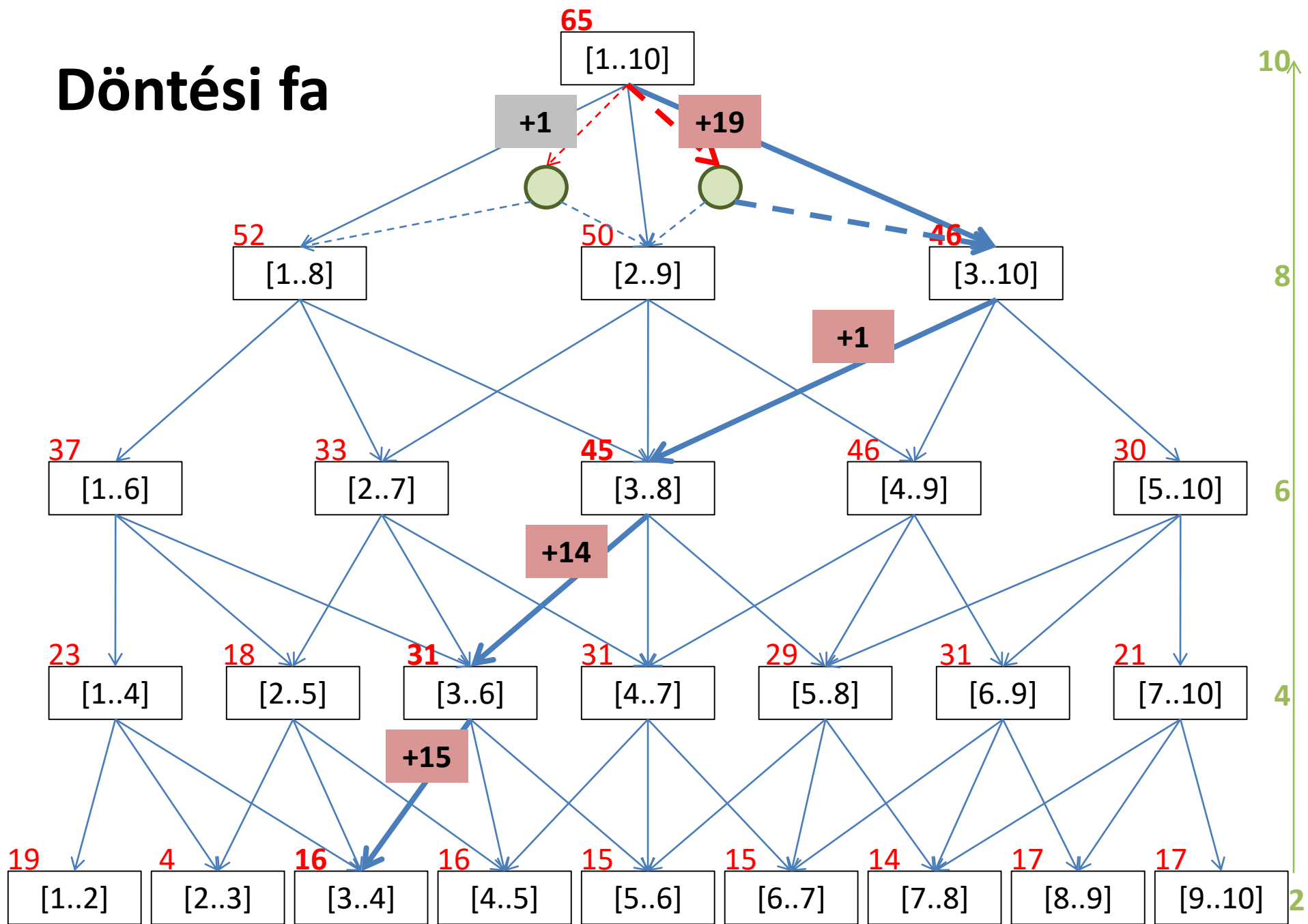
1	1	5	5	4	4	3	3	2	2
19	2	4	16	3	15	4	14	17	1

1	19
1	2
5	4
5	16
4	3
4	15
3	4
3	14
2	17
2	1

	1	2	3	4	5	6	7	8	9	10
1		19		23		37		52	+1	65
2			4		18		33		50	+19
3		0	+16	16	+15	31	+14	45	+1	46
4			0	+4	16	+4	31	+4	46	+4
5				0		15		29		30
6							15		31	
7								14		21
8									17	
9										17
10										

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

Döntési fa



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]);
```

Optimális játék (1)

```
i = 1; j = n; int s1 = 0, s2 = 0;
for( k = 1 ; k <= n/2 ; ++k ){ //fordulonkent
    kiir_aktualis_allas(a,n,i,j,s1,s2);
    //lep PC
    if ( c[i][j] == ooo ){
        ooo
    }
    else{
        ooo
    }

    kiir_aktualis_allas(a,n,i,j,s1,s2);
    //lep USER
    printf("He te LOSER, azaz USER, mit választasz, (b)al vagy (j)obb?\n");
    char c = getch();
    if ( c == 'b' ){
        ooo
    }
    else{
        ooo
    }
}
printf("EREDMENY:\n");
kiir_aktualis_allas(a,n,i,j,s1,s2);
```

Optimális játék (2)

```
i = 1; j = n; int s1 = 0, s2 = 0;
for( k = 1 ; k <= n/2 ; ++k ){ //fordulonkent
    kiir_aktualis_allas(a,n,i,j,s1,s2);
    //lep PC
    if ( c[i][j] == a[i] + mini(c[i+1][j-1],c[i+2][j]) ){
        s1 += a[i]; ++i;
    }
    else{
        ooo
    }

    kiir_aktualis_allas(a,n,i,j,s1,s2);
    //lep USER
    printf("He te LOSER, azaz USER, mit választasz, (b)al vagy (j)obb?\n");
    char c = getch();
    if ( c == 'b' ){
        s2 += a[i]; ++i;
    }
    else{
        ooo
    }
}
printf("EREDMENY:\n");
kiir_aktualis_allas(a,n,i,j,s1,s2);
```

A gyorsaknak!

- Balanced Partition
 - You have a set of n integers each in the range $0 \dots K$. 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:
 - optima miként építhető fel a közvetlen fiúrészfeladatok optimaiból (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 optima mely közvetlen fiú-részfeladatok optimaiból építhető fel, és hogyan?
4. A rekurzív képlet alapján - az egyszerűtől haladva a bonyolult fele - feltöltjük a tömböt a részfeladatok optimum értékeivel.

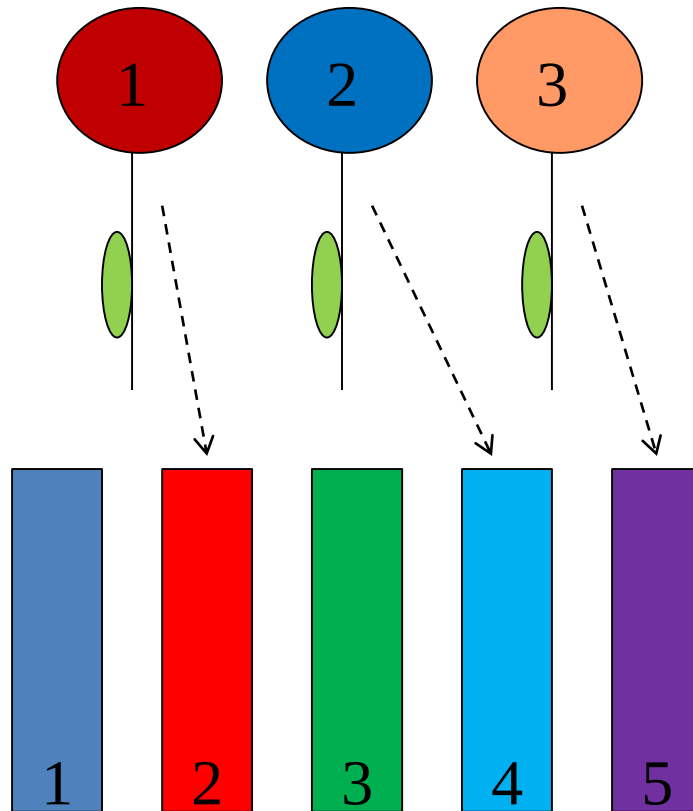
Virágüzlet-feladat

- Egy virágüzlet kirakatában van m váza ($1, 2, \dots, m$ sorrendben) és ezekbe úgy kell elhelyezni az $1, 2, \dots, n$ 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)

c	0	1	2	3	4	5
0						
1						
2						
3						

Meghatározzunk 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:*

- $c[i,j] = \max\{c[i-1,j-1] + e[i,j]; c[i,j-1] + 0\}$

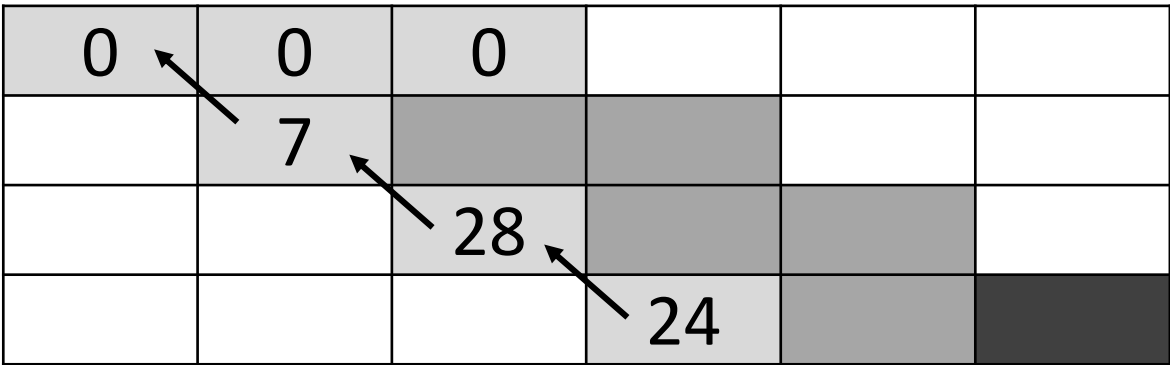
- *A képlet triviális ágai:*

- $c[0,j] = 0;$

- $c[i,i] = c[i-1,i-1] + e[i,i]$

Megírjuk az iteratív algoritmust (1)

c	0	1	2	3	4	5
0	0	0	0			
1		7				
2			28			
3				24		



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 $j \leftarrow 0 \dots m-n$ **végezd** //triviális eset-1

$c[0,j] \leftarrow 0$

vége minden

minden $i \leftarrow 1 \dots n$ **végezd** //triviális eset-2

$c[i,i] \leftarrow c[i-1,i-1] + e[i,i]$

vége minden

minden $i \leftarrow 1 \dots n$ **végezd** //apáról-apára

minden $j \leftarrow i+1 \dots m-n+i$ **végezd**

ha $c[i-1,j-1] + e[i,j] > c[i,j-1]$ **akkor**

$c[i,j] \leftarrow c[i-1,i-1] + e[i,j]$

különben

$c[i,j] \leftarrow c[i,i-1]$

vége ha

vége minden

vége minden

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

c	0	1	2	3	4	5
0	0 ←	0 ↖	0 ↖			
1		7 ←	23 ←	23 ↖		
2			28	28 ←	33 ↖	
3				24	24 ←	53 ↖

Folytatás...