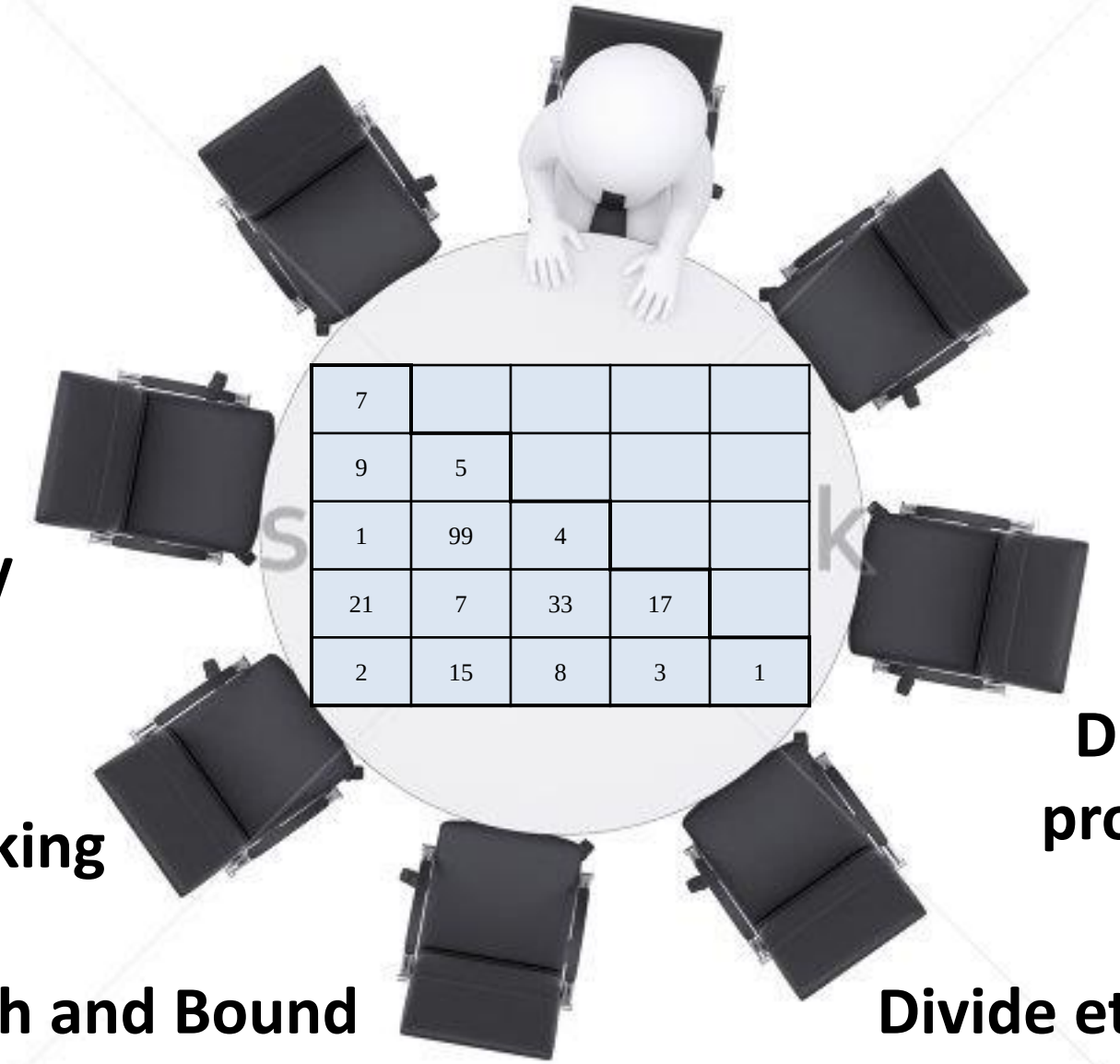
Az algoritmustervezési stratégiák bemutatkoznak

KÁTAI Zoltán

VII. Kiss Elemér programozói tábor

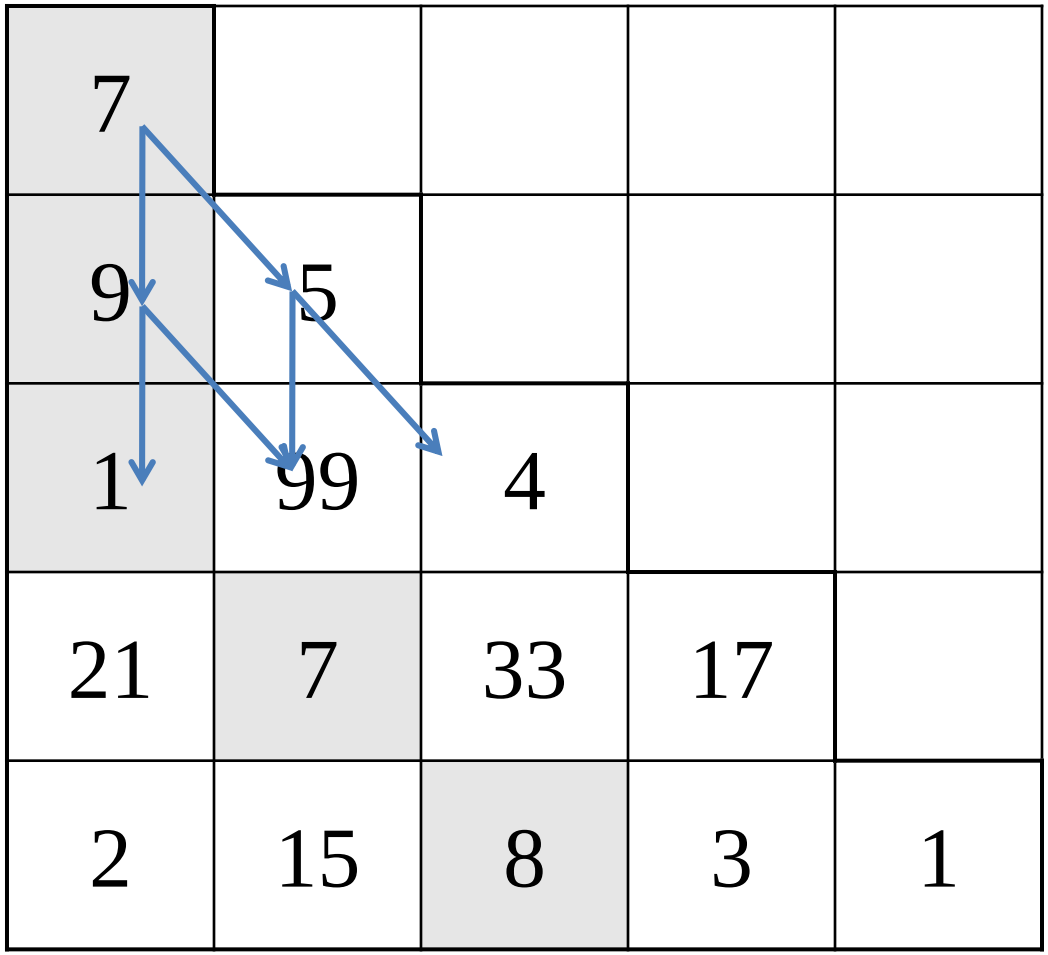
Mezősályi, 2017. március 3-5


7
9 5
1 99 4
21 7 33 17
2 15 8 3 1
Greedy
Backtracking
Branch and Bound
Divide et Impera
Dinamikus programozás

TALKSHOW

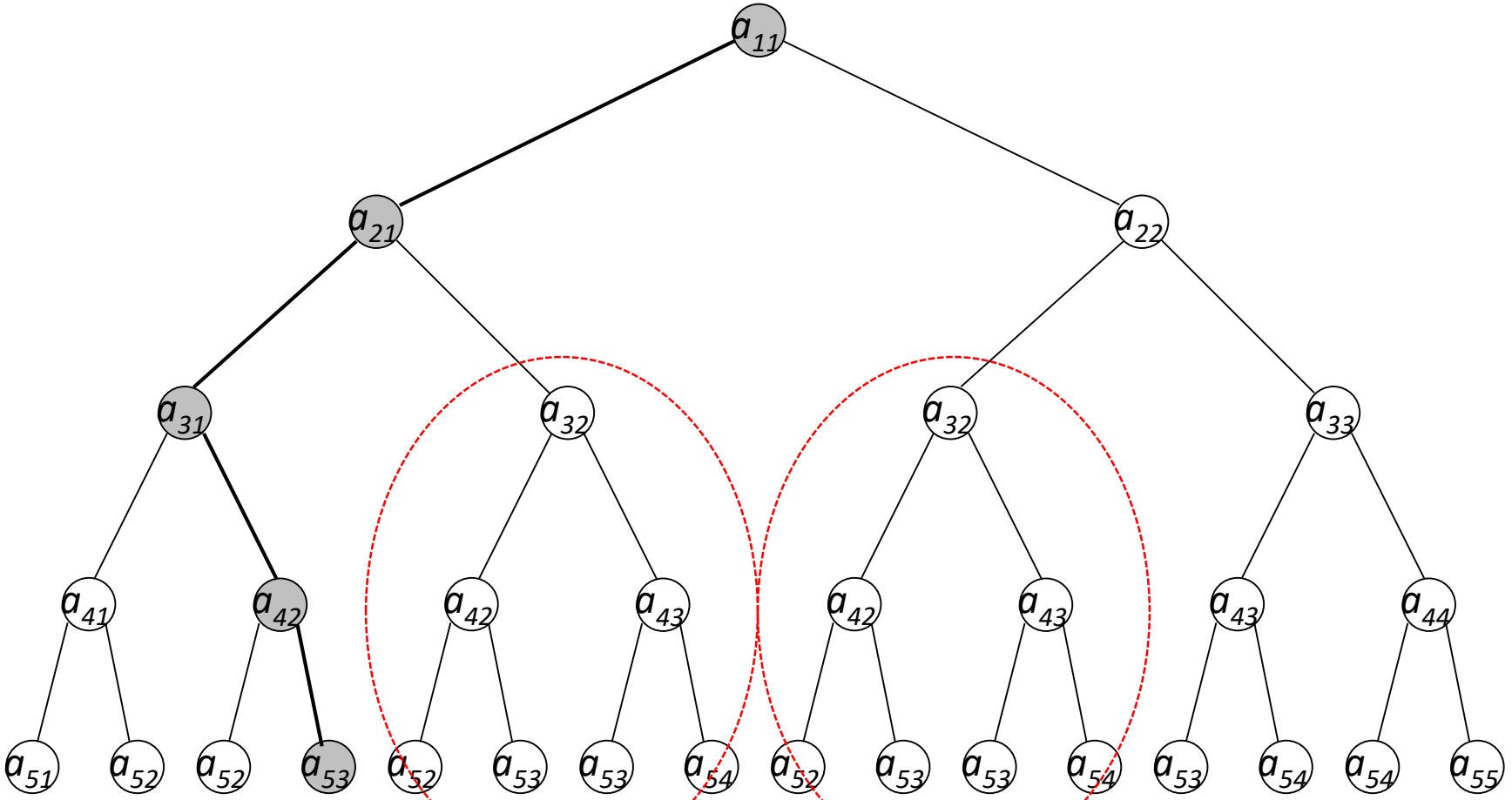
Vitaindító feladat

a



32

A feladat megoldástere



A fa csomópontjainak száma: $1+2+2^2+...+2^{n-1}=2^n-1$.

Különböző részfeladatok száma: $n(n+1)/2$

GREEDY
(*Élj a mának*)

7				
9	5			
1	99	4		
21	7	33	17	
2	15	8	3	1

34

GREEDY
(*Élj a mának*)

```
G(i,j,s)
  s += a[i][j]
  ha i < n akkor
     ha a[i+1][j] < a[i+1][j+1] akkor
          return G(i+1,j,s)
     különben
          return G(i+1,j+1,s)
     vége_ha
  vége_ha
  return s
vége_G
```

```
Beolvas n
Beolvas a[1..n][1..n]
Kiír G(1,1,0)
```

Backtracking

(*Legbizonyosabban úgy találod el a verebet, ha ágyúval lősz a fára*)

7				
9	5			
1	99	4		
21	7	33	17	
2	15	8	3	1

40 53 39 32 … 34

Backtracking

(*Legbizonyosabban úgy találod el a verebet, ha ágyúval lősz a fára*)

```
Bt(i,j,s)
  s += a[i][j]
  ha i == n akkor
     ha s < S akkor
          S = s
     vége_ha
  különben
     Bt(i+1,j,s)
     Bt(i+1,j+1,s)
  vége_ha
vége_Bt
```

```
Beolvas n
Beolvas a[1..n][1..n]
S = ∞
Bt(1,1,0)
Kiír S
```

Backtracking + Branch and bound

Backtracking + Branch and bound

```
Bt(i,j,s)
  s += a[i][j]
  ha s ≥ S  akkor return vége_ha
  ha i == n akkor
     ha s < S akkor
          S = s
     vége_ha
  különben
     Bt(i+1,j,s)
     Bt(i+1,j+1,s)
  vége_ha
vége_Bt
```

```
Beolvas n
Beolvas a[1..n][1..n]
S = ∞
Bt(1,1,0)
Kiír S
```

Branch and bound (best first)

Branch and bound (best first)

1. {}
2. {(1,1,**7**)}
3. {(2,2,**12**), (2,1,**16**)}
4. {(2,1,**16**), (3,3,**16**), (3,2,**111**)}
5. {(3,3,**16**), (3,1,**17**), (3,2,**111**), (3,2,**115**)}
6. {(3,1,**17**), (4,4,**33**), (4,3,**49**), (3,2,**111**), (3,2,**115**)}
7. {(4,2,**24**), (4,4,**33**), (4,1,**38**), (4,3,**49**), (3,2,**111**), (3,2,**115**)}
8. {(5,3,**32**), (4,4,**33**), (4,1,**38**), (5,2,**39**), (4,3,**49**), (3,2,**111**), (3,2,**115**)}
9. {(4,4,**33**), (4,1,**38**), (5,2,**39**), (4,3,**49**), (3,2,**111**), (3,2,**115**)}
10. {(4,1,**38**), (5,2,**39**), (4,3,**49**), (3,2,**111**), (3,2,**115**)}
11. {(5,2,**39**), (4,3,**49**), (3,2,**111**), (3,2,**115**)}
12. {(4,3,**49**), (3,2,**111**), (3,2,**115**)}
13. {(3,2,**111**), (3,2,**115**)}
14. {(3,2,**115**)}
15. {}

Branch and bound (best first)

```
Bb()
   pQ = (1,1,a[1][1]) //prioritás sor
   S = ∞
   amíg NEM üres(pQ) végezd
        (i,j,s) = kivesz_sorelső(pQ)
        ha s ≥ S  akkor continue vége_ha
        ha i == n akkor
              ha s < S akkor
                    S = s
              vége_ha
        különben
              beszúr_sorba(pQ,i+1,j,s+a[i+1][j])
              beszúr_sorba(pQ,i+1,j+1,s+a[i+1][j+1])
        vége_ha
   vége_amíg
   kiír S
vége_Bb
```

```
Beolvas n
Beolvas a[1..n][1..n]
Bb()
```

Divide et impera
(*Oszd meg és uralkodj*)

a

7				
9	5			
1	99	4		
21	7	33	17	
2	15	8	3	1

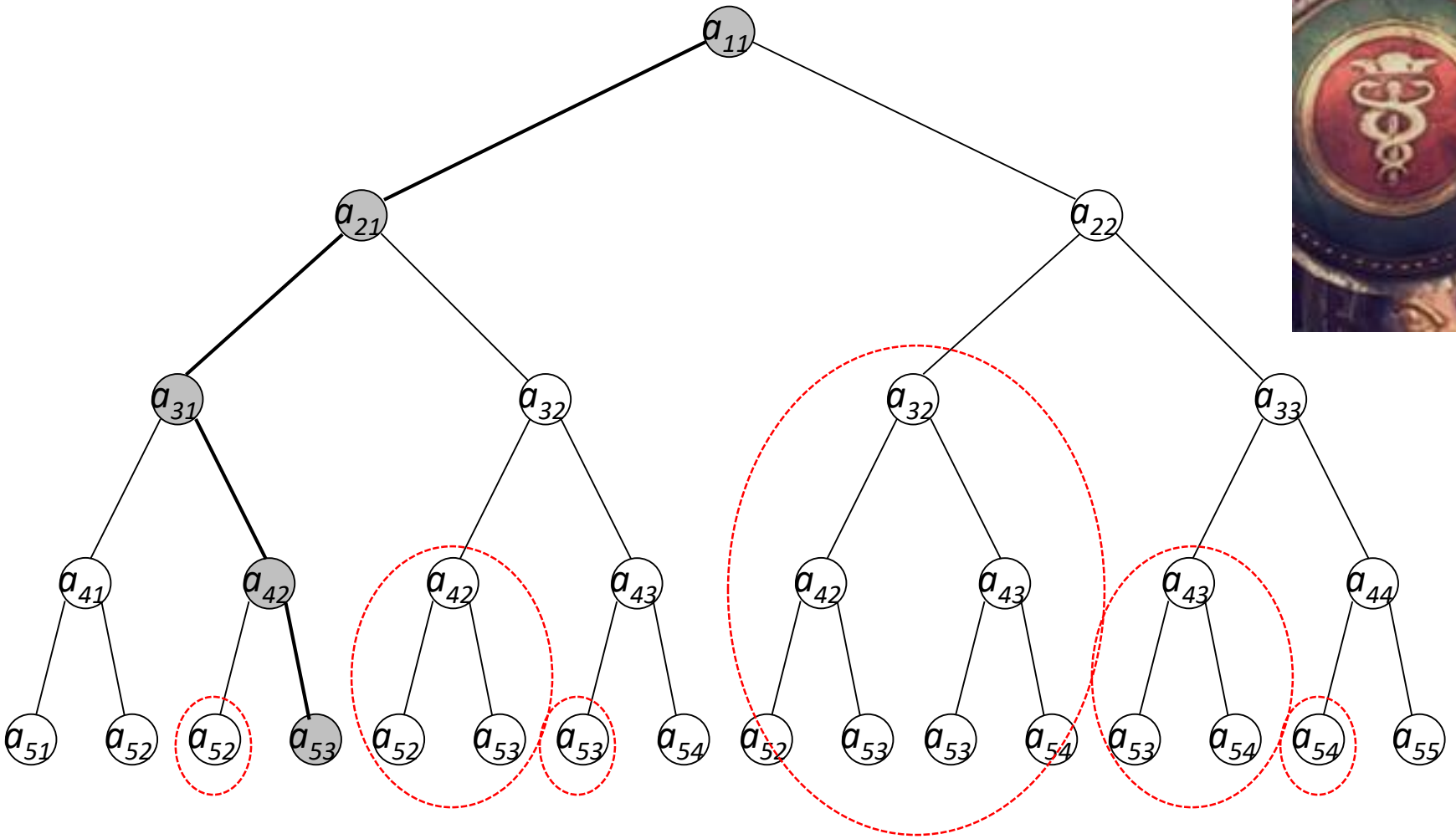
legjobbút(i,j) = a[i][j] + min{legjobbút(i+1,j), legjobbút(i+1,j+1)}

Divide et impera
(*Oszd meg és uralkodj*)

```
Di(i,j)
  ha i == n akkor
     return a[n][j]
  különben
     return a[i][j]+ min{Di(i+1,j),
                         Di(i+1,j+1)}
  vége_ha
vége_Di
```

```
Beolvas n
Beolvas a[1..n][1..n]
Kiír Di(1,1)
```

Divide et impera
(*Oszd meg és uralkodj*)



Divide et impera +
Dinamikus programozás

```
Di_Dp(i,j)
  ha c[i][j] ≠ -1 akkor
     return c[i][j]
  különben
     ha i == n akkor
          return c[n][j] = a[n][j]
     különben
          return c[i][j] = a[i][j]+
                           min{Di_Dp(i+1,j),
                           Di_Dp(i+1,j+1)}
     vége_ha
  vége_ha
vége_Di_Dp
```

```
Beolvas n
Beolvas a[1..n][1..n]
c[1..n][1..n] = -1
Kiír Di_Dp(1,1)
```

Branch and bound +
Dinamikus programozás

```
Bb_Dp()
   pQ = (1,1,a[1][1])
   S = ∞
   amíg NEM üres(pQ) végezd
        (i,j,s) = kivesz_sorelső(pQ)
        ha s ≥ S  akkor continue vége_ha
        ha i == n akkor
              ha s < S akkor
                    S = s
              vége_ha
        különben
              ha … beszúr_sorba(pQ,i+1,j,s+a[i+1][j])
              ha … beszúr_sorba(pQ,i+1,j+1,s+a[i+1][j+1])
        vége_ha
   vége_amíg
   kiír S
vége_Bb_Dp
```

```
Beolvas n
Beolvas a[1..n][1..n]
Bb_Dp()
```

Iteratív dinamikus programozás levelek-gyökér irány

a

7				
9	5			
1	99	4		
21	7	33	17	
2	15	8	3	1

c

32				
25	27			
16	114	22		
23	15	36	18	
2	15	8	3	1

Iteratív dinamikus programozás gyökér-levelek irány

a

7				
9	5			
1	99	4		
21	7	33	17	
2	15	8	3	1

c

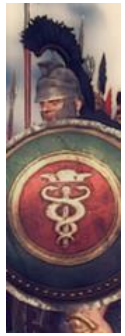
7				
16	12			
17	111	16		
38	24	49	33	
40	39	32	36	34

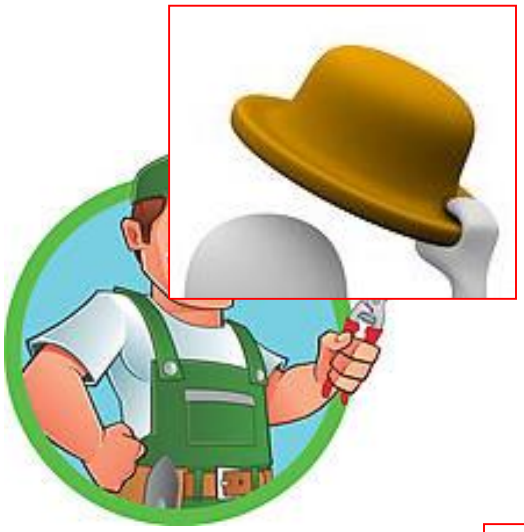
```
#include <time.h>
clock_t kezdet, veg; double idointervallum;
kezdet = clock();   ooo  veg = clock();
idointervallum = (double)(veg-kezdet) / CLOCKS_PER_SEC;
```

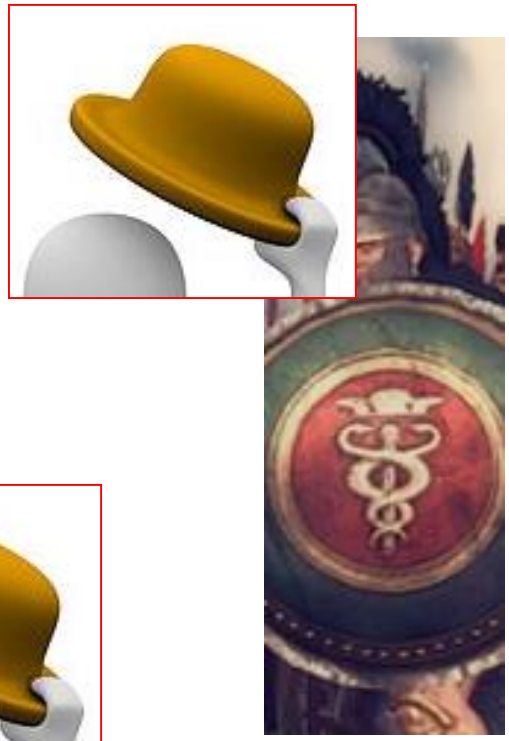
- **G**reedy (n = 3000, t = ?)
- **B**ack**t**racking (n = 27, t = ?)
- **B**ack**t**racking + **B**ranch and **b**ound (n = 37, t = ?)
- **B**ranch and **b**ound (best first) (n = 27, t = ?)
- **D**ivide et **i**mpera (n = 27, t = ?)
- **D**ivide et **i**mpera + **D**imanikus **p**rogramozás (n = 3000, t = ?)
- **B**ranch and **b**ound + **D**imanikus **p**rogramozás (n = 3000, t = ?)
- Iteratív **D**imanikus **p**rogramozás (n = 3000, t = ?)











Jó munkát!