Algoritmusok felülnézetből

3. ELŐADÁS

Sapientia-EMTE

2015-16

Felülnézet módszer (1)

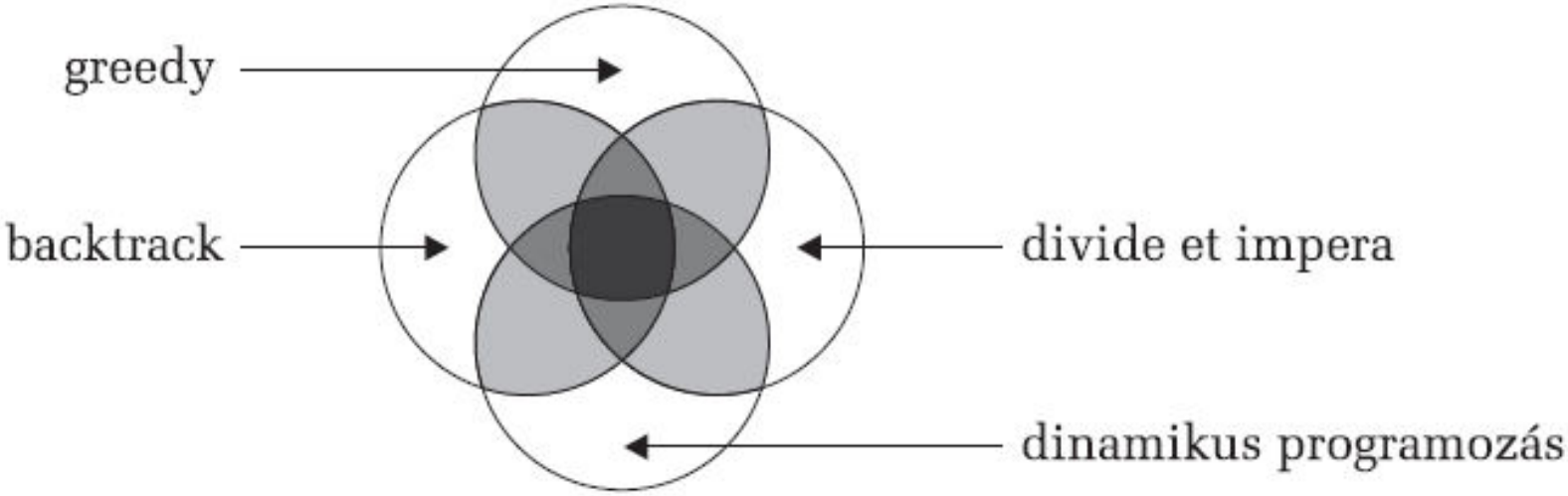
- „Tanítani szinte nem is jelent mást, mint megmutatni, miben különböznek egymástól a dolgok a különböző céljukat, megjelenési formájukat és eredetüket illetően. . . Ezért aki jól megkülönbözteti egymástól a dolgokat, az jól is tanít” (*Comenius*)

Felülnézet módszer (2)

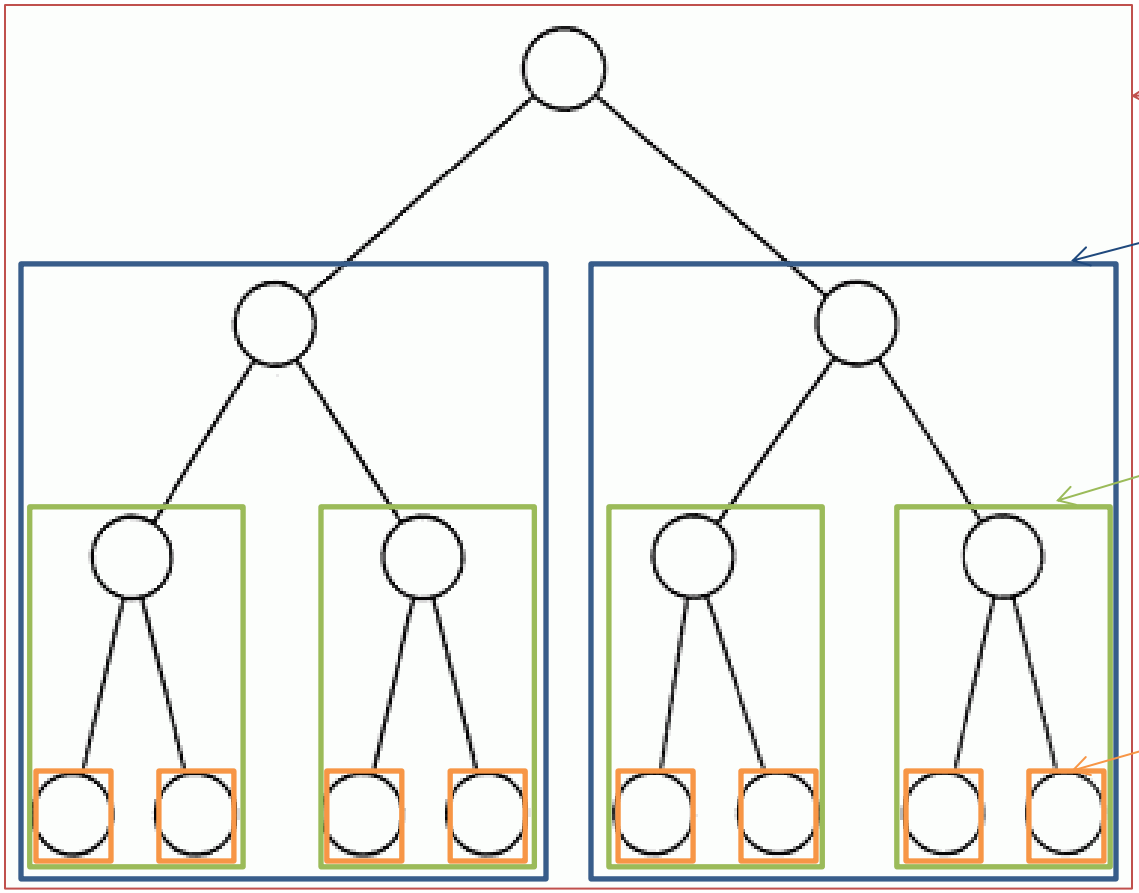
- Mit jelent felülről látni valamit?
  - Egymás mellett látjuk a vizsgálat tárgyát képező entitásokat.
  - Csak az látszik, ami a vizsgálat szempontjából lényeges.
  - Nyilvánvalóak a hasonlóságok és a különbségek, szembetűnők a kapcsolatok.

Felülnézetek az informatikaoktatásban

- Rendezési algoritmusok
  - ugyanazt a feladatot oldják meg
- Programozási technikák



Fastruktúra szerkezetű feladatok



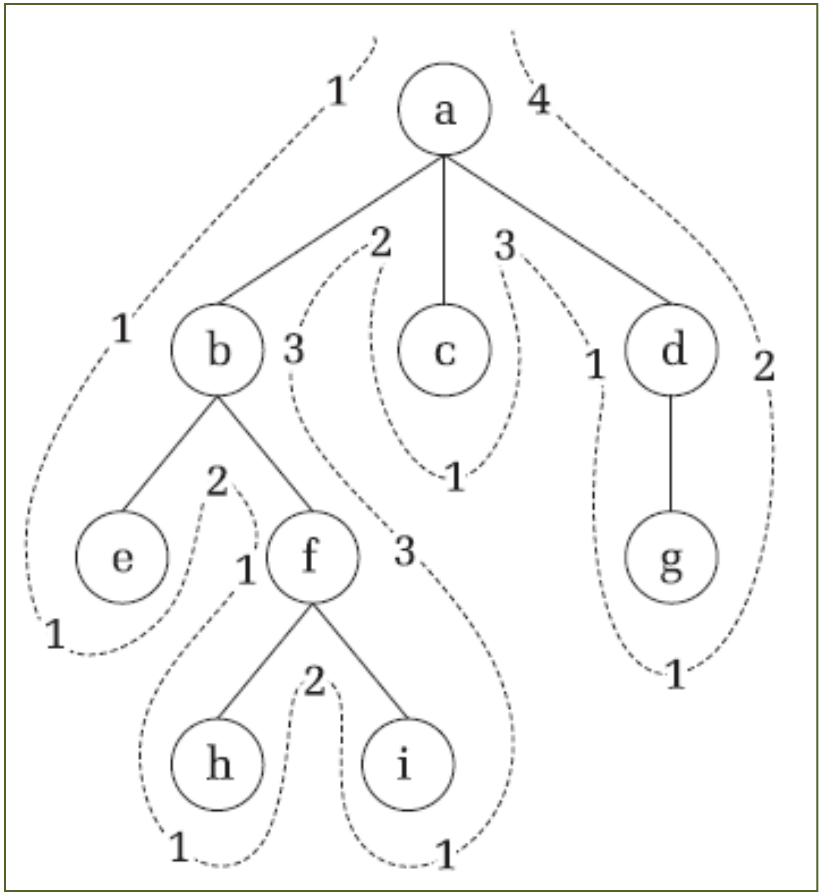
eredeti feladat

fiú részfeladatok

unoka részfeladatok

triviális részfeladatok

Mélységi bejárások



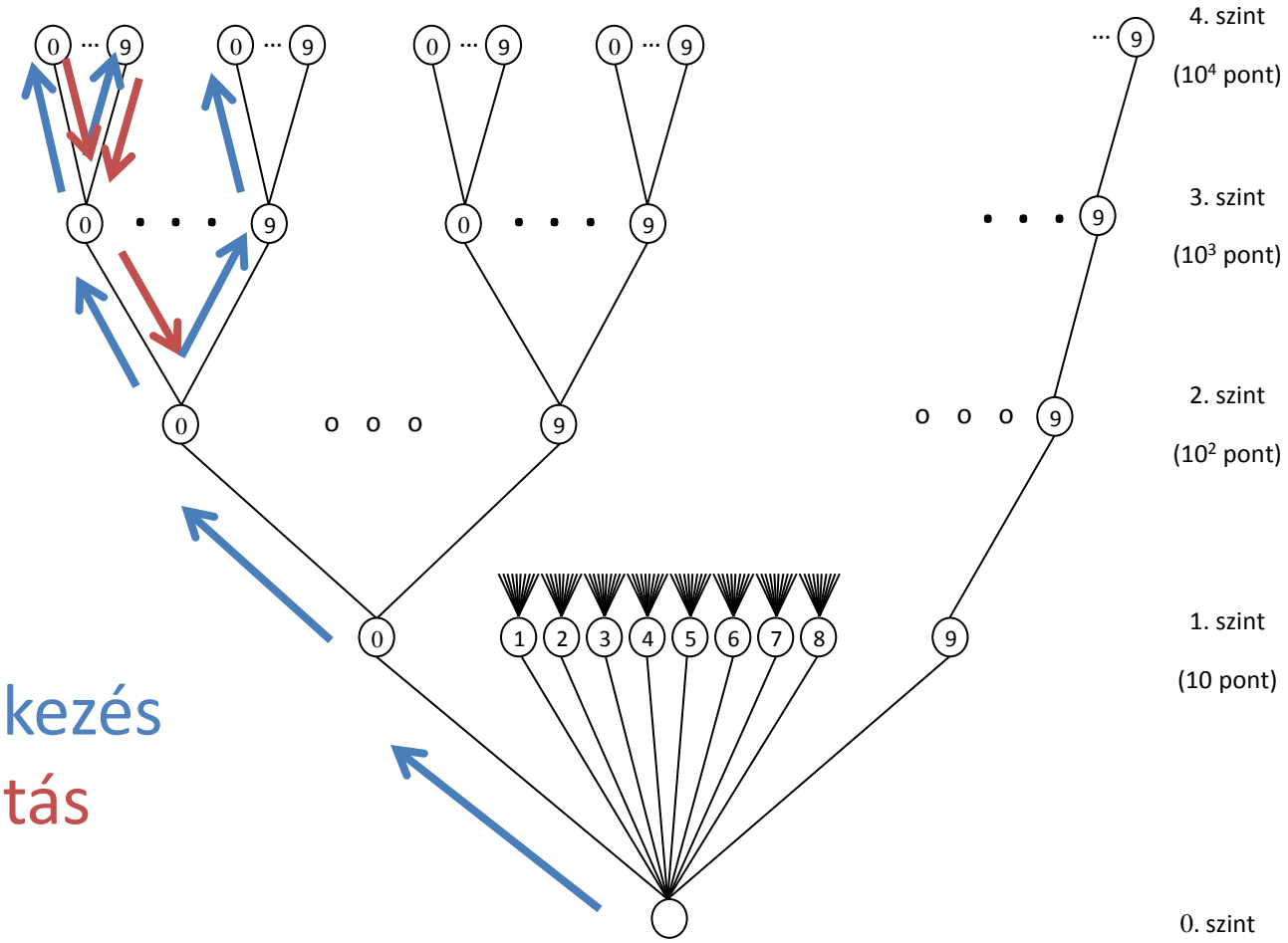
Preorder bejárás szerinti sorrend:
a, b, e, f, h, i, c, d, g

Postorder bejárás szerinti sorrend:
e, h, i, f, b, c, g, d, a

```
preorder(cs)
   meglátogat(cs)
   minden f_i fiára cs-nek végezd
      preorder(f_i)
   vége minden
vége preorder
```

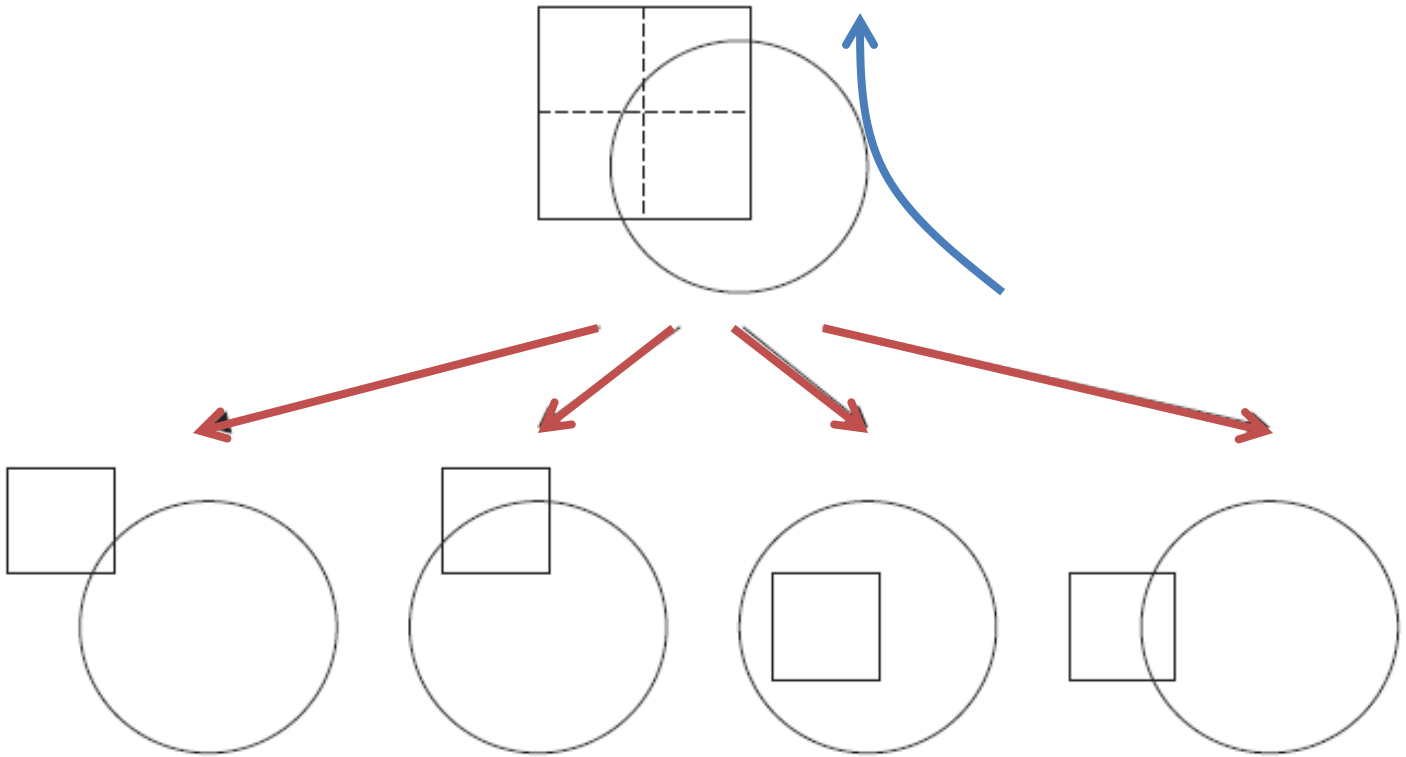
```
postorder(cs)
   minden f_i fiára cs-nek végezd
      postorder(f_i)
   vége minden
   meglátogat(cs)
vége postorder
```

Backtracking: kerékpár-zár



- Építkezés
- Bontás

Oszd meg és uralkodj: Négyzet és kör



- Építkezés (postorder momentumokban)
- Bontás

Backtracking vs. **Oszd meg és uralkodj**

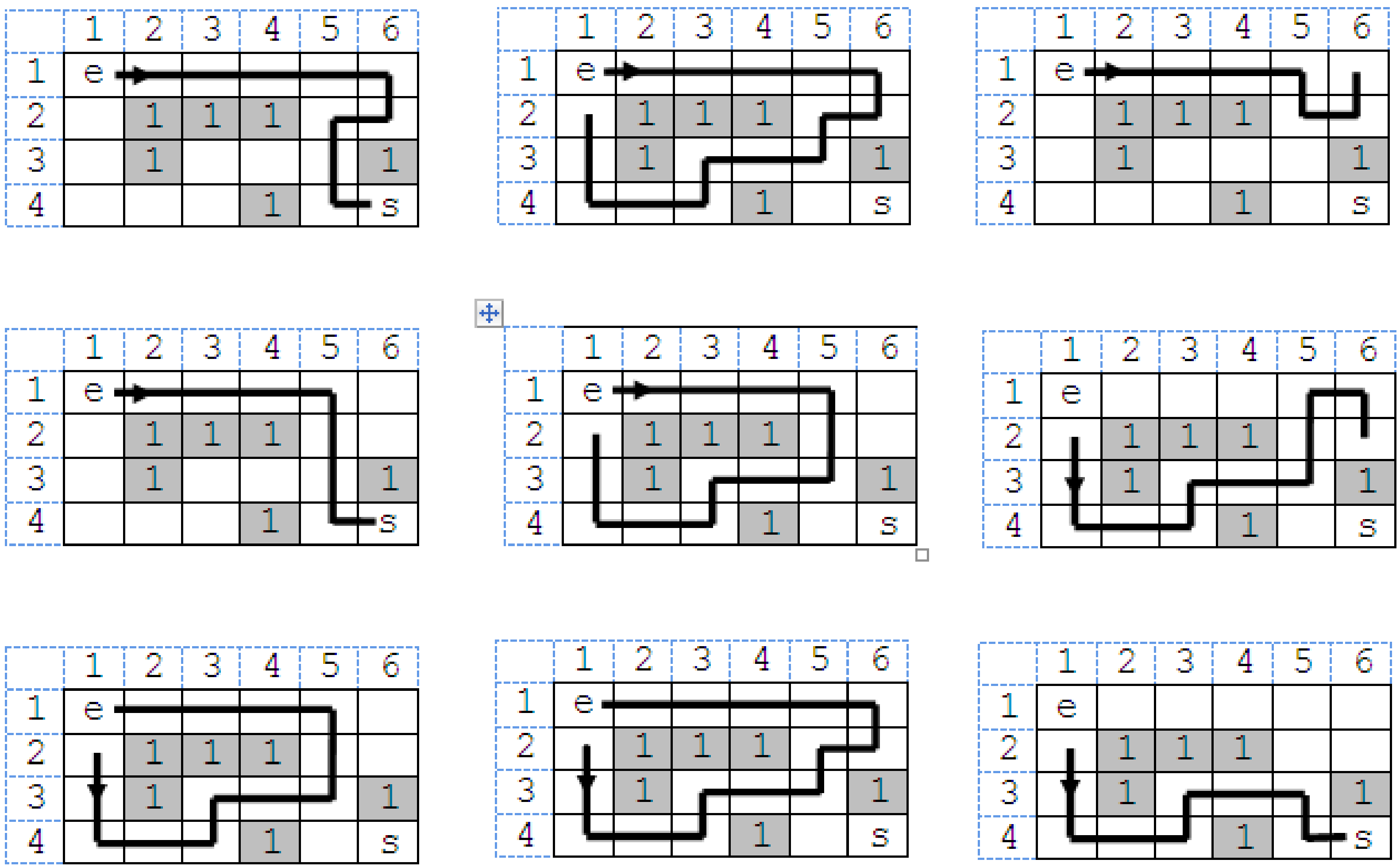
- HASONLÓSÁG
  - Mindkettő mélységében járja be a feladatok szerkezetét ábrázoló fát
- KÜLÖNBSÉGEK
  - Backtracking
    - *előremutató szakaszokon épít*, a visszamutatókon bont *(preorder)*
    - levelekben hirdet megoldásokat (*több* megoldás)
  - Oszd meg és uralkodj
    - előremutató szakaszokon bont, a *visszamutatókon épít (postorder)*
    - gyökérbe visszaérkezve hirdet megoldást (*egy* megoldás)

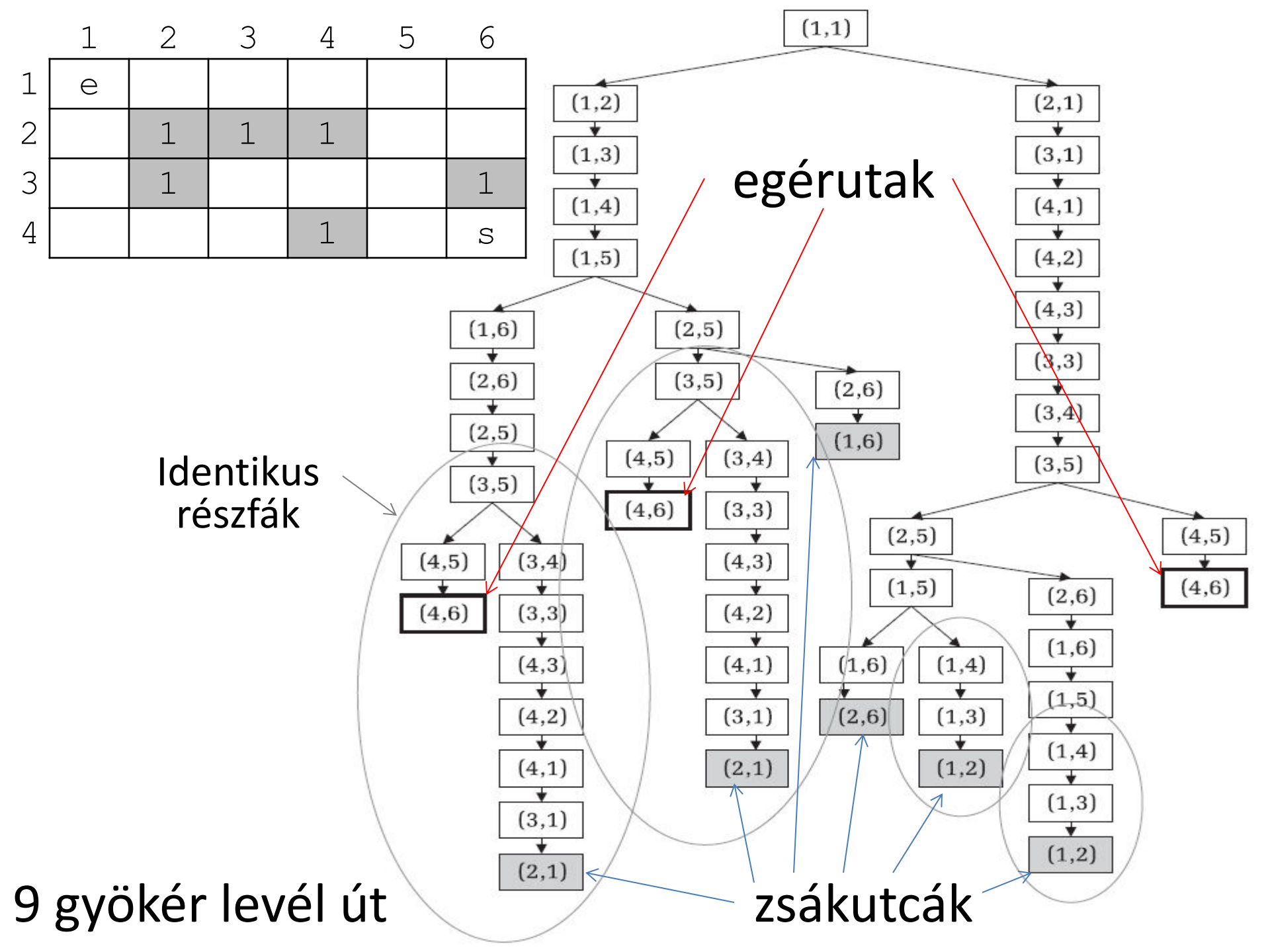
Egér-sajt **optimalizálási** probléma

- Egy labirintusban, amelyet az a[1..n][1..n] bináris mátrix ábrázol, ismert egy egér (xe, ye) és egy darab sajt (xs, ys) pozíciója. Határozzuk meg az egértől a sajthoz vezető legrövidebb utat (a labirintusban négy irányban lehet közlekedni).

	1	2	3	4	5	6
1	e					
2		1	1	1		
3		1				1
4				1		s

Lehetséges utak (9 darab)




1 2 3 4 5 6
1 e
2 1 1 1
3 1 1
4 1 s
(1,1)
(1,2)
(1,3)
(1,4)
(1,5)
(2,1)
(3,1)
(4,1)
(4,2)
(4,3)
(3,3)
(3,4)
(3,5)
(1,6)
(2,6)
(2,5)
(3,5)
(4,5)
(3,4)
(4,6)
(3,3)
(4,3)
(4,2)
(4,1)
(3,1)
(2,1)
(2,6)
(1,6)
(2,5)
(1,5)
(4,5)
(4,6)
(1,6)
(1,4)
(2,6)
(1,3)
(1,2)
(1,6)
(1,5)
(1,4)
(1,3)
(1,2)
egérutak
Identikus részfák
9 gyökér levél út
zsákutcák

backtracking_egér

- *Backtracking stratégia*
  - Generálja az összes hurokmentes utat, amely az egértől a sajthoz vezet, és kiválasztja közülük a legrövidebbet.
- Az eljárás meghívása:

```
backtracking_egér(a,ut,utmin,kmin,xs,ys,xe,ye,0)
```

```
backtracking_egér(a[][],ut[],utmin[],kmin,xs,ys,x,y,k)
    út[k].x = x
    út[k].y = y
    ha x == xs és y == ys akkor // a sajthoz érkeztünk
        ha k < kmin akkor // rövidebb utat találtunk,
            másol(útmin,kmin,út,k)
        vége ha
    különben
        a[x][y] = 1 // a hurkok elkerülése végett
        ha a[x-1][y] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x-1,y,k+1)
        vége ha
        ha a[x][y+1] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x,y+1,k+1)
        vége ha
        ha a[x+1][y] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x+1,y,k+1)
        vége ha
        ha a[x][y-1] == 0 akkor
            backtracking_egér(a,ut,utmin,kmin,xs,ys,x,y-1,k+1)
        vége ha
        a[x][y] = 0 // visszaállítjuk a szabad utat
    vége ha
vége backtracking_egér
```

Oszd_meg_es_uralkodj_egér

- *„Oszd meg és uralkodj” stratégia:*
  - Az egértől a sajthoz vezető legrövidebb út *hosszának* meghatározása visszavezethető az egérrel szomszédos szabad pozíciókból induló –sajthoz vezető– legrövidebb utak hosszának megtalálására.
  - Miután ezek rendelkezésre állnak (jelentés érkezik felőlük), a legrövidebbhez hozzáadva egyet, meg is van a keresett út hossza.
- A függvény meghívása:
  - `oszd_meg_és_uralkodj_egér(a,xs,ys,xe,ye)`

```
oszd_meg_és_uralkodj_egér(a[][],xs,ys,x,y)
    ha x == xs és y == ys akkor // a sajthoz érkeztünk
        return 0
    vége ha
    h1 = ∞
    h2 = ∞
    h3 = ∞
    h4 = ∞
    a[x][y] = 1 // a hurkok elkerülése végett
    ha a[x-1][y] == 0 akkor
        h1 = oszd_meg_és_uralkodj_egér(a,xs,ys,x-1,y)
    vége ha
    ha a[x][y+1] == 0 akkor
        h2 = oszd_meg_és_uralkodj_egér(a,xs,ys,x,y+1)
    vége ha
    ha a[x+1][y] == 0 akkor
        h3 = oszd_meg_és_uralkodj_egér(a,xs,ys,x+1,y)
    vége ha
    ha a[x][y-1] == 0 akkor
        h4 = oszd_meg_és_uralkodj_egér(a,xs,ys,x,y-1)
    vége ha
    a[x][y] = 0 // visszaállítjuk a szabad utat
    return minimum(h1,h2,h3,h4) +1
vége oszd_meg_és_uralkodj_egér
```

Legrövidebb út dúsított „oszd meg és uralkodj” algoritmussal

- Minden cellában minimumot számolunk az odaszületett egerek jelentései között
  - E minimumok tömbjében a mohó útvonal éppen a legrövidebb út lesz

	1	2	3	4	5	6
1	**8**	**7**,∞,∞	**6**,∞,∞	**5**,∞,∞	**4**,∞,∞	**5**,∞,∞,∞
2	∞,∞,**9**				**3**,3,∞	**4**,∞,∞,∞
3	∞,∞,**8**		∞,∞,**4**	∞,∞,**3**	**2**,2,2	
4	∞,∞,**7**	∞,∞,**6**	∞,∞,**5**		**1**,1,1	**0**,0,0