Algoritmusok felülnézetből

Sapientia-EMTE

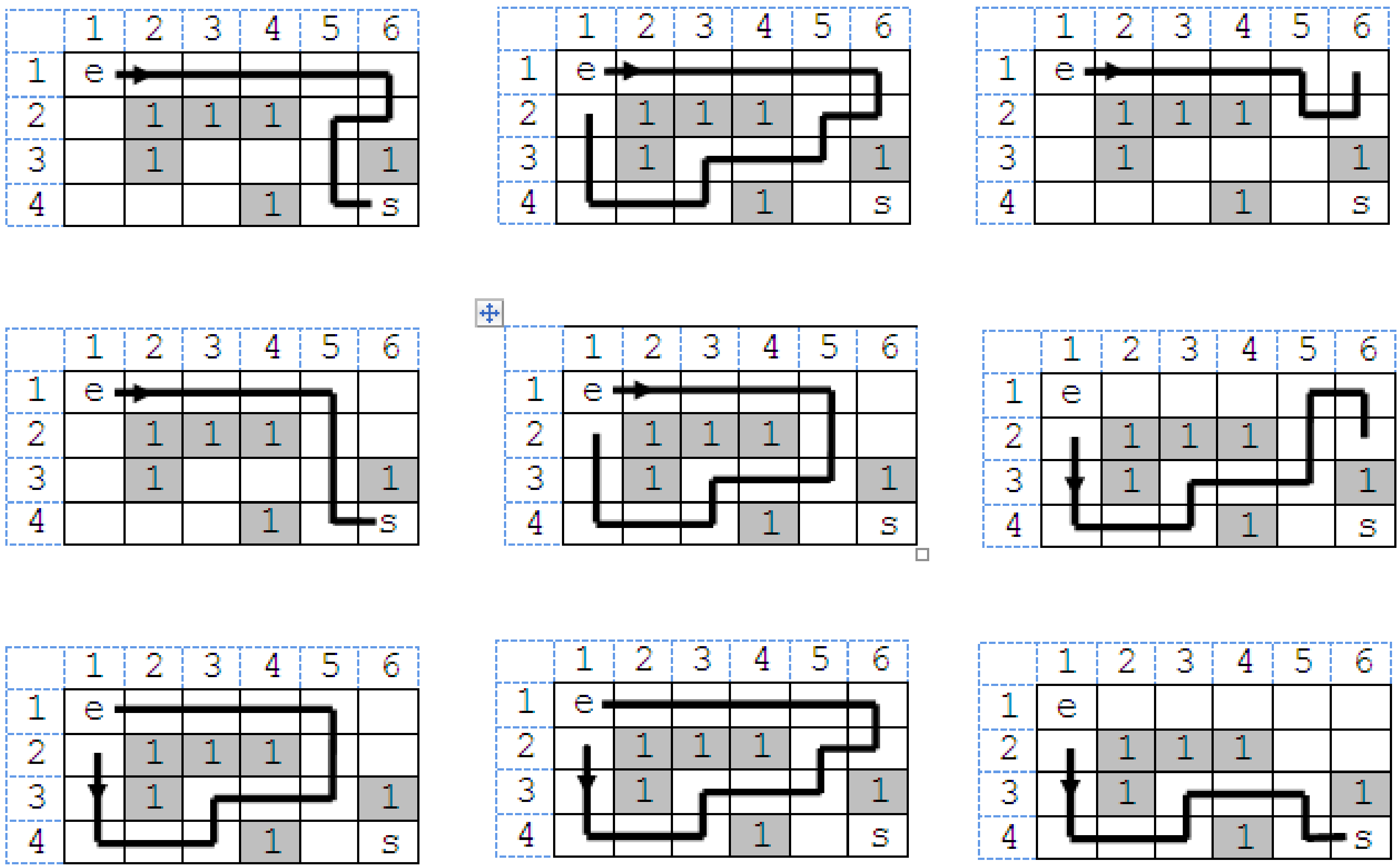
2019-20

Egér-sajt **optimalizálási** probléma

- Egy labirintusban, amelyet az a[1..n][1..m] 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).

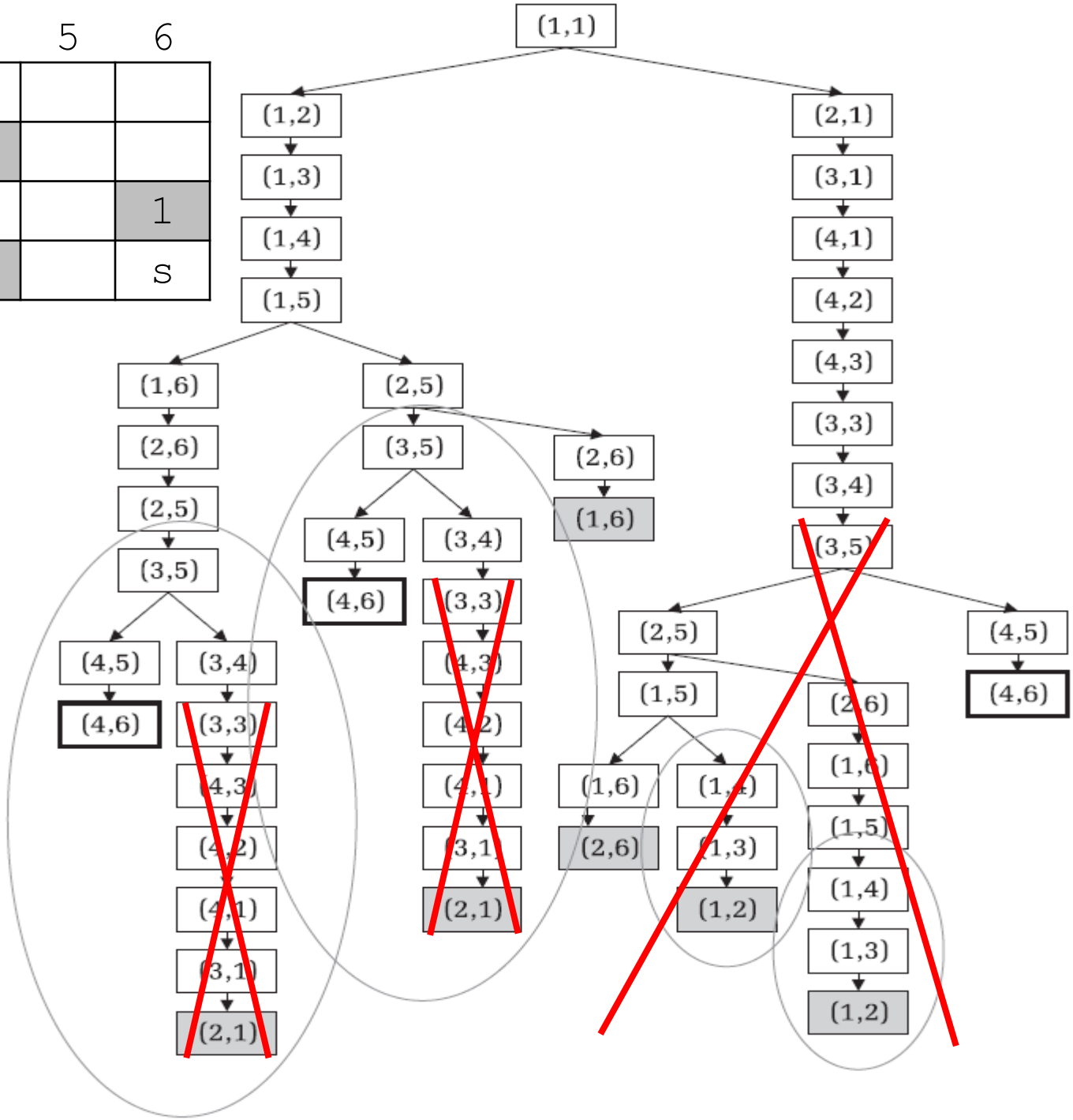
	1	2	3	4	5	6
1	e					
2		1	1	1		
3		1				1
4				1		s

Lehetséges utak (9 darab)



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

DF branch and bound



- Mivel a legrövidebb utak részútjai is legrövidebb utak, ezért minden cellára nyilvántarthatunk egy kurrens optimumot;

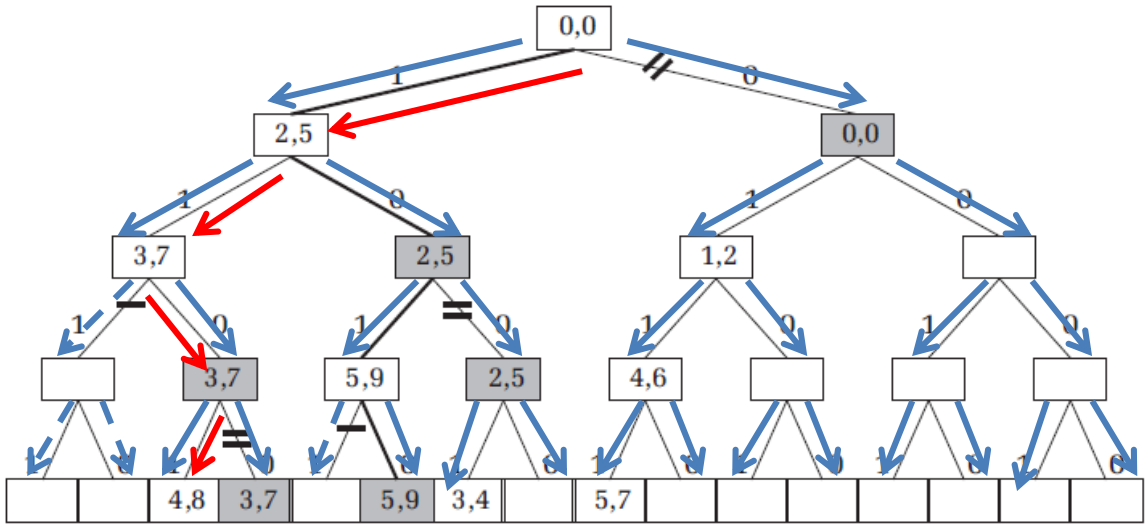
	1	2	3	4	5	6
1	**0**	**1**	**2**	**3**	**4**	**5,**
2	15,13,**1**				7,**5**	**6,**
3	14,12,**2**		10,8,**6**	9,7,**7**	8,**6**	
4	13,11,**3**	12,10,**4**	11,9,**5**		9,**7**	10,**8**

BF - branch and bound

	1	2	3	4	5	6
1	**0**	**1**	**2**	**3**	**4**	**5**
2	**1**				**5**	**6**
3	**2**		**6**	**7**	**6**	
4	**3**	**4**	**5**		**7**	**8**

Backtracking és Greedy házasítás BRANCH and BOUND módra

- Hasonlóságok
  - *mélységükben* viszonyulnak a feladatok szerkezetét ábrázoló fákhoz
  - gyökér-levél irányba építik az optimális megoldás-utat
- Különbségek
  - Backtracking : „teljes” fa / Mohó: egyetlen gyökér levél út

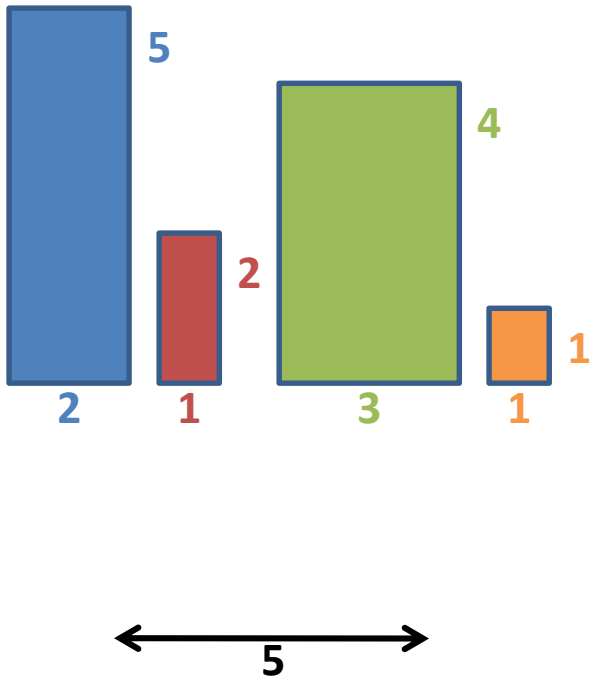


BACKTRACKING és GREEDY kéz a kézben

- Miként növelhető a **backtracking** és **greedy** stratégiák hatékonysága **Branch and bound** szellemben való kombinálásuk révén?

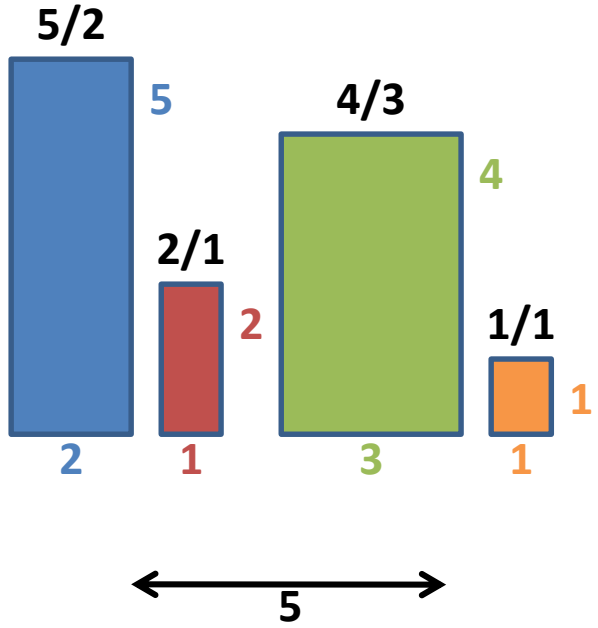
Hátizsák – probléma

- Egy üzletben n tárgy (áru) található, amelyeknek ismert az áruk és a súlyuk. Állapítsuk meg, hogy mely tárgyakat fogja magával vinni egy tolvaj ahhoz, hogy a lehető legnagyobb nyereséggel távozzon (a hátizsákja legtöbb G súlyt bír meg).
  a) a tárgyak elvághatók (folytonos)
  b) a tárgyak nem vághatók el (diszkrét)



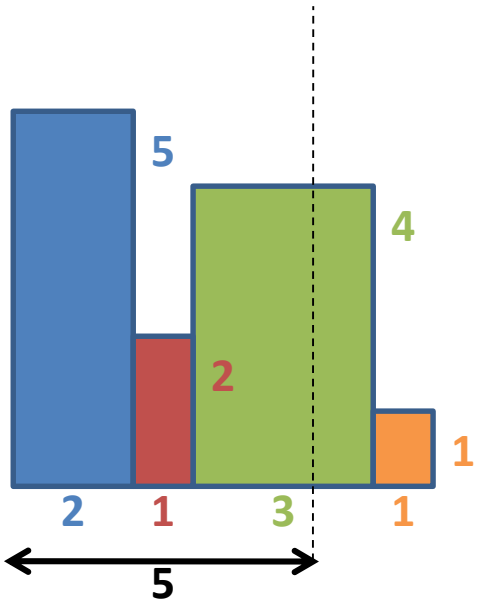
Folytonos változat (mohó-probléma)

- A tárgyakat az ár/súly arány szerinti csökkenő sorrendben próbáljuk betenni a hátizsákba.
- Az első áruból, amelyik nem fér egészében a hátizsákba, levágunk annyit, hogy azzal teljesen megteljen.
- Példa:
  - Bemenet: n = 4, G = 5,
    t[1..4].g = {2, 1, 3, 1},
    t[1..4].a = {5, 2, 4, 1}



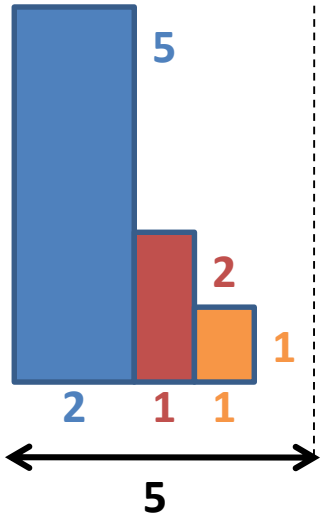
Folytonos változat (mohó-probléma)

- Mohó-sorrend: **1**(5/2) > **2**(2/1) > **3**(4/3) > **4**(1/1)
  - Kimenet: (1, 1, 2/3, 0)
    - 2 + 1 + 3*(2/3) = 5
    - 5 + 2 + 4*(2/3) = **29/3** nyereség

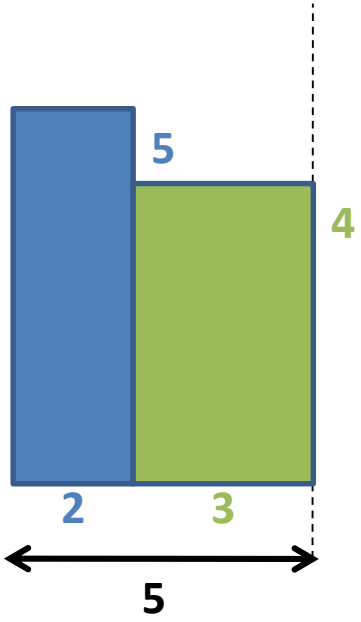


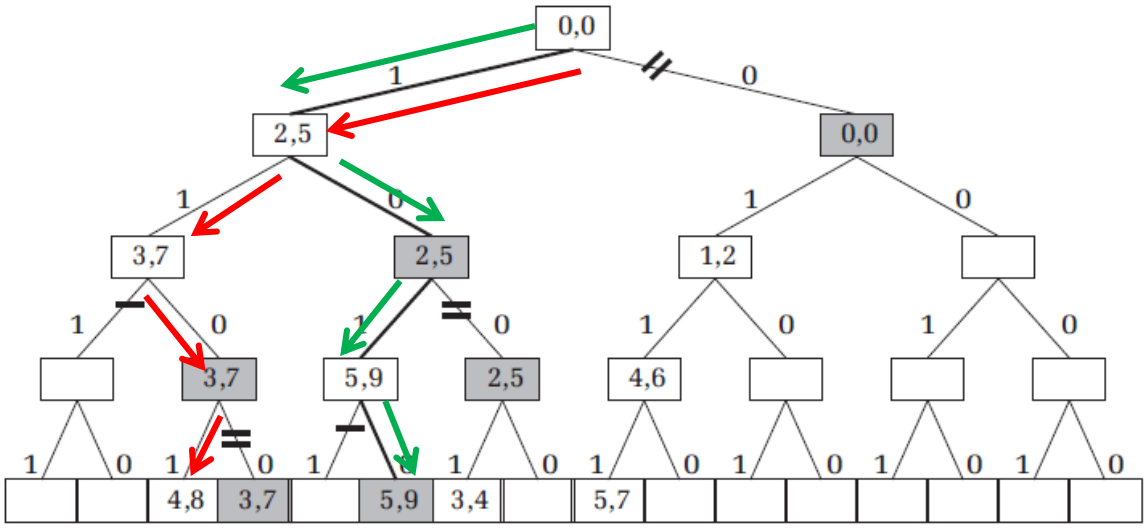
Diszkrét változat

- Mohó stratégia
  - Nyereség: **8**



- Optimális megoldás
  - Nyereség: **9**

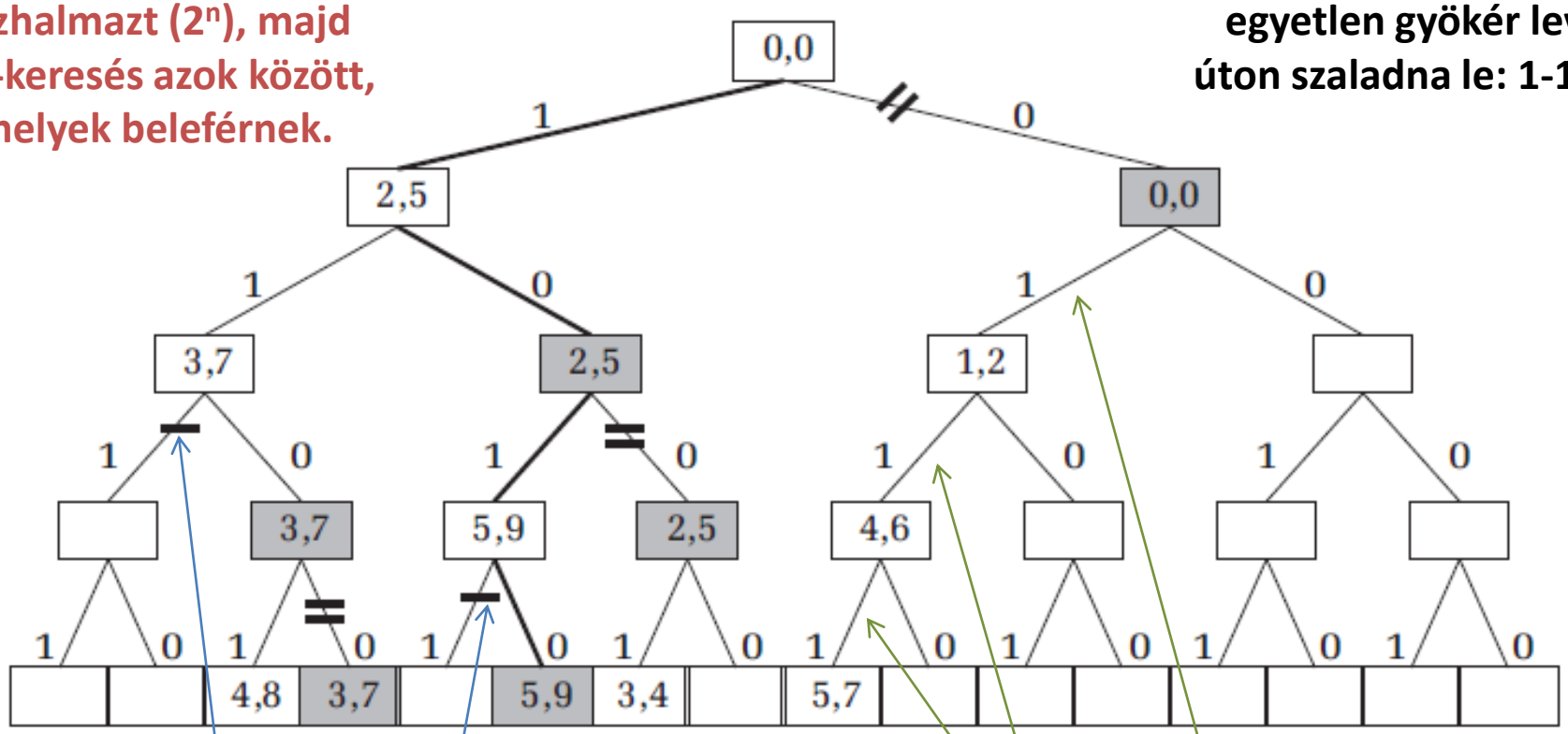




Diszkrét változat (Backtracking)

Generáljuk az összes részhalmazt (2^n), majd max-keresés azok között, amelyek belefférnek.

A mohó stratégia egyetlen gyökér levél úton szaladna le: 1-1-0-1



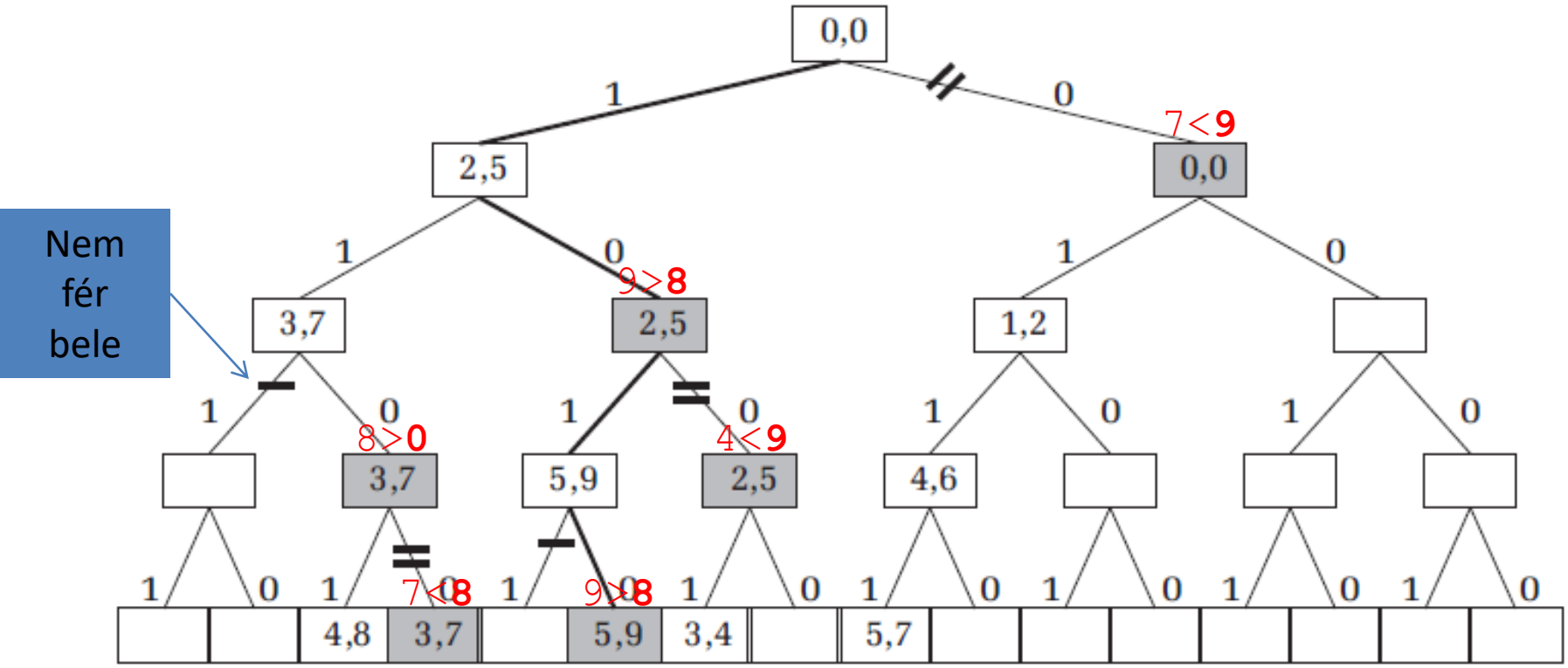
Csak azokat a részhalmazokat generáljuk, amelyek belefférnek (szimpla olló).

Ha az összes többi tárgy belefér, akkor gondolkodás nélkül beletesszük őket.

BACKTRACKING és GREEDY kéz a kézben

- Hogyan lehetne még hatékonyabban optimalizálni a backtracking algoritmust a mohó stratégia segítségével?
  - Foglalkozzon a backtracking is mohó sorrendben a tárgyakkal
    - Elsőnek éppen a mohó megoldást találja meg
  - Mielőtt bejárnánk valamely csomópont jobb fiúrészfáját, először megvizsgáljuk, hogy van-e esély rá, hogy a megoldáslevél ott legyen
    - Mohó folytonos algoritmussal folytatjuk, vizsgálat céljából
    - Ha ezen illegális mohó algoritmus sem talál jobb megoldást, akkor nincs itt az optimális levél (**dupla olló**)

BACKTRACKING és GREEDY kéz a kézben



A szürke pontokra van lefuttatva az illegális greedy.
Amelyek nem kerültek dupla ollóval levágásra, azok részfái potenciálisan tartalmazhatják a megoldás levelet.