

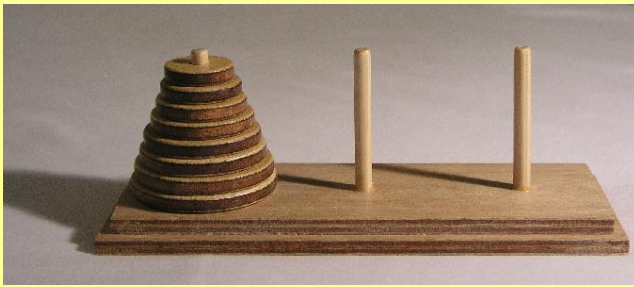
Modellezés

1. Állapottér modell

- ❑ **Állapottér**: a probléma leírásához szükséges adatok által felvett érték-együttesek (azaz **állapotok**) halmaza
 - az állapot többnyire egy **összetett szerkezetű** érték
 - gyakran egy bővebb alaphalmazzal és egy azon értelmezett **invariáns állítással** definiáljuk
- ❑ **Műveletek**: állapotból állapotba vezetnek
 - megadásukhoz: **előfeltétel** és **hatás** leírása
 - invariáns tulajdonságot tartó leképezés
- ❑ **Kezdőállapot(ok)** vagy azokat leíró kezdeti feltétel
- ❑ **Célállapot(ok)** vagy célfeltétel

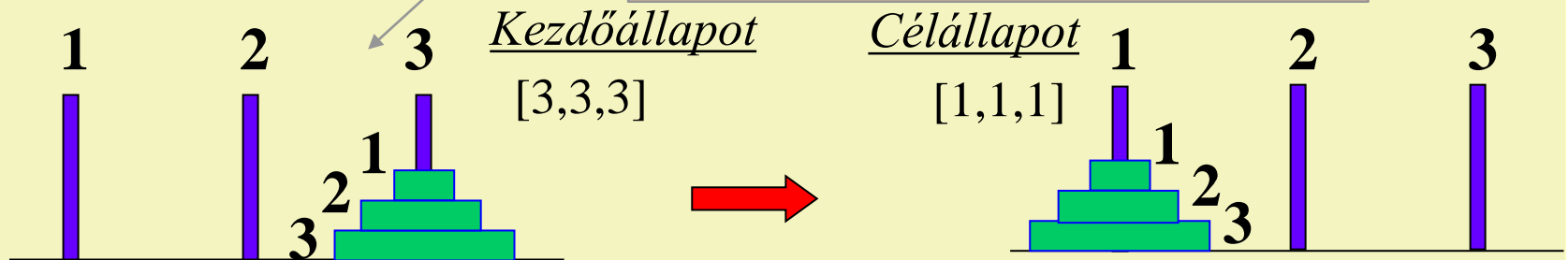
Állapottér modell gráfrepresentációja

- | | | | |
|---|--|---|--|
| □ | Állapottér modell | | Állapot-gráf |
| ○ | állapot | ~ | csúcs |
| ○ | művelet hatása egy állapotra | ~ | irányított él |
| • | egy állapotra véges sok művelet alkalmazható (véges kifokú gráf) | | |
| ○ | művelet költsége | ~ | él költsége |
| • | van a műveltek költségének alsó pozitív korlátja (δ) | | |
| ○ | kezdő állapot | ~ | startcsúcs |
| ○ | célállapot | ~ | célcsúcs |
| □ | Gráf-reprezentáció: állapot-gráf, startcsúcs, célcsúcsok | | |
| ○ | egy műveletsorozat hatása | ~ | irányított út |
| ○ | megoldás | ~ | irányított út a startcsúcsból egy célcsúcsba |



Hanoi tornyai probléma

állapot ~ korongok egy elhelyezkedése



Állapottér: $AT = \{1,2,3\}^n$

$1..n$ intervallummal indexelt egydimenziós tömb, amely elemei az $\{1,2,3\}$ halmazból származnak.

megjegyzés : a tömb i -dik eleme mutatja az i -dik korong rúdjának számát; a korongok a rudakon méretük szerint fentről lefelé növekvő sorban vannak.

Művelet: $Rak(honnan, hova): AT \rightarrow AT$

HA a *honnan* és *hova* létezik és nem azonos, és van korong a *honnan* rúdon, és a *hova* rúd legyen vagy üres vagy felső korongja nagyobb, mint mozgatandó korong (*honnan* rúd felső korongja)

AKKOR $this[honnan \text{ legfelső korongja}] := hova$

$this:AT$ az aktuális állapot

Implementáció

```
template <int n = 3>
class Hanoi {
    int _a[n];          // its elements are between 1 and 3
public:
    bool move (int from, int to) {
        if ((from<1 || from>3 || to<1 || to>3) || (from==to)) return false;
        bool l1; int i;  // l1 ~ 'from' is not empty, i ~ upper disc on 'from'
        for(l1=false, i=0; !l1 && i<n; ++i) l1 = (_a[i]==from);
        if (!l1) return false;
        bool l2; int j;  // l2 ~ 'to' is not empty, j ~ upper disc on 'to'
        for(l2=false, j=0; !l2 && j<n; ++j) l2 = (_a[j]==to);
        if (¬l2 || i<j) { _a[i] = to; return true; } else return false;
    }
    bool final() const { bool l=true; for(int i=0; l && i<n; ++i) l = (_a[i]==1); return l; }
    void init() { for(int i=0; i<n; ++i) _a[i] = 3; }
};
```

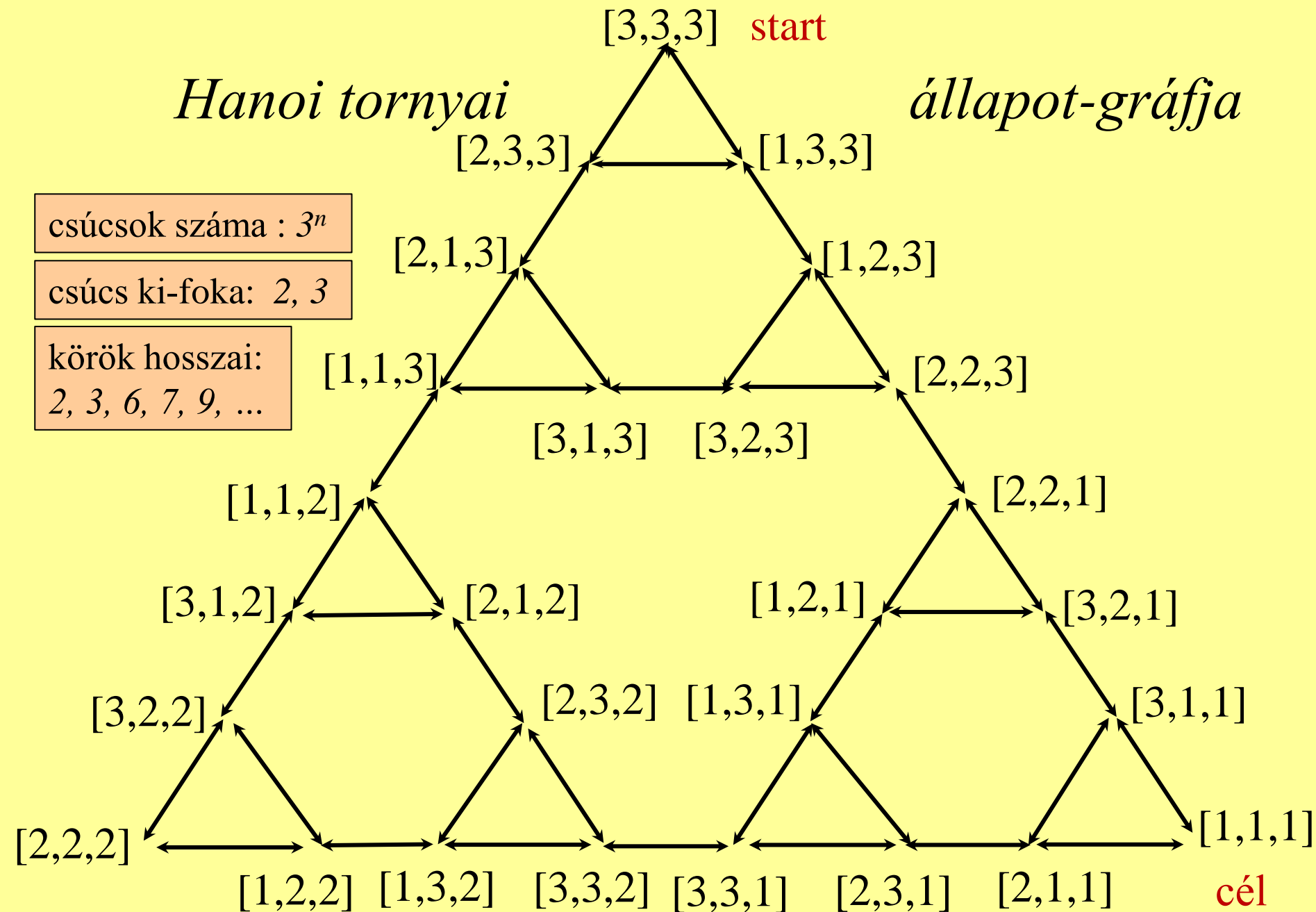
Hanoi tornyai

állapot-gráfja

csúcsok száma : 3^n

csúcs ki-foka: 2, 3

körök hosszai:
2, 3, 6, 7, 9, ...

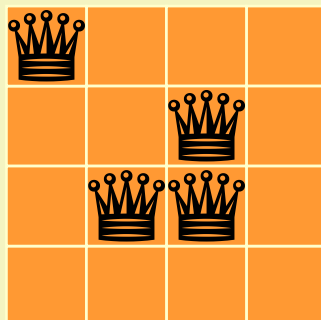




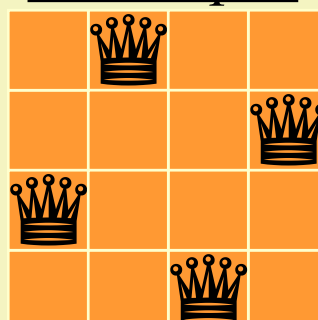
n -királynő probléma 1.

állapot ~ királynők egy elrendezése

általános állapot



Célállapot



Nem ismert,
egy feltétel
ellenőrzésével
dönthető el

Állapottér: $AT = \{ \text{👑}, _ \}^{n \times n}$

kétdimenziós tömb ($n \times n$ -es mátrix),
mely elemei $\{ \text{👑}, _ \}$ halmazbeliek

invariáns: egy állapot (tábla) pontosan n darab királynőt tartalmaz

Művelet: $\text{Áthelyez}(x, y, u, v): AT \rightarrow AT$

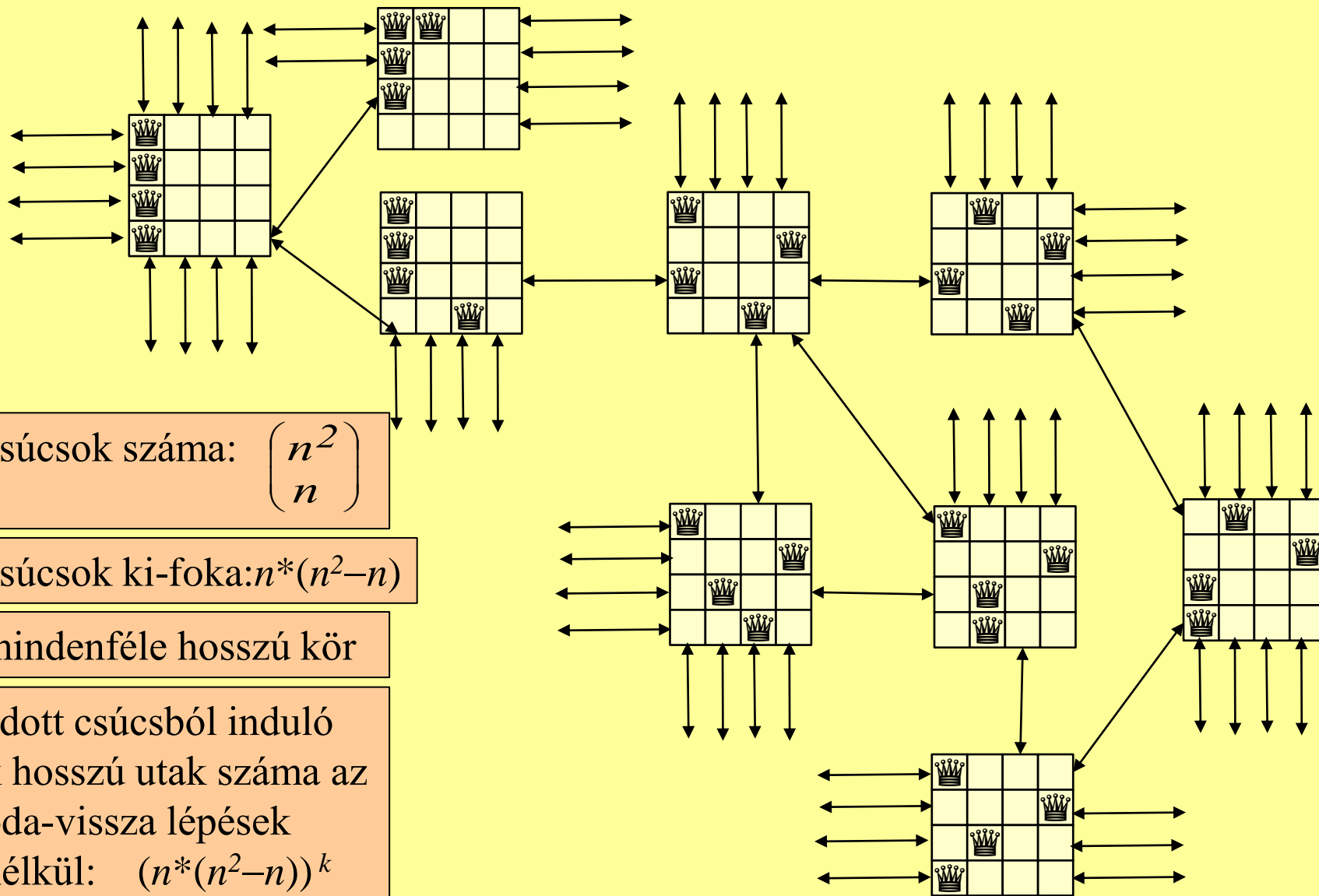
(*this:AT*)

HA $1 \leq x, y, u, v \leq n$ és $\text{this}[x, y] = \text{👑}$ és $\text{this}[u, v] = _$

AKKOR $\text{this}[x, y] \leftrightarrow \text{this}[u, v]$

csere

Állapot-gráf részlet



csúcsok száma: $\binom{n^2}{n}$

csúcsok ki-foka: $n \cdot (n^2 - n)$

mindenféle hosszú kör

adott csúcsból induló
k hosszú utak száma az
oda-vissza lépések
nélkül: $(n \cdot (n^2 - n))^k$

Gregorics Tibor

Mesterséges intelligencia

Állapottér vs. problématér

- Az állapottér és a problématér **kapcsolata szoros**, de ezek **nem azonosak**, hiszen a problématér elemei (a lehetséges válaszok) többnyire a kezdőállapotból kiinduló művelet-sorozatok (utak).
 - A Hanoi tornyai problémára adott lehetséges válasz nem a korongok egy állapota (elrendezése), hanem a kezdő állapotra alkalmazott művelet-sorozat. Ezek között keressük azt (a megoldást), amelyik a célállapothoz vezet.
 - Az n -királynő problémánál igaz ugyan, hogy a problémára adható válasz egy állapot (királynő-elrendezés), de a célállapotot (helyes elrendezést) ebben az esetben is egy alkalmas művelet-sorozattal érhetjük el, azaz végsősoron ilyenkor is a művelet-sorozatok között keresünk, nem az állapotok között.

Reprezentációs gráf bonyolultsága

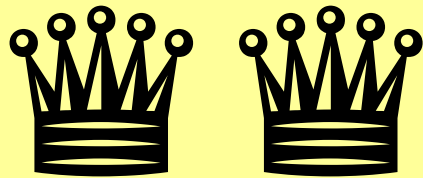
- A reprezentációs gráf bonyolultságától (a startcsúcsból induló utak számától) függ a problémater mérete, amelyen pedig a keresés hatékonysága múlik.



- A bonyolultság a **start csúcsból kivezető utak számától** függ, amely nyilván függvénye a
 - csúcsok és élek számának
 - csúcsok ki-fokának
 - **körök** gyakoriságának, és hosszuk sokféleségének

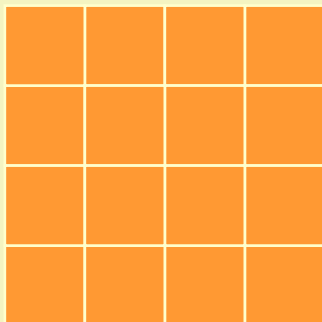
Csökkentsük a problémátér méretét

- Ugyanannak a feladatnak több modellje lehet : érdemes olyat keresni, amely kisebb problémateret jelöl ki.
 - Az n -királynő probléma modelljénél a problémátér mérete (a lehetséges utak száma) óriási. Adjunk jobb modellt!
 - Bővítsük az állapotteret az n -nél kevesebb királynőt tartalmazó állásokkal, és használjunk új műveletet : királynő-felhelyezést (kezdő állás az üres tábla).
 - Műveletek előfeltételének szigorításával csökken az állapotgráf átlagos ki-foka:
 - Sorról sorra haladva csak egy-egy királynőt helyezzünk fel a táblára!
 - Ütést tartalmazó állásra ne tegyünk királynőt!

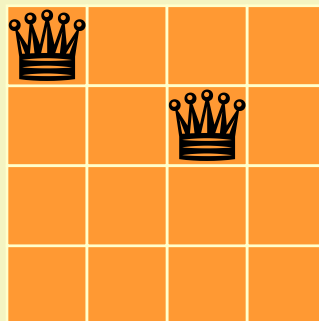


n-királynő probléma 2.

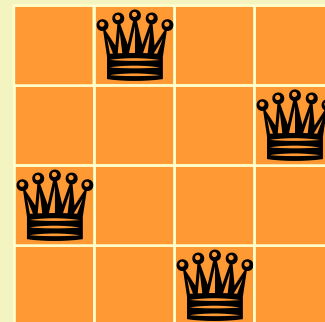
Kezdőállapot:



Közbülső állapot:



Célállapot :



Állapottér: $AT = \{ \text{👑}, _ \}^{n \times n}$

nincs már üres sor és nincs ütés

invariáns: az első néhány sor egy-egy királynőt tartalmaz

Művelet: $\text{Helyez}(\text{oszlop}): AT \rightarrow AT$ (*this*:*AT*)

HA $1 \leq \text{oszlop} \leq n$ és a *this*-beli soron következő üres sor $\leq n$
és nincs ütés a *this*-ben

AKKOR $\text{this}[\text{a *this*-beli soron következő üres sor}, \text{oszlop}] := \text{👑}$

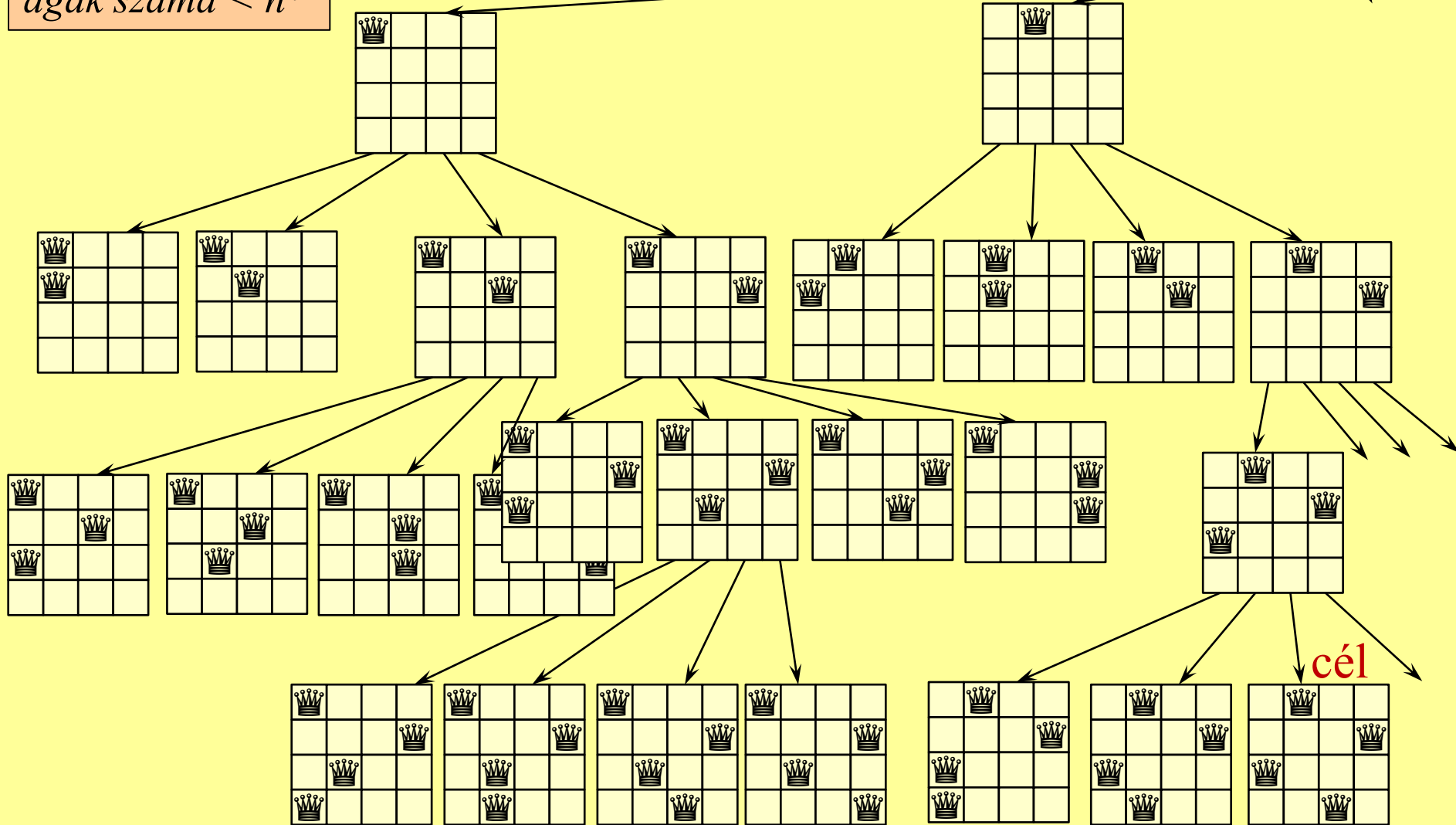
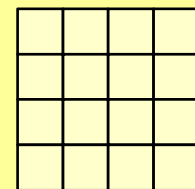
$csúcsok\ száma < (n^{n+1} - 1) / (n - 1)$

$csúcs\ ki-foka: n$

$ágak\ száma < n^n$

Állapot-gráf

start

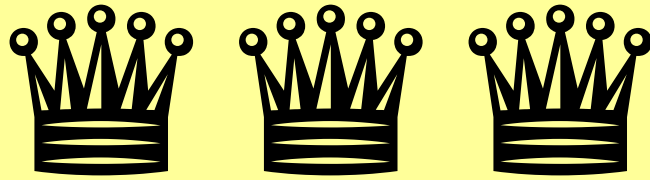


Gregorics Tibor

Mesterséges intelligencia

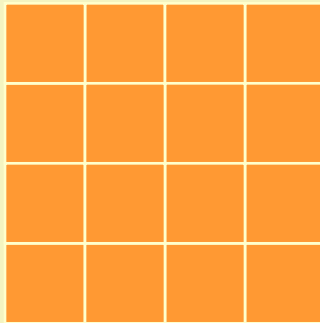
Művelet végrehajtásának hatékonysága

- ❑ A művelet kiszámítási bonyolultsága csökkenthető, ha
 - az állapotokat extra információval egészítjük ki:
egy állapotban a tábla soron következő üres sorának sorszámát eltárolhatjuk a tábla mellett, így új királynő elhelyezésekor ezt nem kell kiszámolni, ugyanakkor egy művelet végrehajtásakor könnyen aktualizálhatjuk (eggyel növeljük).
 - az invariáns szigorításával szűkítjük az állapoteret:
ne engedjük meg ütést létrehozni a táblán, hogy ne kelljen ezt a tulajdonságot külön ellenőrizni. Ennek céljából megjelöljük az ütés alatt álló üres (tehát már nem szabad) mezőket, amelyekre nem helyezhetünk fel királynőt. Egy mező státusza így három féle lehet: szabad, ütés alatt álló vagy foglalt, amelyeket a művelet végrehajtásakor kell karbantartani.



n-királynő probléma 3.

Kezdőállapot:



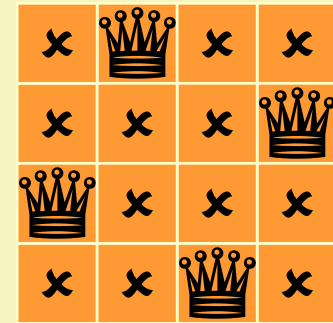
köv_sor = 1

Közbülső állapot:



köv_sor = 3

Célállapot :



köv_sor = 5

Állapottér: $AT = rec(t : \{ \text{queen}, \times, _ \}^{n \times n}, köv_sor : \mathbb{N})$

invariáns:

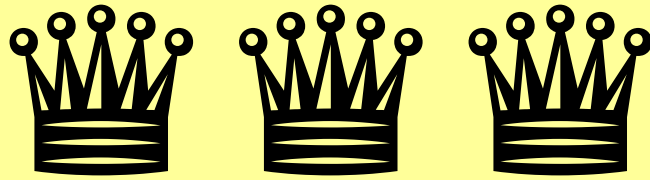
$köv_sor \leq n+1,$

az első $köv_sor-1$ darab sor egy-egy királynőt tartalmaz,
királynők nem ütik egymást,

jelölés :

\times egy királynő által ütött üres mezőt jelöli,

$_$ az ütésben nem álló (szabad) üres mezőt jelöli.



n-királynő probléma 3. folytatás

Művelet: új királynő elhelyezése a soron következő sorba

Helyez(*oszlop*): $AT \rightarrow AT$ (*this:AT*)

HA $1 \leq \text{oszlop} \leq n$ és $\text{this.köv_sor} \leq n$
és $\text{this.t}[\text{this.köv_sor}, \text{oszlop}] = _$

AKKOR

$\text{this.t}[\text{this.köv_sor}, \text{oszlop}] := \text{👑}$

$\forall i \in [\text{this.kövsor} + 1 .. n] :$

$\text{this.t}[i, \text{oszlop}] := \times$

ha $(i \leq n + \text{this.köv_sor} - \text{oszlop})$ akkor $\text{this.t}[i, i - \text{this.köv_sor} + \text{oszlop}] := \times$

ha $(i \leq \text{this.köv_sor} + \text{oszlop} - 1)$ akkor $\text{this.t}[i, \text{this.köv_sor} + \text{oszlop} - i] := \times$

$\text{this.köv_sor} := \text{this.köv_sor} + 1$

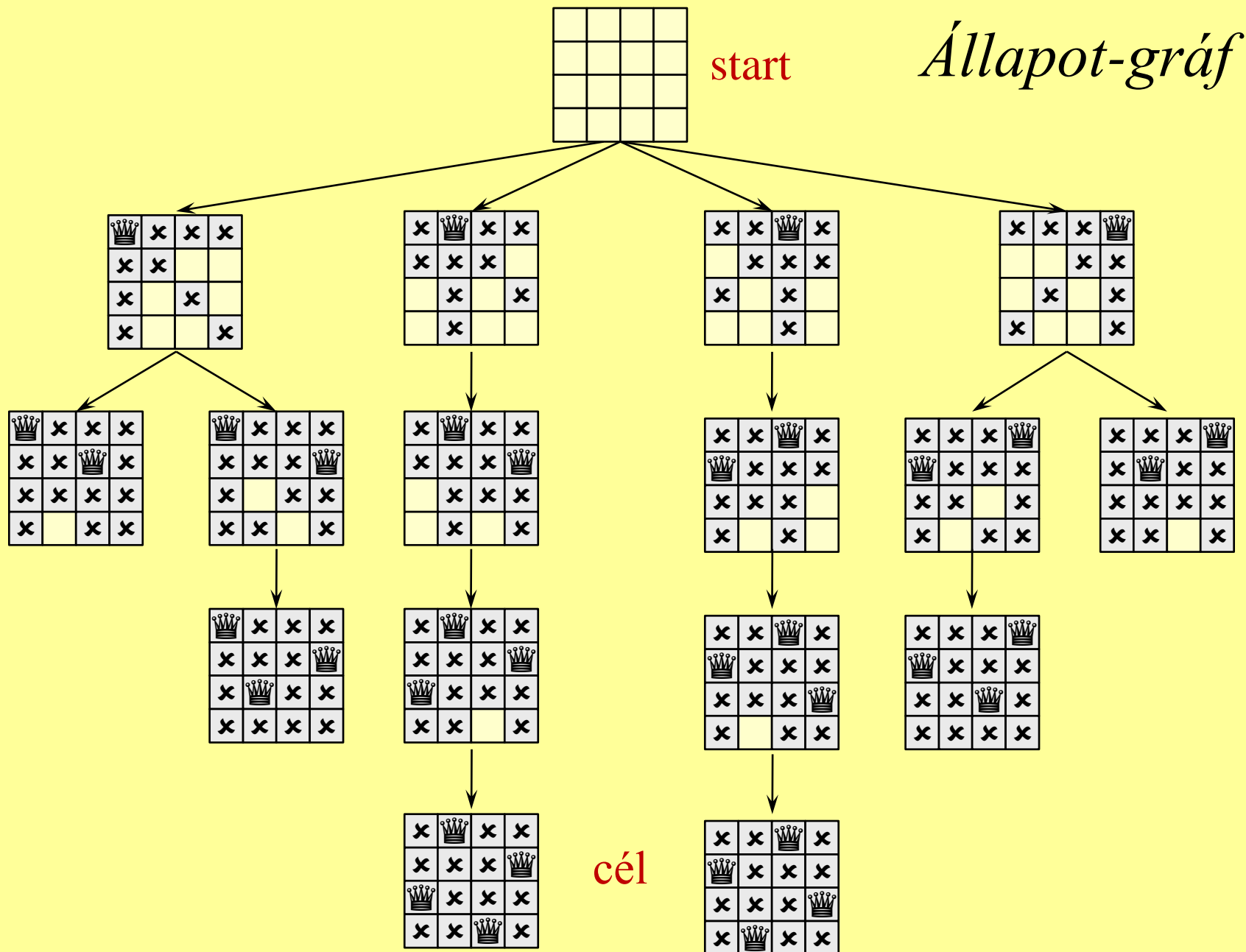
Kezdőállapot: this.t egy üres mátrix, $\text{this.köv_sor} := 1$

Célállapot: $\text{this.köv_sor} > n$

előfeltétel számítás-igénye: konstans

hatás számítás-igénye: lineáris

célfeltétel nagyon egyszerű lett

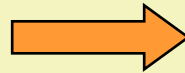


Tologató játék (8-as, 15-ös)

állapot ~ a kirakó egy konfigurációja

kezdőállapot:
tetszőleges

2	8	3
1	6	4
7		5



1	2	3
8		4
7	6	5

célállapot:
szokásos

Állapottér: $AT = \text{rec}(\text{mátrix} : \{0..8\}^{3 \times 3}, \text{üres} : \{1..3\} \times \{1..3\})$

invariáns: egy állapot mátrixának sorfolytonos kiterítése a 0 .. 8 számok egy permutációja, az *üres* hely a 0 elem mátrixbeli sor és oszlopindexe.

Művelet: $\text{Tol}(\text{irány}) : AT \rightarrow AT$

koordinátánkénti összeadás

HA

$\text{irány} \in \{(0,-1), (-1,0), (0,1), (1,0)\}$ és

$(1,1) \leq \text{this.üres} + \text{irány} \leq (3,3)$

(*this*: AT)

AKKOR

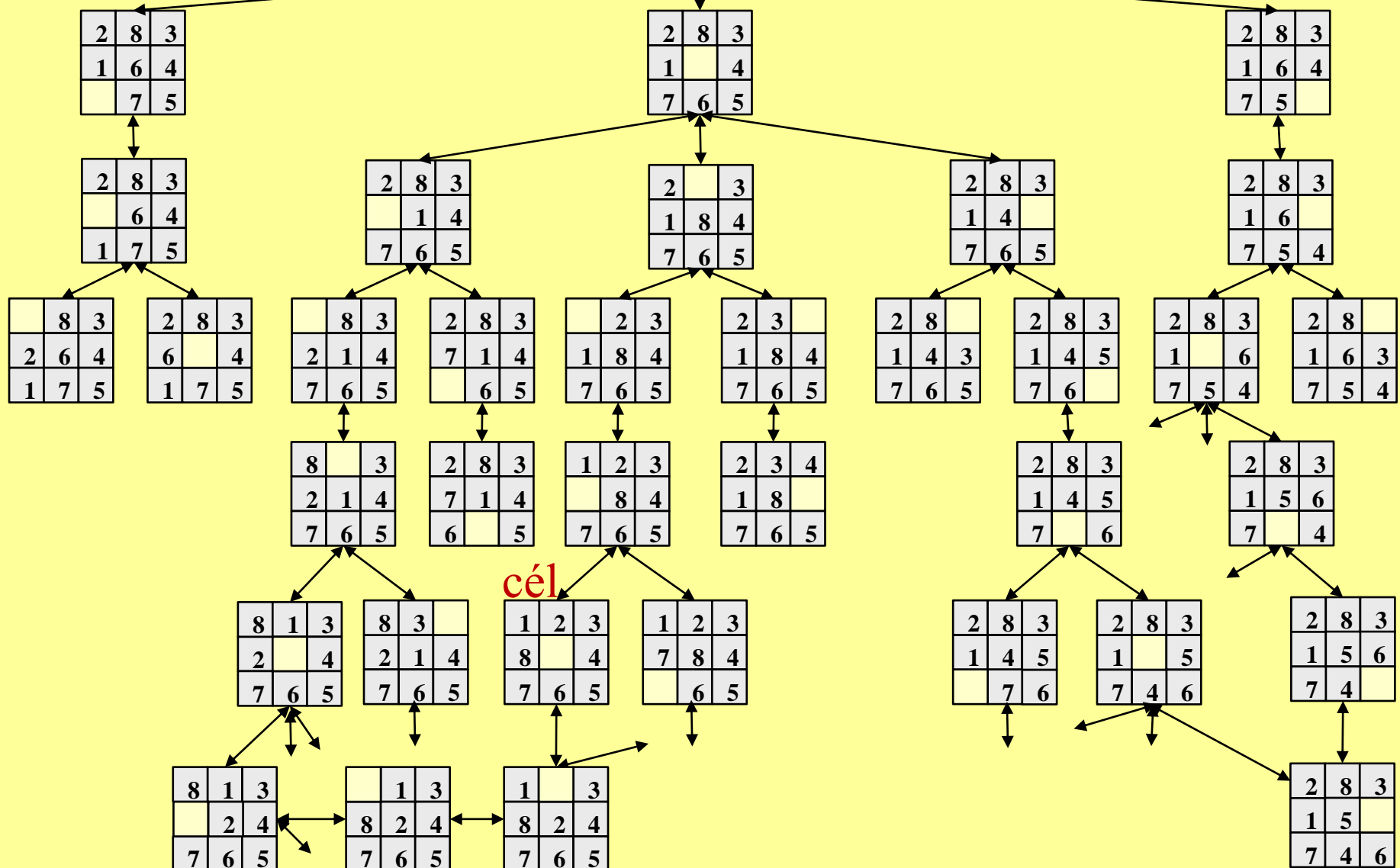
$\text{this.mátrix}[\text{this.üres}] \leftrightarrow \text{this.mátrix}[\text{this.üres} + \text{irány}]$

$\text{this.üres} := \text{this.üres} + \text{irány}$

start

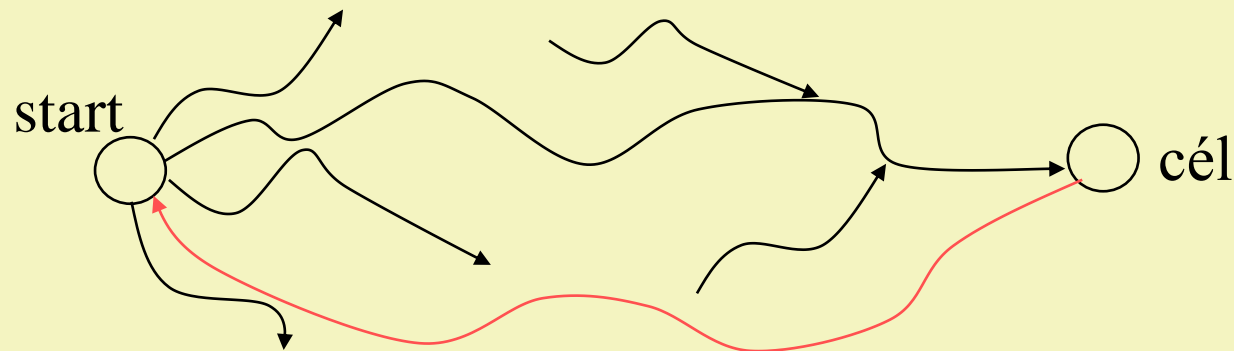
2	8	3
1	6	4
7		5

Állapot-gráf



2. Visszafelé haladó keresés

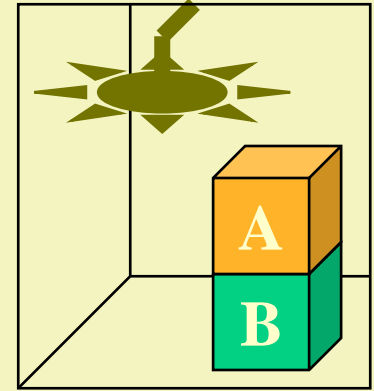
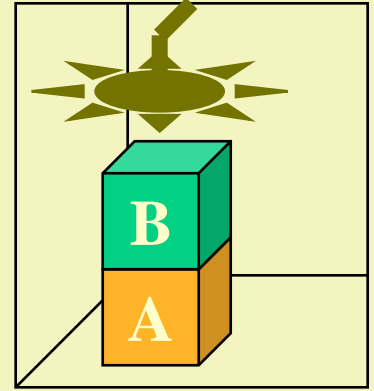
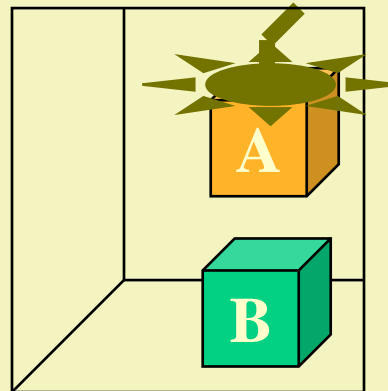
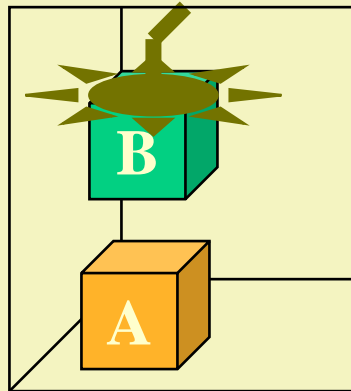
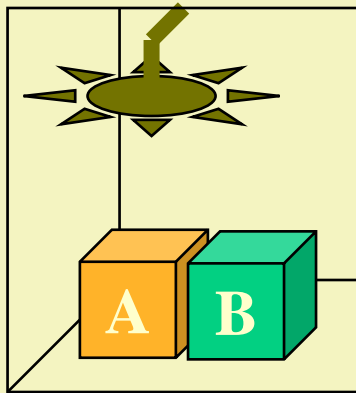
- Ha a megoldás megtalálása a cél felől nézve egyszerűbb, mint a start felől nézve (mert a problémátér így kevesebb alternatívát mutat), akkor érdemes visszafelé, a cél irányából a start felé haladva keresni a megoldást.



- Ügyelni kell azonban arra, hogy az így megtalált megoldási utat fordítva, a starttól a cél irányában kell tudni értelmezni.

Kocka világ probléma

start



cél

út a célból a startba:
megoldás:

levesz(A,B), letesz(A)
felvesz(A), rátesz(A,B)

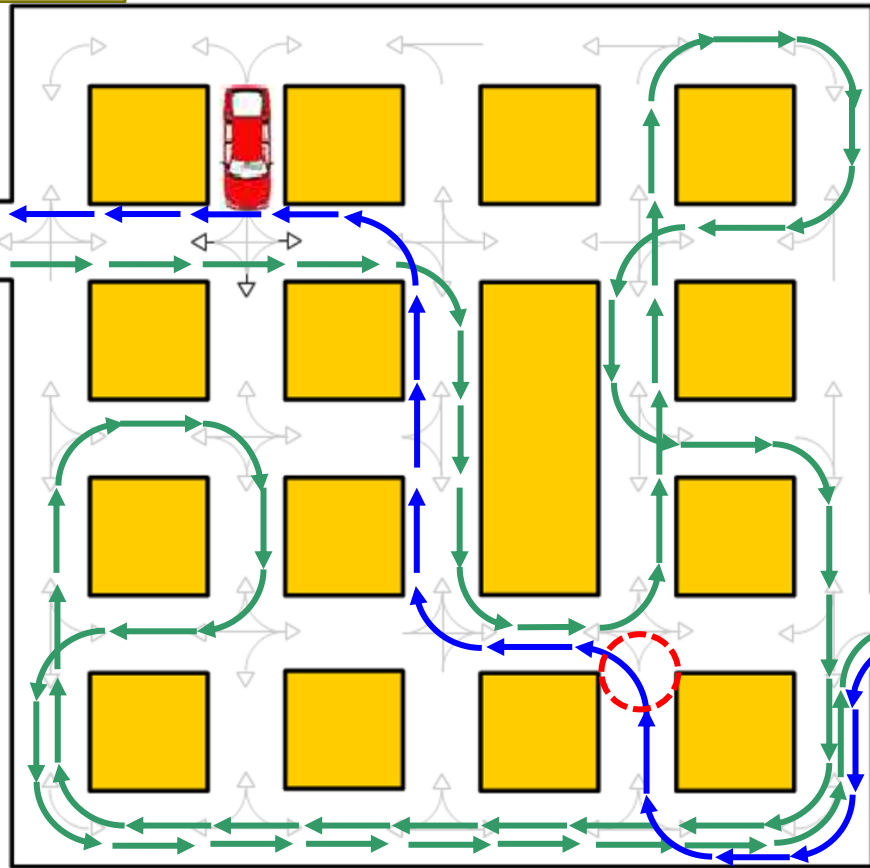
	A cé ton	A sta neg nert ell

A cél felől indított kereséssel előállt (kék) úton nem lehet a start felől végig menni.

start

A start felől indulva nehéz megtalálni a (zöld) utat, mert sok alternatíva közül kell azt kiválasztani.

cél





Kancsók problémája

Három kancsóban együttesen 5 liter bor van. Kezdetben az öt literes van tele, a 3 és 2 literes üres. Töltögetéssel érjük el, hogy a 2 literesbe pontosan 1 liter bor kerüljön!

Állapottér: $AT = \text{map}(\text{key:Keys}, \text{value}:\mathbb{N})$ ahol $\text{Keys} = \{5, 3, 2\}$
invariáns: $\sum_{i \in \text{Keys}} \text{this}[i] = 5$ és $\forall i \in \text{Keys} : \text{this}[i] \leq i$

Kezdőállapot: $[5, 0, 0]$

Célállapot: $[x, y, 1]$

this[5]=5, this[3]=0, this[2]=0

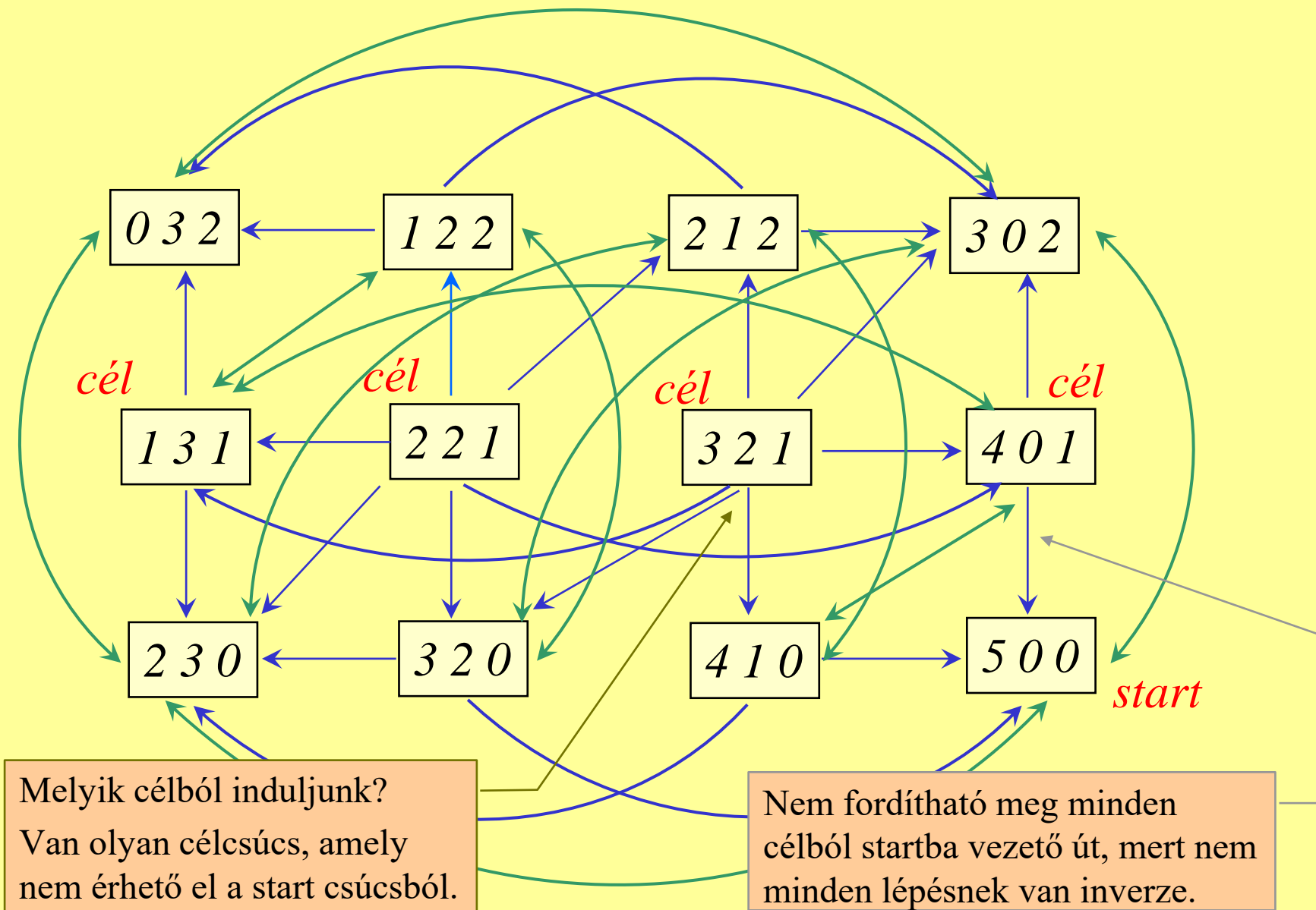
this[2]=1

Művelet: $\text{Tölt}(i, j): AT \rightarrow AT$ ($\text{this} : AT$)

HA $i, j \in \text{Keys}$ és $i \neq j$ és $\min(\text{this}[i], j - \text{this}[j]) > 0$

AKKOR $\text{this}[i], \text{this}[j] := \text{this}[i] - \min(\text{this}[i], j - \text{this}[j]),$
 $\text{this}[j] + \min(\text{this}[i], j - \text{this}[j])$

Kancsók-probléma állapot-gráfja



Visszafelé haladó keresés feltételei

- ❑ A reprezentációs gráf **éleinek kétirányúnak kell lenni** (legalább a visszafelé megtalált megoldási út mentén).
 - Ez állapottér modellezés esetén biztosan teljesül, ha a műveletek van inverze.
- ❑ **Ismerni kell** egy startból elérhető **célállapotot**.

Ha ezek a feltételek nem állnak fenn, de mégis visszafelé haladó keresést szeretnénk végezni, akkor alkalmazzunk probléma redukciót.

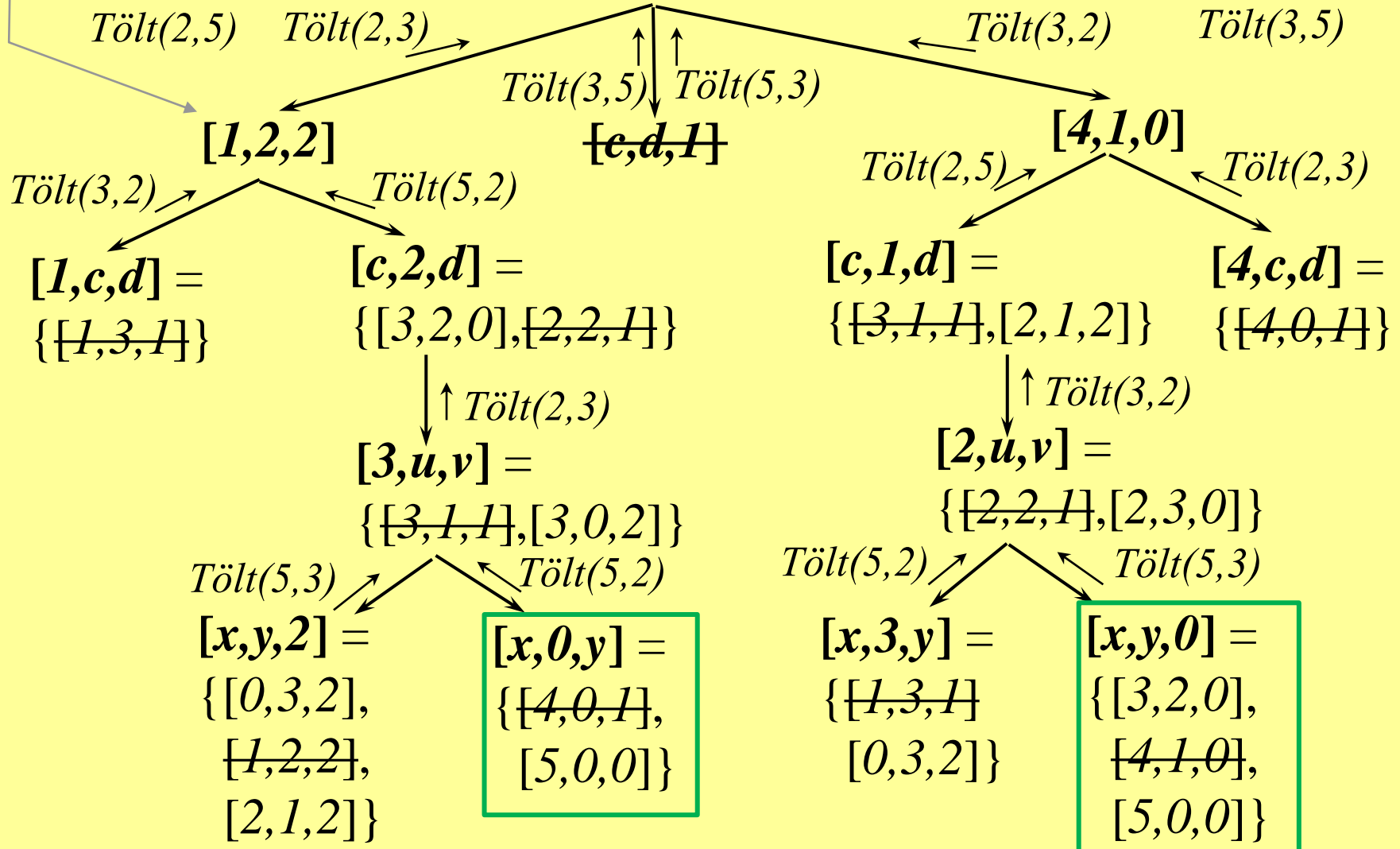
Kancsók probléma redukciós gráfja

Redukció: mely állapotokból lehet egy adott művelettel egy adott állapothalmazba eljutni?

Állapotok helyett, állapotok halmazaival dolgozunk.

célállapotok

$$[a,b,1] = \{[4,0,1], [3,1,1], [2,2,1], [1,3,1]\}$$



invariáns: $\sum_{i \in \text{Keys}} \text{this}[i] = 5$ és $\forall i \in \text{Keys}: \text{this}[i] \leq i$

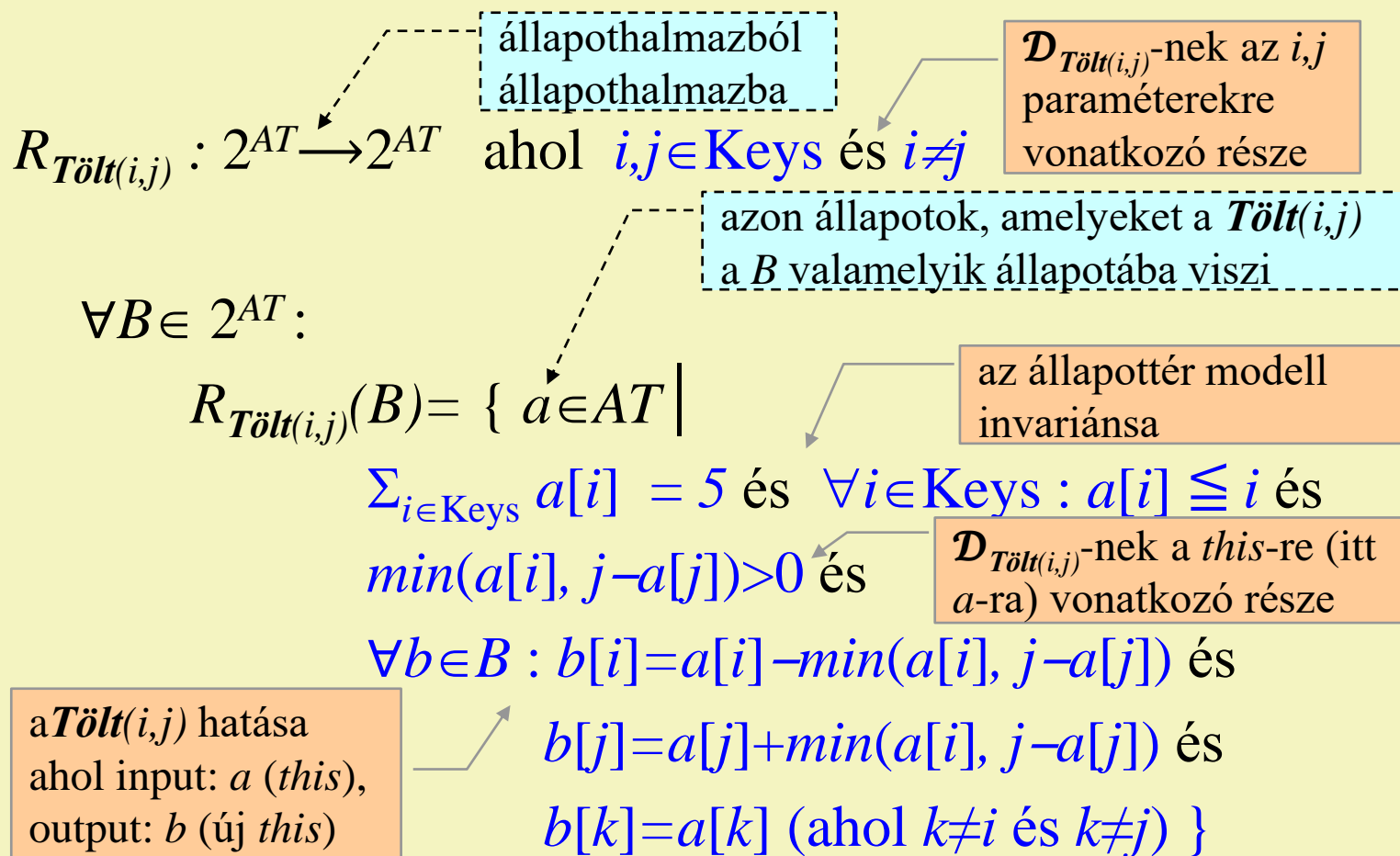
$Tölt(i,j): AT \rightarrow AT$

HA $i,j \in \text{Keys}$ és $i \neq j$ és $\min(\text{this}[i], j - \text{this}[j]) > 0$

AKKOR $\text{this}[i] := \text{this}[i] - \min(\text{this}[i], j - \text{this}[j])$

$\text{this}[j] := \text{this}[j] + \min(\text{this}[i], j - \text{this}[j])$

Kancsó probléma redukciós operátora



Probléma-redukciós modell

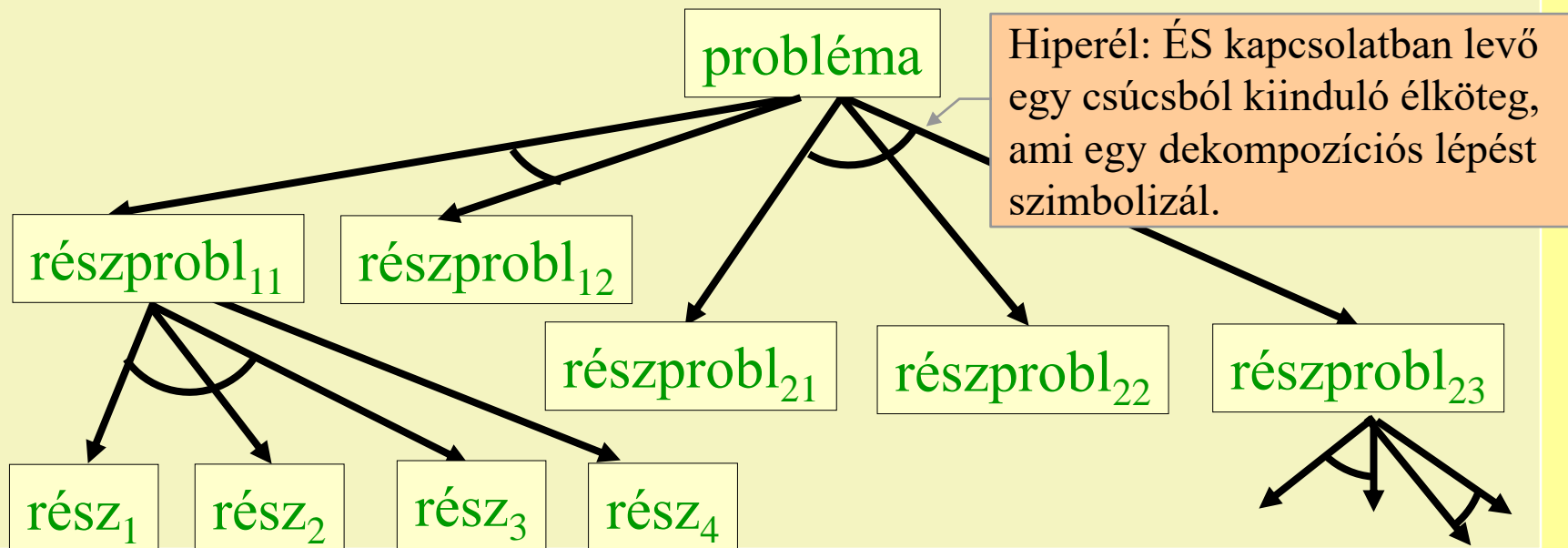
- ❑ Adott a probléma valamelyik **állapottér modellje**.
(állapottér: AT , invariáns: $Inv:AT \rightarrow \mathbb{L}$)
- ❑ Az állapottér modell minden $M:AT \rightarrow AT$ műveletéhez generálunk egy olyan $R_M:2^{AT} \rightarrow 2^{AT}$ leképezést (**redukciós operátor**), hogy az egy B állapot-halmazhoz azt az A állapot-halmazt rendeli, amely bármelyik állapotából az M művelet a B halmaz valamelyik állapotába vezet el.
 - $\mathcal{D}_{R_M} = \{ B \in 2^{AT} \mid B \neq \emptyset \}$
 - $\forall B \in \mathcal{D}_{R_M} : R_M(B) = \{ a \in AT \mid Inv(a) \text{ és } M(a) \in B \}$
- ❑ Egy **célhalmaz** csupa (de nem feltétlenül az összes) célállapotból áll.
- ❑ Egy **kezdőhalmaz** legalább egy kezdőállapotot tartalmaz.

Megjegyzés

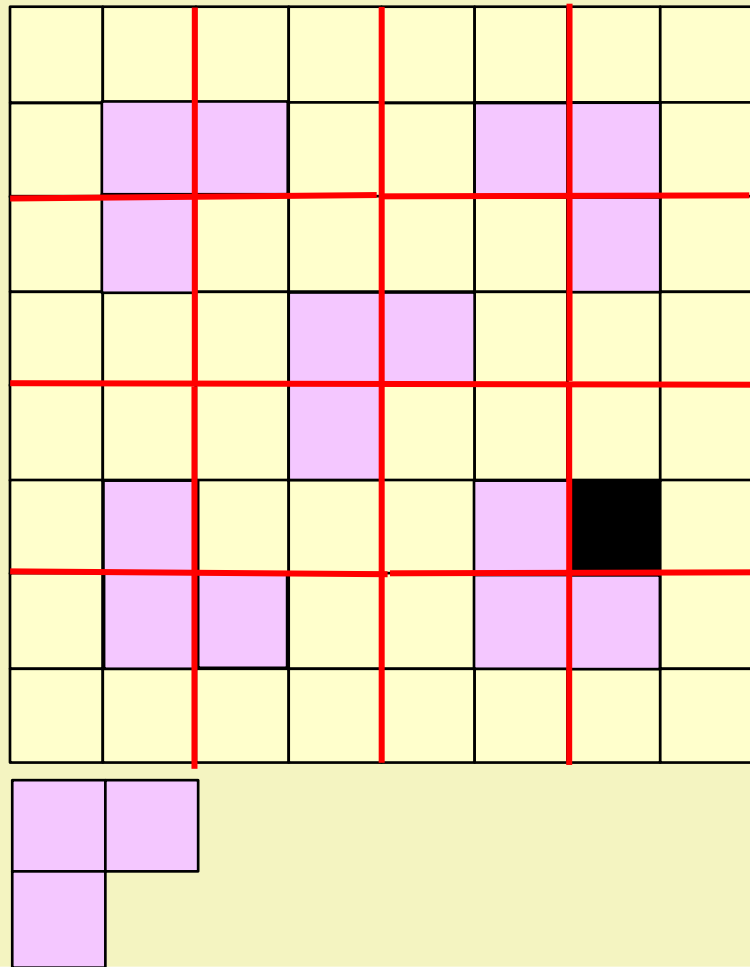
- ❑ A feladat olyan redukciós operátor-sorozat megtalálása, amely **célhalmazból kezdőhalmazba** vezet.
- ❑ A megoldás ezen sorozat redukciós operátorait generáló műveletek **fordított sorrendben kiolvasott sorozata** lesz.
- ❑ A probléma-redukció is modellezhető egy **irányított gráffal**, ahol a csúcsok állapot-halmazokat, az irányított élek a redukálásokat, a **célcsúcs** a célhalmazt, **startcsúcsok** a kezdőhalmazokat szimbolizálják, és a célcsúcsból egy startcsúcsba vezető út keresése a feladat.

3. Probléma dekompozíció

- ❑ Egy probléma dekomponálása során a problémát részproblémákra bontjuk, majd azokat tovább részletezzük, amíg nyilvánvalóan megoldható problémákat nem kapunk.
- ❑ Sok esetben egy problémát többféleképpen is fel lehet bontani részproblémákra.

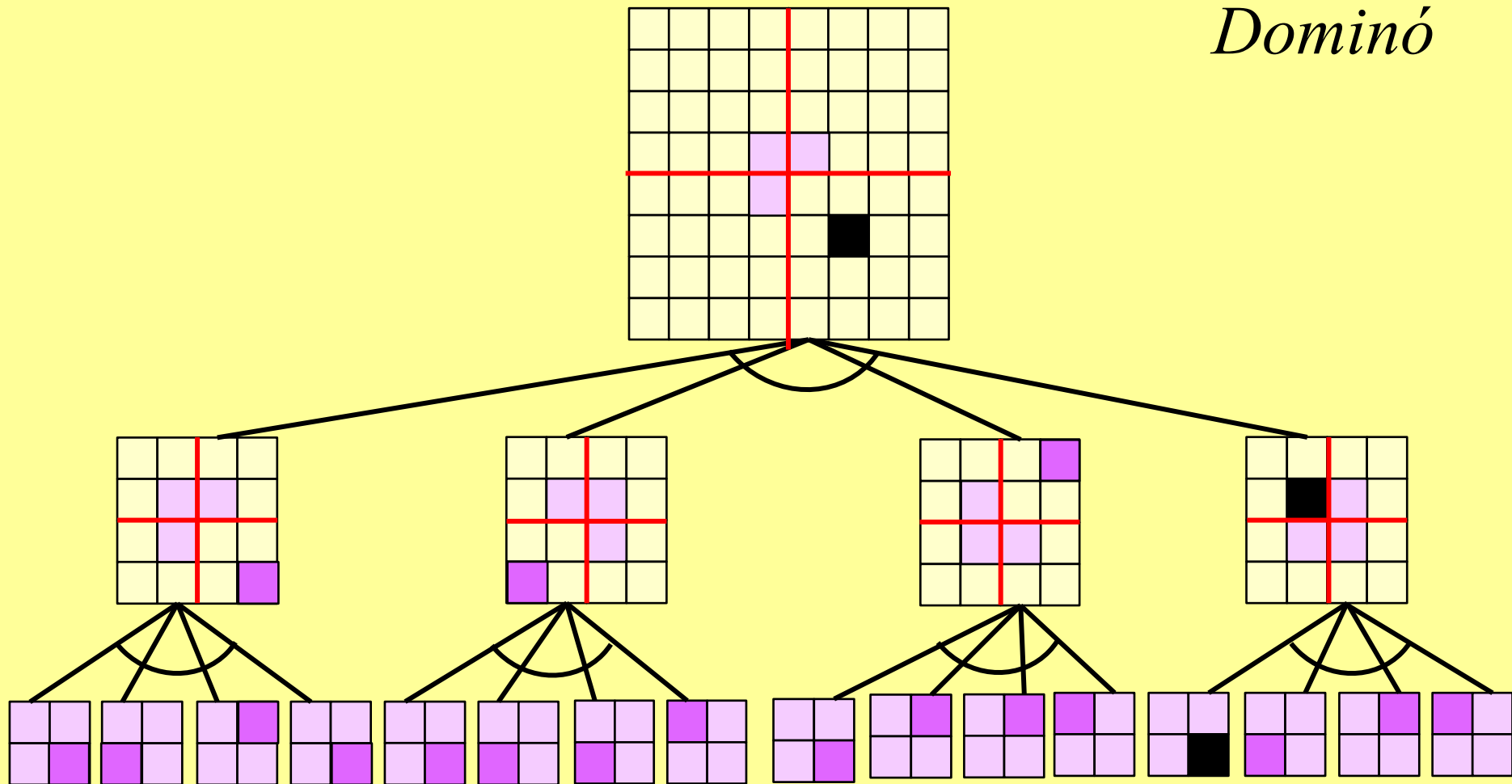


Dominó



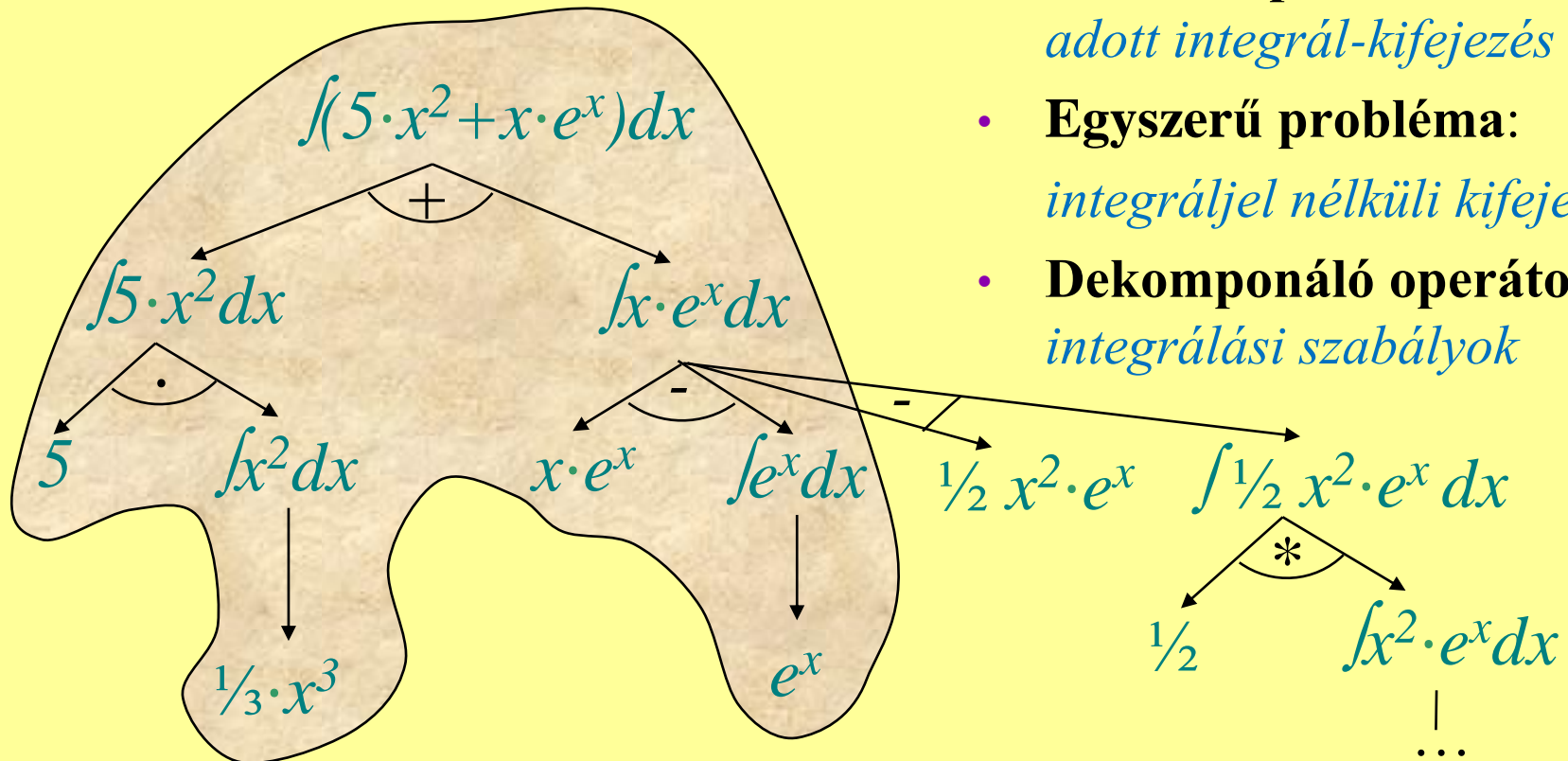
- **Probléma általános leírása:**
 $2^n \times 2^n$ -es tábla egy foglalt mezővel
- **Kiinduló probléma:**
 8×8 -as tábla egy foglalt mezővel
- **Egyszerű probléma:**
 2×2 -es tábla egy foglalt mezővel
- **Dekomponáló operátor:**
felosztja a táblát 4 egyenlő részre és elhelyez középre egy L alakú dominót úgy, hogy az ne fedjen le mezőt abban a részben, ahol a foglalt mező van

Dominó



Megoldás: Egy ÉS/VAGY gráf (fa) szimbolizálja a dekomponáló lépéseket, amely megoldás gráf is egyben. Ennek hiperélei és levélcsúcsai mutatják az L alakú elemek elhelyezését.

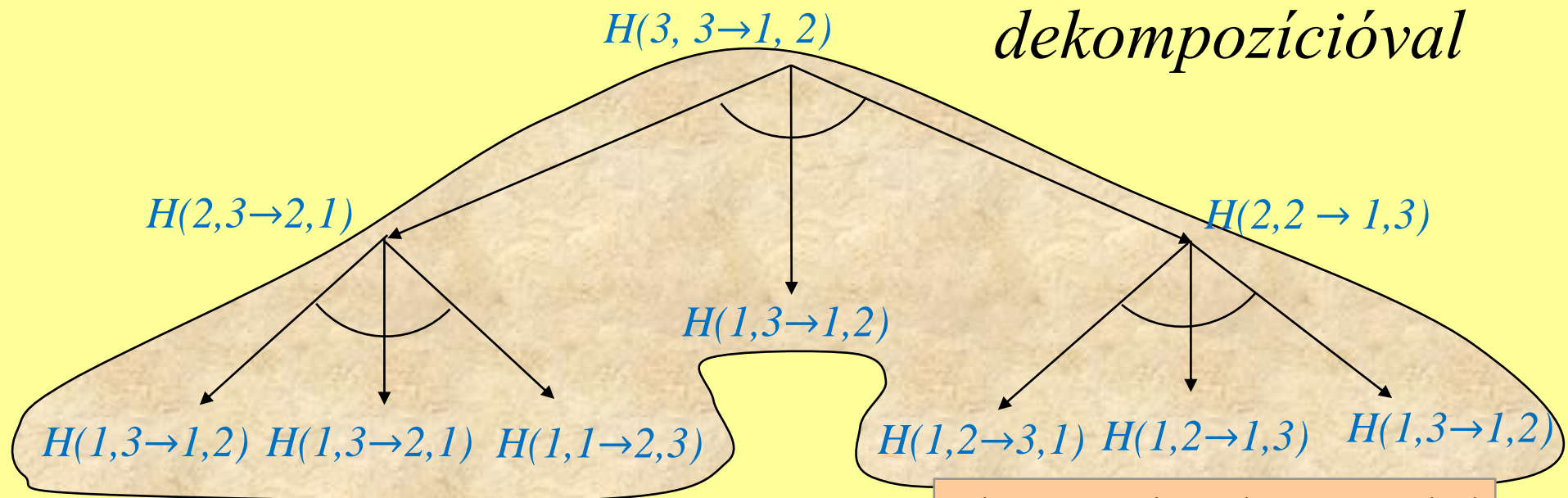
Integrálszámítás



- **Probléma általános leírása:**
szintaktikusan helyes kifejezés
- **Kiinduló probléma:**
adott integrál-kifejezés
- **Egyszerű probléma:**
integráljel nélküli kifejezés
- **Dekomponáló operátorok:**
integrálási szabályok

Megoldás: Egy ÉS/VAGY gráf (fa) egy megoldás gráfjának leveleiben tárolt szimbólumokat kell balról jobbra haladva a megfelelő hiperélekhez rendelt algebrai műveletekkel összekötni ahhoz, hogy az eredeti integrálkifejezés primitív függvényét megkapjuk.

Hanoi tornyai probléma megoldása dekompozícióval



Probléma általános leírása: $H(n, i \rightarrow j, k)$

Kiinduló probléma: $H(3, 3 \rightarrow 1, 2)$

Egyszerű probléma: $H(1, i \rightarrow j, k)$

Dekomponálás: $H(n, i \rightarrow j, k) \sim \langle H(n-1, i \rightarrow k, j), H(1, i \rightarrow j, k), H(n-1, k \rightarrow j, i) \rangle$

Megoldás: A keresett lépéssorozat a reprezentációs fa (ami egyben megoldásgráf is) leveleiből olvashatók ki balról jobbra haladva.

n korongot vigyünk át az i . rúdról a j . rúdra a k . rúd segítségével

eldönthető, hogy megoldható-e

Probléma dekompozíciós modell

- A modellhez meg kell adnunk:
 - a feladat *részproblémáinak általános leírását*,
 - a *kiinduló problémát*,
 - az *egyszerű problémákat*, amelyekről könnyen eldönthető, hogy megoldhatók-e vagy sem, és
 - a *dekomponáló műveleteket*:
 - $D: \text{probléma} \rightarrow \text{probléma}^+$ és
$$D(p) = \langle p_1, \dots, p_n \rangle$$

Probléma dekompozíció ÉS/VAGY gráffal

- Dekompozíciós modell ÉS/VAGY gráf
 - részprobléma ~ csúcs
 - dekomponáló művelet ~ irányított hiperél
hatása egy problémára
 - egy problémára véges sok művelet alkalmazható (véges kifokú gráf)
 - művelet költsége ~ hiperél költsége
 - van a műveltek költségének alsó pozitív korlátja (δ)
 - kiinduló probléma ~ startcsúcs
 - megoldható probléma ~ célcsúcs
- Gráf-reprezentáció: ÉS/VAGY gráf, startcsúcs, célcsúcsok
 - dekompozíciós folyam ~ hiperút
 - megoldás ~ megoldás-gráfból olvasható ki