II. Modellezés

1. Állapottér-reprezentáció

- ***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
- ***Végállapot(ok)*** vagy célfeltétel

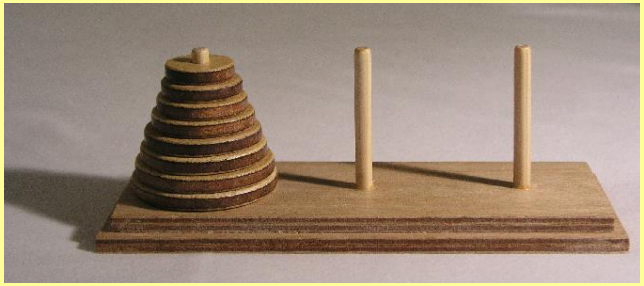
Állapottér-reprezentáció gráf-reprezentációja

- Állapot-gráf

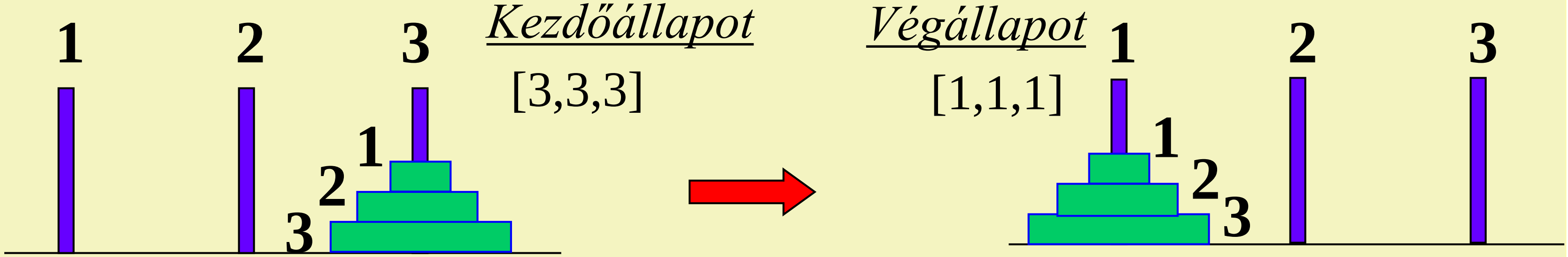
állapot	csúcs
művelet hatása	irányított él
művelet költsége	élköltség

- Reprezentációs gráf

δ-gráf	állapot-gráf
startcsúcs	kezdőállapot
célcsúcsok	végállapotok
irányított út	egy műveletsorozat hatása
irányított út a startból a célba	egy megoldás



Hanoi tornyai probléma



<u>*Kezdőállapot*</u> [3,3,3]

<u>*Végállapot*</u> [1,1,1]

<u>*Állapottér*</u>: ***AT*** **=**$\{1,2,3\}^n$

1..n intervallummal indexelt egydimenziós tömb, amelynek elemei *1,2*,vagy *3* halmazbeliek.

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

<u>*Művelet*</u>: ***Rak****(honnan, hova)*:***AT***→***AT*** *honnan, hova* ∈{*1,2,3*}

HA a *honnan* és *hova* <u>létezik</u> és <u>nem azonos</u>, és <u>van korong</u> a *honnan* rúdon, és a *hova* rúd <u>üres</u> vagy a mozgatandó korong (*honnan* rúd felső korongja) <u>kisebb</u>, mint a *hova* rúd felső korongja,

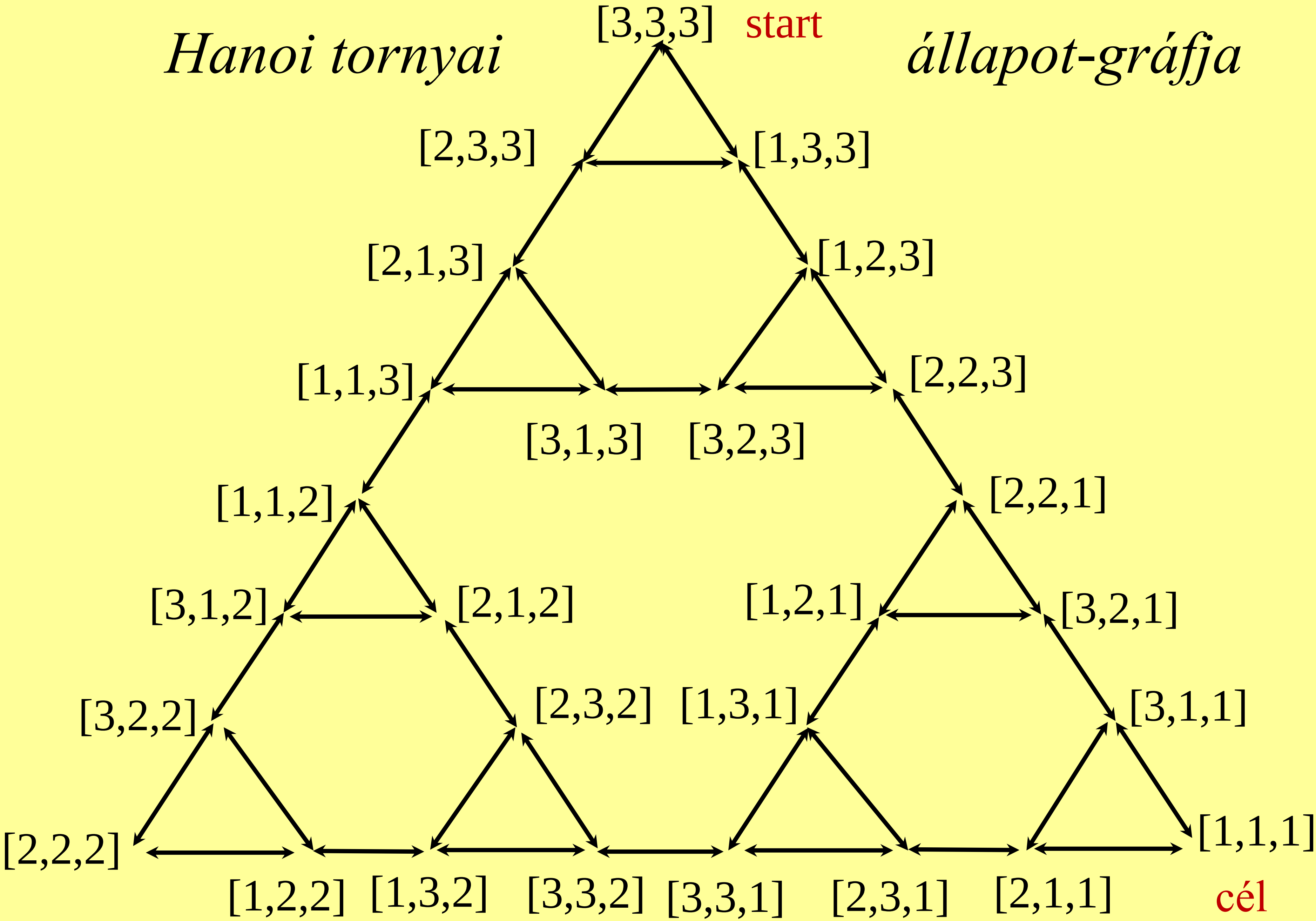
AKKOR *this*[*honnan legfelső korongja*] := *hova*

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) if (l1 = (_a[i]==from)) break;
             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) if (l2 = (_a[j]==to) break;
            if ( ¬l2 || i<j  ){ _a[i] = to;  return true; }   else return false;
      }
      bool final() const { for(int  i=0;i<n;++i) if(_a[i]!=1) return false;   return true; }
      void init()  { for(int  i=0;i<n;++i) _a[i] = 3; }
};
```

Hanoi tornyai állapot-gráfja



Állapottér vs. problématér

- A problématér elemeit többnyire a start csúcsból kiinduló utak szimbolizálják.
  - A Hanoi tornyai problémánál a problématér elemei nem az állapotok (csúcsok), hanem a kezdőállapotból kivezető műveletsorozatok (startcsúcsból induló irányított utak), hiszen a megoldás sem egyetlen állapot, hanem egy kezdőállapotot végállapotba vivő műveletsorozat (startcsúcsból célcsúcsba vezető irányított út).
  - Van amikor a megoldás egyetlen állapot (csúcs), de ebben az esetben is kell találni egy odavezető operátor-sorozatot (utat).
- Az állapottér-reprezentáció és a problématér között szoros kapcsolat áll fenn, de az állapottér többnyire nem azonos a problématérrel.

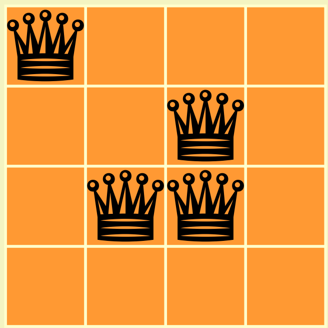
Állapot-gráf bonyolultsága

Állapot-gráf bonyolultsága → Problématér mérete → Keresés számításigénye

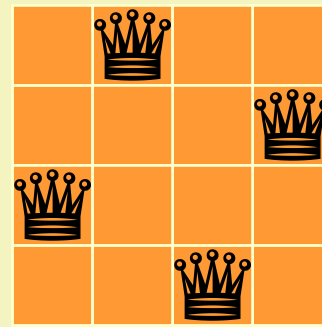
- A start csúcsból kivezető utak száma az oda-vissza lépések nélkül
  Hanoi: a legfeljebb k hosszú utak: $1+2+\ldots+2^k$, azaz $2^{k+1}-1$
  - csúcsok és élek száma
    Hanoi: 3^n csúcs, $3 \cdot \dfrac{3^n - 1}{3 - 1}$ él
  - a csúcsok átlagos ki-foka
    Hanoi: 3
  - körök gyakorisága, és hosszuk sokfélesége
    Hanoi: 2, 3, 6, 7, 9, …

n-királynő probléma 1.

általános állapot



utófeltételnek megfelelő állapot



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

<u>*Állapottér*</u>: ***AT*** = $\{ ♛, \_ \}^{n \times n}$

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

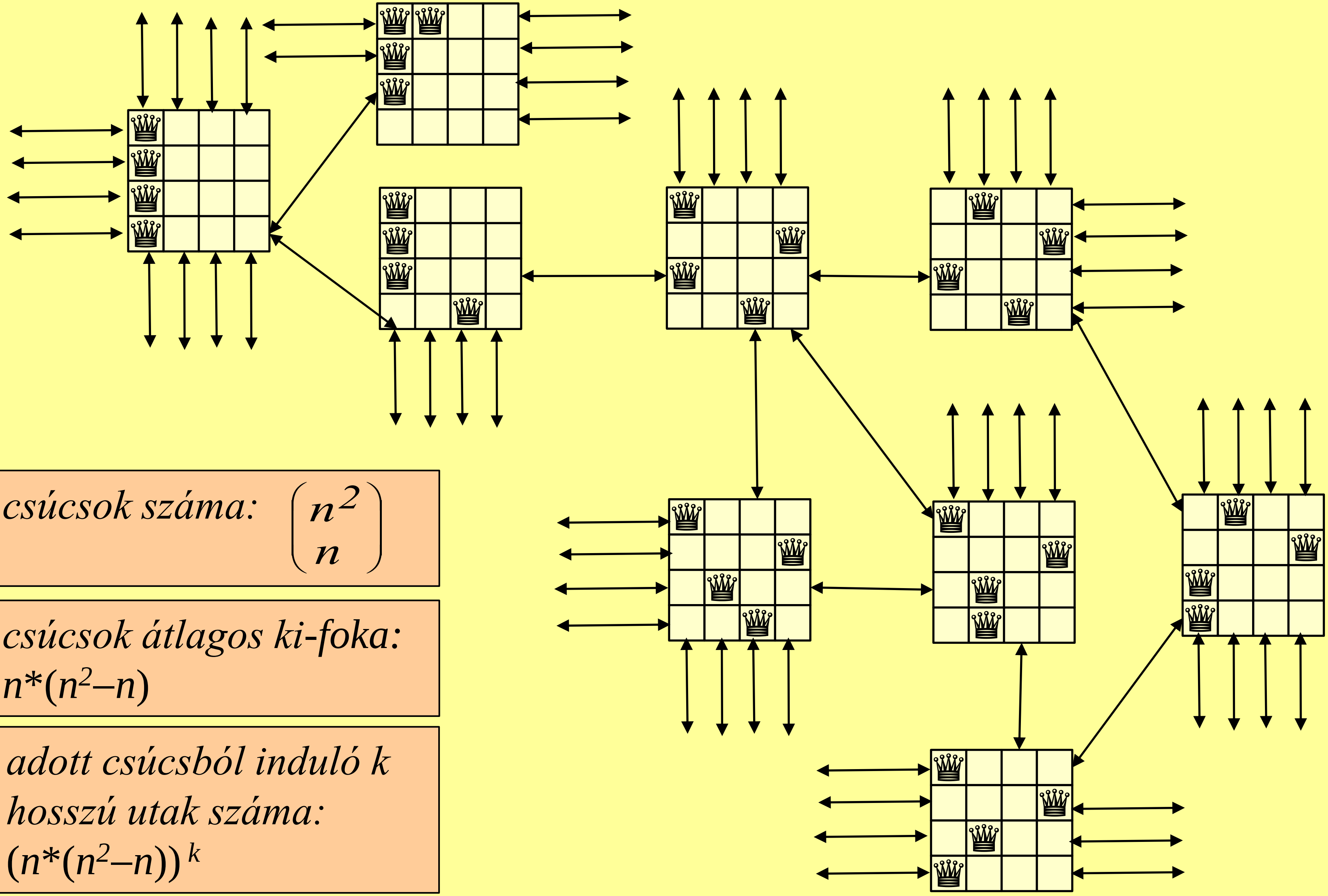
<u>*Művelet*</u>: ***Áthelyez***(x,y,u,v):***AT*** → ***AT*** $x,y,u,v \in [1..\, n]$ (*this*:***AT***)

HA $1 \leqq x,y,u,v \leqq n$ és *this*$[x,y]$= ♛ és *this*$[u,v]$=_

AKKOR *this*$[x,y] \leftrightarrow$ *this*$[u,v]$

csere

Állapot-gráf részlet



csúcsok száma: $\binom{n^2}{n}$

csúcsok átlagos ki-foka: $n*(n^2-n)$

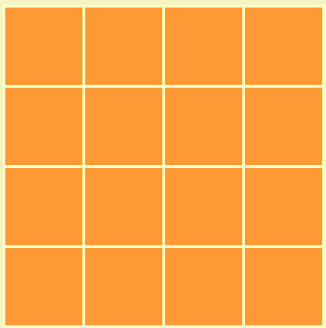
adott csúcsból induló k hosszú utak száma: $(n*(n^2-n))^k$

Csökkentsük a problématér méretét

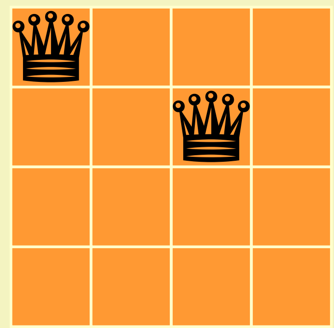
- Ugyanannak a feladatnak több modellje lehet : érdemes olyat keresni, amely kisebb problémateret jelöl ki.
  - Az előző reprezentációnál a problématér mérete, azaz a lehetséges utak száma, óriási.
  - Bővítsük az állapotteret az *n*-nél kevesebb királynőt tartalmazó állásokkal, és használjunk új műveletet : királynő-felhelyezést (kezdő állás az üres tábla). Ekkor a pontosan *n* hosszú utak (ilyen a jó megoldás is) száma: $\binom{n^2}{n} \cdot n!$
  - Műveletek előfeltételének szigorításával csökken az állapot-gráf átlagos ki-foka:
    - Sorról sorra haladva csak egy-egy királynőt helyezzünk fel a táblára! Ekkor az *n* hosszú utak száma: n^n.
    - Ütést tartalmazó állásra ne tegyünk királynőt!

n-királynő probléma 2.

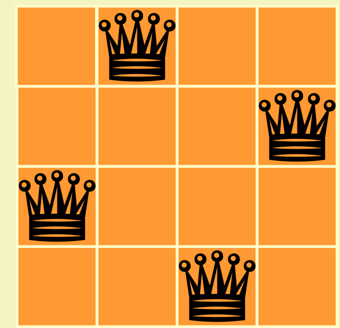
kezdőállapot



közbülső állapot



végállapot



nincs már üres sor és nincs ütés

Állapottér: ***AT*** = {♛, _ }$^{n\times n}$

invariáns: az első néhány sor egy-egy királynőt tartalmaz

Művelet: **Helyez**(*oszlop*): ***AT*** → ***AT*** *oszlop*∈[*1*.. *n*] (*this*:***AT***)

HA *1*≦ *oszlop* ≦*n* és a *this*-beli soron következő üres sor ≦ *n* és nincs ütés a *this*-ben

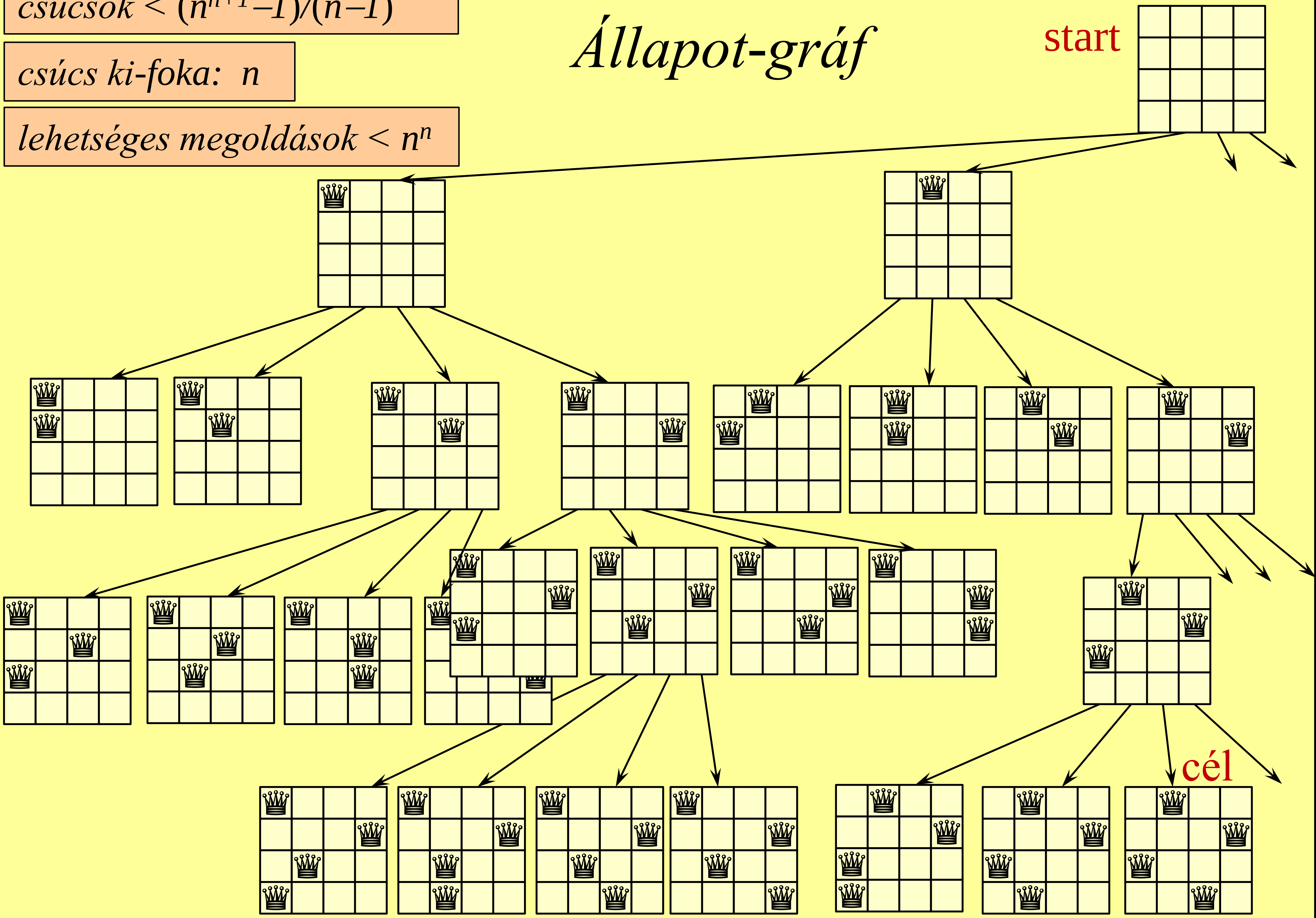
AKKOR *this*[a *this*-beli soron következő üres sor,*oszlop*] := ♛

csúcsok < $(n^{n+1}-1)/(n-1)$

csúcs ki-foka: n

lehetséges megoldások < n^n

Állapot-gráf



Művelet végrehajtásának hatékonysága

- A művelet kiszámítási bonyolultsága csökkenthető, ha az állapotokat extra információval egészítjük ki, vagy az invariáns szigorításával szűkítjük az állapotteret.
- Például
  - A tábla soron következő üres sorának sorszámát eltárolhatjuk a tábla mellett az állapotokban, így új királynő elhelyezésekor ezt nem kell kiszámolni, ugyanakkor könnyen aktualizálhatjuk azt a növelésével egy művelet végrehajtásakor.
  - Ne engedjünk 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 három féle lesz: szabad, ütés alatt álló vagy foglalt, amelyeket a művelet végrehajtásakor kell karbantartani.

n-királynő probléma 3.

kezdőállapot:

kövsor = 1

közbülső állapot:

kövsor = 3

♛	×	×	×
×	×	♛	×
×	×	×	×
×		×	×

végállapot :

kövsor = 5

×	♛	×	×
×	×	×	♛
♛	×	×	×
×	×	♛	×

Állapottér: **AT** = *rec*(*t* :{♛, × , _ }$^{n\times n}$, *kövsor* : ℕ)

invariáns: királynők nem ütik egymást,

kövsor≦*n*+*1*,

az első *kövsor*–*1* darab sor egy-egy királynőt tartalmaz,

× egy királynő által ütött üres mezőt jelöli,

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

n-királynő probléma 3. folytatás

<u>Művelet</u>: új királynő elhelyezése a soron következő üres sor szabad mezőjére

Helyez(*oszlop*): ***AT*** → ***AT*** *oszlop*∈[1.. *n*] (*this*:***AT***)

HA $1 \leqq oszlop \leqq n$ és $this.kövsor \leqq n$

és *this.t*[*this.kövsor*,*oszlop*]=_

előfeltétel számítás-igénye: konstans

AKKOR *this.t*[*this.kövsor*,*oszlop*] := ♛

∀*i*∈[*this.kövsor*+1..*n*] : *this.t*[*i*,*oszlop*]:=✖

this.t[*i*,*i*–*this.kövsor*+*oszlop*]:=✖

this.t[*i*, *this.kövsor*+*oszlop*–*i*]:=✖

hatás számítás-igénye: lineáris

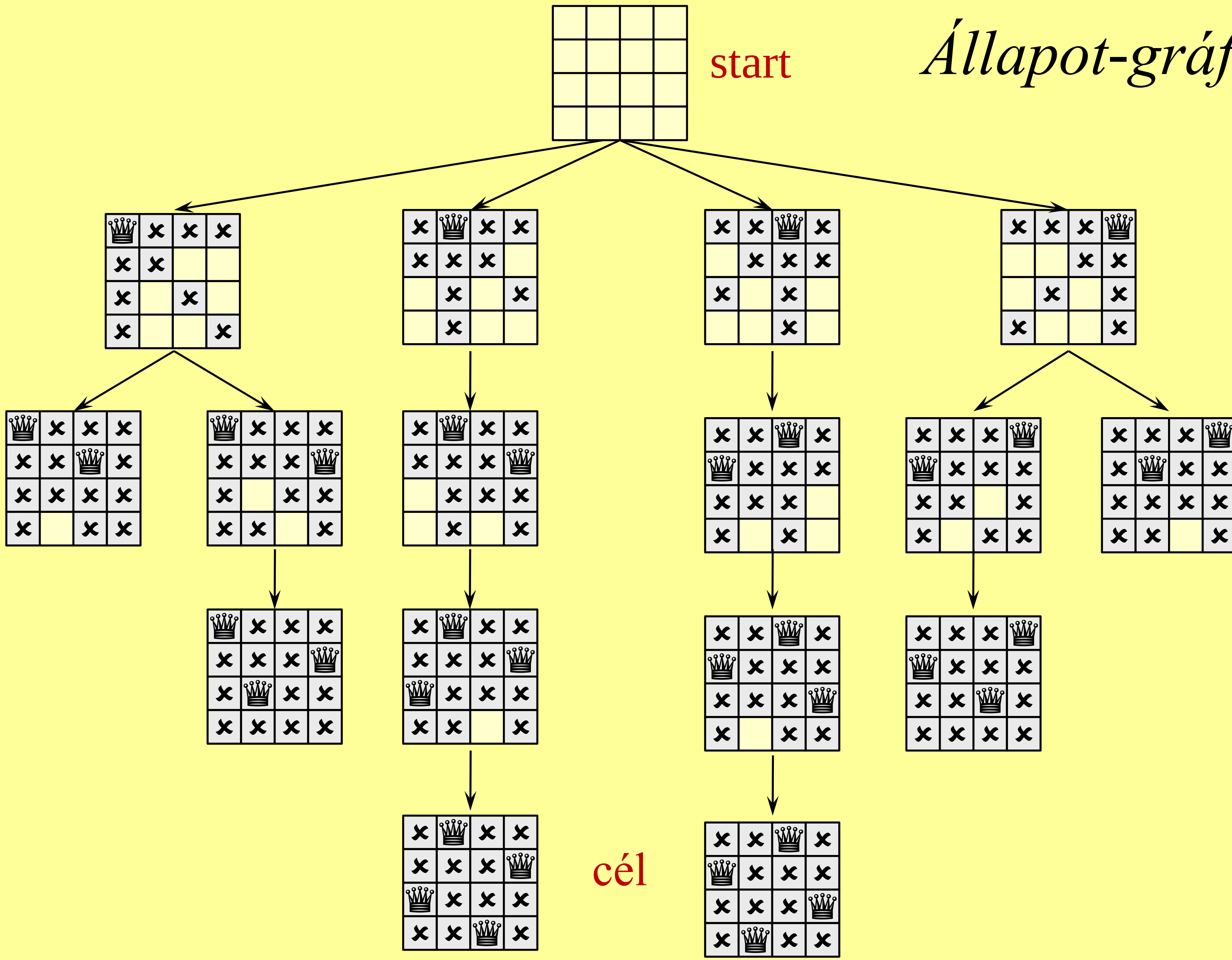
this.kövsor:=*this.kövsor*+1

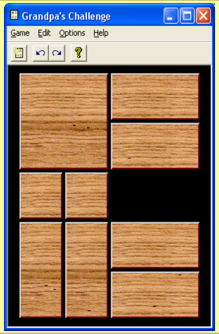
<u>Kezdőállapot</u>: *this.t* egy üres mátrix, *this.kövsor*:=1

<u>Végállapot</u>: *this.kövsor*>*n*

célfeltétel nagyon egyszerű lett

Állapot-gráf





Tologató játék (8-as, 15-ös)

Kezdőállapot: tetszőleges

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

→

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

Végállapot: szokásos

Állapottér: $\boldsymbol{AT} = rec(mátrix :\{0..8\}^{3\times3},\ üres :\{1..3\}\times\{1..3\})$

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

Művelet: **Tol**(*irány*): $\boldsymbol{AT} \to \boldsymbol{AT}$

koordinátánként értendő

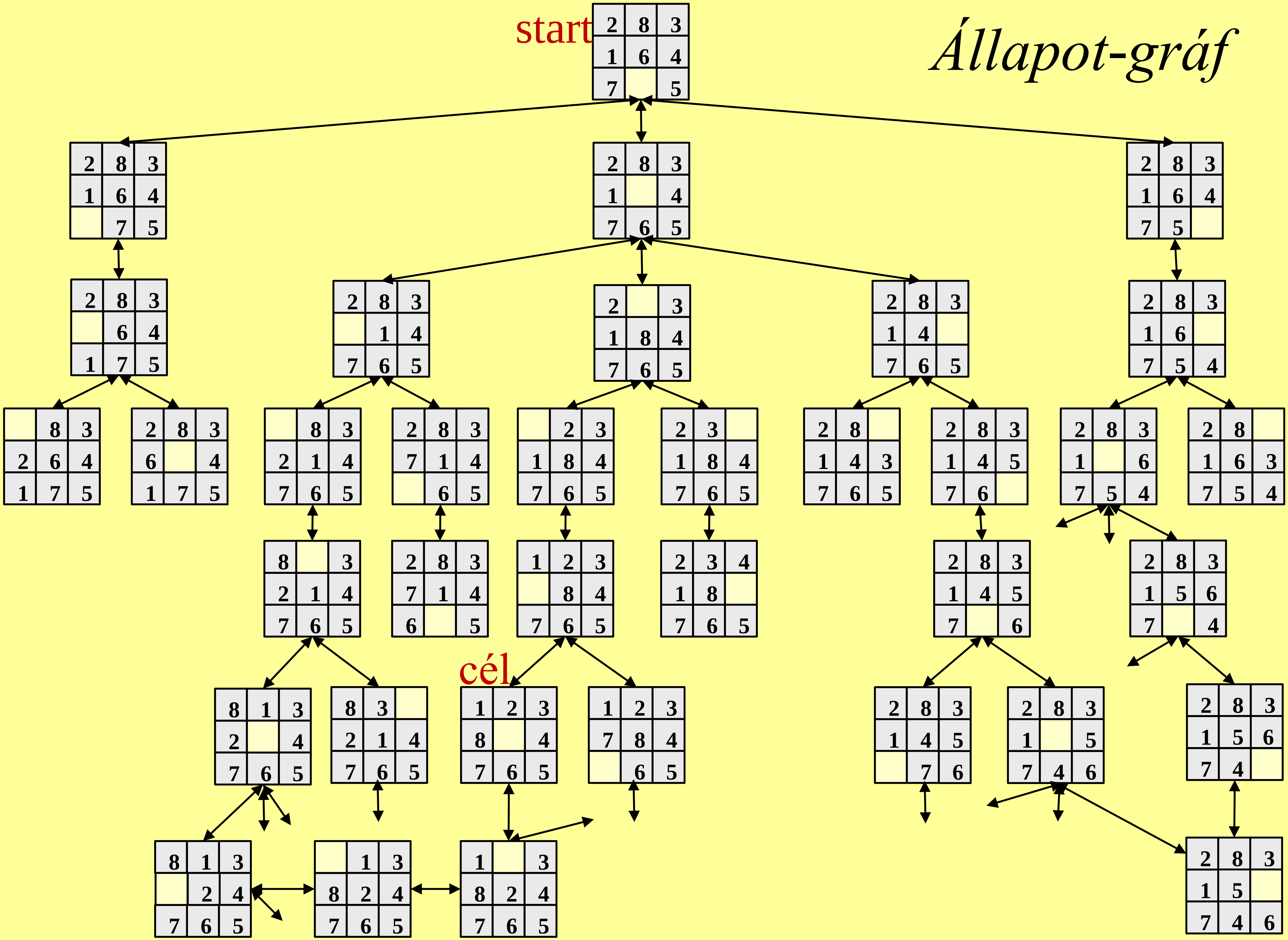
HA $irány \in \{(1,0),(0,1),(-1,0),(0,-1)\}$ és

$(1,1) \leqq this.üres + irány \leqq (3,3)$ (*this*: ***AT***)

AKKOR $this.mátrix[this.üres] \leftrightarrow this.mátrix[this.üres + irány]$

$this.üres := this.üres + irány$

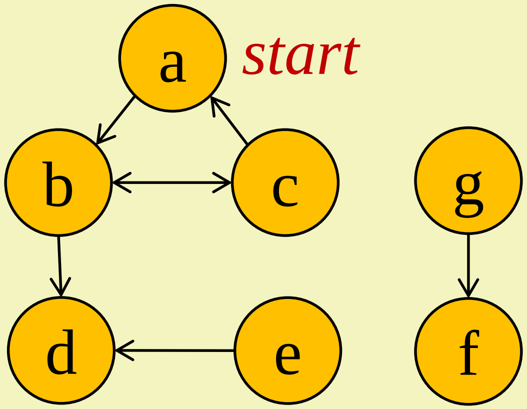
Állapot-gráf



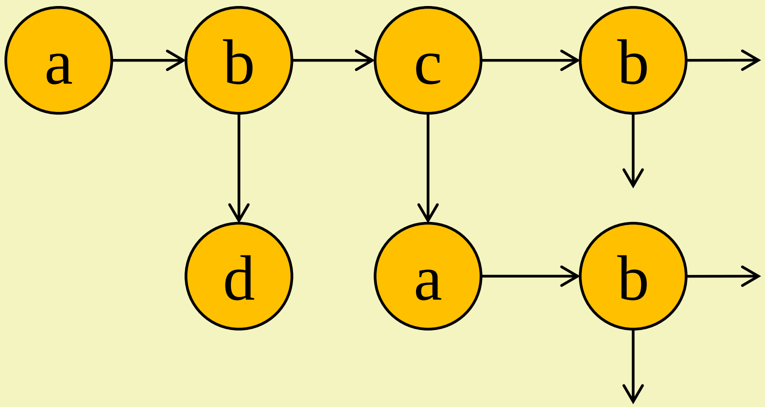
Hogyan „látja” egy keresés a reprezentációs gráfot?

- Egy keresés fokozatosan fedezi fel a reprezentációs gráfot: bizonyos részeihez el sem jut és a felfedezett részt sem feltétlenül tárolja el teljesen. Sőt kifejezetten torzultan „látja” a gráfot, ha egy csúcshoz érve nem vizsgálja meg, hogy ezt a csúcsot korábban már felfedezte-e, hanem ehelyett új csúcsként regisztrálja.

eredeti gráf:



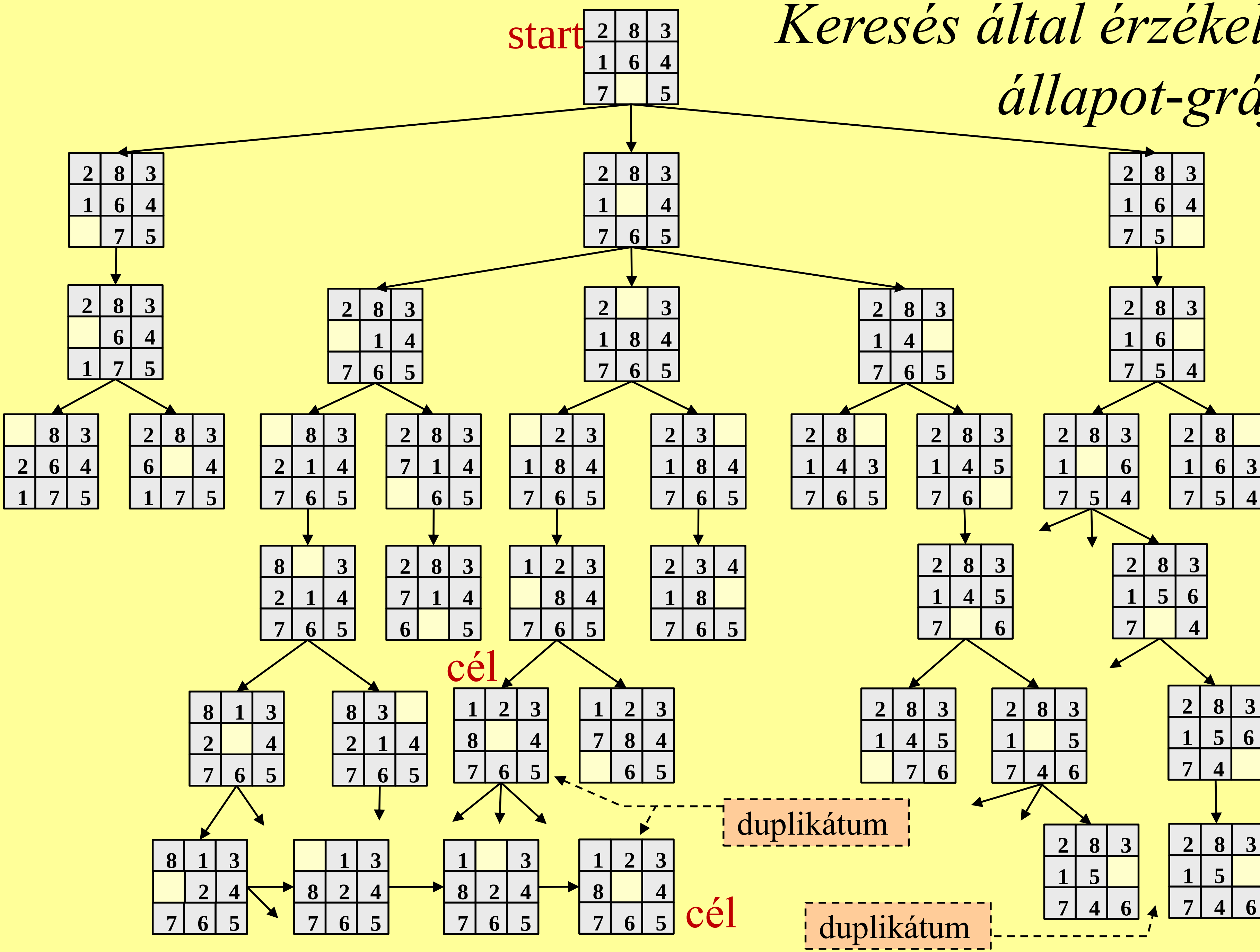
keresés által látott gráf:



Reprezentációs gráf „fává egyenesítése”

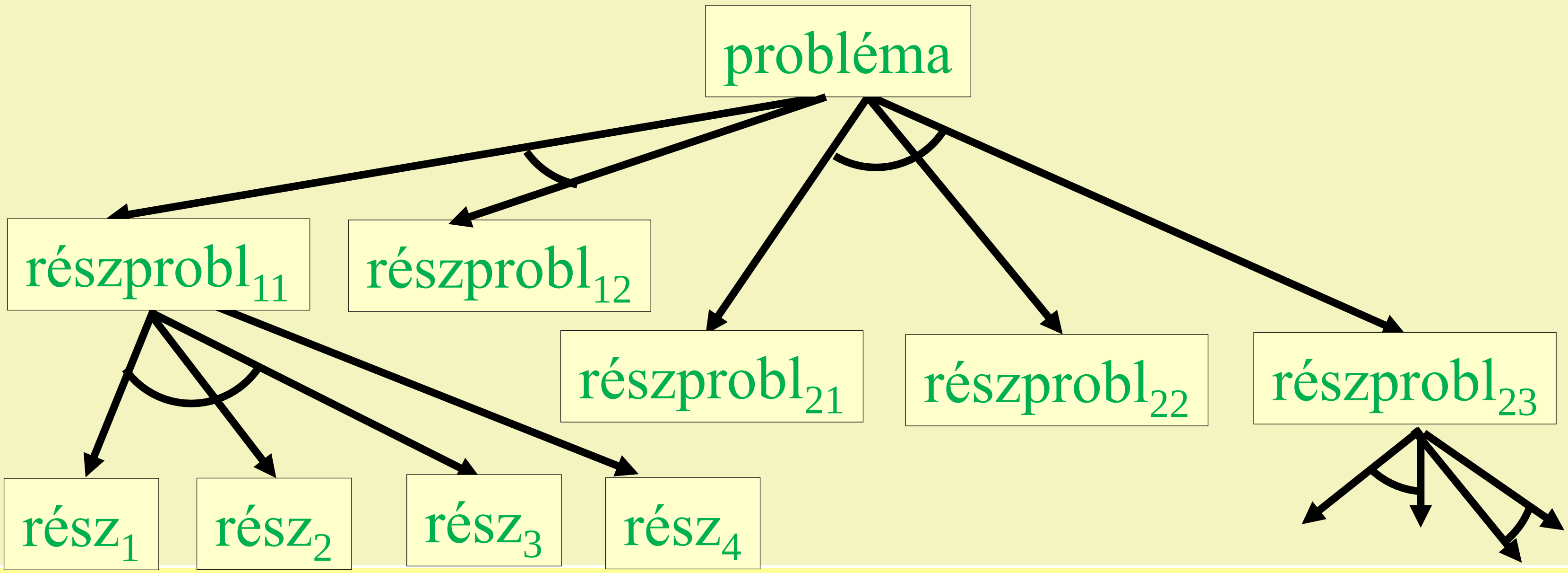
- Ha a keresés nem vizsgálja meg egy csúcsról, hogy korábban már felfedezte-e, akkor az eredeti reprezentációs gráf helyett valójában annak fává kiegyenesített változatában keres.
  Előny: eltűnnek a körök, de a megoldási utak megmaradnak
  Hátrány: duplikátumok jelennek meg, sőt a körök kiegyenesítése végtelen hosszú utakat eredményez
- A kétirányú (oda-vissza) élek különösen megnövelik a kiegyenesített fa méretét. Ezek kiszűrése azonban könnyen beépíthető minden keresésbe a megelőző aktuális csúcs (szülőcsúcs) eltárolásával, így az aktuális csúcsból oda visszavezető él felismerhető és figyelmen kívül hagyható.

Keresés által érzékelt állapot-gráf

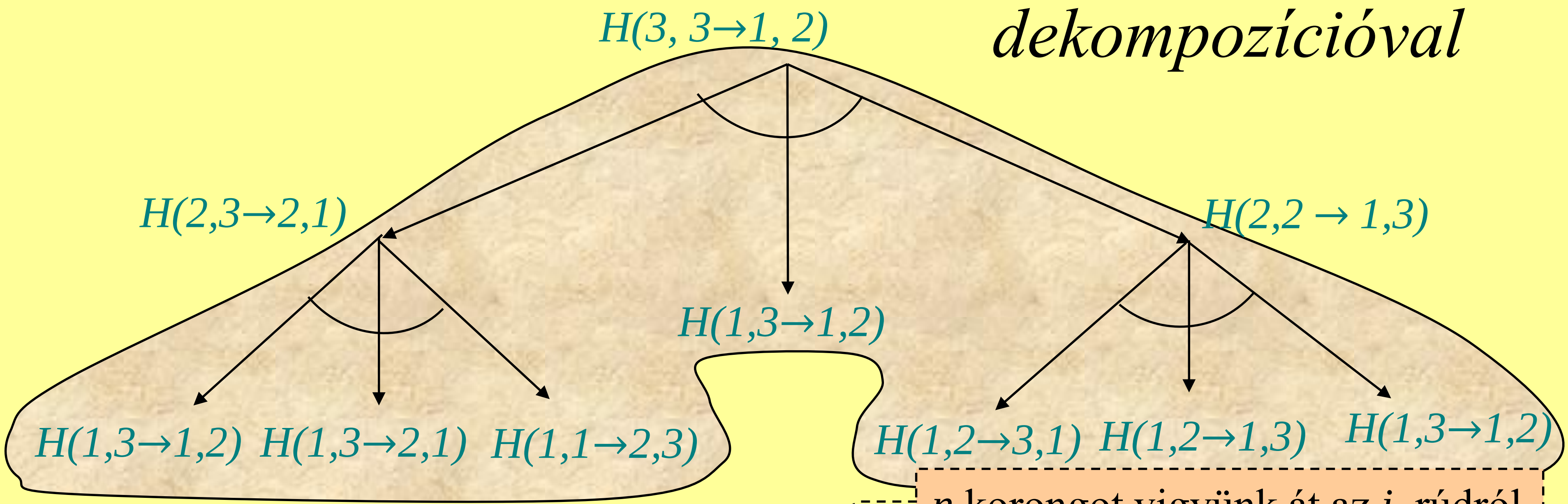


2. Probléma dekompozíció

- Egy probléma dekomponálása során a problémát részproblémákra bontjuk, majd azokat tovább részletezzük, amíg nyilvánvalóan megoldható problémákat nem kapunk.
- Sokszor egy probléma megoldását akár többféleképpen is fel lehet bontani részproblémák megoldásaira.



Hanoi tornyai probléma megoldása dekompozícióval



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

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

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

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

eldönthető, hogy megoldható-e

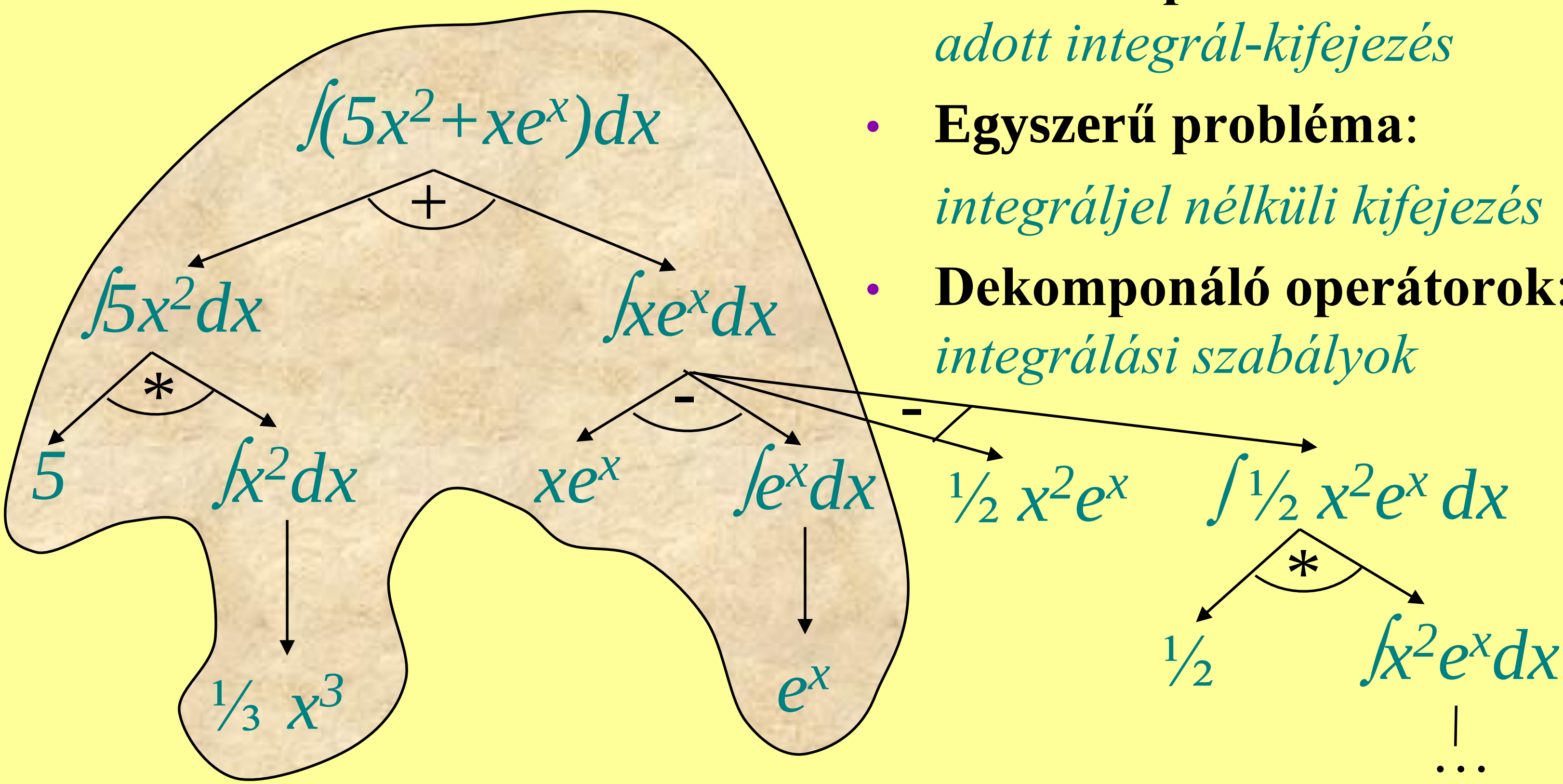
Dekomponálás: $H(n, i{\rightarrow}j, k) \leadsto \langle H(n-1,i{\rightarrow}k, j), H(1, i{\rightarrow}j, k), H(n-1, k{\rightarrow}j, i)\rangle$

Megoldás gráf: kiinduló problémát egyszerű problémákra visszavezető fa

Megoldás: a részfa leveleit kell balról jobbra haladva összeolvasni

Integrálszámítás

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



Megoldás gráf: alternatívákat nem tartalmazó levezetés egy olyan részfa, amelynek belső csúcsaiból egyetlen élköteg indul ki, levelei egyszerű megoldható problémák

Megoldás: a megoldó részfa leveleit balról jobbra haladva kötjük össze a dekomponálások algebrai műveleteivel

Dekompozíciós reprezentáció fogalma

- A reprezentációhoz 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: $\textit{probléma} \to \textit{probléma}^{+}$ és
      $D(p)=\langle p_1, \dots , p_n\rangle$

A dekompozíció modellezése ÉS/VAGY gráffal

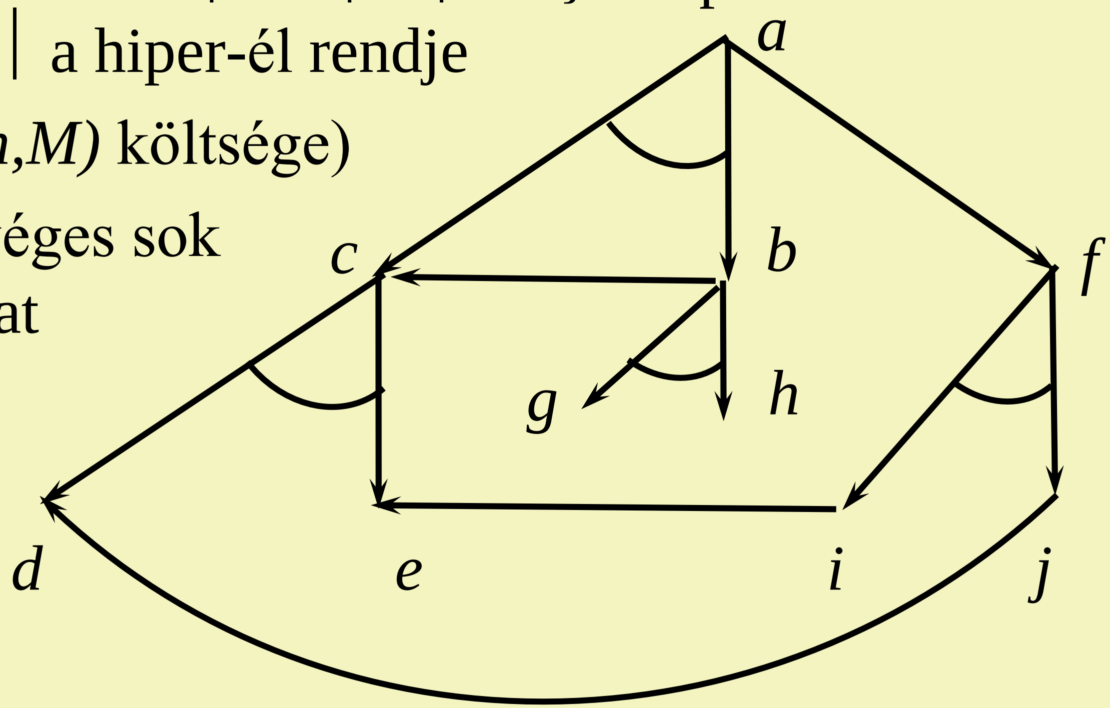
- Egy dekompozíciót egy ÉS/VAGY gráffal szemléltethetünk:
  - a csúcsok részproblémákat jelölnek, a kiinduló problémát a startcsúcs, a megoldható egyszerű problémákat a célcsúcsok.
  - egy dekomponáló művelet hatását egy élköteg mutatja, amely a dekomponált probléma csúcsából a dekomponálással előállított részproblémák csúcsaiba vezet.
    - egy élköteg élei között ún. „ÉS” kapcsolat van: a dekomponált probléma megoldásához annak minden részproblémáját meg kell oldani;
    - egy csúcsból több élköteg is indulhat, hiszen egy probléma többféleképpen dekomponálható. Ezen elkötegek élei között ún. „VAGY” kapcsolat áll fenn: választhatunk, hogy melyik élköteg mentén oldjunk meg egy problémát.

Megoldás-gráf

- Az eredeti problémát egyszerű problémákra visszavezető dekomponálási folyamatot az ÉS/VAGY gráf speciális részgráfja, az ún. megoldás-gráf jeleníti meg, amelyben
  - út vezet a startcsúcsból minden csúcsba, és minden csúcsból egy célcsúcsba
  - egy éllel együtt az összes azzal „ÉS” kapcsolatban álló éleket is tartalmazza (azaz teljes élkötegeket tartalmaz)
  - nem tartalmaz „VAGY” kapcsolatban álló él párokat
- A megoldás a megoldás-gráfból olvasható ki.
- Általában nincs összefüggés a megoldás-gráf költsége (bárhogyan is definiáljuk ezt) és a megoldás költsége között.

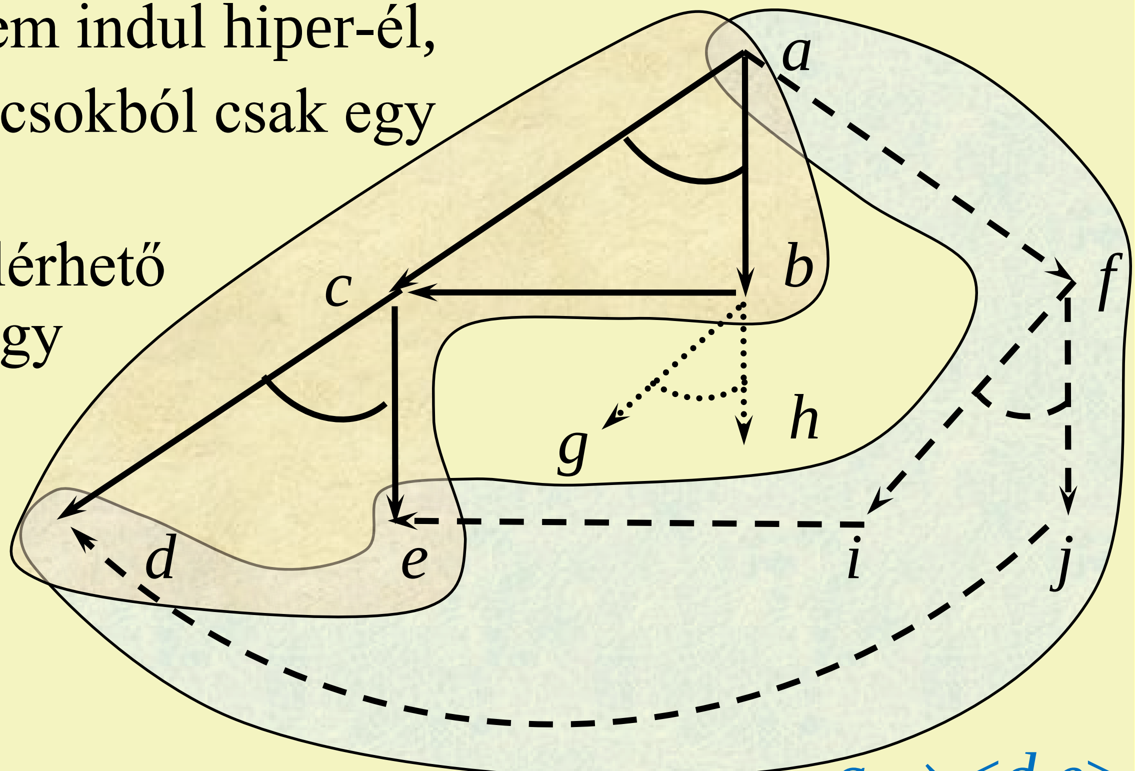
ÉS/VAGY gráfok

1. Az $R=(N,A)$ élsúlyozott irányított hiper-gráf, ahol az
   - N a csúcsok halmaza,
   - $A \subseteq \{ (n,M) \in N \times N^{+} \mid 0 \neq |M| < \infty \}$ a hiper-élek halmaza, $|M|$ a hiper-él rendje
   - ($c(n,M)$ az (n,M) költsége)
2. Egy csúcsból véges sok hiper-él indulhat
3. $(0 < \delta \leq c(n,M))$



Az n csúcsból az M csúcs-sorozatba vezető irányított hiper-út fogalma

- Az $n^\alpha \rightarrow M$ hiper-út ($n \in N$, $M \in N^+$) egy olyan véges részgráf, amelyben
  - *M* csúcsaiból nem indul hiper-él,
  - *M*-en kívüli csúcsokból csak egy hiper-él indul,
  - minden csúcs elérhető az *n* csúcsból egy közönséges irányított úton.
- Hiper-út hossza az éleinek száma.
- Hiper-út költsége is definiálható.

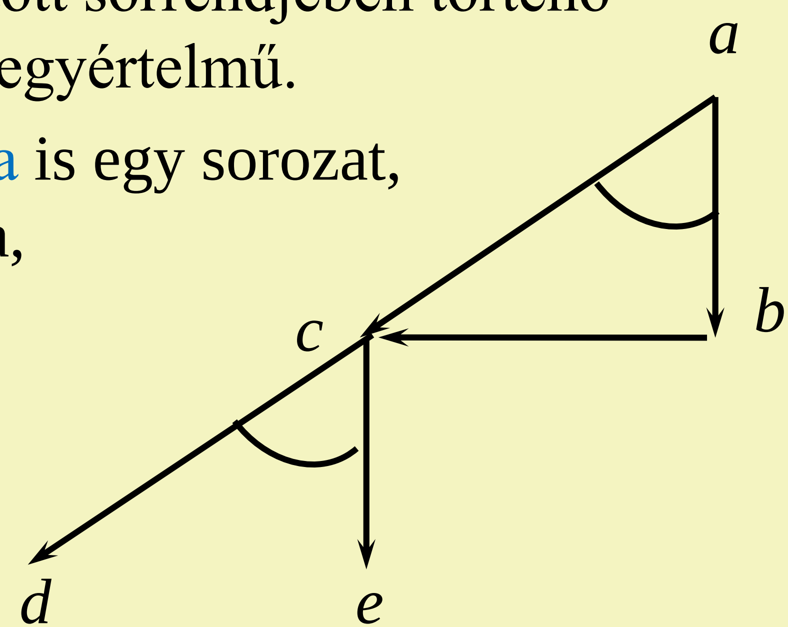


A hiper-út és a megoldás kapcsolata

- Az $n^{\alpha}\rightarrow M$ hiper-út egy egyértelmű haladási irányt jelöl ki az *n* csúcsból az *M* csúcsaiba.
- A probléma dekompozíciónál említett megoldás-gráf nem más, mint egy olyan hiper-út, amely a startcsúcsból egy célcsúcs-sorozatba vezet. Ez a célcsúcs-sorozat adja az eredeti probléma megoldását.
- A megoldás-gráf megtalálásához tehát a startcsúcsból kivezető hiper-utakat kell megvizsgálni.
- A startcsúcsból kivezető hiper-utak szerepe ugyanaz, mint a közönséges irányított gráfokban a startcsúcsból induló közönséges irányított utaké.

Különbség a közönséges út és a hiper-út bejárása között

- Egy közönséges irányított út bejárásán az úton fekvő csúcsoknak az élek által mutatott sorrendjében történő felsorolását értjük. Ez mindig egyértelmű.
- Egy irányított hiper-út bejárása is egy sorozat, de ez csúcs-sorozatok sorozata, ami nem egyértelmű.



Minden lépésben egy csúcs összes előfordulását kicseréljük a csúcsból induló hiper-él végcsúcs-sorozatára

bejárások:

$$<a> \rightarrow <c,b> \rightarrow <c,c> \rightarrow < d,e,d,e >$$

$$<a> \rightarrow <c,b> \rightarrow <d,e,b > \rightarrow <d,e,c> \rightarrow < d,e,d,e >$$

Hiper-út bejárása

- Az $n \to M$ hiper-út egy bejárásán a hiper-út csúcsaiból képzett sorozatoknak a felsorolását értjük:
  - első sorozat: $<n>$
  - a C sorozatot a $C^{k \leftarrow K}$ sorozat követi (ahol C-ben k minden előfordulásának helyére a K sorozatot írjuk), ha a hiper-útnak van olyan (k,K) hiper-éle, ahol $k \in C$ de $k \notin M$.
- Megjegyzés:
  - Egy hiperútnak véges sok véges hosszú bejárása van.
  - Egy $n \to T$ hiper-út (azaz egy megoldás-gráf) bejárásainak utolsó eleme egy csupa célcsúcsot tartalmazó sorozat.

Útkeresés ÉS/VAGY gráfban

- Egy ÉS/VAGY gráf startból induló hiper-útjainak (a potenciális megoldás gráfoknak) a bejárásai közönséges irányított utakként ábrázolhatók, és ezáltal egy közönséges irányított gráfot írnak le, amelynek csúcsai az eredeti ÉS/VAGY gráf csúcsainak véges sorozatai.
- Ha ebben a közönséges gráfban megoldási (azaz csupa célcsúcsot tartalmazó sorozatba vezető) utat találunk, akkor az egyben az eredeti ÉS/VAGY gráf megoldás-gráfja is lesz.
- Az ÉS/VAGY gráfbeli megoldás-gráf keresést tehát közönséges irányított gráfbeli keresést végző útkereső algoritmusokkal végezhetjük el.