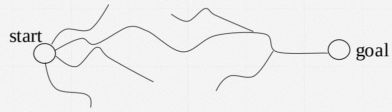
# IV. REDUKCIÓ, DEKOMPOZÍCIÓ

1. Visszafelé haladó keresés
2. Probléma redukció
3. Probléma dekompozíció
4. ÉS/VAGY gráfok

1

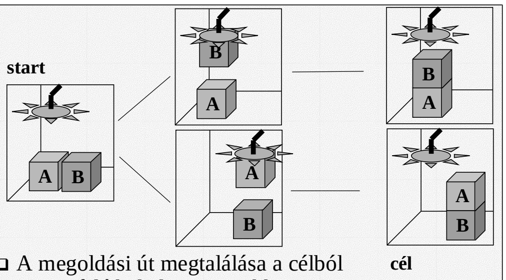
## 1. Visszafelé haladó keresés

- Ha a problématér a cél felől nézve egyszerűbb (kevesebb alternatívát mutat), mint a start felől nézve, akkor érdemes visszafelé haladó keresést alkalmazni.




2

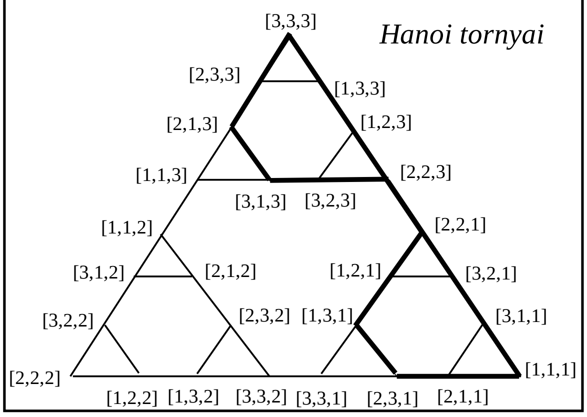
## *Kocka világ probléma*




- A megoldási út megtalálása a célból a start felé haladva gyorsabb.

3

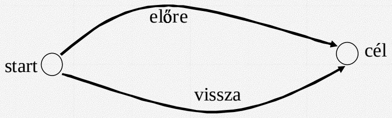
## *Hanoi tornyai*




4

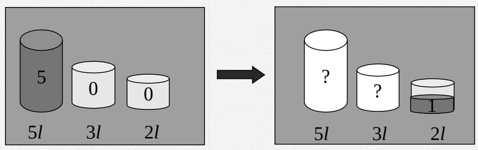
## *Két irányú keresés*

- Akkor alkalmazzuk, amikor se az előre, se a visszafelé haladó keresés nem hatékony, de egyszerre, két irányból indítva egy-egy keresést, hamarabb, olcsóbb megoldást kaphatunk.
- Nem hatékony, ha a két keresés elkerüli egymást.




5

## *Miért nem oldja meg a kancsó-problémát a visszafelé haladó keresés?*




- Kiindulási célállapotot kiválasztása nem egyszerű: Nem elérhető célállapot például a (*2,2,1*).
- A visszafelé haladó kereséssel talált (*4,0,1*)→(*5,0,0*) út nem értelmezhető a feladat megoldásaként.

6

## *Visszafelé haladó keresés feltételei*

- A műveletek invertálhatóak legyenek (legalábbis a visszafelé haladó keresés által alkalmazottak).
- Konkrét célállapotot kell választani. (Ettől a talált megoldás költsége is függ.)
- Hogyan határozható meg egy adott állapothoz az a megelőző állapot, amelyikből az adott állapot egy művelettel előállítható?

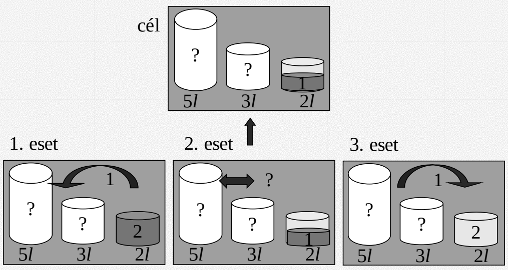
7

## 2. Probléma redukció

- Két kérdésre keressük a választ:
  - Van-e olyan művelet, amellyel elérhető egy éppen vizsgált állapot?
  - Melyik az a megelőző állapot, amelyikből a kiválasztott művelet a jelenlegi állapotba vezet?

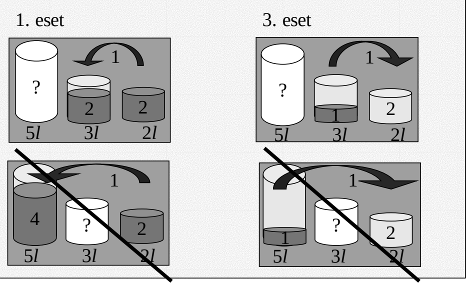
8

## *Kancsó-probléma redukálása*



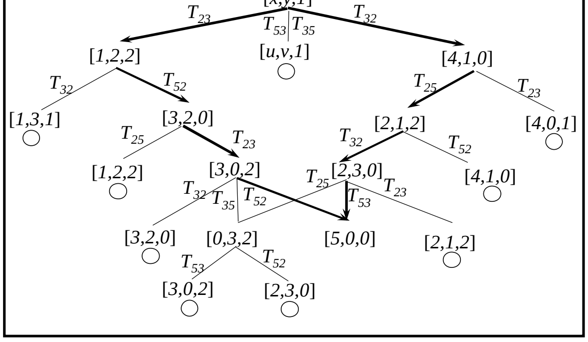

9

## *Megelőző állapotok kiszámolása*

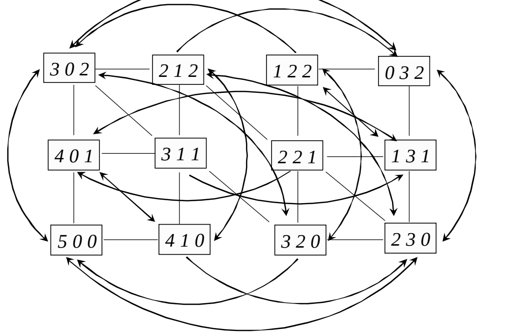



10

## *Kancsók redukciós gráfja*




## *Kancsók állapotgráfja*



## *Redukciós reprezentáció*

- A reprezentációhoz meg kell adnunk a feladat
  - az állapottér-reprezentációját,
  - majd minden művelethez definiálunk egy redukciós műveletet, amely egy állapothoz azokat a megelőző állapotokat rendeli, amelyekből a rögzített művelet az aktuális állapotba vezet.
    - *M* művelethez tartozó redukciós művelet:
      $B_M \subseteq$ *állapot* $\times$ *állapot* és
      $b \in B_M(a)$ ha $M(b)=a$

13

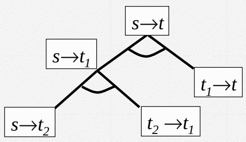
## *Megjegyzés*

- A redukció során eljuthatunk „érdektelen” illetve „hamis” állapot-leírásokhoz.
- Gráfreprezentáció készíthető: ebben kell utat keresni.
- A talált út visszafelé olvasva adja ki a megoldást.
- Kereső rendszer építhető.

14

## *A redukció kétfelé bont egy problémát*

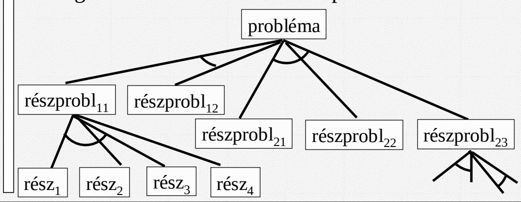
- A redukció során a megoldandó feladatot mindig két részre: egy nyilvánvalóan megoldható és egy további redukálást igénylő részfeladatra bontottuk.




15

## 3. Dekompozíció

- Ennek általánosítása a dekompozíció: egy feladatot több részfeladatra bontunk, majd azokat tovább részletezzük, amíg nyilvánvalóan megoldható feladatokat nem kapunk.

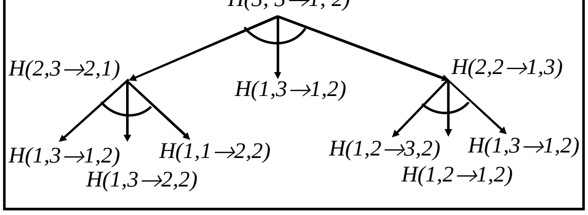



16

## *Hanoi tornyai probléma megoldása dekompozícióval*

*H(n, i→j, k) helyett*

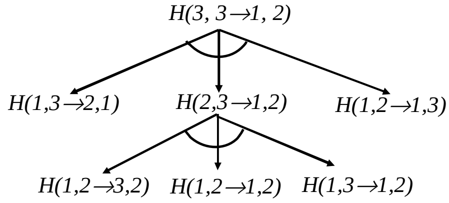
*H(n-1, i→k, j) H(1, i→j, k) H(n-1, k→j, i)*



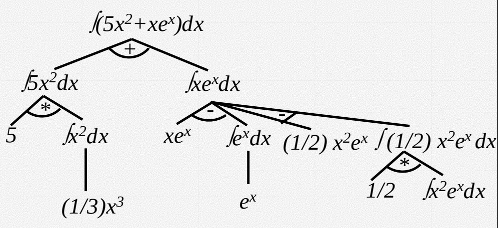

## *Hanoi tornyai probléma megoldása hibás dekompozícióval*

*H(n, i→j, k) helyett*

*H(1, i→k, j) H(n-1, i→j, k) H(1, k→j, i)*



## Integrálszámítás






## Dekompozíciós-reprezentáció

- A reprezentációhoz meg kell adnunk:
  - a feladat részproblémáinak általános leírását,
  - az eredeti problémát,
  - az egyszerű problémákat, amelyekről könnyen eldönthető, hogy megoldhatók-e vagy sem, és
  - a dekomponáló műveleteket:
    - $D$: *probléma* $\to$ *probléma*$^{+}$ és
      $D(p)=<p_1, \dots , p_n>$



## A dekompozíciós-reprezentáció nehéz

- Dekomponáló műveleteket nagyon nehéz megtalálni.
  - Nem minden feladat dekomponálható.
  - Nem biztos, hogy minden dekomponálást észrevettünk.
  - Hamis dekomponáló műveletek.
- Az egyszerű probléma felismerése sem egyértelmű
  $\int sin(x)e^x dx = \dots = sin(x)e^x - cos(x)e^x - \int sin(x)e^x dx$
- A megoldás kiolvasása nem nyilvánvaló.



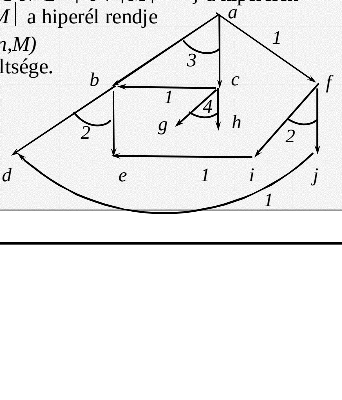
## Gráfreprezentáció

- A feladatnak nem egy közönséges irányított gráf felel meg, hanem egy úgynevezett ÉS/VAGY gráf.
- A megoldást sem egy közönséges irányított út szimbolizálja, hanem egy speciális részgráf: megoldás-gráf
  - A megoldás-gráf egyértelmű haladási irányt jelöl a startcsúcsból célcsúcsokba.
  - A megoldás-gráf a megoldási út ÉS/VAGY gráfokra átvitt általánosítása
- A probléma megoldása a megoldás-gráfból olvasásható ki.



## 4. ÉS/VAGY gráfok

- Az $R=(N,A)$ élsúlyozott irányított hipergráf, ahol az
  - $N$ a csúcsok halmaza,
  - $A \subseteq \{ (n,M) \in N \times 2^N \mid 0 \neq |M| < \infty \}$ a hiperélek halmaza, $|M|$ a hiperél rendje
  - $c(n,M)$ az $(n,M)$ hiperél költsége.
- δ tulajdonság
- σ tulajdonság

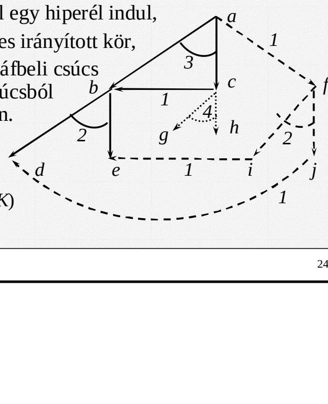





## Az *n* csúcsból az *M* csúcshalmazba vezető irányított hiperút fogalma

- egy véges részgráf ($n^{\alpha} \to M$), amelyben
  - $M$ csúcsaiból nem indul hiperél,
  - a többi csúcsból egy hiperél indul,
  - nincs közönséges irányított kör,
  - bármelyi részgráfbeli csúcs elérhető az $n$ csúcsból közönséges úton.
- Hiperút hossza
- Hiperút költsége:
  - $c^{\alpha}(n,M) := \sum_{(k,K)\in\alpha} c(k,K)$

$a \to \{d,e\}$



## Hiperút bejárása

- Az $n \rightarrow M$ hiperút egy bejárásán a hiperút csúcsaiból képzett halmazoknak olyan felsorolását értjük, amelyben
  - Az első az $\{n\}$ halmaz, a második az $n$ csúcsból kivezető (egyetlen) hiperél utódhalmaza követ.
  - Általában egy $C$ halmaz után a $C$-$\{k\} \cup K$ halmaz következik, ha van olyan $(k,K)$ hiperél az $n \rightarrow M$ hiperúton, hogy $k \in C$ és $k \notin M$.
  - A bejárás utolsó csúcshalmaza az $M$ halmaz.

25

## Megjegyzés

- A bejárás a hiperút összes hiperélét tartalmazó hiperél-sorozat, amelyben ugyanaz a hiperél többször is szerepelhet.
- Az $n \rightarrow M$ hiperút $(k,K)$ hiperéle legfeljebb annyiszor szerepel egy bejárás során, amennyi közönséges út vezet a hiperútban az $n$ csúcsból $k$ csúcshoz. Ezt a számot a $k$ csúcs adott hiperútbeli multiplicitásának nevezzük.
- Egy bejárás véges hosszú.
- Egy hiperútnak véges sok különböző bejárása lehet.

26

## Dekompozíciós gráfreprezentáció

- Egy dekompozíciós-reprezentációhoz tartozó $(R,s,T)$ gráfreprezentációban
  - az $R=(N,A,c)$ egy olyan ÉS/VAGY gráfban (dekompozíciós gráfban), ahol
  - $N$ a részprobémákat,
  - $A$ a dekomponáló műveleteket,
  - $c$ azok költségeit szimbolizálják,
  - $s$ az eredeti problémát,
  - $T$ az egyszerű problémákat jelöli.

27

## Megjegyzés

- A probléma megoldását egy $s \rightarrow M \subseteq T$ hiperút, az úgynevezett megoldás-gráf megtalálása jelenti. Az eredeti probléma megoldása ebből a megoldás-gráfból nyerhető ki.
- A megoldás költsége többnyire nem függ a megoldás-gráf költségétől, ezért nem cél, az optimális megoldás-gráf előállítása.

28

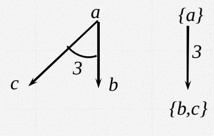
## Keresés ÉS/VAGY gráfban

- Az útkereső algoritmusainkat közönséges gráfokra fogalmaztuk meg. De mivel minden hiperút helyettesíthető valamelyik – közönséges útként ábrázolható – bejárásával, ezért a tanult keresések könnyen adaptálhatók ÉS/VAGY gráfokra.
- Egy ÉS/VAGY gráfon folyó keresés a startcsúcsból kivezető hiperutak (köztük a megoldás-gráfok) bejárásai között folyik.
- Elméletben el készíthetjük ezeket a bejárásokat tartalmazó közönséges gráfot.

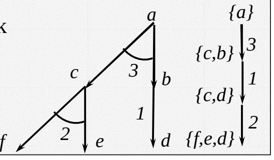
29

## Bejárások ábrázolása közönséges utakkal

Egyetlen hiperélből álló útnak egyetlen bejárása van

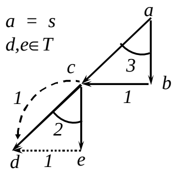



Több hiperélből álló útnak egy bejárása



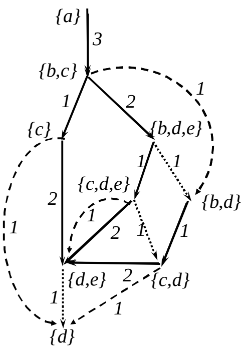

30

## ÉS/VAGY gráf átalakítása

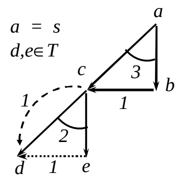



A startból induló hiperutak bejárásait közönséges útként rajzoljuk fel.

Túl sok utat kapunk, ráadásul vannak köztük hamis bejárások is.

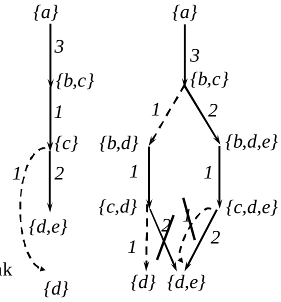



---




Egy hiperútnak elég egy bejárását megadni.

Számunkra csak az $s \to M \subseteq T$ hiperutak a fontosak, ezért a célcsúcsokat nem választjuk ki.




Ha egy csúcshalmazból olyan $k$ csúcsot választottunk, amelyhez a halmazhoz vezető úton korábban már a $(k,K)$ hiperélt illesztettük, akkor itt is csak ezt a hiperélt használhatjuk fel.

---

## Átalakító algoritmus

- Az ÉS/VAGY gráfot az $\{s\}$-sel címkézett startcsúcsból indulva (szélességi bejárással : SOR) építjük fel.
- Kivesszük a SOR-ból a következő $C$ halmazt, $C$-ből pedig egy $k$ nem célcsúcsot (Ha nincs ilyen $k$ csúcs akkor veszünk egy új halmazt a SOR-ból. Ha SOR üres, akkor terminál.)
  - Ha $k$ csúcsot a $C$-hez vezető úton eddig még nem választottuk ki, akkor az összes $(k,K) \in A$ hiperélre előállítunk a $C$-hoz egy $C$-$\{k\} \cup K$-val címkézett utódot,
  - egyébként a $k$ csúcsra a korábban használt $(k,K)$ hiperéllel állítunk elő egyetlen utódot.
  - élköltség: $c(C, C\text{-}\{k\} \cup K) = c(k,K)$
- Célcsúcsok a $\{\ldots t_i \ldots\}$ $(t_i \in T)$ alakú halmazzal címkézettek.

33

---

## Tétel

1. Az átalakítással nyert közönséges gráfok minden megoldási útja egy $s \to M \subseteq T$ hiperútnak egy bejárását írja le.
2. Az átalakítás minden $s \to M \subseteq T$ hiperút valamelyik bejárásához véges lépésben megfeleltet egy közönséges megoldási utat.

- Megjegyzés:
  - Az átalakított gráf egy δ-gráf.
  - Az átalakítást beépítik a keresésekbe.
  - Az átalakítást költségtartóvá is lehet tenni

34

---

## Bizonyítás

1. Az átalakítással nyert gráfban minden $\{s\}$-ből induló útra (annak hossza szerinti teljes indukcióval) beláthatjuk, hogy ő egy startból induló hiperútnak egyik bejárása. A csak $T$ elemeiből álló halmazba futó közönséges út (azaz egy megoldási út) biztos egy $s \to M \subseteq T$ hiperút bejárása.
2. Egy $s \to M \subseteq T$ hiperút minden bejárása véges, és az $\{s\}$-sel kezdődik, amit az algoritmus az első lépés előtt elő is állít. Ha van olyan bejárás, amelynek első $i$ halmaza már előállt, utolsó csúcsa ($C$ halmaz) pedig a SOR-ban található, akkor a $C$ véges lépés múlva kikerül a SOR-ból. Ha $C$ nem csupa célcsúcsból áll (egyébként készen vagyunk), akkor vizsgált bejárás $i+1$-edik halmaza ($C$-$\{k\} \cup K$) bekerül a sorba.

35

---

## Visszalépéses keresés ÉS/VAGY gráfokon

```
Recursive procedure VL2(út) return megoldás
1.    C ← vége(út)
2.    if csupacél(C) then return(nil) endif
3.    if hossza(út) ≥ korlát then return(hiba) endif
4.    if K ∈ maradék(út) then return(hiba) endif
5a.   k ← kivesz-egy-nemcélcsúcsot(C)
5b.   hiperélek ← kivezető-hiperélek(k)
6.    while not üres(élek) loop
7.          (k,K) ← kivesz(hiperélek)
8.          megoldás ← VL2(hozzáfűz(C-{k}∪K, út))
9.          if megoldás ≠ hiba then
                  return(hozzáfűz((C, C-{k}∪K), megoldás))
            endif
10.   endloop
11.   return(hiba)
end
```