Kereső eljárások

- Cél: egy adott problémára a szóbajöhető lehetőségek közül egy adott kritériumrendszernek eleget tévő megoldás megtalálása.

- A keresést az ismeretfeldolgozási eljárások közé soroljuk, mely révén a szemléltetett ismeretet aktívvá tesszük, tudást nyerünk ki.

- Gyakran használják ismeretfeldolgozási eljárásként a keresést következtetőgráfokon következmény előállításra.

dr.Dudás László

Kereső eljárások ..

- A kereséssel történő problémamegoldás lépései
  1. A probléma beazonosítása kereséssel megoldható problémaként
  2. A problémára vonatkozó ismeretek reprezentálása, szemléltetése
  3. A kereső eljárás alkalmazása.

1. Kereséssel megoldható problémák jellemzői

   Egy operátorkészlettel bejárható állapotok mindegyikén értelmezhető egy kritériumfüggvény. A probléma megoldása megfeleltethető a kritériumfüggvény adott értékének, értékhalmazának, vagy extrémumának.

2. A problémára vonatkozó ismeretek reprezentálása

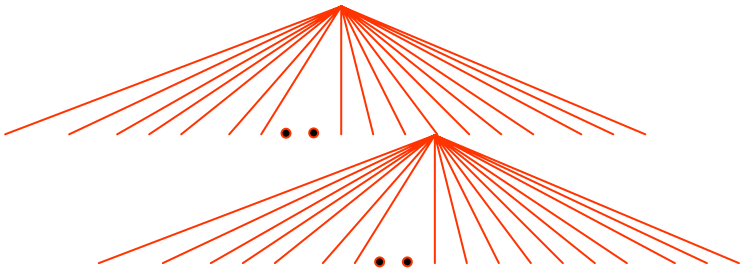
   Lásd a tudásszemléltetés elvárt jellemzőit, Patrick Winston szerint.

   Az ismeretek reprezentálása erősen kihat az ismeretfeldolgozás, jelen esetben a keresés hatékonyságára. Ennek megvilágítására tekintsük a következő példát:

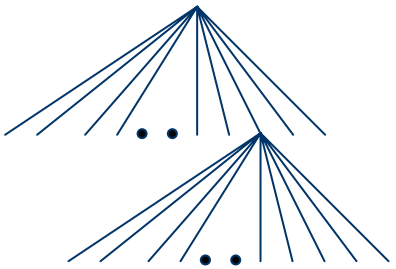
Kereső eljárások ..

- Egy 20*20 méretű négyzetrácsos játéktéren kell az amőba játékban meghatározni azt a helyet, amely a lépésen következő szempontjából jó lépésnek minősül, hat féllépésre előre tekintve.

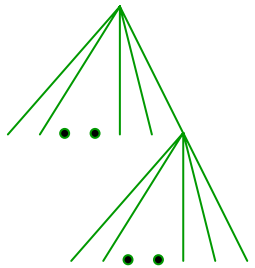
- **Lehetséges reprezentációk**



1. Az összes üres mezőt vizsgáljuk: a lehetőségek száma a játék első lépéseinél százas nagyságrendű, nagyon nagy elágazási tényezőjű fagráffal szemléltethető a hat féllépéses részjátékra.



2. Csak az előző lépés környezetében lévő üres helyeket vizsgáljuk egy 9*9-es mezőben, mivel az utolsónak lerakott jel csak ebben a részben alkothat nyerő ötöst. Az elágazási tényező maximum 80-as.



3. Csak az előző lépésre illesztett csillag alakzat üres helyeire végezzük a vizsgálatot, mivel ezek a helyek vannak közvetlen kapcsolatban az utolsónak lerakott jellel. Az elágazási tényező maximum 32-es.

Megj.: a reprezentációk gyengülésétől sokkal nagyobb előnyt jelent a kivitelezhető keresések hatékonyságnövekedése.
A reprezentációnak összhangban kell lennie a kereső eljárással.

Kereső eljárások ..

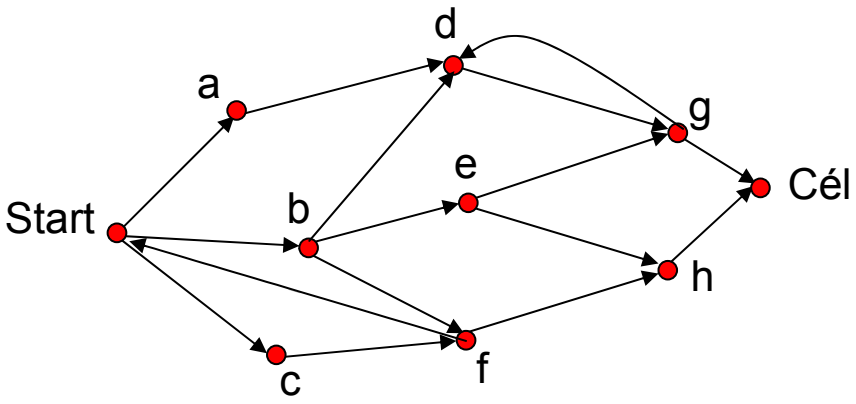
3. A kereső eljárás alkalmazása

A kereső eljárás alkalmazása a feladat megoldására az eljárások jellemzőinek ismeretében történhet. Ilyen közös jellemzők, melyek a megoldandó konkrét probléma függvényei:

- **Teljesség:** ha létezik megoldás, azt az eljárás meg is adja.
- **Optimalitás**: a Start-Cél útvonal egyben költségoptimális is.
- **Időigény**: az algoritmus lépésszámára adott felső korlát.
- **Tárigény**: a megoldás megtalálásához felhasznált memória méretére vonatkozó felső korlát.

A kereső eljárások összetevői

- **Állapottér**, reprezentálása az **állapotok**nak megfelelő csomópontokkal és az állapotok közötti mozgást jelentő **operátorok**nak megfelelő irányított élekkel.
- Kiinduló állapot: **Start**
- A kritériumnak megfelelő állapot/ állapotok: **Cél**.
- **Út, útvonal**: egy állapotból egy másik állapotba átvivő operátorsorozat, ill. érintett csomópontok rendezett listája.
- A keresés **leállási feltétele** előírhatja a kritérium teljesítését, vagy adott tűrésen belüli közelítését. Az utóbbi esetben *kvázioptimális* megoldásról beszélünk, mely gyakorlatilag jó és kompromisszumot jelent a keresés időigénye/költsége és a megoldás minősége között.



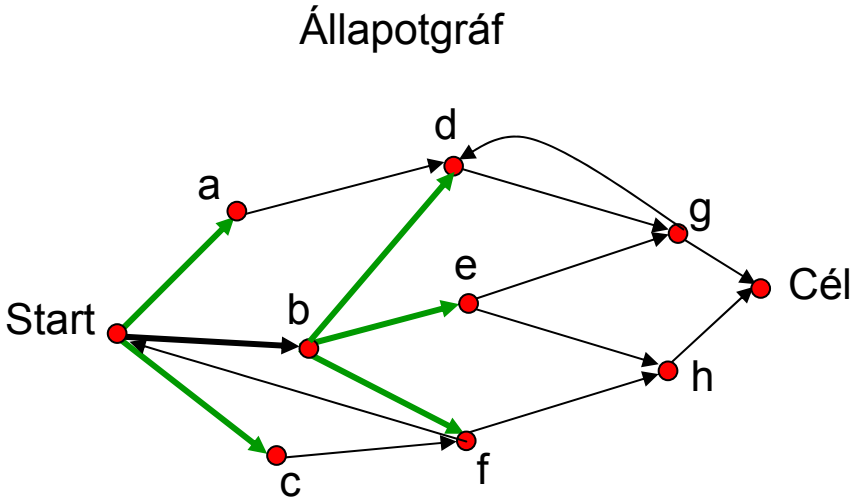
dr.Dudás László

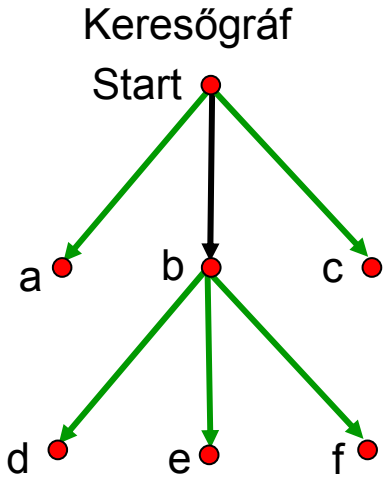
A kereső eljárások összetevői ..

- **Az operátor költsége**: az operátorok a feladat valós tartalmától függő költséggel rendelkezhetnek: pl. legrövidebb út keresése városok között – az operátor költsége az útszakasz hossza. De lehet az operátor költsége érdektelen is, például bűvös kocka kitekerésénél csak a célállapot megtalálása, illetve az odavezető út fontos.
- **Az út költsége**: az úton alkalmazott operátorok költségének összege.
- **A keresés költsége**: a keresés idő- és tárigényéhez kapcsolódó költség.
- **A keresés teljes költsége**: az út költsége + a keresés költsége. Pl. városban történő útkeresés benzinköltsége: az útszakaszokon is fogy a benzin és az elágazásoknál az útválasztási döntés meghozatala ideje alatt is jár a motor.

A kereső eljárások összetevői ..

- **A keresőgráf:** egy fagráf, melynek csúcsa a Start állapot, valamelyik levele pedig a Cél állapot, amennyiben létezik megoldás.
- **Kiterjesztés**: egy állapot kiterjesztése alatt az állapotból egyetlen lépéssel elérhető állapotok feltérképezését, azokba való betekintést, de még bele nem lépést értünk. Megfelel a keresőgráf egy adott pillanatnyi levélállapotától egy szinttel lejjebb lévő állapotok keresőgráfba való megrajzolásának.

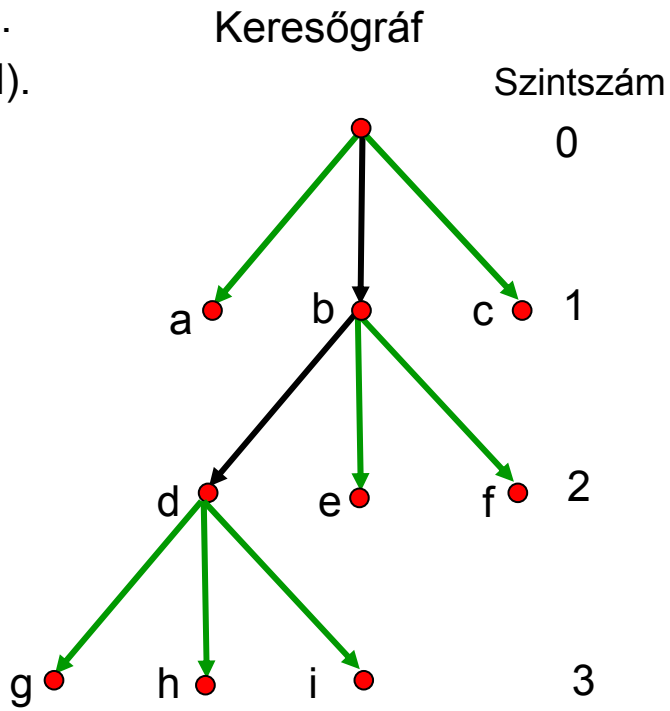




dr.Dudás László

A keresőgráf jellemzői

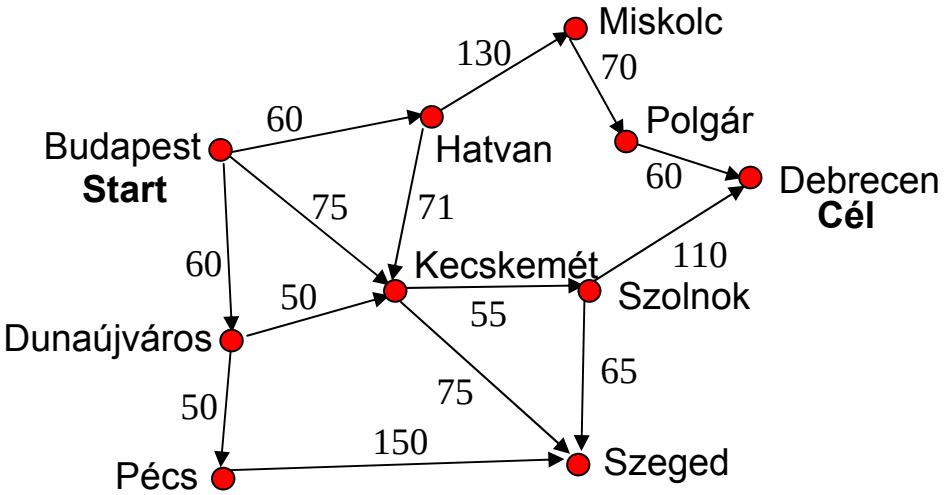
- **Elágazási tényező (branching factor, b):** egy adott csomópontból megtett kiterjesztés ágainak száma.
- **Átlagos elágazási tényező**: több csomópontra, leggyakrabban a teljes keresőgráfra értelmezett jellemző, a figyelembe vett csomópontok elágazási tényezőinek összege osztva a figyelembe vett csomópontok számával.
- A keresőgráf **mélysége** (m): a gráf legmélyebb szintjének száma, a Start a 0. szint.
- A **megoldás mélysége** (depth, d).
- Esetleges **mélységi korlát** (limit, l).
- A keresés **frontja**: az összes, kiterjesztéssel feltárt, de még bele nem lépett csomópont, azaz a keresési fa levelei.
- Az n állapotba vezető út költsége g(n)
- Az n állapotból a Cél elérésének becsült költsége h(n)
- Az n állapoton átvezető út (becsült) költsége f(n). (A*)



dr.Dudás László

Példák kereséssel megoldható feladatokra

- **Útkeresés két város között**



- Probléma: melyik a legrövidebb útvonal Budapestről Debrecenbe?
- Start állapot: Budapesten vagyunk.
- Állapotok: valamelyik városban vagyunk.
- Leállási feltétel: a legrövidebb úton odajutva a Célban, Debrecenben vagyunk már? (A leállási feltétel egyben optimális utat is garantál.)
- Operátorok: utazás a szomszédos városok között.
- Költség: a megtalált Start-Cél útvonal hossza.

dr.Dudás László

Példák kereséssel megoldható feladatokra ..

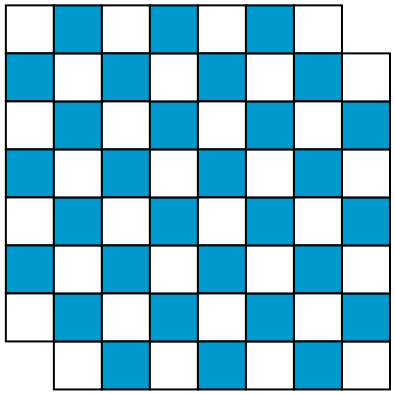
- **Bűvös kocka kitekerése**

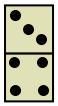
- Probléma: milyen forgatási sorozattal juthatunk vissza az alapállapotba az adott tarka állapotból?
- Start állapot: egy adott tarka állapot.
- Állapotok: élközép és csúcselemek helyzete. (A lapközepek helyben forognak.)
- Leállási feltétel: mind a hat oldalon csak egy-egy szín van.
- Operátorok: a hat lap valamelyikének forgatása +90, vagy –90 fokkal.
- Költség: nincs.

dr.Dudás László

Példák kereséssel megoldható feladatokra ..

- **Lefedési probléma**



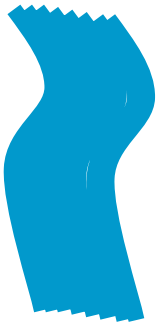


- Probléma: hogyan lehet 31 dominóval lefedni egy, az átellenső sarkain csonka sakktáblát?
- Start állapot: az üres sakktábla.
- Állapotok: mindig eggyel több dominó valamilyen elrendezésben a sakktáblán.
- Leállási feltétel: mind a 62 mező le van fedve, vagy a lefedés lehetetlen voltának megállapítása.
- Operátorok: egy újabb dominó elhelyezése a sakktáblán.
- Költség: nincs.

dr.Dudás László

Példák kereséssel megoldható feladatokra ..

- **A farkas, a kecske és a káposzta**



F, K, Á, C

- Probléma: hogyan tudja az öreg halász a csónakjával átvinni a farkast, a kecskét és a káposztát úgy a folyó jobb partjáról a másikra, hogy a parton magára hagyott farkas ne egye meg a kecskét és a kecske se egye meg a káposztát, ha a csónakjában egyszerre csak egy dolgot tud szállítani?
- Start állapot: Jobb parton a csónak, a farkas, a kecske és a káposzta.
- Állapotok: az előző állapottól egy, vagy két dologban eltérő jobbparti állapot + a jobbparton van a csónak, vagy sem.
- Leállási feltétel: a jobb parton nincs semmi.
- Operátorok: <, >, <F, <K, <Á, F>, K>, Á>. Korlátozás: Ellentett operátorokat nem alkalmazhatunk egymás után.
- Költség: operátorműveletek száma.

Általános kereső eljárás

- A kereső eljárások lényegi lépéseit tartalmazza az általános eljárás, amelytől a különböző eljárások csak kis részben térnek el.

1. A keresési front legyen egyenlő a Start állapottal.
2. Ha a front üres, akkor nincs megoldás, vége.
   Egyébként legyen n a front egyik állapota.
3. Ha n kielégíti a Cél-kritériumot, akkor add vissza a hozzá vezető útvonallal együtt, vége.
4. Egyébként a front n állapota helyére vedd fel a kiterjesztésével adódó új állapotokat és jegyezd fel a hozzájuk vezető útvonalakat.
   Ismételd a 2. ponttól.

- Az algoritmus szabad kezet adó pontja a 2., amelyben arról döntünk, hogy melyik állapot szomszédai irányába terjesszük ki a keresést. Ezen döntést végezhetjük gépiesen, a neminformált eljárások esetében, vagy a keresési feladatra vonatkozó információkra támaszkodva, az informált, vagy másnéven heurisztikus kereső eljárások esetében.

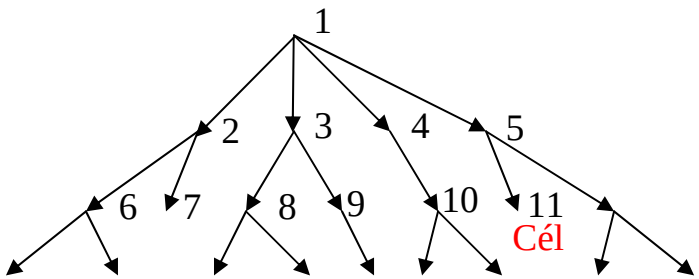
dr.Dudás László

Neminformált kereső eljárások (vak keresés)

- Nincs információnk az aktuális állapot és a cél távolságára (költség, vagy lépésszám). Az eljárás csak arra képes, hogy észrevegye, ha a Cél állapotban van.
- Nem hatékonyak
- Általánosan használhatók.
- Típusok:
  - Szélességben először keresés (breadth-first search)
  - Elágaztatás és ugrálás (branch and bound), vagy másnéven egyenletes költségű keresés (uniform cost search)
  - Mélységben először keresés (depth-first search)
  - Mélységben először keresés mélységi korláttal (depth limited search)
  - Iteratív mélyítés (iterative depth-first search)

Szélességben először keresés

- Egy adott mélységi szint csomópontjainak mindegyikét kiterjeszti, mielőtt a következő mélységi szintre lépne.



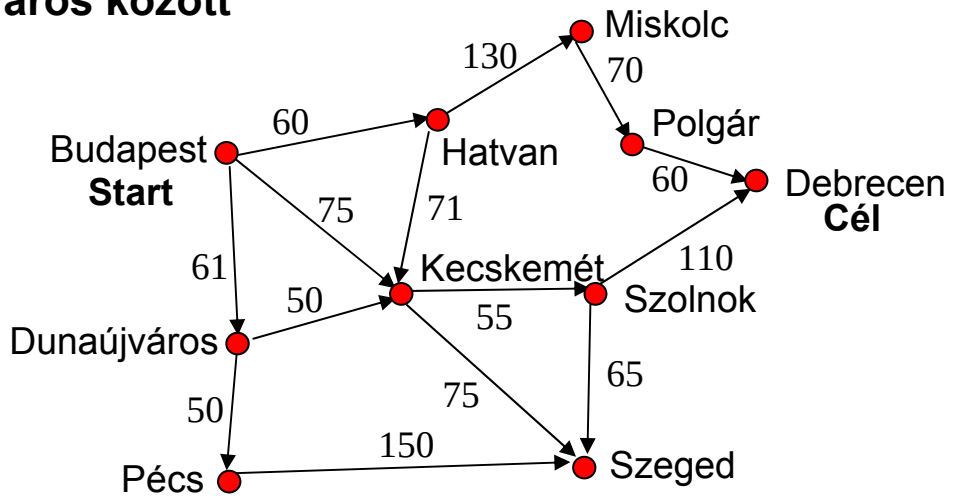
- Az általános kereső eljárás a következőképpen módosul:

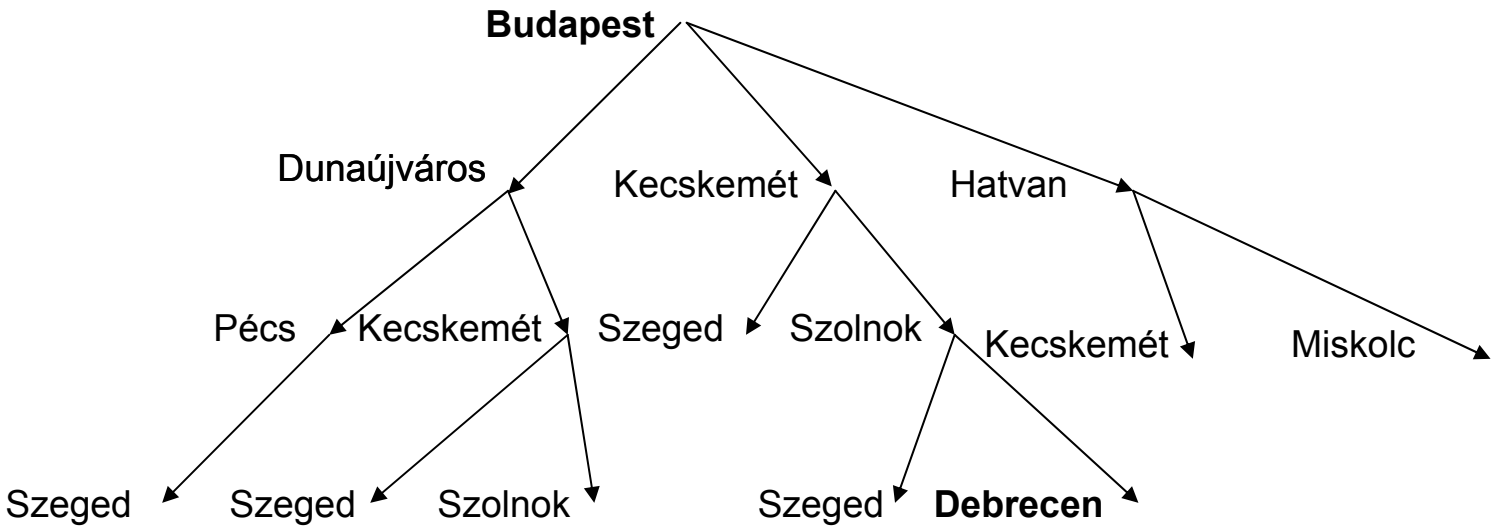
  2. Legyen a front első állapota az n választott állapot.
  4. A kiterjesztéssel kapott új állapotokat csatold a frontlista végéhez.

- Az eljárás
  - Teljes
  - Optimális (amennyiben az útszakaszok egyforma költségűek)
  - Időigénye b^d, (meredeken nő a mélységgel)
  - Tárigénye b^d, (meredeken nő a mélységgel).

Példa a szélességben először keresésre

- **Útkeresés két város között**





dr.Dudás László

Elágaztatás és ugrálás

- A front azon állapotába lép, amelyikhez a legkisebb költségű út vezet a Start-tól.
- Ha az útszakaszok költsége egyforma, akkor a szélességben először keresésre vezet.

- Az általános kereső eljárás a következőképpen módosul:

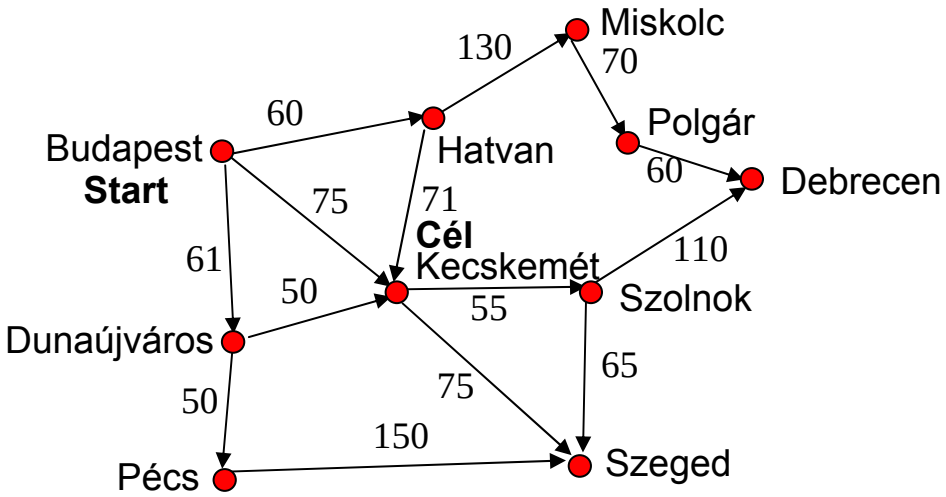
  2. Legyen a front első állapota az n választott állapot.
  4. A kiterjesztéssel kapott új állapotokat add a frontlistához, majd rendezd az állapotokat növekvő költség szerint.

- Az algoritmus leáll, ha Cél-állapotba léptünk, azaz a Front összes állapotába nagyobb költségű út vezet.

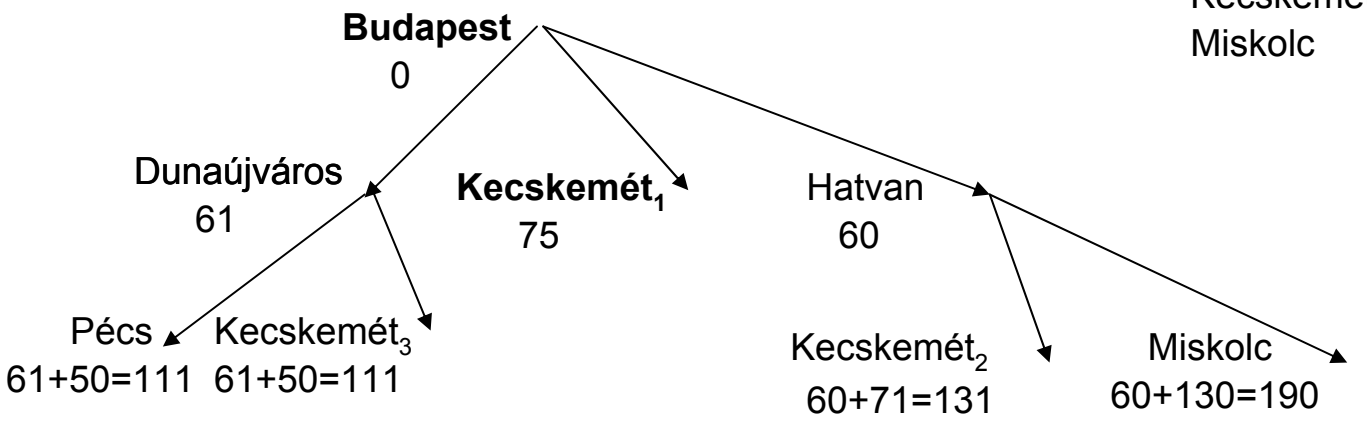
- Az eljárás
  - Teljes
  - Optimális
  - Időigénye $\approx b^d$, (meredeken nő a mélységgel)
  - Tárigénye $\approx b^d$, (meredeken nő a mélységgel).

Példa az *elágaztatás és ugrálás* keresésre

- **Útkeresés két város között**



Sorsz.	Front	Választás
0.	Budapest	Budapest
1.	Dunaújváros	
	Kecskemét$_1$	
	Hatvan	Hatvan
3.	Dunaújváros	
	Kecskemét$_1$	
	Kecskemét$_2$	
	Miskolc	Dunaújv.
4.	Pécs	
	Kecskemét$_3$	
	Kecskemét$_1$	
	Kecskemét$_2$	
	Miskolc	Kecskemét$_1$



dr.Dudás László

Mélységben először keresés

- A keresési fát balra lefelé tartva járja be. Ha nem tud lefelé menni tovább, visszalép a legalsó elágazásig és a következő ágat választja. Visszalépéskor a sikertelen ág állapotait ejti. (Lásd Prolog működését.)

- Az általános kereső eljárás a következőképpen módosul:

  2. Legyen a front első állapota az n választott állapot.
  4. A kiterjesztéssel kapott új állapotokat add a frontlista elejéhez.

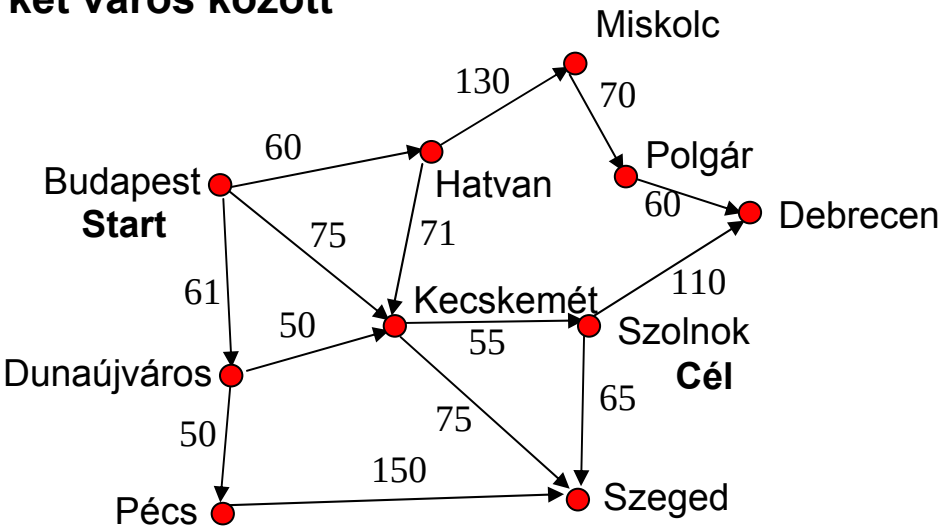
- Az eljárás
  - Teljes, ha nincs végtelen, vagy (a memóriának) túl mély ág.
  - Nem optimális
  - Időigénye $\approx b^m$, (meredeken nő a mélységgel)
  - Tárigénye b*m, (nagyon kis igény!).

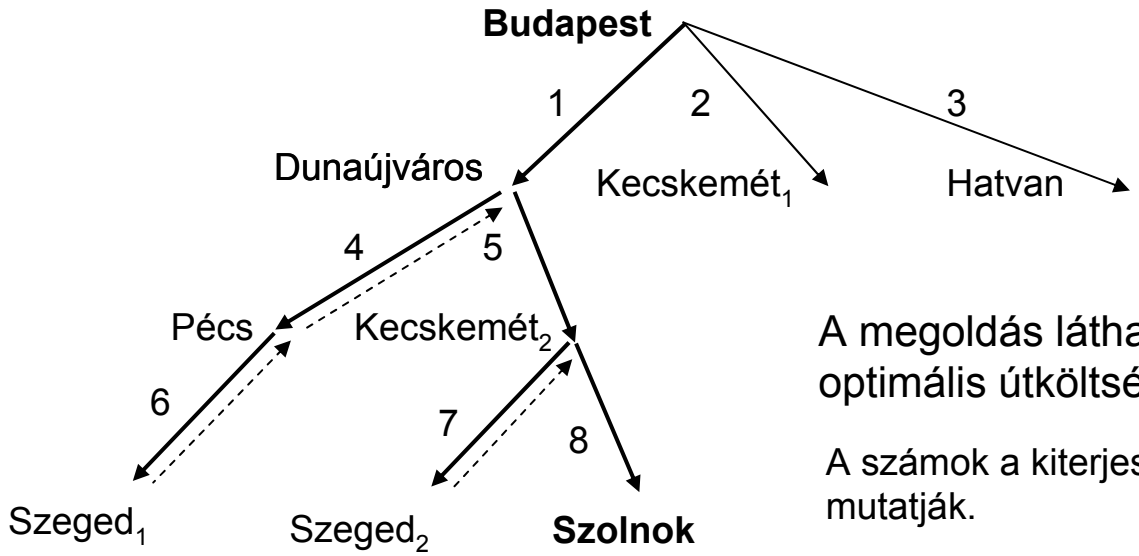
- Csak egyetlen állapotba vezető út állapotait és az út leágazásain található állapotokat kell tárolnia.

dr.Dudás László

Példa a mélységben először keresésre

- **Útkeresés két város között**





A megoldás láthatóan nem optimális útköltségű.

A számok a kiterjesztés sorrendjét mutatják.

Mélységben először keresés mélységi korláttal

- Cél: a mélységben először keresés végtelen, vagy gyakorlatilag végtelen mély ágainak veszélyét elkerülni egy jól megbecsült mélységi korláttal. (Átmenet az informált kereséshez, amennyiben a korlát mélységének megadásához problémafüggő információt is használ.)

- Működése azonos a mélységben először algoritmuséval, de mintha az l mélységi korlát alatti állapotok nem is léteznének.

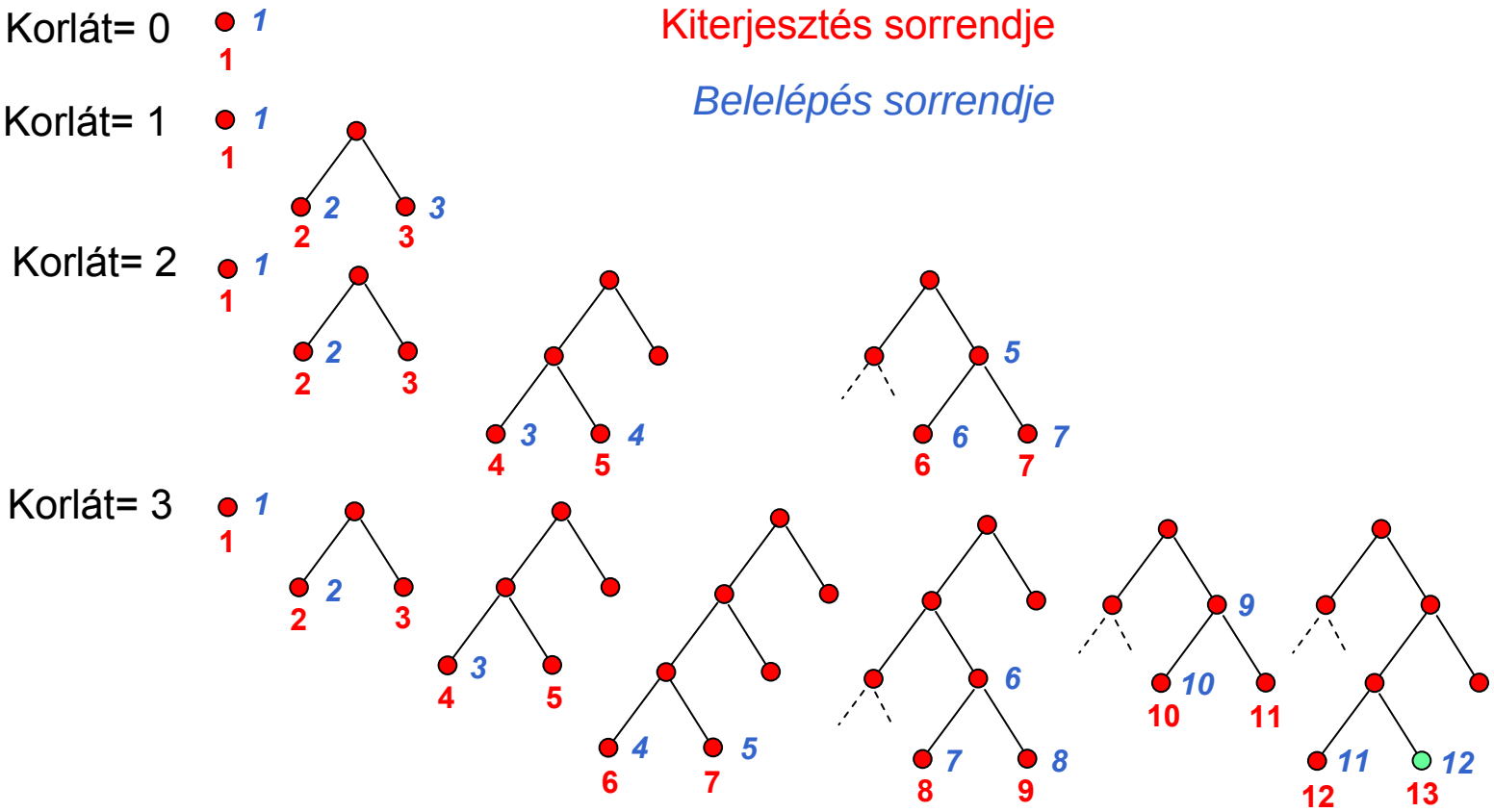
- Az eljárás
  - Teljes, ha l nagyobb, vagy egyenlő, mint a Cél-állapot mélysége.
  - Nem optimális
  - Időigénye b^l, (meredeken nő a mélységi korláttal)
  - Tárigénye b*l, (nagyon kis igény!).

- Konkrét példa lehet: benzinkút keresése egy városban, ha már csak adott liternyi benzinünk van.

dr.Dudás László

Iteratív mélyítés

- Cél: a korlátmegadás problémájának elkerülése azáltal, hogy nulla korlátmélységről indulva egy-egy szinttel növeli a korlátmélységet. Minden egyes korlátmélységnél elvégez egy mélységben először keresést. A korlát mélyítését addig folytatja, amíg megoldást nem talál.



dr.Dudás László

Az iteratív mélyítés algoritmusa

1. Induló korlátmélység l= 0
2. A keresési front legyen egyenlő a Start állapottal.
3. Ha a front üres, akkor növeld egy szinttel a mélységi korlátot, majd ismételd 2.-től.
   Egyébként legyen n a front első állapota.
4. Ha n kielégíti a Cél-kritériumot, akkor add vissza a hozzá vezető útvonallal együtt, vége.
5. Töröld az n állapotot.
   Ha n mélysége kisebb volt mint l, akkor vedd fel a kiterjesztésével adódó új állapotokat a frontlista elejére és jegyezd fel a hozzájuk vezető útvonalakat.
   Ismételd a 3. ponttól.

- Az iteratív mélyítés előnyei
  - Örökli a mélységben először keresés alacsony memóriaigényét.
  - Rendelkezik a szélességben először keresés teljességével.
  - A mélységben először kereséssel szemben nem tárja fel a fa megoldástól mélyebb részeit.
  - Időkorlátos feladatokhoz előnyös, mert a keresés bármikor megszakítható.

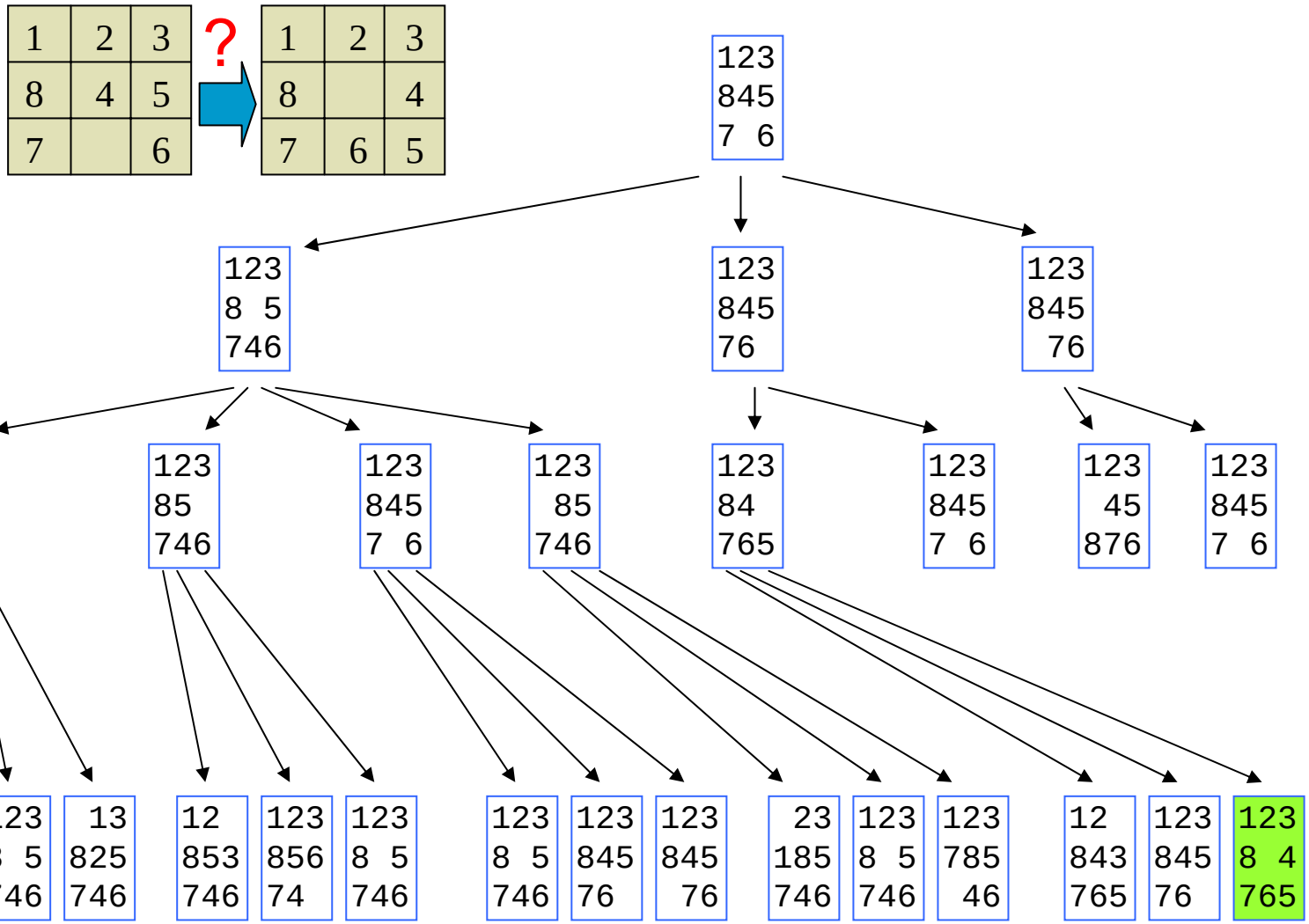
dr.Dudás László

Az iteratív mélyítés tulajdonságai

- Teljes
- Optimális (költség a lépések száma)
- **Időigénye** az újrakezdések ellenére **nagyságrendileg nem változik** a mélységben először b^d értékéhez képest. Erről meggyőz a következő példa: b= 10, d= 5 paraméterek mellett egy egyszerű keresőgráf csomópontjainak száma: 1+10+100+1000+10000+100000 = 111111.
  A legmélyebb szint csomópontjait egyszer hozza létre a kiterjesztés, az eggyel magasabban lévőket kétszer és így tovább. Emiatt a csomópontok száma az iteratív mélyítésnél:
  6+50+400+3000+20000+100000 = 123456,
  azaz nagyságrendileg ugyanaz.
  A b növelésével egyre kisebb a számítási többlet, de b=2 esetén is csak kétszeres.
- Memóriaigénye b*d, nagyon kedvező!

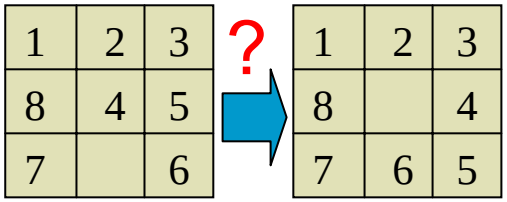
dr.Dudás László

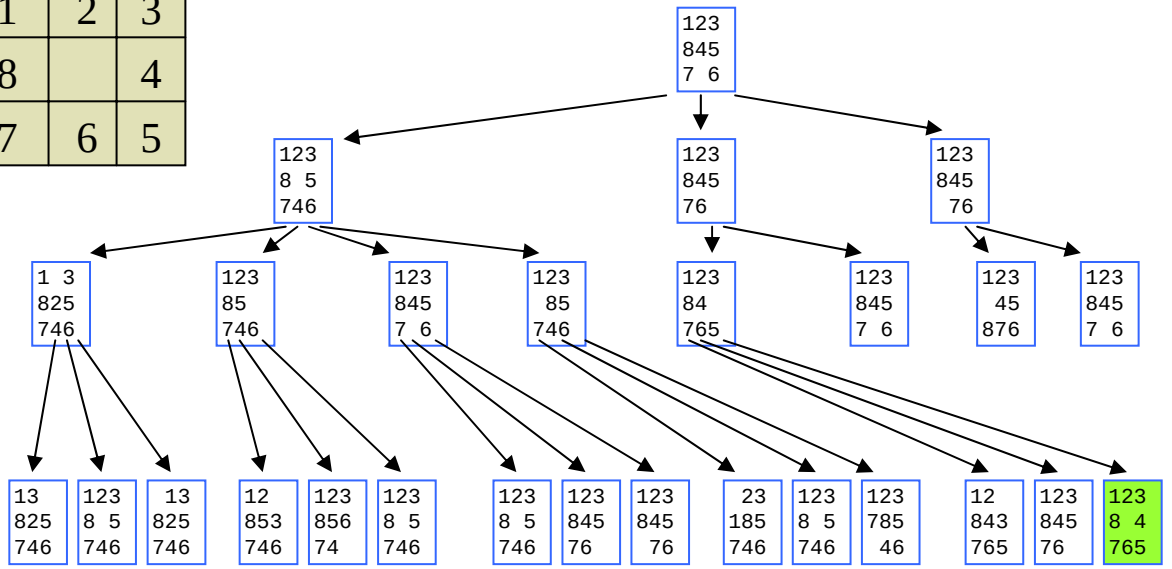
- **Példa az iteratív mélyítésre: 9-ből 8 játék**



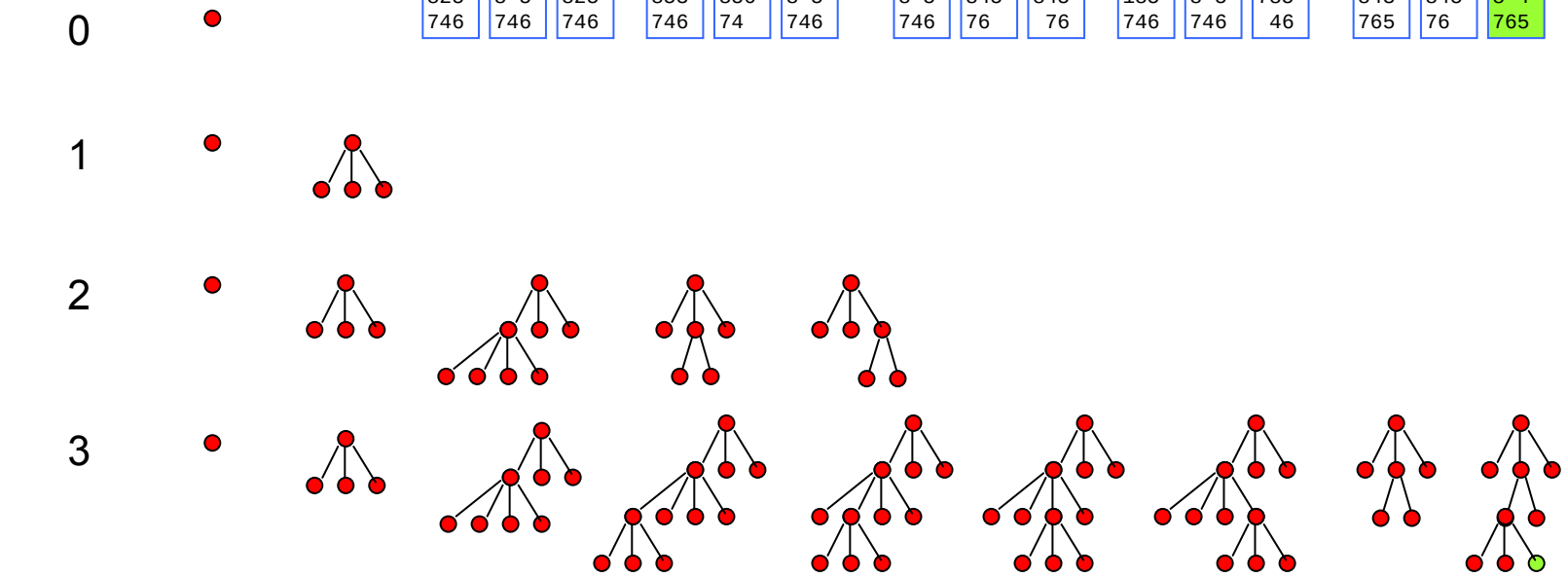
dr.Dudás László

- **Példa az iteratív mélyítésre: 9-ből 8 játék ..**





Korlát szint



dr.Dudás László

Informált kereső eljárások (heurisztikus keresés)

- A front állapotai közül belelépésre azt választjuk, amelyre vonatkozóan a konkrét megoldandó feladatra vonatkozó **információk valamilyen előnyt ígérnek**. Ilyen előny lehet: a célhoz legközelebbinek tűnik; a rajta való áthaladás a legkisebb Start-Cél költséggel kecsegtet; a legnagyobb lépést teszi a Cél irányában, stb.

- Az informált állapotválasztással a keresés hatékonysága nagyban fokozható.

- Az informált kereső eljárásokat gyakran **heurisztikus** kereső eljárásoknak nevezzük, mert a problématerületre vonatkozó tapasztalatra építenek. A heurisztika általában nagyban növeli a hatékonyságot, de tévedhet is.

- A heurisztikát a **h(n) heurisztikus kiértékelő függvény** segítségével számszerűsítjük, számítjuk az n állapot esetében.

- Heurisztikus kiértékelő függvényként gyakran alkalmazzák az n állapotból a Cél elérésének költségére vonatkozó becslést. A jó függvény nem becsüli túl a költséget és a Cél állapotra 0 értéket ad.

Informált kereső eljárások fajtái

- Globális információra támaszkodó eljárások
  - Legjobbat először keresés (Best first search)
  - A és A* keresés (A search, A* search)
  - Iteratív A* algoritmus (IDA* search)

- Lokális információra támaszkodó eljárások
  - Hegymászó keresés (Hill climbing)
  - Szimulált lehűtés (Simulated annealing)
  - Tabu keresés (Tabu search)

- A globális információ az állapottér bármely két pontja között származtatható, leggyakrabban az n állapot és a Cél-állapot között számítjuk és az n állapothoz rendeljük.
  Globális információ felhasználásával megtalálhatjuk a globális optimumot, az optimális költségű utat is.

- A lokális információ az n állapot és közvetlen szomszédai között számítható. Önmagában lokális információ alkalmazásával csak lokális extrémumot garantálnak az eljárások, gyakran azonban annak közelítésével is megelégszünk.

Legjobbat először keresés

- A legjobbat először keresés egy **heurisztikus kiértékelő függvénnyel** becsli az n állapotból **a Cél elérésének költségét** a front összes állapotára és abba az állapotba lép, amelyikre ez a költség a legkisebb. Az eljárás végződhet lokális optimumpontban is.

- Patrick **Winston** az eljárást hegymászókkal magyarázta: Egy hegymászókból álló csapat indul a hegycsúcs meghódítására. A csapat tagjai elágazásoknál szétválnak és rádiótelefonnal tartják a kapcsolatot, hogy mindig az a csapat mászhasson tovább, amely számára a csúcs elérése a legkönnyebbnek tűnik. Ha valamelyik csapat elakad, egy másik mászik tovább.

- Az általános kereső eljárás a következőképpen módosul:

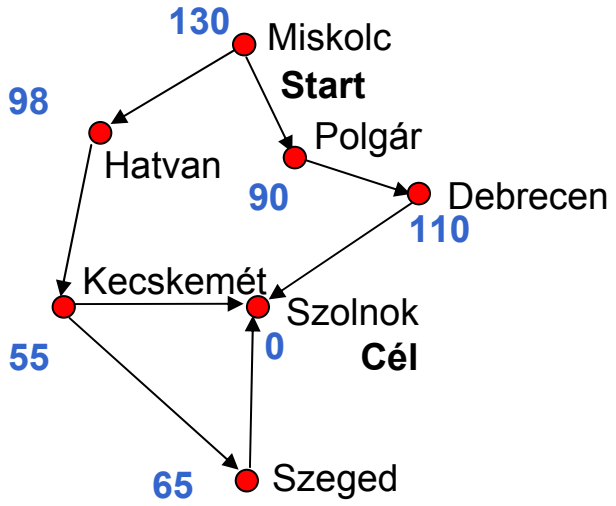
  2. … Legyen a front célhoz legközelebbinek becsült állapota az n választott állapot.

- Az eljárás
  - Teljes
  - Nem optimális
  - Időigénye $\approx b^m$, (meredeken nő a mélységgel)
  - Tárigénye $\approx b^m$, (meredeken nő a mélységgel).

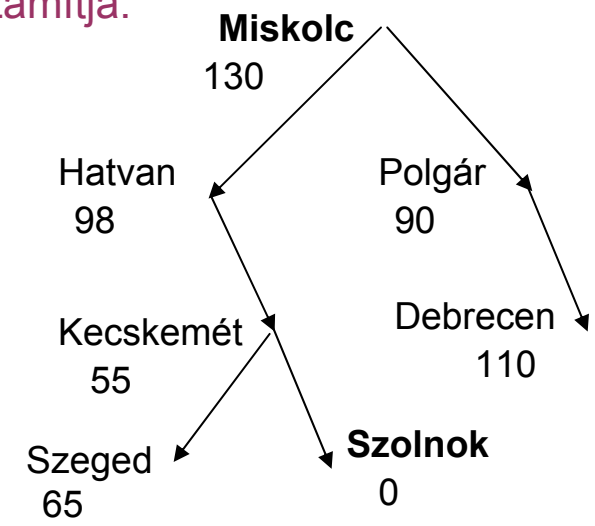
dr.Dudás László

Példa a *legjobbat először* keresésre

- **Útkeresés két város között**



A heurisztikus kiértékelő függvény a légvonalbeli távolságot számítja.



Sorsz.	Front	Választás
0.	Miskolc	Miskolc
1.	Hatvan Polgár	Polgár
3.	Hatvan Debrecen	Hatvan.
4.	Kecskemét Debrecen	Kecskemét
5.	Szeged Szolnok Debrecen	Szolnok

Az A és A* keresés

- Az A keresés számítja a front összes állapotára a g(n) függvénnyel a Start-n távolságokat, valamint ugyanezen állapotokra egy h(n) **heurisztikus kiértékelő függvénnyel** becsli az n állapotból **a Cél elérésének költségét**, majd képezi ezek összegét: f(n)= g(n) + h(n).
  Abba az állapotba lép, amelyikre ez az áthaladó útvonalhossz becsült költsége a legkisebb.
  Az eljárás végződhet lokális optimumpontban is.

- Bizonyítható, hogy amennyiben a h(n) költségbecslő függvény nem becsüli túl a Cél elérésének költségét egyik állapotra sem, akkor az eljárás által adott útvonal egyben optimális is. Az ilyen függvény jele: f ’(n) , az eljárás neve ekkor A*.

- Az általános kereső eljárás a következőképpen módosul:

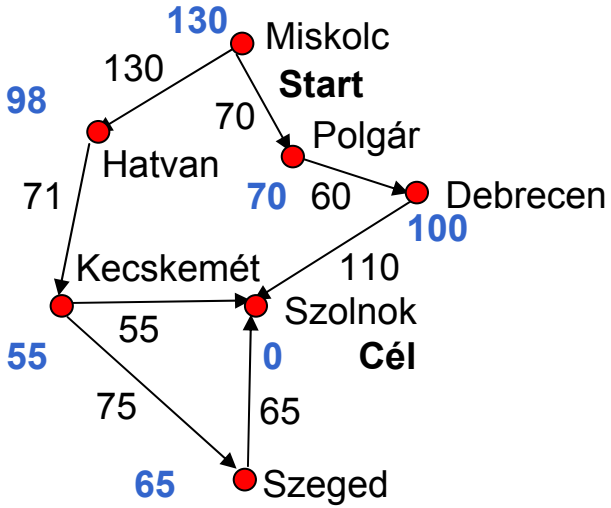
  2. … Legyen a front állapotai közül kiválasztva az, amelynek legkisebb az f ‘(n) függvényértéke.

- Az eljárás
  - Teljes
  - Optimális
  - Időigénye nagy
  - Tárigénye nagy, gyakran túlcsordítja a memóriát.

dr.Dudás László

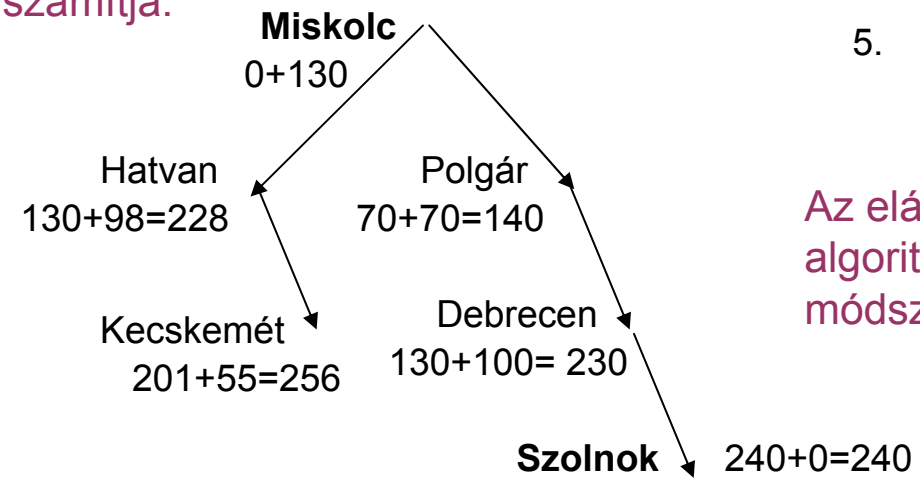
Példa az A* keresésre

- **Útkeresés két város között**



Sorsz.	Front	Választás
0.	Miskolc	Miskolc
1.	Hatvan Polgár	Polgár
3.	Hatvan Debrecen	Hatvan
4.	Kecskemét Debrecen	Debrecen
5.	Kecskemét Szolnok	Szolnok

A heurisztikus kiértékelő függvény a légvonalbeli távolságot számítja.



Az elágaztatás és ugrálás algoritmus és a legjobbat először módszer egyesítése.

dr.Dudás László

Az iteratív A* algoritmus

- Cél: az A* keresés memóriaigényének csökkentése.
- Ötlet: mivel az A* memóriaigénye a kiterjesztett csomópontokkal és a feltárt gráf éleinek számával arányos, csökkentsük a figyelembe vett csomópontok és élek számát azáltal, hogy kezdetben csak a legalacsonyabb áthaladási költségű csomópontokat vegyük figyelembe. Azt, hogy mekkora költség alatt vesszük figyelembe a csomópontokat, egy **jósági küszöb** változóval adjuk meg. Ha ezzel nem adódik útvonal, akkor emeljük a jósági küszöb értékét és egy újabb A* keresést hajtunk végre a sűrűbb gráfon.
- Az eljárás hatékonysága érdekében az új, magasabb küszöböt az előző ciklusban meghatározott értékre emeljük, nem pedig egyenletes lépcsőkkel.
- Az eljárás arra emlékeztet, mint amikor egy város úthálózatáról készült fényképen az egyes útszakaszok az előhívótálban egymás után jelennek meg a képen. Az olcsóbb becsült Start-Cél költségű utak jelennek meg előbb.

dr.Dudás László

Az iteratív A* algoritmus lépései

1. A keresési front legyen egyenlő a Start állapottal.
   Jóságkorlát legyen egyenlő az f(Start) értékkel.
   Újkorlát legyen egyenlő végtelennel.
2. Ha a front üres, akkor nincs megoldás, vége.
   Egyébként legyen n a front egyik állapota.
3. Ha n kielégíti a Cél-kritériumot, akkor add vissza a hozzá vezető útvonallal együtt, vége.
4. Egyébként
   ha f(n)<=Jóságkorlát, akkor a front n állapota helyére vedd fel a kiterjesztésével adódó új állapotokat és jegyezd fel a hozzájuk vezető útvonalakat,
   egyébként legyen Újkorlát új értéke Újkorlát és az f(n) minimuma.
5. Ha a front üres és Újkorlát= végtelennel, nincs megoldás, vége.
6. Jóságkorlát vegye fel az Újkorlát értékét. Újkorlát legyen végtelen.
   Front legyen egyenlő a Start állapottal.
   Ismételd a 2. ponttól.

- Az eljárás
  - Teljes
  - Optimális
  - Időigénye nagy
  - Tárigénye lényegesen kevesebb az A* eljárásétól.

A hegymászó keresés

- Lokális információt használ: csak a szomszédos állapotokat nézi.
- A legjobb szomszédot választja és gyorsan feljut egy közeli lokális csúcspontra.
- Lépései:

1. Legyen n a Start állapot.
2. Ha n Cél-állapot, add vissza a hozzávezető úttal együtt. Vége.
3. Egyébként állítsd elő n kiterjesztett állapotait. Legyen n új értéke ezek közül a legjobb.
   Ismételd 2.-től.

- Az eljárás:
  - Teljessége: amennyiben lokális maximum megfelel, akkor azt megtalálja
  - Nem optimális
  - Időigénye: nagyon alacsony
  - Tárigénye: nagyon kicsi, mert nem tárol keresőgráfot.
- Probléma: sikere erősen függ a felület alakjától.

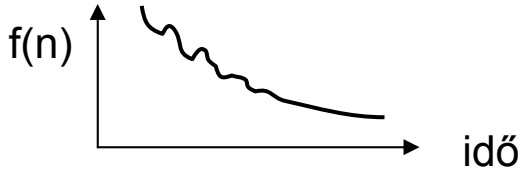
Szimulált lehűtés

- Ötlet: a fémek lehűlés során rácsszerkezetbe dermednek, melynek rendezettsége és finomsága a lehűtés sebességétől függ, lassú lehűtés magasabb rendezettséget eredményez.
- Cél: a lokális csúcsokról való elkerülés azáltal, hogy megadott valószínűséggel lefelé haladó, rontó lépések is lehetnek. A rontó lépések valószínűsége a hőmérséklet csökkenésével csökken.
- Kellően lassú lehűlés esetén megtalálja a globális maximumot.
- Nagyméretű feladatokra is hatékony.
- Közelítő eredményt ad, melyet a leállási feltétel teljesülésekor ér el. A leállási feltétel általában azt írja elő, hogy az utolsó néhány hőmérsékleten a célfüggvény értéke megadott kis értéken belül maradjon.

A szimulált lehűtés algoritmusa, minimalizáló

1. Legyen n a kezdő állapot,
   T_0 a kezdő hőmérséklet,
   L_0 a kezdő folyamathossz,
   k=0 a ciklusváltozó kezdőértéke.
2. Ismételd L_k-szor:
   Legyen r az n egyik szomszédja,
   ha f(r) <= f(n) akkor n vegye fel r értékét,
   egyébként ha exp((f(n) – f(r))/T_k) > Véletlenszám akkor n vegye fel r értékét.
3. Inkrementáld a k-t,
   számítsd T_k új értékét k és T_k-1 függvényében,
   számítsd az L_k új értékét k és L_k-1 függvényében.
4. Ha a leállási feltétel nem teljesül, menj 2-re.
5. Add vissza az n megoldást, vége.

- A folyamathossz nem csökkenő, a hőmérséklet szigorúan csökkenő sorozat.
- *Véletlenszám* egy 0-1 intervallumbeli egyenletes eloszlású véletlenszám.
- Az f(n) függvényérték globális minimumhoz való konvergálását egyre kisebb romlási szakaszok jellemzik.



A tabu keresés

- Ötlet:minimalizáló lokális algoritmusnak, ha megengedjük, hogy megpróbáljon kimászni egy lokális gödörből, előfordulhat, hogy a hegygerinc elérése előtt újra a kedvezőbb, kisebb függvényértékek irányába fordul és újra lemegy a völgybe. Akadályozzuk meg ezt egy rövidtávú memóriaként működő listával, mely tárolja a tiltott, tabunak minősülő visszautakat.

- Az algoritmus:

1. Legyen a kezdeti állapot n,
   n_eddigi_legjobb vegye fel n értékét,
   k ciklusváltozó legyen 1,
   a Tabulista legyen üres.
2. Ha a megállási kritérium teljesül add vissza az n_eddigi_legjobb állapotot, vége.
3. Egyébként n_legjobb_a_kiterjesztésben legyen egyenlő a legkisebb f() értéket adó állapottal a kiterjesztett elemek és a Tabulista elemeinek különbségét adó állapotok közül.
4. Frissítsd a Tabulistá-t az nlegjobb_a_kiterjesztésben elemmel,
   n vegye fel n_legjobb_a_kiterjesztésben értékét.
5. Ha n_legjobb_a_kiterjesztésben kisebb, mint n_eddigi_legjobb, akkor
   n_eddigi_legjobb vegye fel n_legjobb_a_kiterjesztésben értékét.
6. Inkrementáld a k ciklusváltozót és ismételd a 2. ponttól.

dr.Dudás László

Tabu keresés, megjegyzések

- Megállási kritérium lehet:
  - A k ciklusváltozó elér egy korlátot.
  - n_eddigi_legjobb adott számú ismétlésben nem javult.
  - A kiterjesztett elemek és a Tabulista különbséghalmaza üres.
- n_eddigi_legjobb az adott k iterációban vizsgált összes állapot közül a legjobb.
- n_legjobb_a_kiterjesztésben a k. ciklusban kiterjesztett szomszédok és a Tabulista elemeinek különbséghalmazában a legjobb f() értéket adó állapot.
- Az eljárás matematikai háttérrel nem bír, ennek ellenére nagyméretű feladatokban bizonyította erejét.
- A Tabulista elemszáma rögzített, a frissítés a legrégebben bekerült állapotot írja felül.
- Valós alkalmazásokban a Tabulista lépéssorozatok fordított listájának bizonyos jellemzőit tárolja a memóriatakarékosság érdekében.

dr.Dudás László