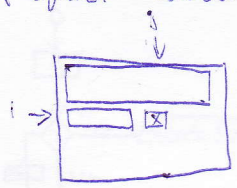
sorfolytonos kitöltés



$\forall i \forall j \; (a_{ij} \neq 0 \Rightarrow \forall \ell_1, \ell_2 (\ell_1 < i \vee (\ell_1 = i \wedge \ell_2 < j) \Rightarrow a_{\ell_1,\ell_2} \neq 0)$

Megoldóalgoritmusok

- adatbázis: a feltárt gráf távolt része
- műveletek: hogyan manipuláljuk az adatbázist
- vezérlő: hogyan alkalmazzuk a műveleteket az adatbázison

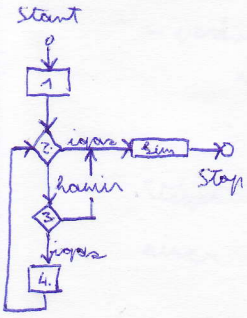
Tulajdonságok

- célállapot / megoldás áll elő
- irányítatlan (rendszematikus) / irányított (heurisztikus)
- teljes-e az algoritmus
- alkalmazható-e köröket v. hurkokat tartalmazó gráfon?

Nem módosítható megoldóalgoritmusok

Próba hiba módszer

- adatbázis: az aktuális állapotot tárolja
- műveletek: az állapottér-reprezentáció operátorai
- vezérlő:



1) az adatbázis inicializálása
az aktuális állapot legyen a kezdőállapot
2) célállapot-e az aktuális állapot
- igen: sikeresen véget, célállapot állt elő
- nem: folyt. 3.
3) tárolt-e alkalmazható operátorok az aktuális állapotra?
- nem: sikertelenül befejeződik
- igen: folyt. 4.
4) Egy alkalmazható operátor választása, az alkalmazása során előállt állapot legyen az aktuális állapot
5) folyt. 2.

Hegymászó módszer

- vezérlő:

4) $h : A \to \mathbb{R}$

$$h(a) = \begin{cases} 0, & \text{ha } a \in C \\ \infty, & \text{ha nem érhető el új állapot} \\ ?, & \text{egyébként} \end{cases}$$

A h fv. alapján választ egy operátort.

n-királynő heurisztika

$(a_1, a_2, a_3, a_4) \in A$

$\sum_{i=1}^{4} sgn(a_i)$

$h((a_1,a_2,a_3,a_4)) = 4 - \sum_{i=1}^{4} sgn(a_i)$

3-korsó probléma

$(a_1, a_2, a_3) \in A$

$$h((a_1,a_2,a_3)) = \begin{cases} 0, & \text{ha } (a_1,a_2,a_3) \in C \\ 4 - \left(\sum_{i=1}^{3} sgn(a_i)\right) & \text{egyébként} \end{cases}$$

(nem üres korsók száma)

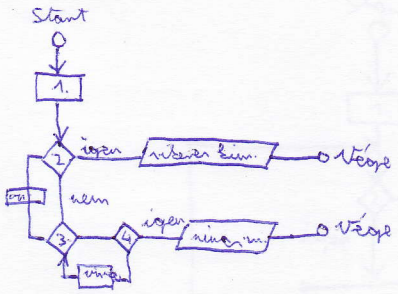
Backtrack algoritmusok

Alap backtrack

- adatbázis: aktuális út, aminek végén az aktuális áll, és mely a kezdőállapotból indul
  csúcsonként tároljuk:
  - állapot
  - szülő
  - előállító operátor
  - ki nem próbált, de alkalmazható, vagy sikertelenül alkalmazott operátorok

- műveletek
  - az állapottér-reprezentáció operátorai
    az akt. út hossza nő → az eredményt az akt. út végére fűzzük
  - visszalépés
    eltávolítjuk az akt. állapotot az út végéről

- vezérlő
  1) az adatbázis inicializálása
     $(kezdő, -, -, \{o \in O \mid kezdő \in dom(o)\})$
  2) Célállapot-e az aktuális állapot?
     - igen: sikeresen vége, megoldás áll elő
     - nem: folyt. 3.
  3) Vannak-e még ki nem próbált, de alkalmazható operátorok az aktuális állapotban?
     - igen: alkalmazunk egyet, folyt. 2.
     - nem: folyt. 4.
  4) Egyelemű-e az aktuális út (van-e hová visszalépni)?
     - igen: vége, nincs megoldása a problémának
     - nem: visszalépés: folyt. 3.



Véges, körmentes állapottér-gráf esetén garantáltan megoldást talál, vagy felismeri, ha nincs megoldás.

Útihossz-korlátos backtrack

- adatbázis
  - tárolja az aktuális út hosszát
  - operátor alkalmazása esetén +1
  - visszalépés esetén −1
- vezérlő:
  - új visszalépési feltétel: ha elértük az útihossz-korlátot, visszalépés

Véges gráfban véges útihossz esetén garantáltan befejezi a működést. Nem garantált a megoldás.

Körmentesített backtrack algoritmus

- vezérlő:
  - új visszalépési feltétel: ha az akt. állapot szerepel már az akt. úton, visszalépés

Véges gráfban garantáltan megoldást talál, vagy felismeri, ha nincs megoldás.

Íny és korlát algoritmus
optimális megoldás keresésére

- adatbázis:
  - költség / útihossz-korláttal egészül ki
- vezérlő:
  - új visszalépési feltétel:
    - ha az aktuális út hossza elérte a korlátot, visszalépés következik
    - ha célállapot az akt. állapot, a megoldást tároljuk, a korlát = aktuális út hossza, visszalépés

Keresőfával megoldáskereső

- Adatbázis:
  - részfát tárol, gyökere a start csúcs
  - csúcsonként:
    - állapot
    - szülőcsúcs
    - előállító operátor
    - státusz: nyílt/zárt
    - ...

- műveletek:
  - kiterjesztés

    kizárólag nyílt státuszú csúcs terjeszthető ki, zárt lesz

    a kiterjesztett csúcsra alkalmazott operátor eredményeképpen új nyílt csúcsok állhatnak elő

- vezérlő:
  1: adatbázis inicializálása (a kezdőállapottal)

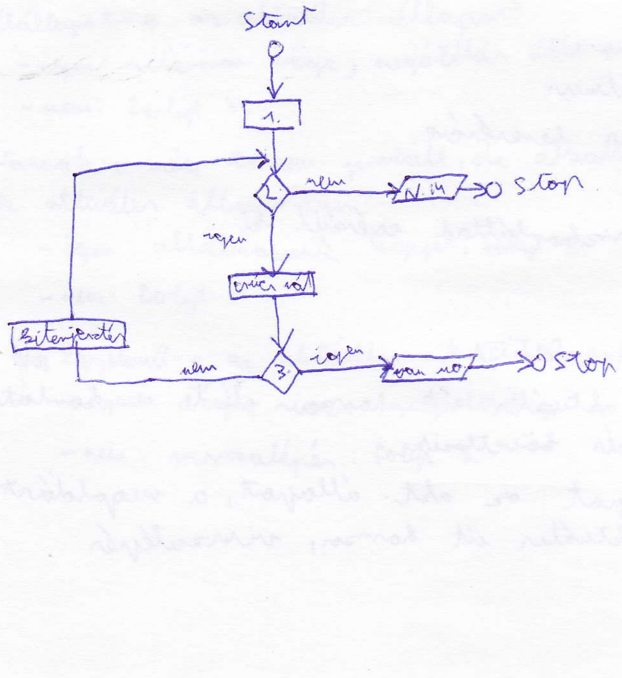
  (kezdő, –, –, nyílt, ...)

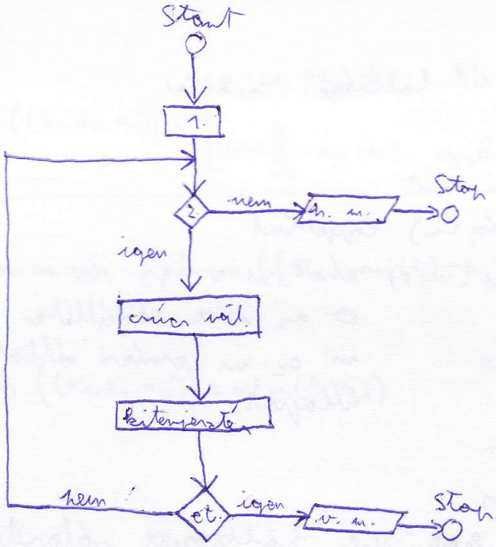
  2: vannak-e nyílt csúcsok
  - nem: vége, a problémának nincs megoldása
  - igen: válasszunk egy nyílt csúcsot, folyt. 3.

  3: Terminális csúcs-e, célállapot-e?
  - igen: sikeres vége, megoldás áll elő
  - nem: kiterjesztjük a csúcsot, folyt. 2.

Tesztelés előrehozása:

nem a kiterjesztésre kijelölt csúcsot vizsgáljuk, hanem a célfeltételvizsgálat a kiterjesztés eredményeként létrejövő csúcsokban megy végbe





Szélességi és mélységi keresők

Adatbázis: csúcsonként mélységi számot tárol

$$\text{mélység}(n) = \begin{cases} 0, & \text{ha } n = \text{start} \\ \text{mélység}(m)+1 & \text{egyébként} \end{cases}$$

(m: az n szülője)

Vezérlő: kiterjesztésre kiválasztás:

- szélességi kereső: a legkisebb mélységi számút választja
- mélységi kereső: a legnagyobb mélységi számút választja

Implementációs következmény: Ha a nyílt csúcsokat külön listában tároljuk, mélységszám szerint rendezett listában, a szélességi kereső a lista elejéről, a mélységi kereső a lista végéről választ, a rendezettség megtartása érdekében az új csúcsokat a lista végére vesszük fel

Ha a kiterjesztés során olyan csúcsot találunk, ami már szerepel az adatbázisban (kör vagy hurok), akkor a csúcsot eldobjuk.

Véges állapottéren garantáltan megoldást talál, vagy felismeri, hogy nincs megoldása.

Alkalmazható az előrehozott tesztelés technikája (hamarabb befejeződik a keresés)

Optimális keresés

adatbázis: csúcsonként az aktuális út költségét tárolja

költség: $A: o \to \mathbb{R}^+$

útköltség$(n) = \begin{cases} 0, \text{ ha } n = \text{start} \\ \text{útköltség}(m) \text{ egyébként} \\ + \text{költség}(m', o) \end{cases}$

ahol m a n szülője
o az n-et előállító operátor
m' az m csúcs által reprezentált állapot

vezérlő: kiterjesztésre kiválasztás
az optimális keresés a legkisebb költségűt választja

ha a kiterjesztés során olyan csúcs áll elő, ami az adatbázisban már szerepel, az állapotot nem tároljuk

Best-first

adatbázis: csúcsonként heurisztika

vezérlő: kiterjesztésre kiválasztás
minimális heurisztikájút választja

mivel a heurisztika nem kínál optimális megoldást, így az előzőhöz hasonló technikája alkalmazható.

A algoritmus alap A, A^*, monoton A

adatbázis:

csúcsonként:
- állapot
- szülő csúcs
- előállító operátor
- státusz: nyílt/zárt
- útköltség (nemnegatív valós szám)

útköltség(N) = útköltség(N) + heurisztika(N)

útköltség$(N) = \begin{cases} 0, \text{ ha } N = \text{start} \\ \text{útköltség}(m) + \text{költség}(m', o) \text{ egyébként} \end{cases}$

N szülője
az m által reprez. állapot
az N-et előállító operátor

$\text{útköltség}(N) = \begin{cases} \text{heurisztika}(N), \text{ ha } N = \text{start} \\ \text{útköltség}(M) + \text{heurisztika}(N) + \text{költség}(m, o) - \text{heurisztika} \end{cases}$

N szülője
az M által reprez. állapot
az N-et előállító operátor

3-korsó probléma

$$\text{heurisztika}((a_1,a_2,a_3)) = \begin{cases} 0, & \text{ha } a_1=4 \\ 4-\sum_{i=1}^{3} \operatorname{sgn}(a_i) & \text{egyébként} \end{cases}$$

$$\text{költség}((a_1,a_2,a_3), o_{ij}) = \min(\{a_i, \max(H_j) - a_j\})$$

$$\text{heurisztika}((a_1,a_2,a_3)) = \operatorname{abs}(4-a_1)$$

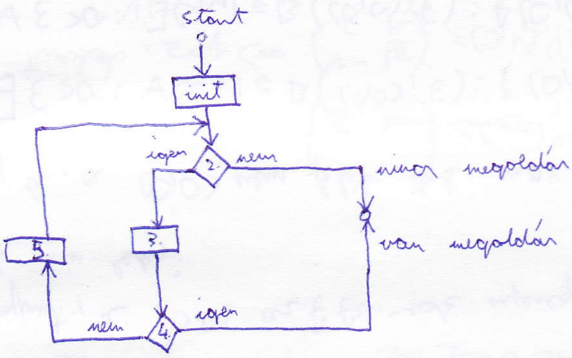
alsó becslés (A^*)

nem monoton

monotonitás:

$h(N) \leq h(M)$ $\quad m \Rightarrow n$

Vezérlő:



2. van-e nyílt csúcs?
3. kiterjesztésre kiválasztás
4. Célfeltétel teljesül?
5. kiterjesztés

Ha kiterjesztés során olyan állapot áll elő, ami már szerepel az adatbázisban, és az új út kisebb összköltségű

- ha a tárolt csúcs nyílt (nyílt csúcs probléma)

  frissítjük a nyílt csúcsot:
  - új szülő
  - új előállító operátor
  - új összköltség
- ha a tárolt csúcs zárt (zárt csúcs probléma):

  frissítjük és nyílttá tesszük