

Logikai programozás

5.

ISMÉTLÉS

Fibonacci sorozat:

a/

fibonacci(1,1).

fibonacci(2,1).

fibonacci(N,F) :- N > 1, N1 is N - 1, N2 is N - 2,
fibonacci(N1, F1), fibonacci(N2, F2),
F is F1 + F2.

ISMÉTLÉS

Fibonacci sorozat:

b/

fibonacci(N, I, Elozo, Utolso) :- I =< N ,

Mostani is Elozo + Utolso,

Ujl is I + 1,

fibonacci(N, Ujl, Utolso, Mostani).

fibonacci(____,____,____).

ISMÉTLÉS

Fibonacci sorozat:

c/

fibonacci(N,Elozo, Utolso) :- N > 0 ,

Mostani is Elozo + Utolso, UjN is N - 1,

fibonacci(UjN, Utolso, Mostani).

fibonacci(____,____,____).

Hogyan írathatjuk ki az első N Fibonacci számot?

Hogyan határozhatjuk meg az N-edik Fibonacci számot?

Hogyan indíthatjuk?

ISMÉTLÉS

Fibonacci sorozat:

b/ (1. javítás)

```
fibonacci(N,I,Elozo, Utolso) :- I =< N ,
                                writef('%w. elem: %w\n',[I,Utolso]),
                                Mostani is Elozo + Utolso, Ujl is I + 1,
                                fibonacci(N, Ujl, Utolso, Mostani).
fibonacci(_,_,_,_).
```

Indítás:

```
fibonacci(N) :- writef('Az első %w Fibonacci szám: \n', [N]),
                fibonacci(N, 1, 0, 1).
```

ISMÉTLÉS

Fibonacci sorozat:

b/ (2. javítás)

```
fibonacci(N,I,Elozo, Utolso, F) :- I < N ,
                                    Mostani is Elozo + Utolso, Ujl is I + 1,
                                    fibonacci(N, Ujl, Utolso, Mostani, F).
fibonacci(_,_,_,_ , F, F).
```

Indítás:

```
fibonacci(N,F) :- writef('A(z) %w. Fibonacci szám: \n', [N]),
                  fibonacci(N, 1, 0, 1,F).
```

Miért kell a plusz paraméter?

ISMÉTLÉS – NÉHÁNY LISTA PÉLDA

Listaelemek száma:

$\text{hossz}([], 0)$.

$\text{hossz}([_|T], H) :- \text{hossz}(T, \text{Thossz}), H \text{ is } \text{Thossz} + 1$.

$\text{hossz}(L, H) :- \text{hossz}(L, 0, H)$.

$\text{hossz}([_|T], \text{Eddig}, H) :- Uj \text{ is } \text{Eddig} + 1, \text{hossz}(T, Uj, H)$.

$\text{hossz}([], H, H)$.

ISMÉTLÉS – NÉHÁNY LISTA PÉLDA

Hány db, adott feltételnek megfelelő elem van egy listában:

$\text{hanydb}([], 0)$.

$\text{hanydb}([F|T], H) :- \text{hanydb}(T, \text{Tdb}), \text{feltetel}(F), H \text{ is } \text{Tdb} + 1, !$.

$\text{hanydb}([_|T], H) :- \text{hanydb}(T, H)$. és ha egyértelmű megoldást szeretnénk? (; hatására sincs több)

$\text{hanydb}([], 0)$.

$\text{hanydb}([F|T], H) :- \text{hanydb}(T, \text{Tdb}), !,$

(
feltetel(F), H is Tdb + 1

;

H is Tdb

).

Itt és a továbbiakban

feltetel(F) :- ...

a feltételt tartalmazó

szabály.

ez is jó: feltetel(F) -> H is Tdb + 1

ISMÉTLÉS – NÉHÁNY LISTA PÉLDA

Hány db, adott feltételnek megfelelő elem van egy listában:

hanydb(L, H) :- hanydb(L, 0, H),!.

hanydb([F|T], Eddig, H) :- feltetel(F), Uj is Eddig + 1,
hanydb(T, Uj, H).

hanydb([_|T], Eddig, H) :- hanydb(T, Eddig, H).

hanydb([], H, H).

és ha egyértelmű megoldást
szeretnénk?(; hatására sincs több)

hanydb([F|T], Eddig, H) :- (feltetel(F) -> Uj is Eddig + 1

;
Uj is Eddig),
hanydb(T, Uj, H).

hanydb([], H, H).

LISTA PÉLDA - BŐVÍTÉS

Hány db, tetszőleges feltételnek megfelelő elem van egy listában:

hanydb(L, H, Feltetel) :- hanydb(L, 0, H, Feltetel),!.

hanydb([F|T], Eddig, H, Feltetel) :-
(current_atom(Feltetel),
Term =.. [Feltetel,F],
Term -> Uj is Eddig + 1
;
Uj is Eddig),
hanydb(T, Uj, H, Feltetel).

hanydb([], H, H,_).

?- hanydb([1,2,3,4,5,6], Db, feltetel).

current_atom

=.. Id. HELP

KITÉRŐ – MEGJEGYZÉS A HF-EKHEZ

```
%
% Get element from index, get index from element or get boolean
from index and element.
% Only the first match!
%
% Get element from index: get(+list, -element, +index).
% Get index from element: get(+list, +element, -index).
% Get boolean from index and element: get(+list, +element, +index).
%
```

Fontos a szép és szabályos kommentezés.

ISMÉTLÉS – ÚTKERESÉS

Az útkeresési példa és javítása

```
start:- write('Melyik városból akarsz indulni?'),
        read(Start),
        write('Hova akarsz eljutni?'), read(Cel),
        ut(Start,Cel).
```

```
ut(Start,Cel) :- repulo(Start,Cel).
```

```
ut(Start,Cel) :- repulo(Start,Uj_varos), ut(Uj_varos,Cel).
```

ÚTKERESÉS – JAVÍTÁS

Az útkeresési példa és javítása

```
ut(Start,Cel,_,[Start,Cel]) :- repulo(Start,Cel).
```

```
ut(Start,Cel,Eddigvizsgalt , [Start|Utvonal] ):-
    repulo(Start,Uj_varos),
    \+(member(Uj_varos,Eddigvizsgalt)),
    ut(Uj_varos,Cel,[Uj_varos|Eddigvizsgalt],Utvonal).
```

```
start:- write('Melyik városból akar indulni?'), read(Start),
        write('Hova akar eljutni?'), read(Cel),
        ut(Start,Cel, [Start,Cel],Megoldas),
        writeln(Megoldas), fail.
start.
```

KERESÉSEK

A keresési algoritmus (esetleges javításokkal) tetszőleges gráf esetén is alkalmazható.

Az

```
ose(A,B) :- el(A,B).
ose(A,B) :- el(A,C), ose(C,B).
```

keresés tetszőleges fa-gráf esetén alkalmazható.

```
:-use_module(arpadhaz).
%:-use_module(kiralynok).
%:- use_module(vasut).
%:-use_module(farmer).
```

KERESÉSEK

:- module(arpadhaz,[el/2]).

el(INNEN,IDE):- szuloje(INNEN,IDE).

szuloje('álmos', 'árpád').

szuloje('árpád', zolta).

szuloje('árpád', tarhos).

szuloje('árpád', 'üllő').

szuloje('árpád', jutas).

szuloje(tarhos,tevel).

...

KERESÉSEK

:- module(vasut,[el/2]).

el(INNEN,IDE):- vonal(INNEN,IDE).

el(INNEN,IDE):- vonal(IDE,INNEN).

vonala('Budapest', 'Székesfehérvár').

vonala('Budapest', 'Pusztaszabolcs').

vonala('Pusztaszabolcs', 'Börgönd').

vonala('Börgönd', 'Székesfehérvár').

vonala('Börgönd', 'Szababattyán').

vonala('Székesfehérvár', 'Szababattyán').

vonala('Szababattyán', 'Csajág').

...

VIGYÁZAT!
Nem fa-gráf!

KERESÉSEK

Feladat: hogyan lehet egy 4-szer 4-es sakktáblán elhelyezni négy királynőt úgy, hogy ne üssék egymást. (Alapfeladat: igazi sakktáblán 8 királynő.)

FONTOS MEGJEGYZÉS:

Most a gráf szemléltetése céljából konkrét adatokkal írjuk meg a modult, de NE szokjon hozzá, hogy konkrét adatokkal ír programot!!!

A feladat ügyesebben (és elegánsabban, jobban) is megoldható, ha általánosan fogalmazzuk meg a táblára pakolás szabályát.

HF: egy ennél ügyesebb, célirányos megoldás.

KERESÉSEK

`:- module(kiralynok,[el/2]).`

`el(tabla(0,0,0,0),tabla(X,0,0,0)) :- member(X,[1,2,3,4]).`

`el(tabla(X,0,0,0),tabla(X,Y,0,0)) :- member(Y,[1,2,3,4]),
X\=Y, abs(X-Y)\=1.`

`el(tabla(X,Y,0,0),tabla(X,Y,Z,0)) :- member(Z,[1,2,3,4]),
X\=Z, abs(X-Z)\=2,
Y\=Z, abs(Y-Z)\=1.`

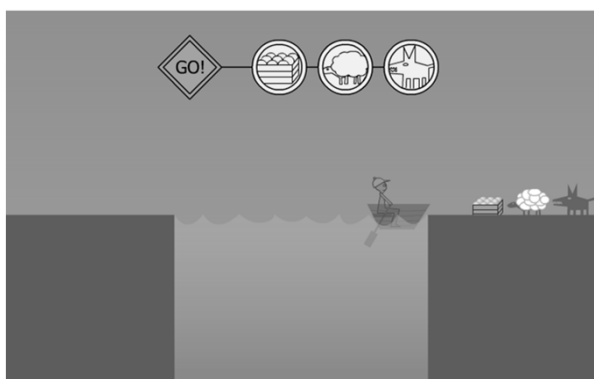
`el(tabla(X,Y,Z,0),tabla(X,Y,Z,Q)) :- member(Q,[1,2,3,4]),
X\=Q, abs(X-Q)\=3,
Y\=Q, abs(Y-Q)\=2,
Z\=Q, abs(Z-Q)\=1.`

KERESÉSEK

Probléma: A farmernek át kell vinnie a folyón a farkast, kecskét, ill. a káposztát úgy, hogy
a/ rajta kívül legföljebb egy valami fér el a csónakban
b/ távollétében nehogy kárt tegyen a farkas a kecskében, ill. a kecske a káposztában.

KERESÉSEK

Kecske-farkas-káposzta:



<http://www.plastelina.net/examples/games/game1.html>

KERESÉSEK

:- module(farmer,[el/2]).

el(EGYIK,MASIK) :- atvisz(EGYIK,MASIK),
 \+ hibas(EGYIK),
 \+ hibas(MASIK).

atvisz: hatására EGYIK állapotból a MASIK-ba kerülünk

allapot: azt mutatja, hogy a farmer, a farkas, a kecske ill. a káposzta a folyó melyik oldalán van.

Pl. állapot(bal,bal,jobb,bal) jelentése: a farmer, a farkas és a káposzta a folyó baloldalán, a kecske a folyó jobboldalán van.

KERESÉSEK

hibás az az állapot, ahol a kecske és a káposzta, illetve a farkas és a kecske a folyó azonos oldalán van, a farmer viszont az ellenkező oldalon.

hibas(allapot(Farmer,_,Kecskekaposzta,Kecskekaposzta)) :-
 ellenkezo(Farmer,Kecskekaposzta) .

hibas(allapot(Farmer,Farkaskecske,Farkaskecske,_)) :-
 ellenkezo(Farmer,Farkaskecske) .

ellenkezo(bal,jobb).

ellenkezo(jobb,bal).

KERESÉSEK

atvisz(allapot(Farmer1,Farkas,Kecske,Kaposzta),
 allapot(Farmer2,Farkas,Kecske,Kaposzta)):-
 ellenkezo(Farmer1,Farmer2).

*/*a farmer egyedül megy át a túloldalra */*

atvisz(allapot(Farmer1,Farkas1,Kecske,Kaposzta),
 allapot(Farmer2,Farkas2,Kecske,Kaposzta)):-
 ellenkezo(Farmer1,Farmer2),
 ellenkezo(Farkas1,Farkas2).

*/*a farmer a farkast viszi át */*

KERESÉSEK

atvisz(allapot(Farmer1,Farkas,Kecske1,Kaposzta),
 allapot(Farmer2,Farkas,Kecske2,Kaposzta)):-
 ellenkezo(Farmer1,Farmer2),
 ellenkezo(Kecske1,Kecske2).

*/*a farmer a kecskét viszi át */*

atvisz(allapot(Farmer1,Farkas,Kecske,Kaposzta1),
 allapot(Farmer2,Farkas,Kecske,Kaposzta2)):-
 ellenkezo(Farmer1,Farmer2),
 ellenkezo(Kaposzta1,Kaposzta2).

*/*a farmer a káposztát viszi át */*

VIGYÁZAT! Ez sem fa-gráf!

KERESÉSEK

Szélességi keresés:

Az adott csúcs összes rákövetkezőjét vizsgáljuk, és addig nem megyünk egy szinttel lejjebb, amíg mindet meg nem néztük. (Felfogható párhuzamos keresésnek is.)

KERESÉSEK

```
szelessegi(Start,Cel):- szelessegi([Start],Cel,[Start]).
```

```
szelessegi([Start|_],Start,_).
```

```
szelessegi([Start|Csucsk],Cel,Utvonal):-
    findall(Csucs,( el(Start,Csucs),
                    \+ member(Csucs,Utvonal)
                    ), Kovetkezo),
    append(Csucsk,Kovetkezo,UjCsucsk),
    append(Utvonal,UjCsucsk,UjUtvonal),
    szelessegi(UjCsucsk,Cel,UjUtvonal).
```

```
findall(Valtozo, feltetel(Valtozo), Lista) - részletesen később
```

KERESÉSEK

HF: bővítse ki az előző programot pl. ezekkel a tényállításokkal:

el(a,b).

el(a,c).

el(b,d).

el(b,e).

el(c,f).

el(c,g).

el(d,h).

el(d,i).

el(e,j).

Próbálja ki nyomkövetéssel pl. a

:- szelessegi(a,g) célt.

(vagy más hasonlót)