PROLOG

DEKLARATÍV PROGRAMOZÁS 2. NHF

KEMPING

Jankó András

NHVU6N

2023.11.03

1. Követelmények elemzése & algoritmus ismertetése

A házi feladatban a híres Tents Puzzle problémát kellett megoldani Prolog nyelven, amit pontosabban ismertetett a házi feladat leírása a hivatalos tárgyoldalon. A feladat leírásában definiált bemenetre kell megmondani az összes lehetséges megoldást az adott rejtvényre. A következőkben ismertetem a megoldás menetét egy konkrét példán keresztül:

A kiadott tesztesetek közül a test04d.txt-ben szereplő esetet fogom bemutatni, amiben a következő mátrixot olvassa be a teszt keretrendszer:

```
   3 0 3 0 3
 3 - F - F -
 0 F - - - F
 3 - - - F -
 0 F - - - F
 3 - F - F -
```

Tehát a beolvasás nem része jelen programnak, hanem a teszt keretrendszer végzi azt. Ezután át lesz alakítva a következő formába:

Bemenet:

TentsPerRow: [3,0,3,0,3]

TentsPerColumn: [3,0,3,0,3]

Trees: [1-2,1-4,2-1,2-5,3-4,4-1,4-5,5-2,5-4]

Kimenet:

Solution: [w,w,s,n,w,s,n,e,e][e,e,n,s,w,n,s,w,w]

A program az utóbbi formában várja a rejtvényt. A következő lépésekben kiszámoljuk a sorok, oszlopok, fák hosszát, majd generálunk egy fa index listát.

RowLength: 5

ColumnLength: 5

TreeLength: 9

TreeIndexes: [1,2,3,4,5,6,7,8,9]

Ebben a lépésben legeneráljuk az összegfeltételeket a get_all_sum_conditions/3 és a get_sum_condition/4 predikátumokkal:

AllSumConditions:[col(1,3),col(2,0),col(3,3),col(4,0),col(5,3),row(1,3),row(2,0),row(3,3),row(4,0),row(5,3)]

Majd a generate_initial_dir_lists/6, check_dirs_of_tree/5 és valid_direction/7 függvényekkel legeneráljuk a kezdeti iránylistát. Ebben a listában azok az irányok szerepelnek az egyes fákra, ahova kerülhet a sátor hozzá képest, és csak azokat az eseteket zárjuk ki, ahol átlógna a pálya szélére vagy egy másik fába ütközne.

PossibleDirections:[[s,w,e],[s,w,e],[n,s,e],[n,s,w],[n,s,w,e],[n,s,e],[n,s,w],[n,w,e],[n,w,e]]

Ezt az iránylistát próbáljuk majd meg szűkíteni a továbbiakban. A choice_point_filter prédikátumban kiválasztunk az eredeti irányok közül egy tömböt (az 1. indexűt), amire a továbbiakban megpróbálunk egy sátorszűkítést végrehajtani:

PossibleDirections:[[s,w,e],[s,w,e],[n,s,e],[n,s,w],[n,s,w,e],[n,s,e],[n,s,w],[n,w,e],[n,w,e]]

DirArr(Kiválasztott): [s,w,e]

Az előbbi műveletet minden indexre megismételjük amíg el nem fogynak a fák, és az összegfeltételek, ha elfogytak az eredményül kapott listát kisimítjuk, és megkapjuk az eredményt:

Flatten: [[w],[w],[s],[n],[w],[s],[n],[e],[e]] -> [w,w,s,n,w,s,n,e,e] (ez csak az egyik megoldás)

A fenti ciklusban a következőket hajtjuk végre: A kiválasztott tömbre a filter/5 predikátumban megpróbálkozunk először egy sátorszűkítéssel (tent_filter) majd egy összegszűkítéssel (sum_filter). A sátorszűkítés a következő módon zajlik:

Megnézzük van-e olyan fa, aminek csak egy lehetséges iránya van a find_single_element_index/4 predikátummal.

Erre az indexre lefuttatunk egy sátorszűkítést. (tent_filter/4) A place_tent_for_single_option/8, tent_coordinate/3, direction_filter/4, neighbor_filter/5, is_neighbor/3, filter_element_from_list/4 predikátumokkal más iránylistákból is elhagyjuk azokat, amik az adott sátor mezőjére, vagy a max 8

szomszédjára esnek. Majd ennek a fának az indexét kivesszük a TreeIndexes-ből. (filter_element_from_list/4)

Végezetül pedig megpróbálkozunk még egy sátorszűkítéssel, ezt addig csináljuk amíg nem marad olyan fa, amelynek egy lehetséges iránya van. Ha ez nem vezet megoldásra tehát van olyan fa, aminek az iránylistájában több irány szerepel megpróbálkozunk a következő módszerrel:

Az összegszűkítés a következő módon zajlik: select(SumCondition, AllSumConditions, NewAllSumConditions): Végig megyünk az összes összegfeltételen.

SumCondition: col(1,3) -> Kiválasztott

AllSumConditions[col(1,3),col(2,0),col(3,3),col(4,0),col(5,3),row(1,3),row(2,0),row(3,3),row(4,0),row(5,3)] -> Összes

NewAllSumConditions[col(2,0),col(3,3),col(4,0),col(5,3),row(1,3),row(2,0),row(3,3),row(4,0),row(5,3)] -> Szűkítés után

És megpróbálkozunk a szűkítéssel a sum_filter/4 predikátummal. A szűkítés során a biztos (B), és esetleg (E) fa-szám értékeket a check_certain/6, és check_probable/7 predikátumokkal kapjuk meg, és ezekkel az értékekkel a check_sumcondition/7 predikátum leszűkíti az eredményhalmazt. Majd meghívjuk a következő összegfeltételre a szűkítést, és így tovább amíg el nem fogynak vagy megoldásra nem jutunk.

2. Tervezés, az egyes eljárások, függvények specifikálása

```
% TentsPerRow    -> list(int):            Sátrak soronkénti számát
tartalmazó lista.              PL: [1,1,0,3,0]
% TentsPerColumn -> list(int):            Sátrak oszloponkénti számát
tartalmazó lista.           PL: [1,0,2,0,2]
% Trees          -> list(int-int):        Fák sorát és oszlopát
azonosító párok lexikografikusan. PL: [1-2,3-3,3-5,5-1,5-5]
% Solution       -> list(list(direction)): A fákhoz tartozó lehetséges
sátorpozíciók.              PL: [e,s,n,n,n]
% A program bemenete.
satrak(satrak(TentsPerRow, TentsPerColumn, Trees), Solution) :-

------------------------------------------------------------------------

% Előállítja az összegfeltételeket.
get_all_sum_conditions(TentsPerRow, TentsPerColumn, Return) :-

------------------------------------------------------------------------

% Rekurzívan előállítjuk az egyes Elementeket (col(1, 2)) mikor a
lista feje egy nemnegatív integer.
get_sum_condition(_, [], _, []) :-
get_sum_condition(Type, [TentsPerHead | TentsPerTail], CurrentIndex,
[Element | Result]) :-
get_sum_condition(Type, [_ | TentsPerTail], CurrentIndex, Result) :-

------------------------------------------------------------------------

% Előállítjuk a kezdeti iránylistákat, itt csak az "Out of bounds" és
a másik fába ütközés hibákkal törődünk. Acc-ban tároljuk az
eddigieket.
% PL: Trees = [2-2, 3-3] Matrix = 3x3 -> Result = [[n, s, w, e], [n,
w]]
% Alapeset mikor a 2.Trees üres akkor a végére értünk.
generate_initial_dir_lists(Trees, [], RowLength, ColumnLength,
Accumulator, Result) :-
generate_initial_dir_lists(Trees, [CurrentTreeCoords |
RemainingTreeCoords], RowLength, ColumnLength, Accumulator, Result) :-

------------------------------------------------------------------------

% EGY! fára megnézi a lehetséges irányokat.
check_dirs_of_tree(Trees, TreePair, RowLength, ColumnLength, Flags) :-

------------------------------------------------------------------------

% Megnézzük egy lehetséges irányra, hogy nem esik e kívül a mátrixon
vagy nem ütközik-e másik fába.
```

```
valid_direction(Trees, Row, Col, RowLength, ColumnLength, Dir-(DRow,
DCol)) :-

--------------------------------------------------------------------

% Kiválasztunk egy pontot az iránylistából, és megpróbálkozunk vele
sátor, és összegszűkítésre.
% Ha elfogytak a fa indexek, és az összegfeltételek megkaptuk a
megoldást (ha van).
choice_point_filter(Dir, CurrentIndex, Trees, PossibleDirections, [],
[], Result) :-
choice_point_filter(Dir, CurrentIndex, Trees, PossibleDirections,
TreeIndexes, AllSumConditions, Return) :-

--------------------------------------------------------------------

% Kicserél egy adott indexű elemet egy adott elemre.
% replace_element_in_list([a, b, c, d], 3, x, Result) -> Result = [a,
b, x, d]
replace_element_in_list(List, Index, Element, Result) :-
replace_element_in_list_helper([], _, _, []).
replace_element_in_list_helper([_|Tail], 1, Element, [Element|Tail])
:-
replace_element_in_list_helper([Head|Tail], Index, Element,
[Head|Result]) :-

--------------------------------------------------------------------

% Egy nested listából flattened listát csinál. [1, [2, [3, 4], 5] ->
[1, 2, 3, 4, 5]
flat(NestedList, FlattenedList) :-
flat(OneElementList, [OneElementList]).

--------------------------------------------------------------------

% Iránylista szűkítés, amivel először iránylista, majd összegfeltétel
alapján próbálunk meg szűkíteni a lehetőségek halmazán.
filter(0, PossibleDirections, TreeIndexes, AllSumConditions,
PossibleDirections-TreeIndexes-AllSumConditions) :-
filter(Trees, PossibleDirections, TreeIndexes, AllSumConditions,
Return) :-

--------------------------------------------------------------------

% Megtalálja annak a fának az indexét, amelyiknek egy lehetséges
iránya van.
find_single_element_index([], _, _, _) :-
find_single_element_index(_, [], _, _) :-
```

```
find_single_element_index(TreeIndexes, [DirList | RemainingDirLists],
Index, Result) :-

------------------------------------------------------------------------

% Sátorszűkítést hajtunk végre, ilyen akkor fordul elő mikor egy fa
iránylistájában már csak egy elem van.
tent_filter(Trees, TargetIndex, PossibleDirections,
NewPossibleDirections) :-
% Ha nincs már több fa koordináta megállunk, amúgy meg rekurzívan
megnézzük az összeset.
place_tent_for_single_option(Trees, [], _, _, [], _, Return, Return)
:-
place_tent_for_single_option(Trees, [CurrentTreeCoords |
RemainingTreeCoords], CurrentIndex, TargetIndex, [CurrentDirArray |
RemainingDirArrays], InitialDirArrays, Accumulator, Return) :-

------------------------------------------------------------------------

% Szabályok sátorkoordináta származtatásra a fa koordinátákból.
tent_coordinate(TreeY-TreeX, n, TentY-TentX) :-
tent_coordinate(TreeY-TreeX, e, TentY-TentX) :-
tent_coordinate(TreeY-TreeX, s, TentY-TentX) :-
tent_coordinate(TreeY-TreeX, w, TentY-TentX) :-

------------------------------------------------------------------------

% Minden fára és iránylistára lefutttatjuk a sátorszűkítést.
direction_filter(Trees, DirArrays, Coord, Return) :-
direction_filter_helper([], [], _, Accumulator, Accumulator) :-
direction_filter_helper([HeadTrees | TailTrees], [HeadDirArrays |
TailDirArrays], Coord, Accumulator, Solution) :-

------------------------------------------------------------------------

% Kiszűri azokat az irányokat a kapott irány alapján, amelyeknek van
sátorszomszédja.
neighbor_filter(Tree, [], Coord, Return, Return) :-
neighbor_filter(Tree, [Head | Tail], Coord, Return, Result) :-

------------------------------------------------------------------------

% Leellenőrzi hogy egy sátornak van-e oldal vagy sarokszomszédja, ami
másik sátor.
is_neighbor(ParamRow-ParamCol, Row-Col, 1) :-
is_neighbor(_, _, 0).

------------------------------------------------------------------------
```

```
% Egy adott elem összes példányának eltávolítása a listából
filter_element_from_list(Element, List, Accumulator, FilteredList) :-
filter_element_helper(_, [], Accumulator, Accumulator) :-
filter_element_helper(Element, [Element|Tail], Accumulator,
FilteredList) :-
filter_element_helper(Element, [Head|Tail], Accumulator, FilteredList)
:-

------------------------------------------------------------------------

% Összegfeltételek alapján próbáljuk meg szűkíteni az iránylistát.
sum_filter(Trees, SumCondition, PossibleDirections,
NewPossibleDirections) :-

------------------------------------------------------------------------

% Megkeressük a biztos, és esetleges irányokat
get_certain_and_probable(_, _, [], [], CheckedArrs, CertainCount,
ProbableCount, CertainCount-ProbableCount-CheckedArrs).
get_certain_and_probable(IsRowOrCol, RowOrColTentNum, [HeadTrees |
TailTrees], [HeadDirections | TailDirections], CheckedArrs,
CertainCount, ProbableCount, Return) :-

------------------------------------------------------------------------

% Megnézi az adott iránylistából melyek azok, amelyek adott oszlopba
vagy sorba mutatnak biztosan.
check_certain(_, _, [], _, Return, Return).
check_certain(IsRowOrCol, Nummer, [HeadDirections | TailDirections],
TreePair, _, IsCertain):-

------------------------------------------------------------------------

% Megnézi az adott iránylistából melyek azok, amelyek adott oszlopba
vagy sorba mutatnak remélhetőleg.
check_probable(_, _, [], _, ProbableDirections, Return, Return-
ProbableDirections).
check_probable(IsRowOrCol, Nummer, [HeadDirections | TailDirections],
TreePair, ProbableDirections, Return, IsProbable) :-

------------------------------------------------------------------------

% Biztos (B) és Esetleg (E) logika a tárgyoldalról
check_sumcondition(B, _, B_and_E, Db, _, _, []) :-
check_sumcondition(B, _, B_and_E, Db, PossibleDirections,
ReturnCheckedArrs, Return) :-
check_sumcondition(B, _, _, Db, PossibleDirections, ReturnCheckedArrs,
Return) :-
```

```
------------------------------------------------------------------------

% Az iránylistákból eltávolítja az ElementsToKeep-ben szereplő
irányokon kívüli irányokat.
remove_directions_reverse([], [], NewDirections, Return) :-
remove_directions_reverse([HeadDirections | TailDirections], [_-0 |
RemainingDirections], NewDirections, Return) :-
remove_directions_reverse([HeadDirections | TailDirections], [_-
ElementsToKeep | RemainingDirections], NewDirections, Return) :-

------------------------------------------------------------------------

% Az iránylistákból eltávolítja az ElementsToRemove-ban szereplő
irányokat.
remove_directions([], [], NewDirections, Return) :-
remove_directions([HeadDirections | TailDirections], [_-0 |
RemainingDirections], NewDirections, Return) :-
remove_directions([HeadDirections | TailDirections], [_-
ElementsToRemove | RemainingDirections], NewDirections, Return) :-

------------------------------------------------------------------------

% Eltávolítja az első listából az összes 2. listában NEM szereplő
elemet.
filter_list_by_another([], _, []).
filter_list_by_another([Head|Tail], SecondList, Result) :-
filter_list_by_another([Head|Tail], SecondList, [Head|Result]) :-

------------------------------------------------------------------------

% Eltávolítja az első listából az összes 2. listában szereplő elemet.
filter_out_elements([], _, []).
filter_out_elements([Head|Tail], FilterList, Result) :-
filter_out_elements([Head|Tail], FilterList, [Head|Result]) :-

------------------------------------------------------------------------
```

3. Tesztelési megfontolások

A tárgyhonlapon lévő teszt környezetet használtam, a forráskódomat bemásoltam a satrak_keret mappába, majd a <sicstus --nologo --noinfo -l ksatrak.pl --goal 'teljes_teszt(120),halt.'> paranccsal indítottam el a teszteket. A legutolsó kimenete a következő volt:

```
Feladvany: tests/test01d.txt,   3*3,   3 fa,    6 osszf, megoldas:    1 db,    0.000
sec HELYES
Feladvany: tests/test02d.txt,   4*4,   4 fa,    8 osszf, megoldas:    1 db,    0.000
sec HELYES
Feladvany: tests/test03d.txt,   4*4,   4 fa,    0 osszf, megoldas:    3 db,    0.000
sec HELYES
Feladvany: tests/test04d.txt,   5*5,   9 fa,   10 osszf, megoldas:    2 db,    0.000
sec HELYES
Feladvany: tests/test05d.txt,   6*6,   9 fa,   12 osszf, megoldas:    1 db,    0.000
sec HELYES
Feladvany: tests/test06d.txt,   8*8,  15 fa,   10 osszf, megoldas:    1 db,    0.000
sec HELYES
Feladvany: tests/test07d.txt, 10*10,  20 fa,   20 osszf, megoldas:    2 db,    0.000
sec HELYES
Feladvany: tests/test08d.txt, 12*12,  35 fa,    6 osszf, megoldas:    2 db,    0.000
sec HELYES
Feladvany: tests/test09d.txt, 12*15,  42 fa,   19 osszf, megoldas:    1 db,    0.015
sec HELYES
Feladvany: tests/test10d.txt, 15*15,  52 fa,   19 osszf, megoldas:    2 db,    0.031
sec HELYES
Feladvany: tests/test11d.txt, 15*15,  40 fa,   30 osszf, megoldas:    2 db,    0.016
sec HELYES
Feladvany: tests/test12d.txt, 15*20,  32 fa,   35 osszf, megoldas:    4 db,    0.000
sec HELYES
Feladvany: tests/test13d.txt, 15*20,  43 fa,   35 osszf, megoldas:    3 db,    0.875
sec HELYES
Feladvany: tests/test14d.txt, 15*15,  52 fa,   30 osszf, megoldas:    2 db,    0.000
sec HELYES
Feladvany: tests/test15d.txt, 15*20,  65 fa,   35 osszf, megoldas:    1 db,    0.156
sec HELYES
Feladvany: tests/test16d.txt, 15*15,  31 fa,   20 osszf, megoldas:    8 db,    0.000
sec HELYES
Feladvany: tests/test17d.txt, 15*20,  59 fa,   20 osszf, megoldas:    1 db,    0.297
sec HELYES
Feladvany: tests/test18d.txt, 15*20,  50 fa,   22 osszf, megoldas:   12 db,    0.047
sec HELYES
Feladvany: tests/test19d.txt, 20*20,  80 fa,   21 osszf, megoldas:    2 db,    3.391
sec HELYES
Feladvany: tests/test20d.txt, 20*20,  80 fa,   40 osszf, megoldas:    1 db,    1.156
sec HELYES
```

```
Feladvany: tests/test21d.txt, 20*20,  80 fa,   26 osszf, megoldas:    4 db,    3.266
sec HELYES
Feladvany: tests/test22d.txt, 20*20,  79 fa,   40 osszf, megoldas:    2 db,    0.047
sec HELYES
Feladvany: tests/test23d.txt, 20*20,  55 fa,   40 osszf, megoldas:    6 db,    0.219
sec HELYES
Feladvany: tests/test24d.txt, 20*20,  81 fa,   30 osszf, megoldas:    4 db,    0.281
sec HELYES
Feladvany: tests/test25d.txt, 25*20, 102 fa,   45 osszf, megoldas:    1 db,    0.078
sec HELYES
Feladvany: tests/test26d.txt, 20*20,  86 fa,   40 osszf, megoldas:    6 db,    0.062
sec HELYES
Feladvany: tests/test27d.txt, 25*25, 128 fa,   50 osszf, megoldas:    1 db,    0.110
sec HELYES
Feladvany: tests/test28d.txt, 25*25, 125 fa,   50 osszf, megoldas:    1 db,    0.671
sec HELYES
Feladvany: tests/test29d.txt, 20*20,  72 fa,   32 osszf, megoldas:    2 db,    0.234
sec HELYES
Feladvany: tests/test30d.txt, 30*30, 180 fa,   60 osszf, megoldas:    1 db,    0.297
sec HELYES
Feladvany: tests/test31d.txt, 20*20,  72 fa,   36 osszf, megoldas:    1 db,    0.047
sec HELYES
Feladvany: tests/test32d.txt, 20*20,  72 fa,   40 osszf, megoldas:    1 db,    0.047
sec HELYES
Feladvany: tests/test33d.txt, 20*20,  72 fa,   31 osszf, megoldas:    1 db,    0.157
sec HELYES
Feladvany: tests/test34d.txt, 20*20,  73 fa,   20 osszf, megoldas:   16 db,    2.078
sec HELYES
Feladvany: tests/test35d.txt, 20*20,  72 fa,   27 osszf, megoldas:    1 db,    0.515
sec HELYES
Feladvany: tests/test36d.txt, 25*25, 124 fa,   50 osszf, megoldas:    4 db,    7.952
sec HELYES
```

4. Kipróbálási tapasztalatok

A programot lokálisan futtattam, és VS Code-ot használtam a fejlesztéshez, valamint a tárgyoldalon lévő InstallSICStus-4.8.0-x64-VC16.exe állományt telepítettem.