Artificial Intelligence

Assignment 3

Matthias Horbach

mhorbach@uni-koblenz.de

Institute of Web Science and
Technologies Department of Computer
Science University of Koblenz-Landau

Submission until: 16.05.2020, 23:59

Tutorial on: 17.05.2020

Abhinav Ralhan (221202684), abhinavr8@uni-koblenz.de
Hammad Ahmed (221202832), hammadahmed@uni-koblenz.de
Muhammad Asad Chaudhry (221202659), maac@uni-koblenz.de
Vishal Vidhani (221202681), vvidhani@uni-koblenz.de

1 Lists in Prolog (5 Points)

Develop a prolog predicate lists intersect/2, which tests for any two lists if at least one element is contained in both lists. Hints:

- The /2 after the predicate name indicates the arity of the predicate to be developed.
- You are allowed to use the built-in predicate member/2.

For example, the following queries should result in answer true:

- lists intersect([1,2,3],[3,4,7])
- lists intersect([1,2,3],[a,1,b,c])
- lists intersect([1,2,3],[3,4,1,7])

And the following queries should result in answer false:

- lists intersect([1,2,3],[a,b,c])
- lists intersect([],[a,b,c])

Solution

```
lists_intersect(List1, List2):-
   member(Element,List1),
   member(Element,List2).
```

2 Search Problems (20 Points)

Encode the following problems as search problems as in the lecture: De ne what a state is, the transition function as well as the initial state and set of goal state.

1. The 15 puzzle (https://en.wikipedia.org/wiki/15 puzzle). While the classic variant of the puzzle is played on a 4x4 board with 15 tiles, you can restrict yourselves to a 2x2 board with 3 tiles with this initial position:

2	3
	1

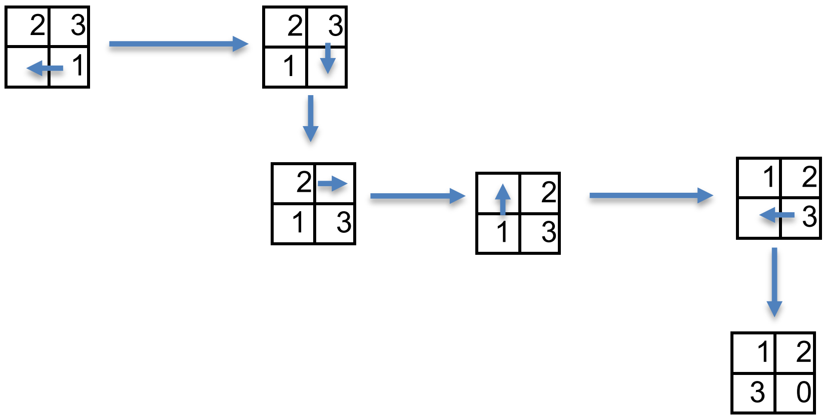
Solution:

The search problem tuple P = (S, O, I, G), so the solution π for P is sequence π = (S1, S2, …, Sn), where as

S is set of states.
O is state transition
I is initial states
G is get of goal states

S1,….,Sn ∈ S, S1 = I and Sn ∈ G
(Si, Si+1) ∈ O

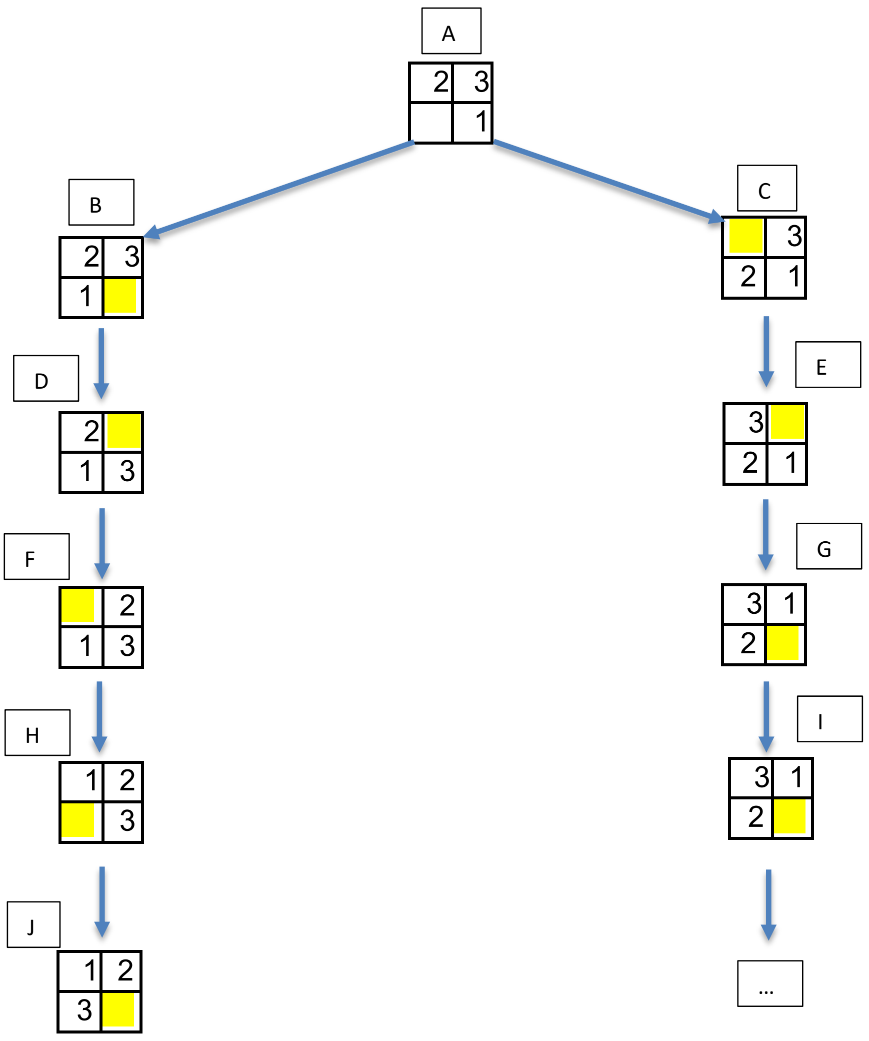
therefore, 2x2 board with 3 tiles have in total (n^2 – 1) tiles from 1,2,3 and 0 for the empty space. So,



S_1 = {0,1,2,3}
O_1 = {([[2,3],[1,0]], [[2,0],[1,3]], [[0,2],[1,3]], [[1,2],[0,3]], [[1,2],[3,0]])}
I_1 = {[[2,3],[0,1]]}
G_1 = {[[1,2],[3,0]]}

P = ({0,1,2,3}, {([[2,3],[1,0]], [[2,0],[1,3]], [[0,2],[1,3]], [[1,2],[0,3]], [[1,2],[3,0]])}
, {[[2,3],[0,1]]}, {[[1,2],[3,0]]})

So, the minimal no. of sequence on the problem P is 5 states, i.e left, down, right, up, left. We can also solve this problem by using the BFS algorithm.



Explored	Frontier
-	A
A	B, C
A, B	C, D
A, B, C	D, E
A, B, C, D	E, F
A, B, C, D, E	F, G
A, B, C, D, E, F	G, H
A, B, C, D, E, F, G	H, I
A, B, C, D, E, F, G, H	I, J
A, B, C, D, E, F, G, H, I	J, …
A, B, C, D, E, F, G, H, I, J	Goal node found!!!!!

So, the runtime complexity for this algo is $O(n^{e+1})$

2. Four gnomes G_1; G_2; G_3; G_4 are on the right side of a wobbly bridge. Its dark and the group only has one torch. The task is to get everyone to the left side. At most two gnomes can cross the bridge together. They need the torch in order to cross the bridge. The torch burns for 60 minutes. G_1 needs 5 minutes to get across, G_2 needs 10, G_3 needs 20 and G_4 needs 25 minutes. If two gnomes cross the bridge together, they need as much time as the slowest of them.

Solutions:

Tuples (L,R,o,T)
where L - set of gnomes on the Left side of bridge
R - set of gnomes on the right side of bridge
o - position of the torch
T - elapsed time of burning a torch

And our P = (S,O,I,G), where as

S is set of states.
O is state transition
I is initial states
G is get of goal states

So the final possible solution will be:

G1 = (L1,R1,o1,T1)
G2 = (L2,R2,o2,T2)
G3 = (L3,R3,o3,T3)
G4 = (L4,R4,o4,T4)

S(G1G2G3G4) = {(L, R, o, T)| L,R ⊆ {G1, G2, G3, G4},
O(1,2…6) = {(
({}, {G1, G2, G3, G4}, r, 0),
({G1, G2}, {G3, G4}, l, 10),
({G2}, {G1, G3, G4}, r, 15),
({G2, G3, G4}, {G1}, l, 40),
({G3, G4}, {G1, G2}, r, 50),
({G1, G2, G3, G4}, {}, l, 60),
)}
I(G1G2G3G4)= ({}, {G1, G2, G3, G4}, r, 0)
G(G1G2G3G4)= {({G1, G2, G3, G4}, {}, l, 60)}

Therefore, P = {(L, R, o, T), {(
({}, {G1, G2, G3, G4}, r, 0),
({G1, G2}, {G3, G4}, l, 10),
({G2}, {G1, G3, G4}, r, 15),
({G2, G3, G4}, {G1}, l, 40),
({G3, G4}, {G1, G2}, r, 50),
({G1, G2, G3, G4}, {}, l, 60),
)}, ({}, {G1, G2, G3, G4}, r, 0), {({G1, G2, G3, G4}, {}, l, 60)}}

3 Search in Prolog (5 Points)

Implement a 2x2 version of the 15 puzzle from the previous exercise in Prolog.
Bonus task (no points): Use Prolog to verify that the target position is reachable from

2	3
	1

Hint: You will notice quickly that a naive implementation does not terminate. You may use any built-in predicates that might help you, e.g. length/2 modeling the length of a list, for example length([X,Y],2) is true.

Solution:

```
% Define the initial state
initial_state([[2, 3], [0, 1]]).

% Define the goal state
goal_state([[1, 2], [3, 0]]).

% Define the valid moves
move([[A, B], [C, 0]], [[A, B], [0, C]]). % Move the tile down
move([[A, 0], [C, B]], [[A, C], [0, B]]). % Move the tile right
move([[0, A], [C, B]], [[C, A], [0, B]]). % Move the tile left
move([[A, B], [0, C]], [[A, B], [C, 0]]). % Move the tile up

% Define the search algorithm
search(State, _, []) :- goal_state(State).
search(State, Visited, [Move | Moves]) :-
   move(State, NextState),
   \+ member(NextState, Visited), % Avoid loops
   search(NextState, [NextState | Visited], Moves),
   Move = NextState.

% Predicate to solve the puzzle
solve_puzzle(Moves) :-
   initial_state(InitialState),
   search(InitialState, [InitialState], Moves).
```

Important hints

- Always include all names of all group members that helped solving the excercises on your PDF. Only those will receive points for solving the excercises.
- By handing in this sheet, you con rm that you solved these excercises yourself. If the situation occurs that two groups have identical solutions, both groups will get zero points.
- Your SVN-Repositories can be accessed via

  https://svn.uni-koblenz.de/mhorbach/ai23/[yourGroupName]

  You can use the subfolder workspace to share data among your group; this folder's content will not be graded. Submit your solution in the subfolder solutions with speaking name such as assignment3.pdf. You tutor will upload their notes in the subfolder comments.

  Note that you do not have access to the repository's base directory.
- Format: All solutions must be contained in PDF documents (including source code). Additionally, source code must be provided as plain les that are readable via a standard text editor.
- Please make sure that all your programs can be run without errors. Comments on your souce code will be in the annotated PDF that we create during excercise corrections.
- Do not use any mutated vowels or special characters in your source code. Also, do not use those or spaces in le names.