COSC1125/1127 Artificial Intelligence

School of Computer Science and IT
RMIT University
Semester 1, 2017

Tutorial Sheet 2
Search II

Exercises

1. Consider this problem: We have one 3 litre jug, one 5 litre jug and an unlimited supply of water. The goal is to get exactly one litre of water into either jug. Either jug can be emptied or filled, or poured into the other.

   For this problem give:

   (a) An appropriate data structure for representing a state.

   Answer

   The representation of the state can be simply: (nL, nR), where nL is the number of litres in the 3 litre jug, and nR is number of litres in the 5 litre jug.

   (b) The initial state.

   Answer

   The initial state is $(0, 0)$.

   (c) The final states (there are 2).

   Answer

   The goal state can be either $(1, 0)$ or $(0, 1)$. The wording of the question is a bit ambiguous, because it doesn't state whether the other jug must be empty or not, so $(1, nR)$ or $(nL, 1)$ should be acceptable as well.

   (d) A specification of the operators (or actions) which includes the preconditions that must be satisfied before the operator can be used and the new state generated.

Answer

Operators could be (using FOL sentences):

$\underline{Fill(X)}$
Action: fill jug X
Precondition: jug X is not full
State generated: jug X is full - $(3, nR)$ or $(nL, 5)$

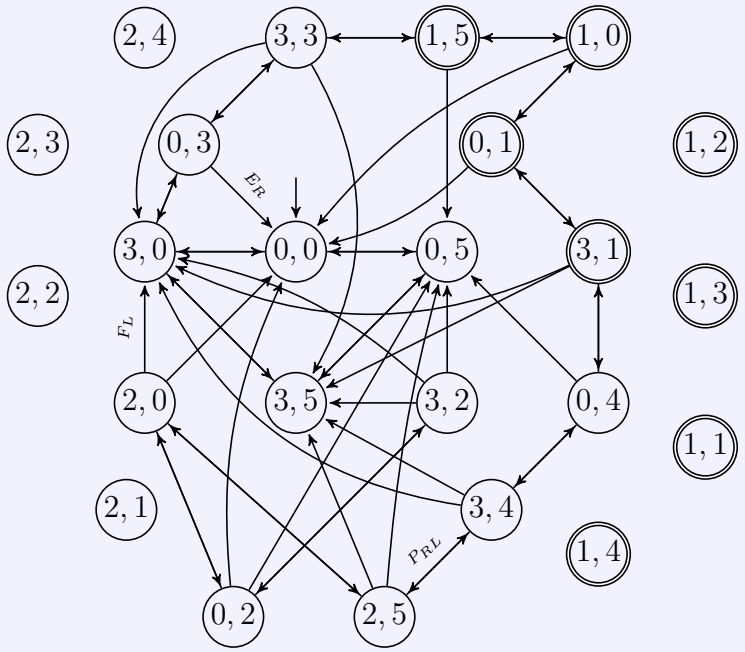
$\underline{Pour(X, Y)}$
Action: pour jug X into jug Y until either X is empty or Y is full
Precondtion: jug X is not empty and jug Y is not full
State Generated:

- jug X is empty and jug Y is partially full - $(0, nR)$ or $(nL, 0)$, or
- jug X is partially full and jug Y is full - $(3, nR)$ or $(nL, 5)$

$\underline{Empty(X)}$
Action: empty jug X
Precondition: jug X is not empty
State generated: jug X is empty - $(0, nR)$ or $(nL, 0)$

(e) Draw the full state space.

Answer



State transition labels:
F_X fill jug X
E_X empty jug Y
P_{XY} pour jug X into jug Y

Note: For clarity, not all state transition labels are not shown on this diagram, usually you should show all state transition labels above all arcs. Observe also that states that cannot be reached from the initial state, such as (1,4), are still part of the full state space. If you are not going to show inaccessible states, you should explicitly state so.

(f) What is the solution to the problem.

Answer

$(0,0) \rightarrow (3,0) \rightarrow (0,3) \rightarrow (3,3) \rightarrow (1,5) \rightarrow (1,0)$

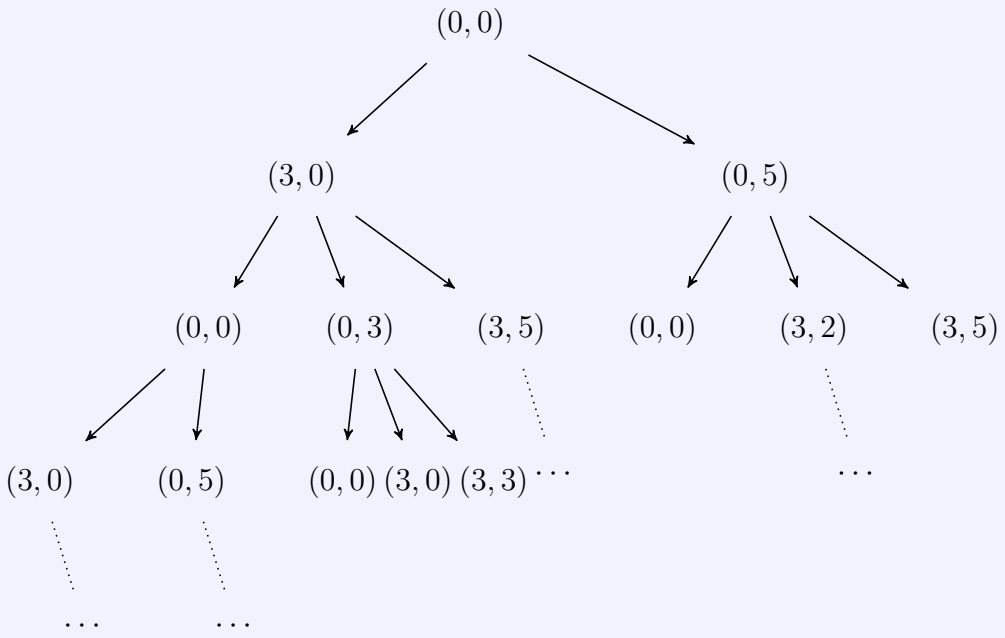
2. In the previous exercise, a representation for states and the full state space were developed. For the same problem, apply search strategies and note:

   - *The order in which nodes are created in memory.*
   - *The nodes that are not created in memeory at all.*

   for the following search strategies:

   (a) Breadth first search with no checking for duplicate states.

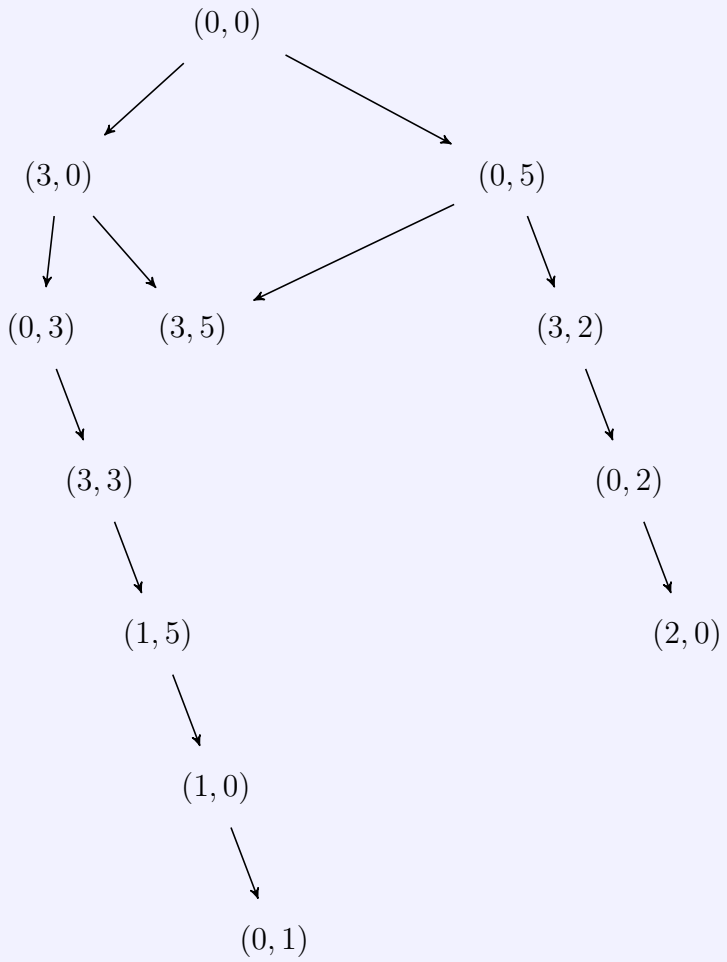
Answer



Basically you need you generate every possible state from a parent state, and you do this state by state (from left to right), and by level (from top down). Note that operators applied are not shown over each arc, but you should do so normally.

(b) Breadth first search with checking for duplicate states.

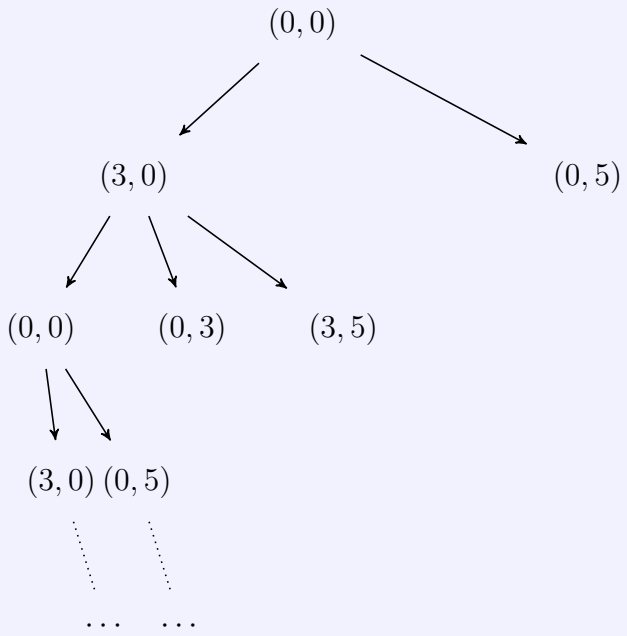
Answer



Note that you still expand nodes level by level, but just not generating nodes that are already there.

(c) Depth first search with no checking for duplicate states.

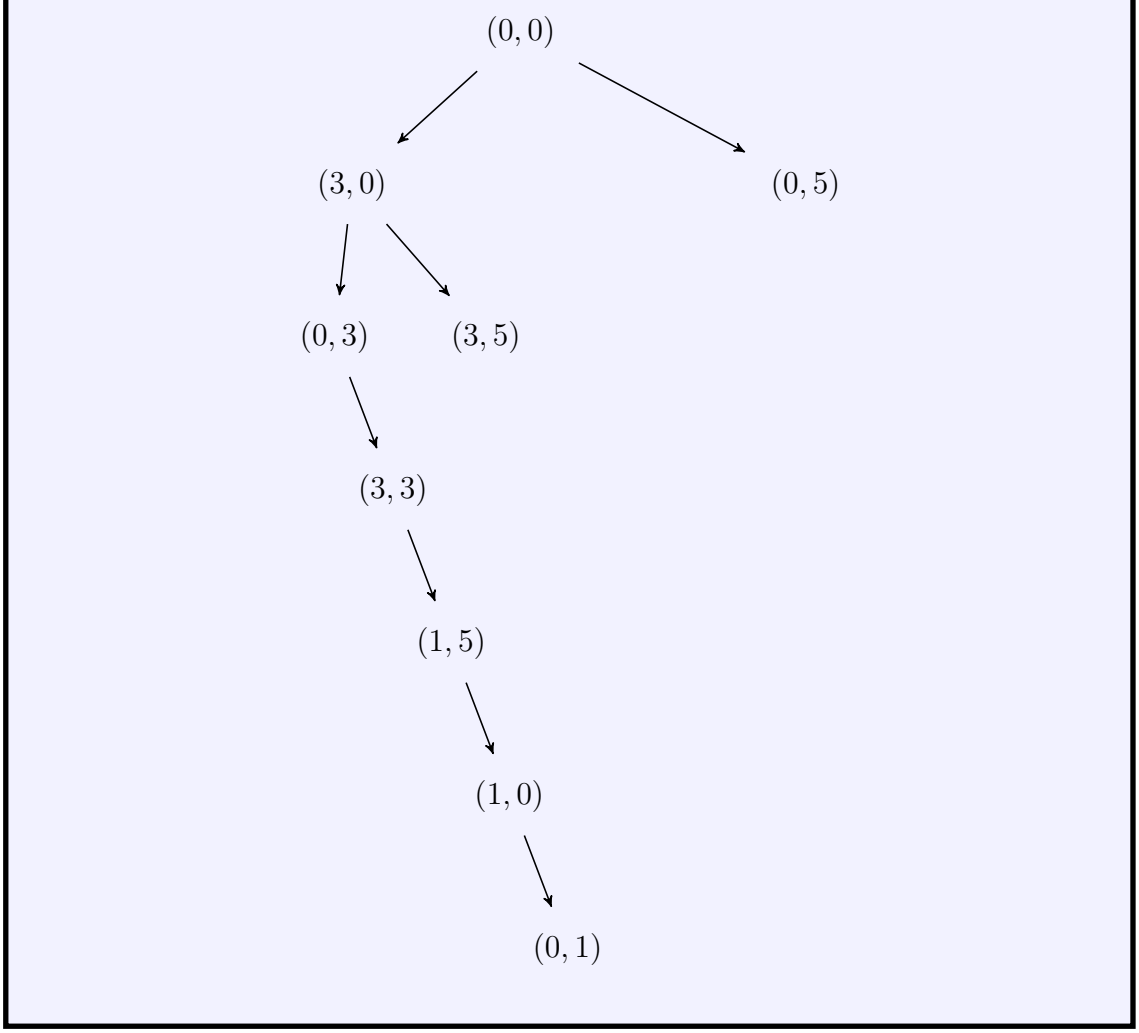
Answer



Expand nodes in the order as displayed in (a), ie. DFS here keeps expanding the left-most branch branch of the search tree. Since it is not checking for duplicate states, it can go infinitely deep. If a stack data-structure is used, then it will expand the right-most branches.

(d) Depth first search with checking for duplicate states.

Answer



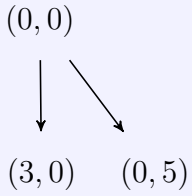
(e) Iterative deepening with no checking for duplicate states.

Answer

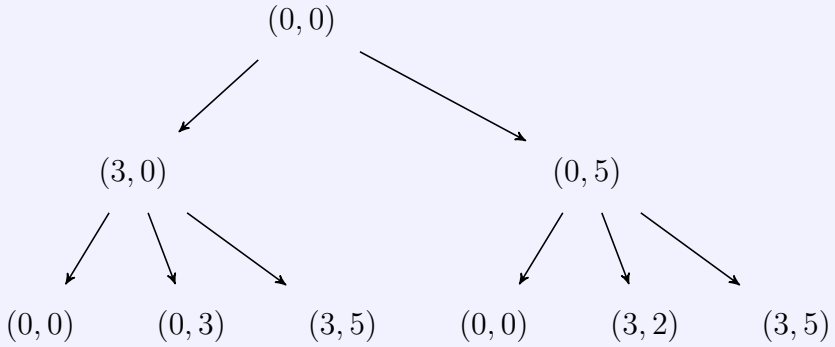
Step 1:

$(0, 0)$

Step 2:



Step 3:



This goes for as many more steps as is required to find the goal.

(f) Iterative deepening with checking for duplicate states.

Answer

Similar to (e) without showing the duplicated states.

(g) Is bi-directional search possible for this problem?

Answer

Yes, it is possible to use bi-directional search, since you know both the initial and goal states. Bi-directional search basically means that you expand nodes from the initial and goal nodes simultaneously (with or without checking for duplicates), until the two search trees are connected to form a single path from the initial to the goal state. One caveat to this is that, without being given the full state space, it can be difficult to search backwards as new transitions (eg. 'unfill', 'unempty' and 'unpour') need to be defined which are not all that intuitive.

3. Consider the problem of getting from Arad to Bucharest in Romania. For this problem give:

   - Search state descriptions.

Answer

A possible state description could be $(\langle city \rangle, g(n))$, where $g(n)$ is the total path cost to that node.

- Initial search state.

Answer

$(Arad, 0)$

- Final search state.

Answer

$(Bucharest, g(n))$, where $g(n)$ is total path cost from Arad to Bucharest.

- Operators.

Answer

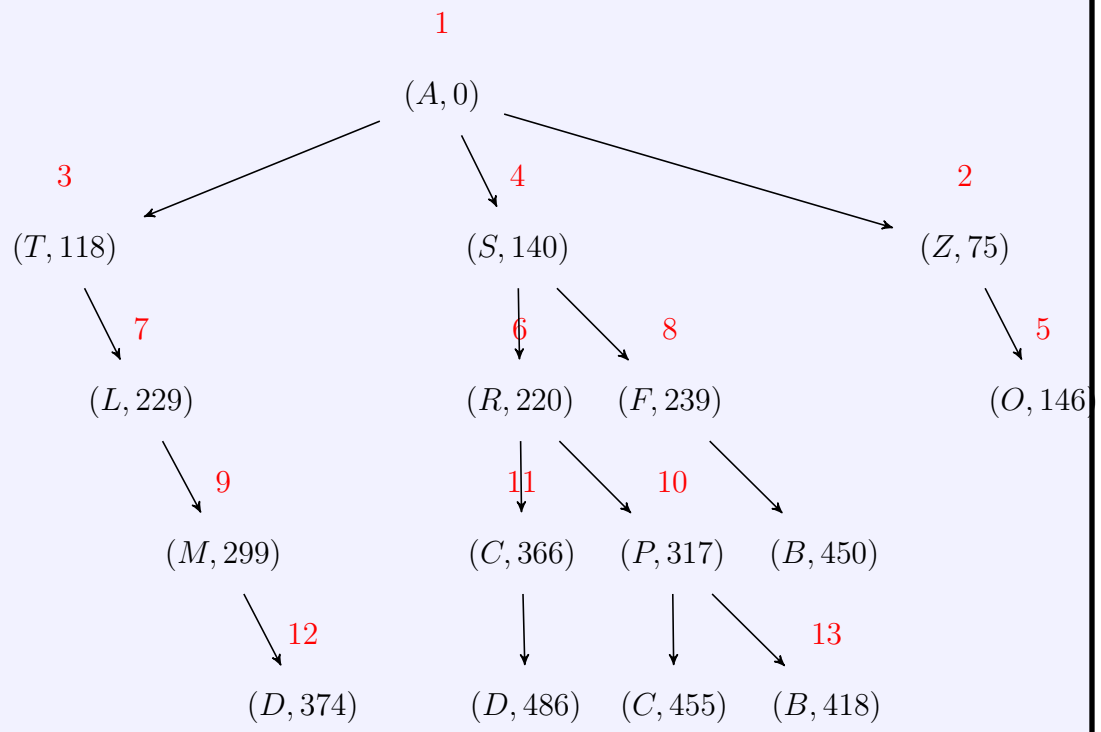
Only one operator exists (using FOL sentences):

$\underline{Go(X, Y)}$
Action: travel from city X to city Y
Precondition: currently in city X and city X has an edge connecting it to city Y
State generated: in city Y - $(Y, g(n))$, where $g(n))$ is total path cost to city Y.

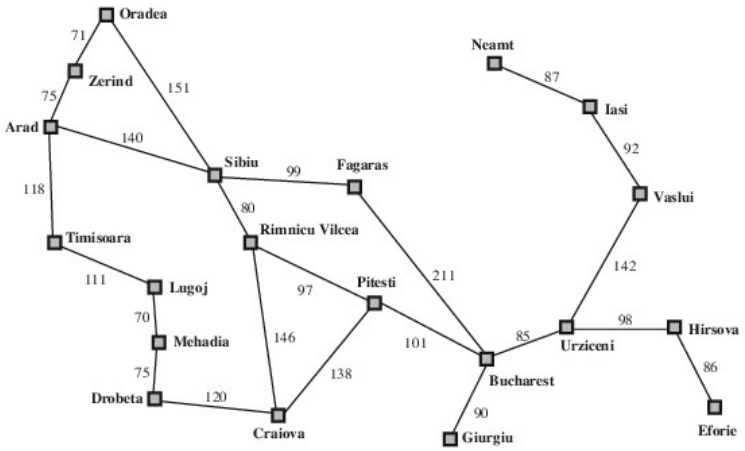
The reason $Go(X)$ is not used, is that it does not take into account the fact that we must travel between connected cities.

- The part of the search space that is realized in memory and the order of node expansion if uniform cost search is used.

Answer



Cities are represented by their first letter and the red numbers indicate the order in which nodes are expanded.



4. Finally, get your hands dirty by doing this fun Lab-Search sheet.

5. *You say more?* Lots of cool exercise in RN book, chapter 3....