COSC1125/1127 Artificial Intelligence

School of Computer Science and IT
RMIT University
Semester 1, 2019

Tutorial Sheet 2
Search II

Exercises

1. **From Tutorial 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.

   (b) The initial state.

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

   (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.

   (e) What is the solution to the problem.

   Answer

   See solution to tutorial 1.

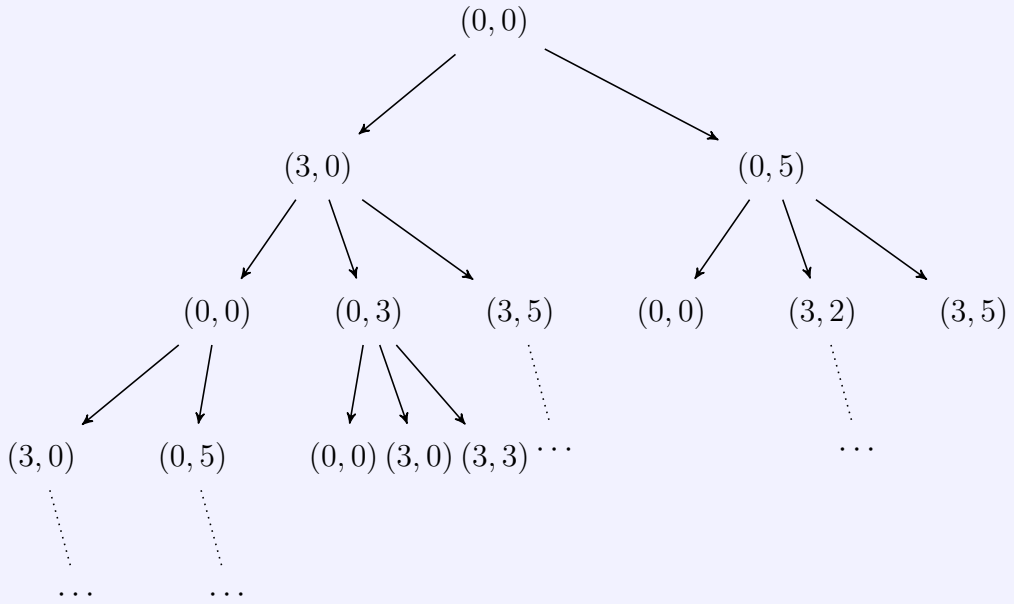
2. In the previous exercise, a representation for states and the full state space were developed. For the same problem, apply search strategies and note:

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

   for the following search strategies:

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

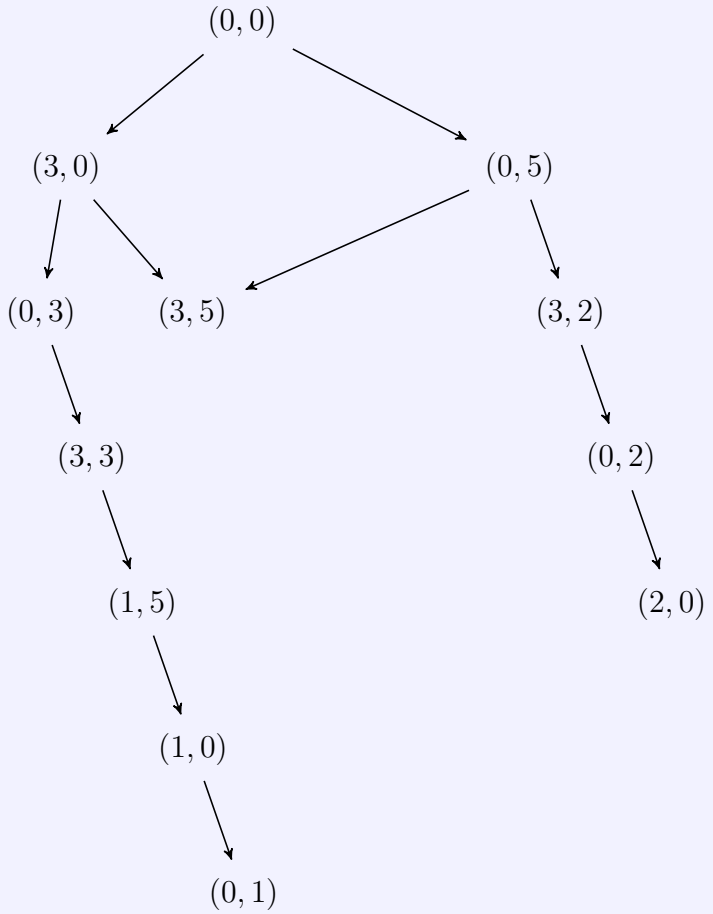
Answer



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.

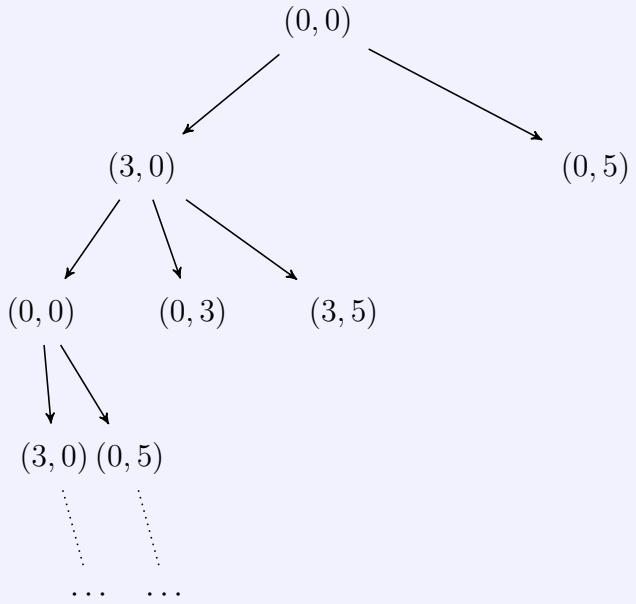
Answer



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.

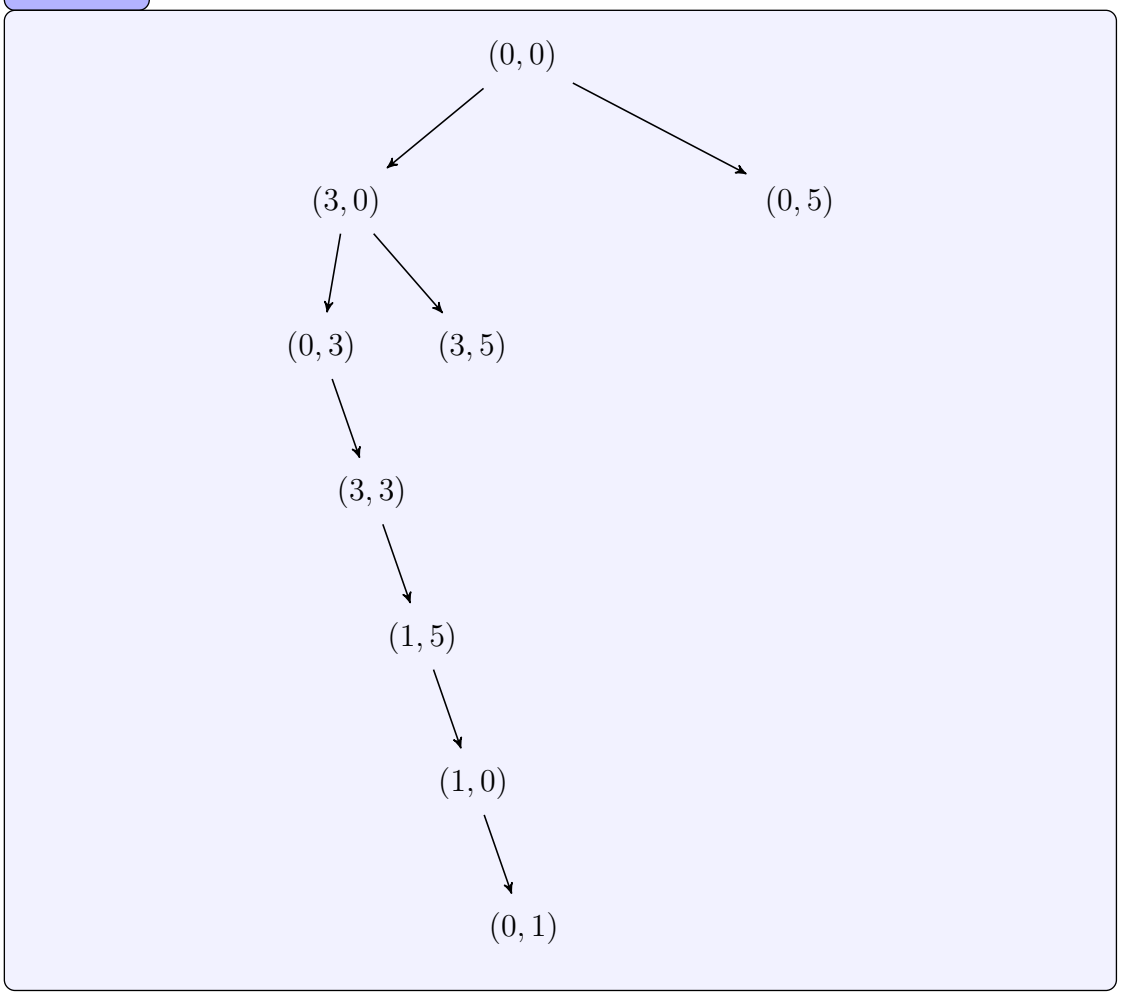
Answer



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

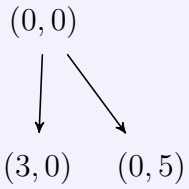


(e) Iterative deepening with no checking for duplicate states.

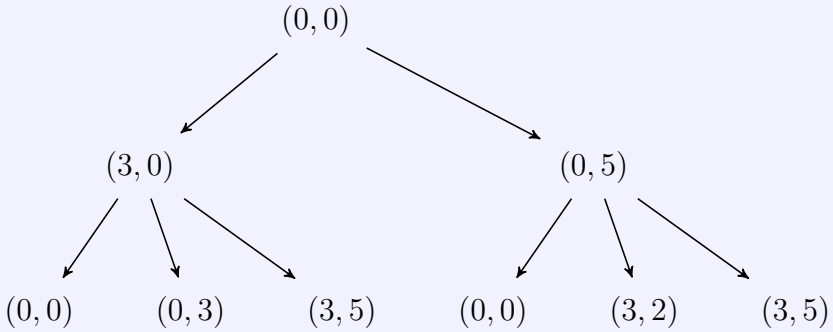
Answer

Step 1:

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.

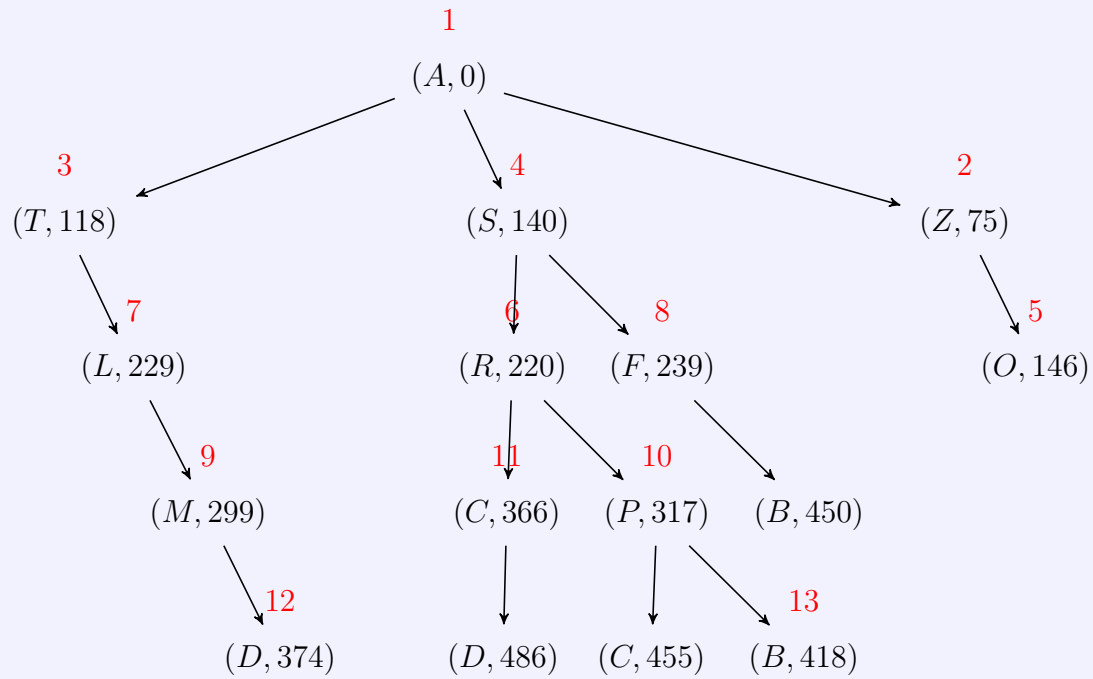
(g) Is bi-directional search possible for this problem?

Answer

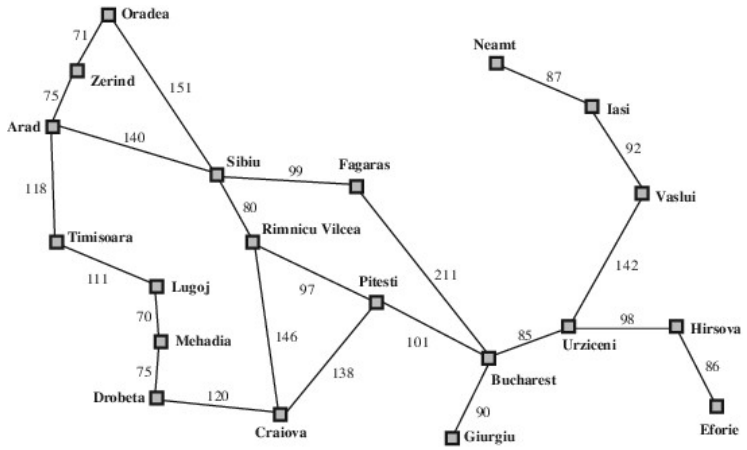
Yes, it is possible to use bi-directional search, since you know both the initial and goal states. Bi-directional search basically means that you expand nodes from the initial and goal nodes simultaneously (with or without checking for duplicates), until the two search trees are connected to form a single 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 from Tutorial 1. Provide 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. What are the dimensions we judge the various search algorithms? Discuss each of them for each algorithm.

Answer

Dimensions: Time required, space required, completeness, and optimality.
All search algorithms are worst-case exponential in time, but they vary across the other properties! Check slides and book.

5. (RN) Which of the following are true and which are false? Explain your answers.

   - Depth-first search always expands at least as many nodes as A search with an admissible heuristic.

- $h(n) = 0$ is an admissible heuristic for the 8-puzzle.
- A* is of no use in robotics because percepts, states, and actions are continuous.
- Breadth-first search is complete even if zero step costs are allowed.
- Assume that a rook can move on a chessboard any number of squares in a straight line, vertically or horizontally, but cannot jump over other pieces. Manhattan distance is an admissible heuristic for the problem of moving the rook from square A to square B in the smallest number of moves.

Answer

- *False*: a lucky DFS might expand exactly d nodes to reach the goal. A largely dominates any graph-search algorithm that is guaranteed to find optimal solutions.
- *True*: $h(n) = 0$ is always an admissible heuristic, since costs are nonnegative.
- *False*: A* search is often used in robotics; the space can be discretized or skeletonized.
- *True*: depth of the solution matters for breadth-first search, not cost.
- *False*: a rook can move across the board in move one, although the Manhattan distance from start to finish is 8.

6. If you finish this sheet, you can start with the heuristic search algorithms in Tutorial Sheet 3. :-)
7. Finally, get your hands dirty by doing this fun Lab-Search sheet.
8. *You say more?* Lots of cool exercise in RN book, chapter 3....