National University of Singapore
School of Computing
CS3243 Introduction to AI

Tutorial 2: Uninformed Search

Issue: January 22, 2015 Due: February 6, 2015

Important Instructions:

- *Your solutions for this tutorial must be TYPE-WRITTEN.*
- *Make TWO copies of your solutions: one for you and one to be SUBMITTED TO THE TUTOR IN CLASS. Your submission in your respective tutorial class will be used to indicate your CLASS ATTENDANCE. Late submission will NOT be entertained.*
- *YOUR SOLUTION TO QUESTION* 1 *will be GRADED for this tutorial.*
- *You may discuss the content of the questions with your classmates. But everyone should work out and write up ALL the solutions by yourself.*

1. The Missionaries and Cannibals problem is usually stated as follows (refer to page 115 of the textbook, i.e., AIMA 3rd edition). Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side, without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place.

   Questions:

   - Give the representation of a state in this problem;
   - Using the state representation defined above, specify the initial state and goal state.
   - Define its actions (to simplify your problem, you can ignore the possibility of illegal states); and
   - Using the state representation and actions defined above, specify the transition model/function T (i.e., when each of the actions defined above is applied to a current state, what is the resulting next state?). To simplify the problem, you can ignore the possibility of illegal states.

In this problem, a state is an ordered sequence of three numbers (M, C, B)*, where:* M *is the number of missionaries on the bank of the river from which the missionaries and cannibals started,* C *is the number of cannibals on the bank of the river from which the missionaries and cannibals started, and* B *is the number of boats on the bank of the river from which the missionaries and cannibals started. Note that* $M = \{0, 1, 2, 3\}; C = \{0, 1, 2, 3\}; B = \{0, 1\}$.

That is, the initial state is $(3, 3, 1)$ *and the goal state is* $(0, 0, 0)$.

There are 5 *actions:*

(a) Let a_1 *denote the action of 'the boat carrying 2 missionaries':*
$T((M, C, 0), a_1) = (M + 2, C, 1), 0 \leq M \leq 1$
$T((M, C, 1), a_1) = (M - 2, C, 0), 2 \leq M \leq 3$

(b) Let a_2 *denote the action of 'the boat carrying 1 missionary':*
$T((M, C, 0), a_2) = (M + 1, C, 1), 0 \leq M \leq 2$
$T((M, C, 1), a_2) = (M - 1, C, 0), 1 \leq M \leq 3$

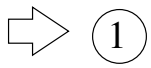
(c) Let a_3 *denote the action of 'the boat carrying 2 cannibals':*
$T((M, C, 0), a_3) = (M, C + 2, 1), 0 \leq C \leq 1$
$T((M, C, 1), a_3) = (M, C - 2, 0), 2 \leq C \leq 3$

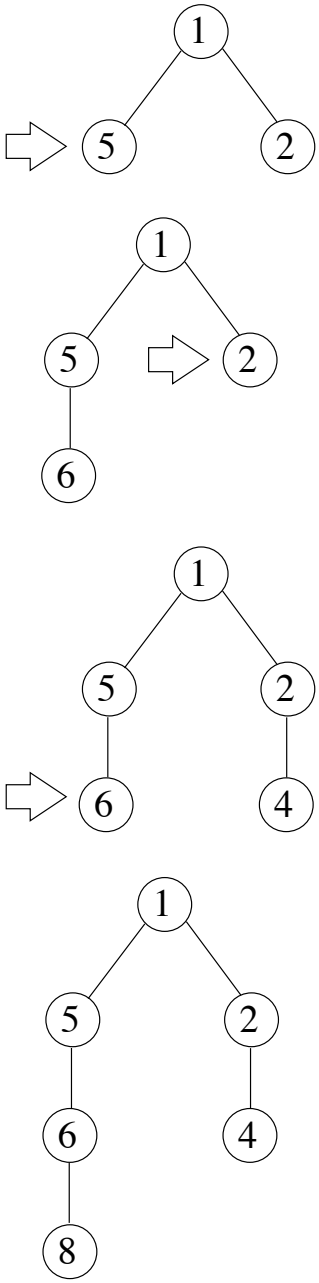
(d) Let a_4 *denote the action of 'the boat carrying 1 cannibal':*
$T((M, C, 0), a_4) = (M, C + 1, 1), 0 \leq C \leq 2$
$T((M, C, 1), a_4) = (M, C - 1, 0), 1 \leq C \leq 3$

(e) Let a_5 *denote the action of 'the boat carrying 1 missionary and 1 cannibal':*
$T((M, C, 0), a_5) = (M + 1, C + 1, 1), 0 \leq M \leq 2, 0 \leq C \leq 2$
$T((M, C, 1), a_5) = (M - 1, C - 1, 0), 1 \leq M \leq 3, 1 \leq C \leq 3$

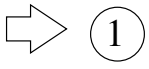
2. Consider the vacuum world problem with the state space shown in Figure 1. With the state numbers assigned in Figure 4.9 (page 134) of AIMA 3rd edition, let the initial state be state 1 and the goal state be either state 7 or state 8. Assume that the order of expansion of actions is S, R, L.

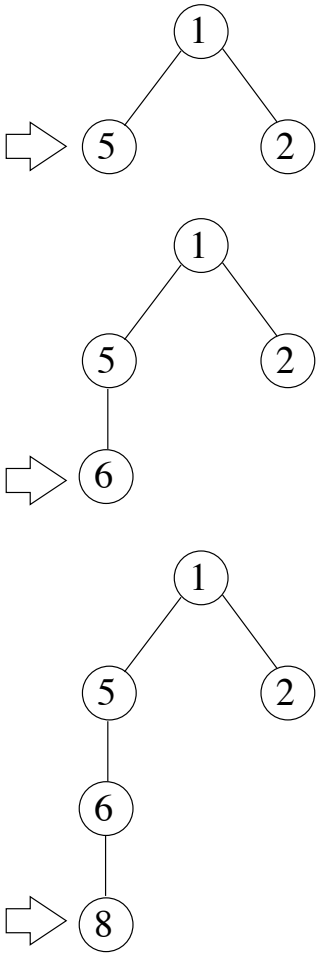
   (a) Give a trace of the breadth-first search algorithm in the style of Figures 3.11 and 3.12 of AIMA 3rd edition. That is, show the search tree at each stage (all repeated states are eliminated).





(b) Give a similar trace of the depth-first search algorithm.





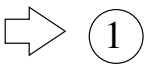
(c) Which of these two search algorithms is better for this problem? Why? Is one search strategy always better than the other in general?

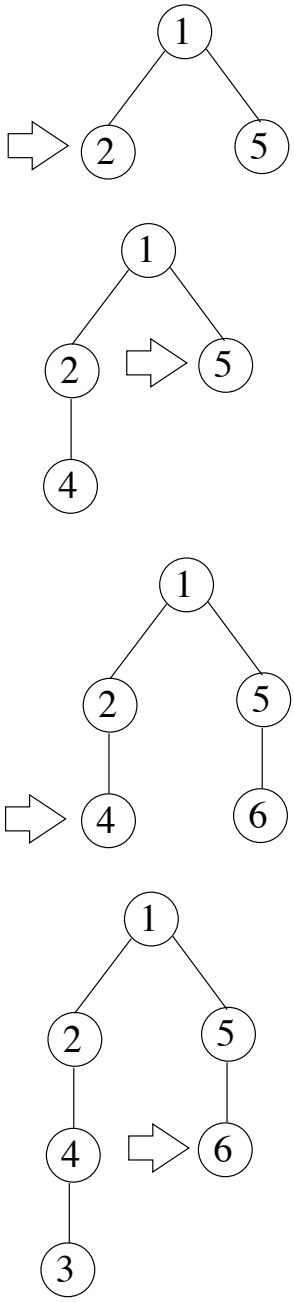
DFS is better for this problem. This is because there are multiple solutions distributed across different leaves at about the same depth, and the search tree has a finite maximum depth.

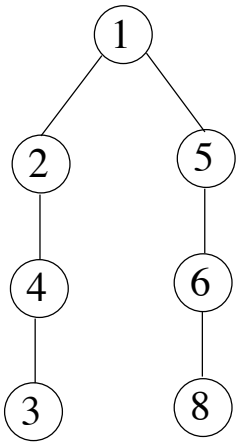
In general, it is not always the case that DFS is better than BFS, or vice versa. The comparison between the two along the dimensions of optimality, completeness, time and space complexity is described in detail in the lecture notes and textbook.

(d) Give similar traces of breadth-first search and depth-first search when the order of expansion of the actions is R, L, S.

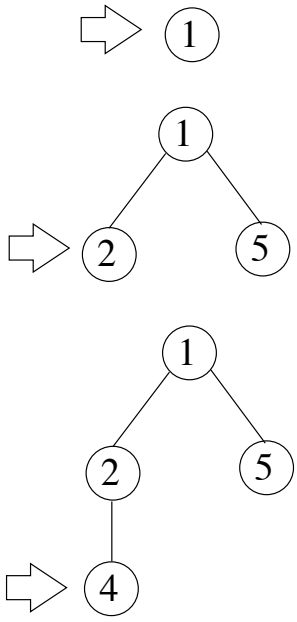
Breadth-first search (R,L,S):

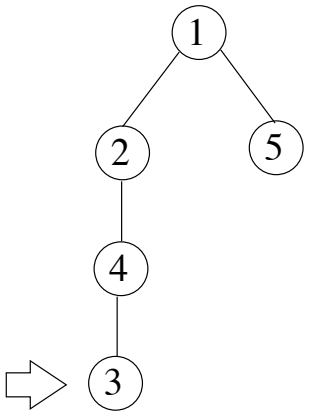


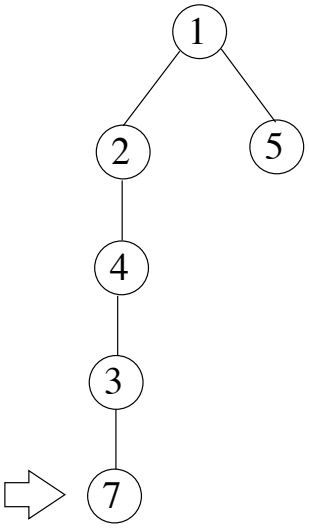

1
2
5
1
2
5
4
1
2
5
4
6
1
2
5
4
6
3



Depth-first search (R,L,S):



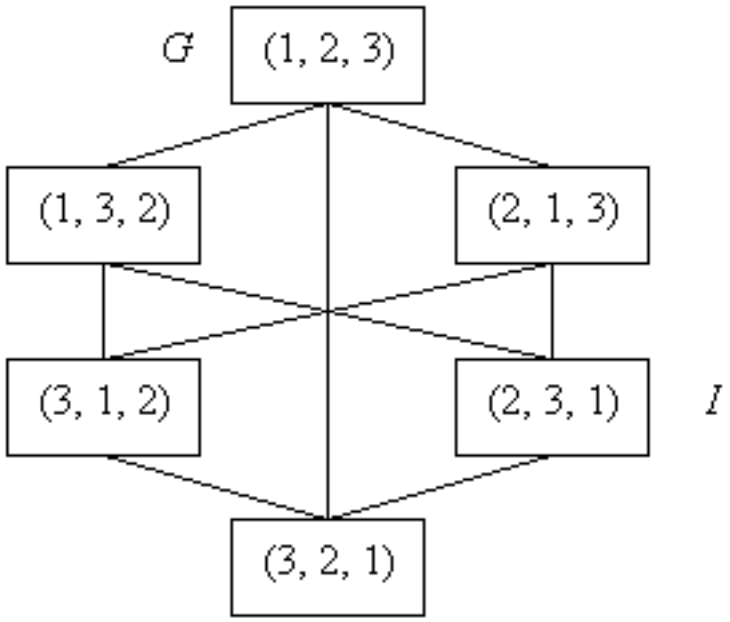




3. Sorting as Searching. We can sort a list of objects using only the operation of swapping two objects in the list. With this in mind, can you cast the sorting problem as a searching problem?

   Draw the state space when sorting the list of numbers $(2, 3, 1)$ in ascending order. What is the minimum number of swaps required? Is the state space fully observable? Deterministic? Episodic? Static? Discrete? Justify your answers when necessary.

   Suppose the input to the sorting problem is a list L of n objects. Then the state space is the set of permutations of L. The swap operations define the edges: suppose $s = (s_1, s_2, \ldots, s_n)$ and $t = (t_1, t_2, \ldots, t_n)$ are two states, then there is an undirected edge linking s and t iff there exists two distinct indices i and j such that $s_i = t_j$ and $s_j = t_i$, and $s_k = t_k$ for all indices k not equal to i or j. Thus the initial state is L and the goal states are the states $(s_1, \ldots, s_n)$ satisfying $s_1 \leq s_2 \leq \ldots \leq s_n$.



The following diagram shows the state space when sorting the list (2,3,1). The initial state is labeled I and the goal state G. As can be seen, we need a minimum of two swaps. The state space is fully observable, deterministic, episodic, static, and discrete.

4. Describe a state space in which iterative deepening search performs much worse than depth-first search.

   Consider a state space with branching factor b such that all nodes at depth d are solutions and all nodes at shallower depths are not solutions.

   Number of search nodes generated by DFS (`GRAPH-SEARCH` *implementation) =* $O(bd)$
   Number of search nodes generated by IDS = $O(b^{d-1})$

5. Consider the graph shown in Figure 2. Let S be the initial state and G be the goal state. The cost of each action is as indicated.

   (a) Give a trace of uniform-cost search.

   If we apply the `TREE-SEARCH` *algorithm, the following are the nodes in frontier at the beginning of the loop and at the end of each iteration of the loop (A node n is followed by its path cost in parenthesis):*

frontier
S(0)
A(1) B(5) C(15)
S(2) B(5) G(11) C(15)
A(3) B(5) B(7) G(11) C(15) C(17)
S(4) B(5) B(7) G(11) G(13) C(15) C(17)
B(5) A(5) B(7) B(9) G(11) G(13) C(15) C(17) C(19)
A(5) B(7) B(9) G(10) S(10) G(11) G(13) C(15) C(17) C(19)
⋮

which is somewhat cumbersome. So, instead, we try the `GRAPH-SEARCH` *algorithm, which is modified to include an extra check in case a shorter path to a frontier state is discovered (see Fig.* 3.14 *on page* 84 *of AIMA* 3*rd edition). We obtain the following, which shows the nodes in frontier and explored at the beginning of the loop and at the end of each iteration of the loop (A node n is followed by its path cost in parenthesis):*

frontier	explored
S(0)	
A(1) B(5) C(15)	S
B(5) G(11) C(15)	S, A
G(10) C(15)	S, A, B

(b) When A generates G which is the goal with a path cost of 11, why doesn't the algorithm halt and return the search result since the goal has been found? With your observation, discuss how uniform-cost search ensures that the shortest path solution is selected.

After G is generated, it will be sorted together with the other nodes in frontier. Since node B has lower path cost than G, it is then selected for expansion. Node B then expands to G with a path cost of 10*, which gives the solution. By always selecting the node with the least path cost for expansion (i.e., nodes are expanded in increasing order of path cost), uniform-cost search ensures that the first goal node selected for expansion is an optimal (i.e., shortest path) solution. A more detailed proof is provided on page* 85 *of AIMA* 3*rd edition.*

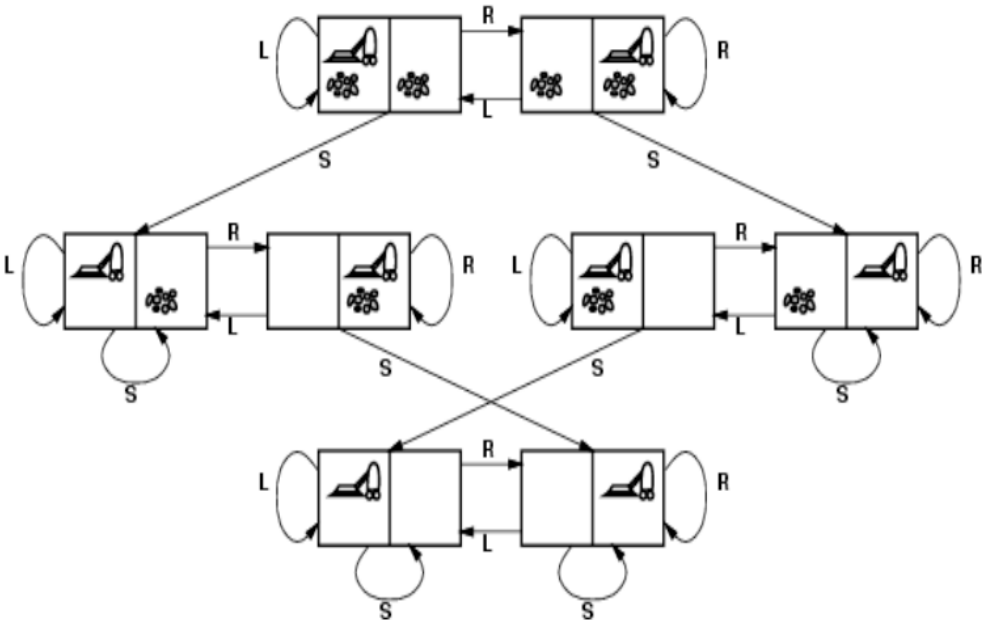


Figure 1: The state space for the vacuum world.

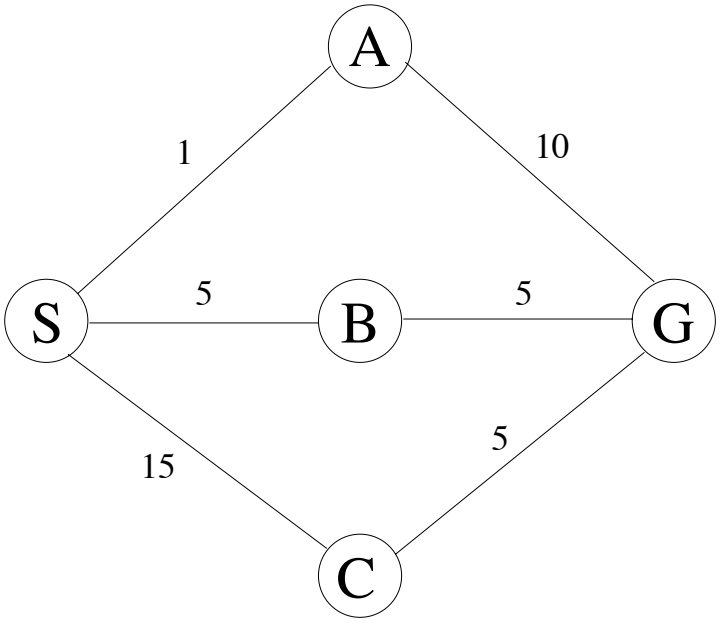


Figure 2: Graph of routes between S and G.