National University of Singapore
School of Computing
CS3243: Foundations of Artificial Intelligence
Solutions for Tutorial 2

Readings: AIMA Chapters 3

1. The Missionaries and Cannibals problem is usually stated as follows (refer to page 90 of the textbook). 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.

   Give the representation of a state in this problem and define its actions (ignoring 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 start state is $(3, 3, 1)$ *and the goal state is* $(0, 0, 0)$*.*

   There are 5 actions:

   (a) *The boat carries 2 missionaries:*
   $(M, C, 0) \rightarrow (M + 2, C, 1), 0 \leq M \leq 1$
   $(M, C, 1) \rightarrow (M - 2, C, 0), 2 \leq M \leq 3$

   (b) *The boat carries 1 missionary:*
   $(M, C, 0) \rightarrow (M + 1, C, 1), 0 \leq M \leq 2$
   $(M, C, 1) \rightarrow (M - 1, C, 0), 1 \leq M \leq 3$

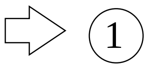
   (c) *The boat carries 2 cannibals:*
   $(M, C, 0) \rightarrow (M, C + 2, 1), 0 \leq C \leq 1$
   $(M, C, 1) \rightarrow (M, C - 2, 0), 2 \leq C \leq 3$

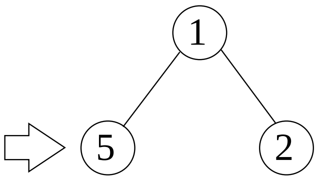
   (d) *The boat carries 1 cannibal:*
   $(M, C, 0) \rightarrow (M, C + 1, 1), 0 \leq C \leq 2$
   $(M, C, 1) \rightarrow (M, C - 1, 0), 1 \leq C \leq 3$

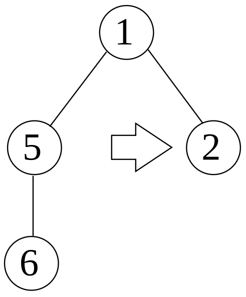
   (e) *The boat carries 1 missionary and 1 cannibal:*
   $(M, C, 0) \rightarrow (M + 1, C + 1, 1), 0 \leq M \leq 2, 0 \leq C \leq 2$
   $(M, C, 1) \rightarrow (M - 1, C - 1, 0), 1 \leq M \leq 3, 1 \leq C \leq 3$

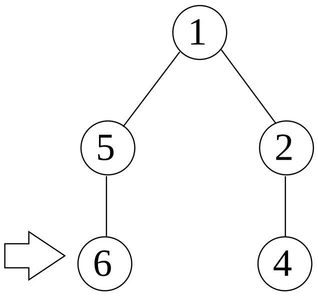
2. Consider the vacuum world problem with the state space shown in Figure 1. With the state numbers assigned in Figure 3.20 (pp 84) of AIMA, let the initial state be state 1 and the goal state be either state 7 or state 8. Assume that the order of expansion of actions is S, R, L.

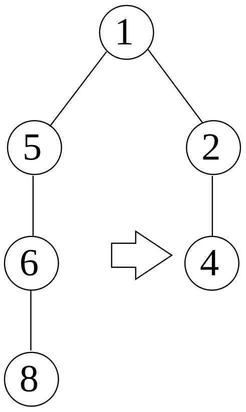
   (a) Give a trace of the breadth-first search algorithm in the style of Figure 3.10 of AIMA. That is, show the search tree at each stage (all repeated states are eliminated).

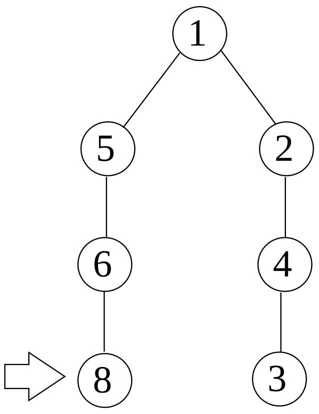

1


1
5
2

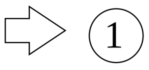

1
5
2
6

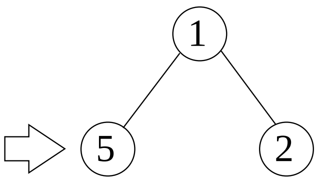

1
5
2
6
4

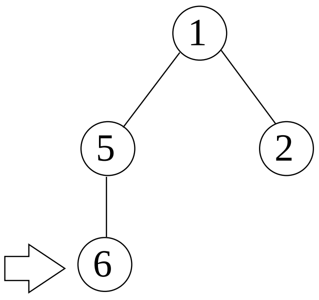

1
5
2
6
4
8

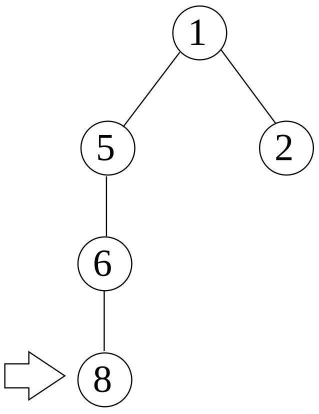


(b) Give a similar trace of the depth-first search algorithm.









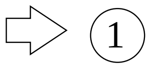
(c) Which of these two search algorithms is better for this problem? Why? Is one search strategy always better than the other in general?

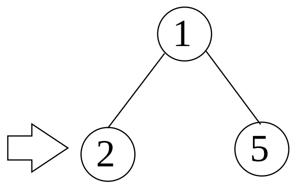
DFS is better for this problem. This is because there are multiple solutions distributed across different leaves at about the same depth, and the search tree has a finite maximum depth.

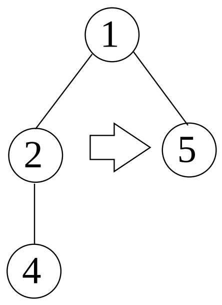
In general, it is not always the case that DFS is better than BFS, or vice versa. The comparison between the two along the dimensions of optimality, completeness, time and space complexity is described in detail in the textbook.

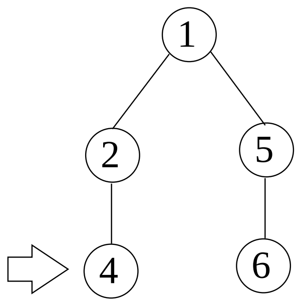
(d) Give similar traces of breadth-first search and depth-first search when the order of expansion of the actions is R, L, S.

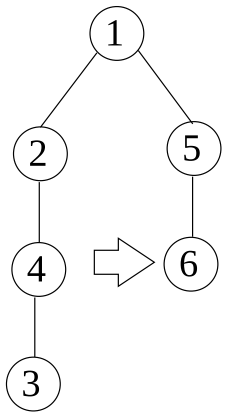
Breadth-first search (R,L,S):

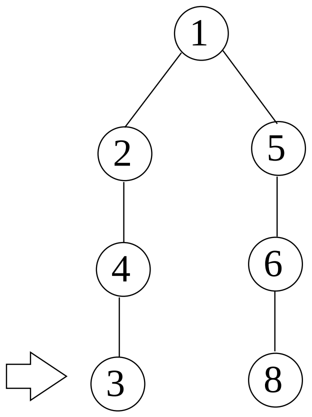


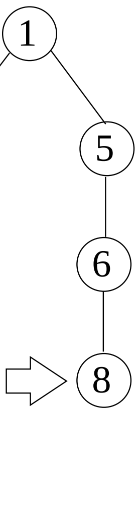




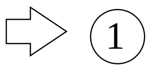


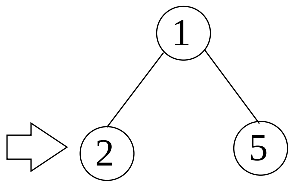


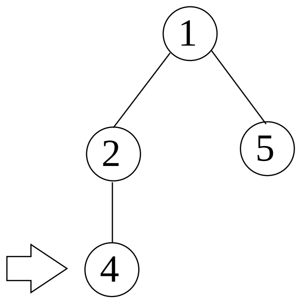


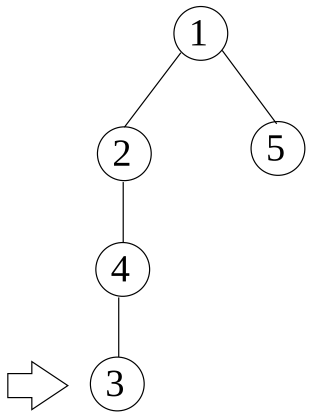


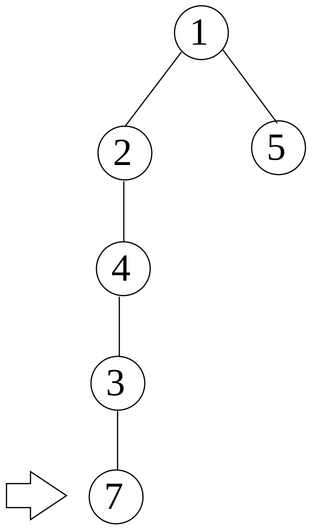
Depth-first search (R,L,S):





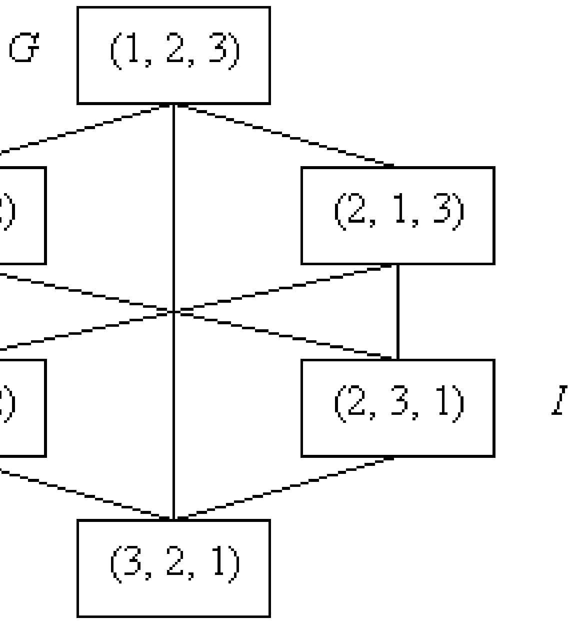






3. Draw the state space when sorting the list of numbers (2,3,1) in ascending order. What is the minimum number of swaps required? Is the state space fully observable? Deterministic? Episodic? Static? Discrete? Justify your answers when necessary.

   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 $s = (s_1, s_2, ..., s_n)$ and $t = (t_1, t_2, ..., t_n)$ 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 $(s_1, ...s_n)$ satisfying $s_1 \leq s_2 \leq ... \leq s_n$.



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 $= O(d)$
   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 fringe 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)*

fringe
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, and obtain the following are the nodes in fringe and closed 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)*

fringe	closed
S(0)	
A(1) B(5) C(15)	S
S(2) B(5) G(11) C(15)	S, A
B(5) G(11) C(15)	S, A
G(10) S(10) G(11) 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 fringe. Since nodes S and B have lower path cost than G, they are 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 cost for expansion, uniform-cost search ensures that the solution it finds is optimal.

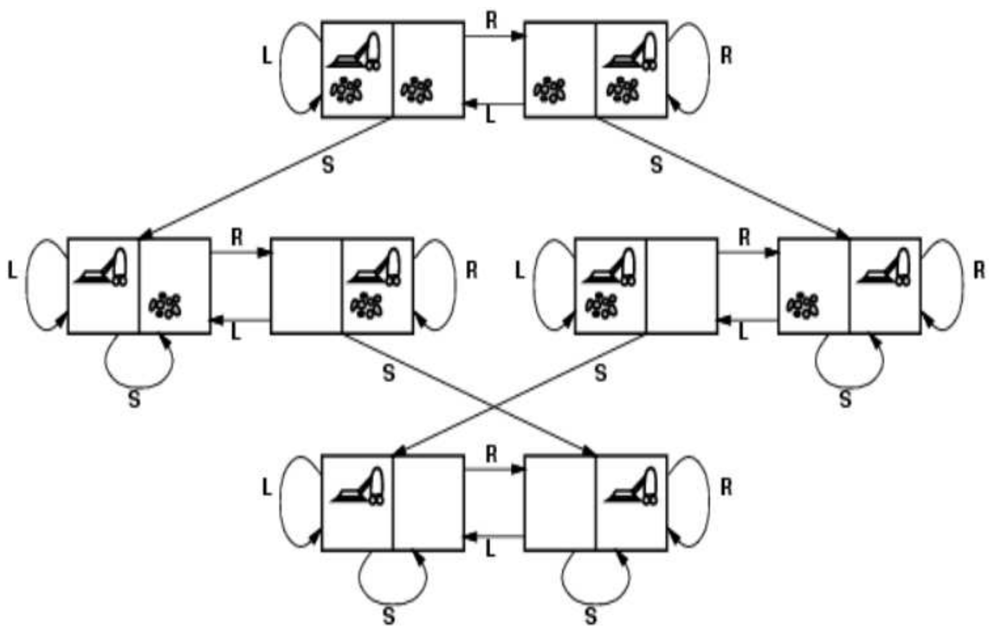


Figure 1: The state space for the vacuum world.

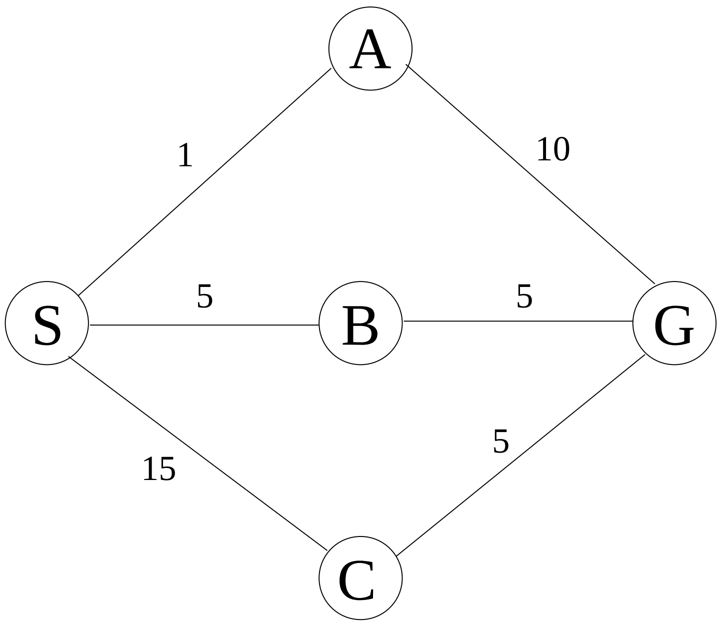


Figure 2: Graph of routes between S and G.