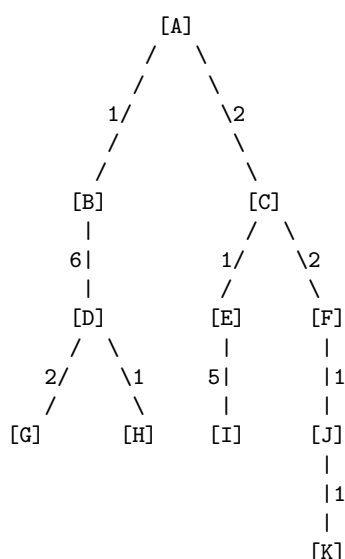


1. (from Exercise 1.2 of Nilsson's *Principles of Artificial Intelligence*) Consider the water-jug problem:

Given a 5-liter (Nilsson was metric in 1980!) jug filled with water and an empty 2-liter jug, how can one obtain precisely 1 liter in the two-liter jug? Water may either be discarded or poured from one jug into another; however, no more than the initial 5 liters is available.

- (a) [6 marks] Specify a problem representation for the water-jug problem. Recall that a problem representation consists of a set of states (which are data structures), a set of operators (which transform states to states), an initial state, and a goal condition. As this is a decision problem, costs can be ignored. Remember to specify the preconditions for your operators.
- (b) [6 marks] Draw the search tree that is implicit in your problem representation. Do not include any cycles in your search tree; that is, do not display any node that has the same state as one of its ancestor nodes. Indicate which node is the initial node and indicate which nodes satisfy the goal condition.
- (c) [1 marks] In layperson's English describe every solution to the waterjug problem that is present in your search tree.
2. (a) [14 marks] Specify a problem representation for the travelling salesman problem (TSP).¹ Of course there are many ways that this could be done. Your task is to do it in such a way that you can use the same operators for every instance of the problem. Thus, the graph cannot be encoded in the operators; it must be encoded in the states. Since the TSP is an optimisation problem, be sure to indicate the operator costs.
- (b) [4 marks] Illustrate your method by drawing the search tree for a small instance of the TSP. In the search tree indicate the initial state, the cost of each arc, all goal states, and the optimal path(s).
3. [4 marks] Consider the following search space, in which every state is shown as a box containing the name of the state. Every arc is labeled with the cost of traversing the arc. A is the initial state, and I and K are goal states.



Show how each of the following search methods finds a solution in this search space by writing down, in order, the names of the nodes removed from the list of states. Assume that the search runs until it returns a goal.

¹Recall that an instance of the TSP is a complete, undirected, weighted graph. A solution to the instance is a cycle through every vertex of the graph that has minimum cost (the sum of the weights on the arcs comprising the cycle).

- (1) Breadth first search
 - (2) Uniform cost search
 - (3) Depth first search
 - (4) Iterative deepening search
4. [4 marks] Of all the search methods we have examined, which would you use to find all goals in a finite search tree? Justify your answer.
5. You are given a search tree containing at least one goal. Consider the following search methods: depth-first search, breadth-first search, depth-limited search, iterative deepening and uniform cost search.
- (a) [2 marks] Which of these search methods is guaranteed to find a goal?
 - (b) [2 marks] Assuming the search tree is finite, which of these search methods is guaranteed to find a goal?
 - (c) [2 marks] Assuming the search tree is finite, which of these search methods is guaranteed to find one of the shallowest goals (that is, it is guaranteed to find a goal and the first goal it finds is guaranteed to be among the shallowest)?
 - (d) [2 marks] Assuming the search tree is finite, which of these search methods is guaranteed to find one of the goals with least cost (that is, it is guaranteed to find a goal and the first goal it finds is guaranteed to have a path cost not more than that of any other goal)?
6. [4 marks] (from problem 3.13 of Russell and Norvig's book *Artificial Intelligence: A Modern Approach*) Describe a search tree in which iterative deepening search performs much worse than depth-first search.
7. [2 marks] Who first used iterative deepening search and for what was it used?
8. [challenge problem] There is very simple way to implement depth-first search with a recursive procedure. When depth-first search is implemented this way it is often referred to as *backtracking*. Unlike the depth-first algorithm we have seen, the backtracking procedure does not explicitly use a list L to store all the unexpanded nodes.
- (a) Specify the backtracking procedure.
 - (b) How does backtracking get away without using L ? That is, without L , how does backtracking remember all the unexpanded nodes?