CS4725/CS6705 - Fall 2017

Assignment # 1

SAMPLE SOLUTIONS

---

3. **(12 marks)** This exercise deals with the formal specification of a task as a search problem. You are not being asked to **solve** this problem, but just to set it up as a formal problem so that it **could** be solved using a search algorithm.

   Consider the following problem. You have three jugs, with capacities of 12 litres, 8 litres and 3 litres. They have no measuring markers on them. You also have access to a water faucet. At any time, you can perform any one of the following tasks that are applicable.

   - fill one of the three jugs (to the top) with water from the faucet
   - empty the entire contents of one of the jugs onto the ground
   - pour from one jug into a second jug until the second jug is full (leaving any remaining water in the first jug) or until the first jug is empty

   Your goal is to end up with **exactly one litre** in one of the jugs.

   Suppose that you have decided to use (x, y, z) as your state representation, where x is the number of litres of water in the 12-litre jug, y is the number of litres of water in the 8-litre jug, and z is the number of litres of water in the 3-litre jug.

   Provide a precise formulation of the task as a search problem.

   (a) (1 mark) What is the initial state?

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

   (b) (1 mark) What is the goal test?

   For a given state (x, y, z), the goal test would be to check whether $(x == 1)$ or $(y == 1)$ or $(z == 1)$.

   (c) (1 mark) What is the cost of a given path (sequence of actions)?
   (There are different possible answers here, but you can just provide one possible approach and give a brief justification.)

   The simplest path cost function would just be to count the number of actions performed. However, it is quite reasonable to specify that certain actions would have a higher cost than others. For example, we might state that each action has a cost equal to the amount of water poured during that action. Another option would be to worry only about the total amount of water poured from the faucet (i.e., transferring water from one jug to another is free, since we only want to count the total amount of water used). Another option would be to have a similar measure, but to add a high penalty for any water that is wasted by pouring it onto the ground.

(d) Suppose that you have decided on the following list of possible actions: $F12$ (filling the 12-litre jug), $F8$, $F3$, $E12$ (emptying the 12-litre jug), $E8$, $E3$, $12T8$ (transferring from the 12-litre jug to the 8-litre jug), $12T3$, $8T12$, $8T3$, $3T12$, and $3T8$.

i. (3 marks) For each action, specify how you would check if the action is allowed or possible in the current state. (For example, "if $y = 0$")

ii. (6 marks) For each action, specify how the next state $s' = (x', y', z')$ would be computed from the current state $s = (x, y, z)$.

(Note that this could be as simple as "$s' = (x, 5, z)$" or could be something more complicated like "if _____ then $s' =$ _____ else $s' =$ _____".)

Let $s = (x, y, z)$ be the current state. The table below shows the conditions that must be satisfied in order for each action to be allowed and the resulting state after the action is performed.

Action	Conditions	Resulting state
$F12$	$x < 12$	$(12, y, z)$
$F8$	$y < 8$	$(x, 8, z)$
$F3$	$z < 3$	$(x, y, 3)$
$E12$	$x > 0$	$(0, y, z)$
$E8$	$y > 0$	$(x, 0, z)$
$E3$	$z > 0$	$(x, y, 0)$
$12T8$	$x > 0$ and $y < 8$	if $x + y \leq 8$ then $(0, x + y, z)$ else $(x + y - 8, 8, z)$
$12T3$	$x > 0$ and $z < 3$	if $x + z \leq 3$ then $(0, y, x + z)$ else $(x + z - 3, y, 3)$
$8T12$	$y > 0$ and $x < 12$	if $y + x \leq 12$ then $(x + y, 0, z)$ else $(12, y + x - 12, z)$
$8T3$	$y > 0$ and $z < 3$	if $y + z \leq 3$ then $(x, 0, y + z)$ else $(x, y + z - 3, 3)$
$3T12$	$z > 0$ and $x < 12$	if $z + x \leq 12$ then $(z + x, y, 0)$ else $(12, y, z + x - 12)$
$3T8$	$z > 0$ and $y < 8$	if $x + y \leq 8$ then $(x, z + y, 0)$ else $(x, 8, z + y - 8)$

4. **(11 marks)** Suppose that you have a 1 x 4 board containing 2 green marbles and 1 red marble, in the initial state pictured below.

A goal state is defined as one in which there are no green marbles anywhere to the left of the red marble. There are two possible types of actions:

- A marble can *slide* to the left or right by one position into a blank square. **Cost: 1**
- A marble can *jump* over one other marble to its left or right, provided that there is a blank square at the destination. **Cost: 2**

(a) (2 marks) Suppose that we use the representation $GGRO$ to represent the initial state above, where G represents a green marble, R a red marble, and O an open space. Using this representation, list all of the possible states in the state space for this problem. Circle the states that are goal states.

Here is a complete list of possible states for this problem, with the goal states enclosed in a box:

GGOR, GGRO, GOGR, GORG, GRGO, GROG,
OGGR, OGRG, [ORGG], [RGGO], [RGOG], [ROGG]

(b) (6 marks) Use the **uniform-cost search** algorithm to find a lowest-cost solution to the problem.

- Draw your search tree. As we did in class, list the order in which nodes are chosen for expansion and show the contents of the priority queue after each expansion. Each item in the priority queue should be a state name, along with the path cost from the root node to that node – *e.g.*, $GROG^3$.
- When a node is generated that has already been expanded, or if a node with the same state is already on the priority queue with a lower or equal cost, then include the node in your tree, but put the state name in parentheses – *e.g.*, $(GGRO)$, and do not add it to the priority queue.
- When adding nodes to the priority queue, break any ties alphabetically. For example, $GORG^3$ would come before $GROG^3$.

The search tree and priority queue contents are shown in a separate file posted on Desire2Learn.

(c) (2 marks) What is the solution path found by the algorithm and what is its path cost?

The solution path found is $GGRO \rightarrow GORG \rightarrow GROG \rightarrow ORGG$, with a path cost of 5.

(d) (1 mark) What is the branching factor of your search tree?

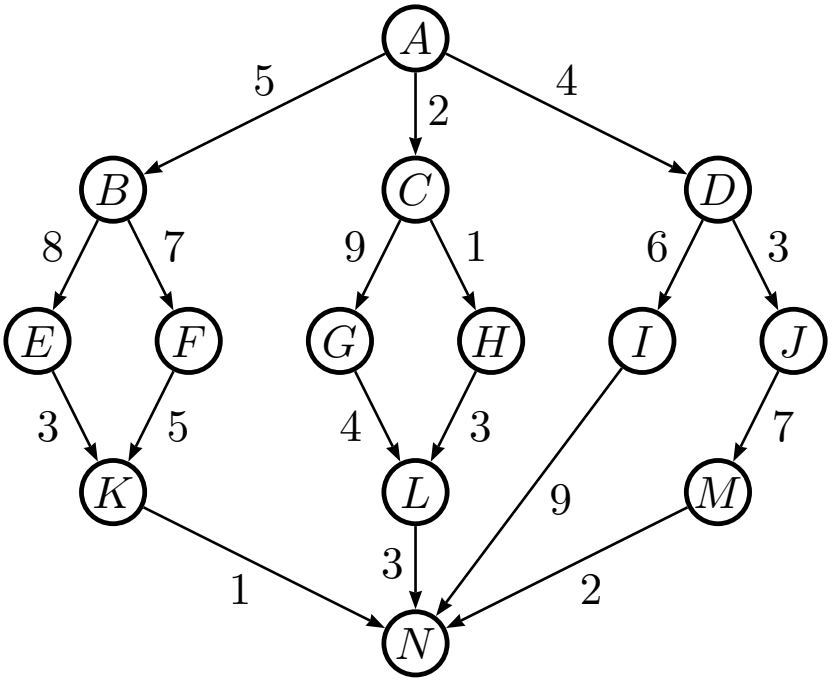
The maximum number of children (successors) for any node in the search tree is 3, and so the branching factor is 3.

5. **(20 marks)** Consider the search problem involving getting from A to N in the diagram pictured below. Edges are labelled with the cost associated with following that edge.

   For parts (d) and (e), use the following heuristic function h:

Node n	A	B	C	D	E	F	G	H	I	J	K	L	M	N
$h(n)$	9	11	8	11	2	4	3	4	6	7	1	2	1	0

   For questions below that ask for a path, please give answers in the form $A \to D \to J \to M \to N$.



   (a) (4 marks) Depth-first search
   (Successors of the same node should be added to the frontier so that they will be selected in alphabetical order.)

   - In what order would nodes be selected for expansion?
   - What solution path would be returned by the algorithm and what is its cost?

   The order of expansion would be A, B, E, K, N.

   The solution path returned would be $A \to B \to E \to K \to N$, with a cost of 17.

(b) (4 marks) Breadth-first search
(Successors of the same node should be added to the frontier so that they will be selected in alphabetical order.)

- In what order would nodes be selected for expansion? *(Recall from our discussion in class that BFS can stop when a goal node is generated, rather than waiting until one is selected for expansion.)*
- What solution path would be returned by the algorithm and what is its cost?

The order of expansion would be $A, B, C, D, E, F, G, H, I$.

The solution path returned would be $A \to D \to I \to N$, with a cost of 19.

(c) (4 marks) Uniform cost search
(If two or more nodes have the same g value, then use alphabetical order to break ties.)

- In what order would nodes be selected for expansion?
- What solution path would be returned by the algorithm and what is its cost?

The order of expansion would be A, C, H, D, B, L, J, N.

The solution path returned would be $A \to C \to H \to L \to N$, with a cost of 9.

(d) (4 marks) Greedy search, with the heuristic function h given above
(If two or more nodes have the same h value, then use alphabetical order to break ties.)

- In what order would nodes be selected for expansion?
- What solution path would be returned by the algorithm and what is its cost?

The order of expansion would be A, C, G, L, N.

The solution path returned would be $A \to C \to G \to L \to N$, with a cost of 18.

(e) (4 marks) A^* search, with the heuristic function h given above
(If two or more nodes have the same f value, then use alphabetical order to break ties.)

- In what order would nodes be selected for expansion?
- What solution path would be returned by the algorithm and what is its cost?

The order of expansion would be A, C, H, L, N.

The solution path returned would be $A \to C \to H \to L \to N$, with a cost of 9.

6. **(5 marks)** In our discussions, we have always assumed that actions have nonnegative costs. Write a short proof that, if actions could have arbitrarily large negative costs, then any optimal search algorithm would have to explore the entire state space.

   Hint: Structure this as a proof by contradiction. Suppose that a search algorithm could find an optimal goal node G that has a path cost C, **without** having expanded at least one node N in the search space.

   Proof (by contradiction): Suppose not. Suppose that a search algorithm could find an optimal goal node G that has path cost C, **without** having explored the entire state space.

   Let N be one of the nodes that was not expanded during the search, and suppose it has a path cost D. It is possible that, from node N, there exists an action with a negative cost that leads directly to some goal node G'. More specifically, suppose that the action that takes us from N to G' has a cost that is less than $C - D$. Therefore, this goal node G' will then have a path cost less than $D + (C - D)$, i.e. less than C, which proves that the path to the goal node G was not, in fact, optimal.