OSU CSE 3521/5521
Homework #1: Problem Set

Release Date: September 4th, 2020

Submission Instructions

Due Date: September 21st (23:59 ET), 2020

Submission: Please submit your solutions in a single PDF file named HW_1_name.number.pdf (e.g., HW_1_serai.1.pdf) to Carmen. You may write your solutions on paper and scan it, or directly type your solutions and save them as a PDF file. *Submission in any other format will not be graded.*

Collaboration: You may discuss with your classmates. However, you need to write your own solutions and submit separately. Also in your written report, you need to list with whom you have discussed for each problem. Please consult the syllabus for what is and is not acceptable collaboration.

1 Search [12 points]

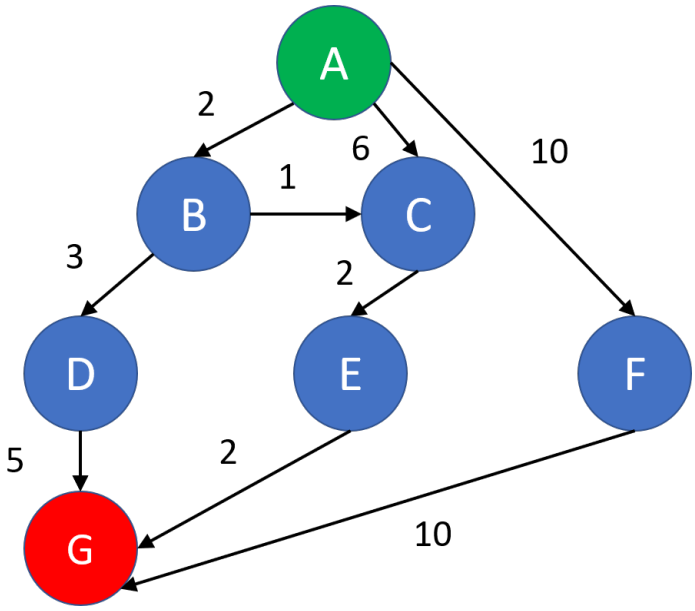

Figure 1: State space graph

Figure 1 shows a state space graph, in which state A is the start state and state G is the goal state. Each edge is directed and associated with a cost. Please apply the five *tree* search algorithms — DFS, BFS, UCS, greedy, and A* — you learned in the lectures to find the solution.

1. Please apply DFS to search the solution. Please consider that your fringe is implemented by stack and you always push the successor states following the alphabet order (A-to-Z). For example, if you are expanding node A, then you will push its three successor states in the order of B, C, F into the stack.

   (a) How many nodes do you need to expand (including expanding A and and the node containing G) until you find the solution? **[1 points]**

   (b) What is the solution (i.e., state sequence) outputted by DFS? **[1 points]**

2. Please apply BFS to search the solution. Please consider that your fringe is implemented by queue and you always enqueue the successor states following the alphabet order (A-to-Z). For example, if you are expanding node A, then you will enqueue its three successor states in the order of B, C, F into the stack.

(a) How many nodes do you need to expand (including expanding A and and the node containing G) until you find the solution? **[1 points]**

(b) What is the solution (i.e., state sequence) outputted by BFS? **[1 points]**

3. Please apply UCS to search the solution. If there are two candidate nodes (i.e., partial plans) whose costs are the same, expand the one whose number of states is smaller. For example, if A-B-D (cost 5) and A-B-C-E (cost 5) are both in your fringe, you will expand A-B-D before expanding A-B-C-E.

   (a) How many nodes do you need to expand (including expanding A and and the node containing G) until you find the solution? **[1 points]**

   (b) What is the solution (i.e., state sequence) outputted by UCS? **[1 points]**

4. Please apply greedy search to search the solution. Please apply the following heuristic h: the minimum number of edges to travel to achieve G. For example, h(B) is 2. If there are two candidate nodes (i.e., partial plans) whose heuristic values are the same, expand the one whose number of states is smaller. For example, if A-B-D ($h = 1$) and A-B-C-E ($h = 1$) are both in your fringe, you will expand A-B-D before expanding A-B-C-E.

   (a) How many nodes do you need to expand (including expanding A and and the node containing G) until you find the solution? **[1 points]**

   (b) What is the solution (i.e., state sequence) outputted by greedy search? **[1 points]**

5. Please apply A* search to search the solution. Please apply the following heuristic h: the minimum number of edges to travel to achieve G. For example, h(B) is 2. If there are two candidate nodes (i.e., partial plans) whose $g + h$ values are the same, expand the one whose number of states is smaller. For example, if A-B-D ($g+h = 6$) and A-B-C-E ($g+h = 6$) are both in your fringe, you will expand A-B-D before expanding A-B-C-E.

   (a) How many nodes do you need to expand (including expanding A and and the node containing G) until you find the solution? **[1 points]**

   (b) What is the solution (i.e., state sequence) outputted by A* search? **[1 points]**

6. (a) What is the cost of the optimal solution? **[1 points]**

   (b) Which algorithm can achieve it? (Write down all the algorithms that can achieve it) **[1 points]**

2 Logic [3 points]

Please show that $\neg(A \wedge B \wedge C)$ is the same as $\neg A \vee \neg B \vee \neg C$, by filling in the truth table in Figure 2. T: True; F: False. Please make sure you answer all the 16 cells correctly. No partial grades will be given for this question.

A	B	C	¬(A∧B∧C)	¬A∨¬B∨¬C
T	T	T		
T	T	F		
T	F	T		
T	F	F		
F	T	T		
F	T	F		
F	F	T		
F	F	F		

Figure 2: Truth table