OSU CSE 3521
Homework #1: Problem Set

Release Date: September 7th, 2021

Submission Instructions

Due Date: September 21st (23:59 ET), 2021

Submission: Please submit your solutions in a single PDF file named HW1_name_number.pdf (e.g., HW1_chao_209.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.*

We highly recommend that you write down the derivation of your answers, and highlight your answers clearly!

Collaboration: You may discuss with your classmates at a very high level. However, you need to write your own solutions and submit them separately by yourself. Also in your written report, you need to list with whom you have discussed for each problem (please do so in the first page). Please consult the syllabus for what is and is not an acceptable collaboration.

1 Search [20 points]

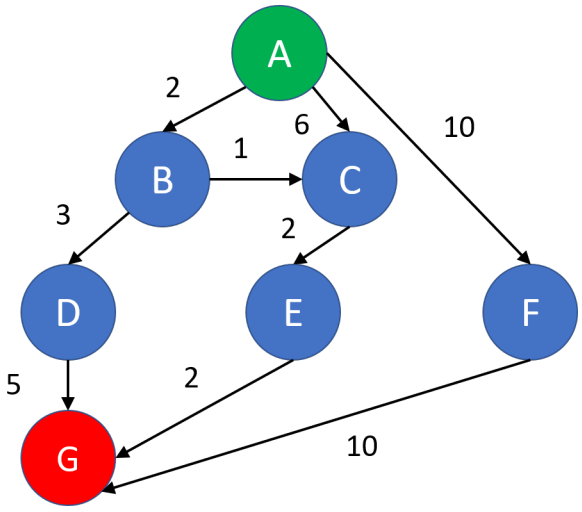

Figure 1: State space graph

Figure 2 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 (not graph search) algorithms — DFS, BFS, UCS, greedy, and A* — that you learned in the lectures to find the solution. Please read the hints at the end of this question before answering the question.

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? **[1.5 points]**

   (b) What is the solution (i.e., state sequence) outputted by DFS? Your solution should contain both states A and G. **[1.5 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 queue.

   (a) How many nodes do you need to expand (including expanding A and and the node containing G) until you find the solution? **[1.5 points]**

(b) What is the solution (i.e., state sequence) outputted by BFS? Your solution should contain both states A and G. **[1.5 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 first. 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? **[1.5 points]**

   (b) What is the solution (i.e., state sequence) outputted by UCS? Your solution should contain both states A and G. **[1.5 points]**

4. Please apply greedy search to search the solution. Please apply the following heuristic h: the minimum number of edges to travel through 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 first. 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? **[1.5 points]**

   (b) What is the solution (i.e., state sequence) outputted by greedy search? Your solution should contain both states A and G. **[1.5 points]**

5. Please apply A* search to search the solution. Please apply the following heuristic h: the minimum number of edges to travel through 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 first. 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? **[1.5 points]**

   (b) What is the solution (i.e., state sequence) outputted by A* search? Your solution should contain both states A and G. **[1.5 points]**

6. (a) What is the cost of the optimal solution? **[2.5 points]**

   (b) Which algorithm can achieve it? (Write down all the algorithms that can achieve it) **[2.5 points]**

Hint:

- Since you are working on tree searches, you may expand multiple nodes (i.e., partial paths) that have the same last states and you should count all of them. For example, you may expand A-B-C and then A-C and you should count both of them.
- As an example of how to count how many nodes you expand, if you expand "A", and then "F" (i.e., A-F), and then "B" (i.e., A-B), and then "G" (i.e., A-F-G) in the search process, then you expand 4 nodes. You expand a node (i.e., A-F) only when you check if its last state (i.e., "F") is a goal or not (if not, adding the successors into the fringe), not when you put that node (i.e., A-F) into your fringe.
- A solution is the state sequence (in the problem set) or action sequence (in the programming set) that can directly lead you from the start state to the goal state. It is NOT the sequence of nodes that you expand in your search process. For instance, in the above example, you expand 4 nodes, but your solution is A-F-G.

2 Adversarial Search [20 points]

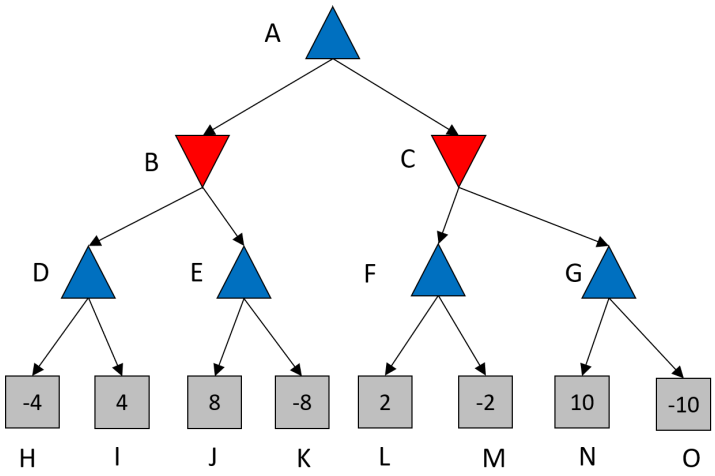

Figure 2: An adversarial search tree

Figure 2 shows an adversarial search tree, in which the blue nodes (i.e., "A", "D", "E", "F", "G") are max nodes and the red nodes (i.e., "B", "C") are min nodes. The gray nodes are the goal states, each has its utility value (the *higher*, the better). Each max or min node has two actions: "go left" or "go right". For example, if you are at "A", then you can go left to arrive at "B" or go right to arrive at "C".

Note that, you are provided with this entire adversarial search tree, which has not been traversed/expanded yet. Your job is to do minimax search, determining each node's value and finding out the minimax solution from "A" to the goal states.

1. Please perform the **adversarial (minimax) search** to

   (a) fill in the value of each node; [**3.5 points**]

A	B	C	D	E	F	G

   (b) find out the minimax predicted state-sequence. A predicted state-sequence is a "path" from "A" to the gray (leaf) nodes. For example, "A"-"C"-"G"-"N" (or equivalently, right-right-left) can be a valid predicted state-sequence BUT it may not be the minimax predicted state-sequence. [**2.5 points**]

2. Please apply the α-β **pruning** to redo the minimax search. Note that, in α-β pruning, you are expanding/traversing the tree in a depth-first-search fashion, in which you will always choose the left action first and then the right action. In other words, the node expansion order will be "A"-"B"-"D"-"H"-"I"-"E"-"J"-"K"-"C"-"F"-"L"-"M"-"G"-"N"-"O", but some nodes might be pruned/skipped without being visited. Every min or max node will have its (α, β), which are initialized by its parent and may be updated when its children are being expanded. The initial (α, β) to the node "A" is $(-\infty, \infty)$.

   Please see the lecture slides for more details. Specifically, please follow the "Alpha-Beta Implementation" slide to update each min or max node's value and its (α, β). The slide provides an recursive implementation for the depth-first-search fashion. If you are visiting a leaf node, you simply return its utility value. Note that, (α, β) in that slide are called/passed by values.

   (a) Fill in the value of each node. If a node is pruned (i.e., no need to visit according to α-β pruning), please put X as the answer for that node. For a gray node, if it is visited, simply fill in its utility value. Note that, the values of the nodes can be different from the previous question. **[7.5 points]**

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

   (b) Find out the minimax predicted state-sequence. A predicted state-sequence is a "path" from "A" to the gray (leaf) nodes. For example, "A"-"C"-"G"-"N" (or equivalently, right-right-left) can be a valid predicted state-sequence BUT it may not be the minimax predicted state-sequence. **[3 points]**

3. Now let us consider the **expectimax search** by changing each red min node to an expectation node. The probability for the left and right actions are 50% and 50%. Please fill in the value of each node. **[3.5 points]**

A	B	C	D	E	F	G

3 Logic-1 [4 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 3. T: True; F: False. Please make sure that 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 3: Truth table

4 Logic-2 [6 points]

Suppose that the following logical sentences are all True and they are in your knowledge base.

$$\begin{aligned} C \wedge F &\Rightarrow \neg B \\ B &\Rightarrow F \\ B &\Rightarrow C \end{aligned} \tag{1}$$

Question: Is B True or False?

Hint: If B is True, given that $B \Rightarrow C$ is True, then C is True.
Please write down your derivation (no more than 10 lines), not just the answer.