CSE 415 Autumn 2025 Assignment 3

Last name:_____ First name:_____

UWNetID: _____

Due Monday night October 27 via Gradescope at 11:59 PM. You may turn in either of the following types of PDFs: (1) Scans of these pages that include your answers (handwriting is OK, if it's clear), or (2) Documents you create with the answers, saved as PDFs. When you upload to Gradescope, you'll be prompted to identify where in your document your answer to each question lies.

Do all four exercises. These are intended to take 30-60 minutes each if you know how to do them. Each is worth 25 points. If any corrections have to be made to this assignment, these will be posted in ED.

This is an *individual-work* assignment. Do not collaborate on this assignment. Do *not use AI* to answer the questions for you.

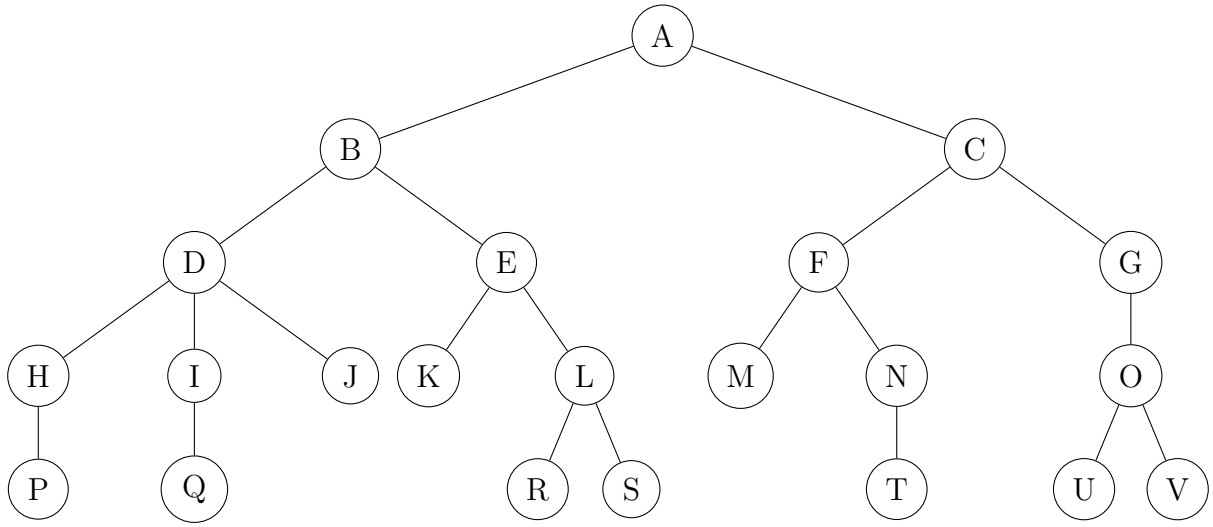
Prepare your answers in a neat, easy-to-read PDF. Our grading rubric will be set up such that when a question is not easily readable or not correctly tagged or with pages repeated or out of order, then points will be deducted. However, if all answers are clearly presented, in proper order, and tagged correctly when submitted to Gradescope, we will award a 5-point bonus. (Thus the maximum points for A3 will be 105).

If you choose to typeset your answers in Latex using the template file for this document, please put your answers in blue while leaving the original text black.

Version 25-10-15. If any corrections to this document are required, this will be announced in ED.

1 Basic Search (Eric)

Use the following tree to answer the questions below comparing Breadth First Search, Depth First Search, and Iterative-Deepening Depth First Search. Assume that children are visited from left to right.



(a) (2 points) Write out the order that the nodes are expanded in using Breadth First Search, starting from A and searching to O.

(b) (2 points) Write out the order that the nodes are expanded in using Depth First Search, starting from A and searching to O.

(c) (2 points) Write out the order that the nodes are expanded in using Iterative-Deepening Depth First Search, starting from A and searching to O. If a node is repeated, make sure to include it each time it is expanded.

(d) (3 points) Which of the three search algorithms (BFS, DFS, IDDFS) has the smallest maximum size of the open list while searching from A to R? What is the maximum size of the open list for that algorithm?

(e) (1 points) True or False: BFS, DFS, and IDDFS will each return the same path on this tree starting at A searching to R.

(f) (1 points) True or False: Given the same start and goal state, BFS, DFS, and IDDFS will always return the same path for any search graph (not necessarily a tree).

(g) (4 points) What are the advantages of using IDDFS over DFS and BFS for very large or infinite graphs in terms of (i) memory usage, (ii) completeness, and (iii) computation time?

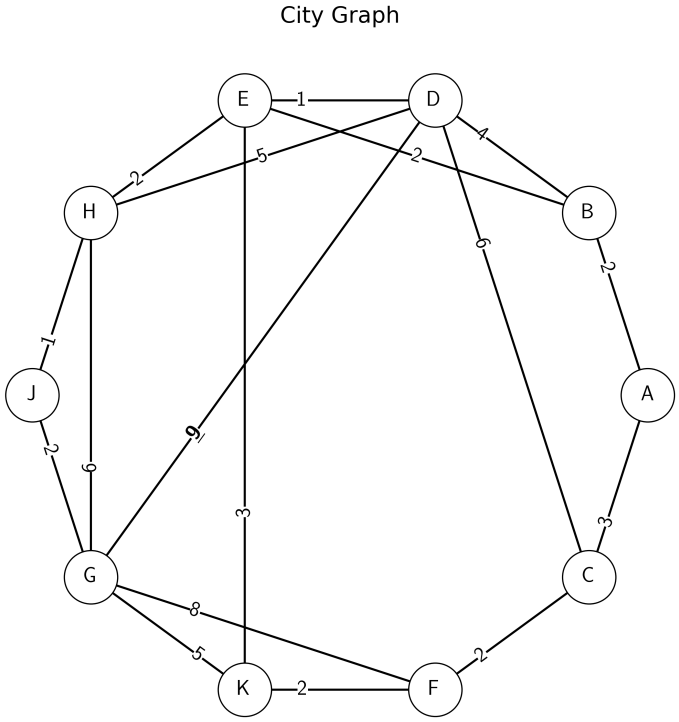
(h) (4 points) In terms of complexity, explain why IDDFS is considered to have the same time complexity as BFS even though it may revisit nodes multiple times.

(i) (3 points) Suppose you are searching a tree where the goal might be at any depth, including very deep or potentially infinite depth. Which of the three search algorithms (BFS, DFS, IDDFS) would guarantee to find the goal if it exists? Justify your answer with a one sentence explanation.

(j) (3 points) If the tree structure is unknown and could be either very wide or very deep, which of the three search algorithms (BFS, DFS, IDDFS) is the most "robust" choice that handles both cases reasonably well? Justify your answer with a one sentence explanation.

2 A* Operation (Sahil)

(25 points) This exercise is intended to reinforce your understanding of the mechanics of graph-search algorithms with and without heuristics. Consider the following graph, which represents cities and the distances between them. Each circle represents a city, and the edges between them represent roads from one city to another. The number next to an edge indicates the travel distance along that edge. For example, in the graph, the edge from city A to city B is labeled with 2, which means the travel distance from city A to city B is 2 units. $\underline{9}$ is underlined to avoid confusion with 6. So G-H is 6 units, C-D is also 6 units, and D-G is $\underline{9}$ units.



The problem is to find a lowest-distance path from city A to city G. You'll solve this problem first using Uniform Cost Search (UCS) and then solve it again using A* Search. For the A* search, you are given a heuristic, which estimates the distance from each city to the goal, city G.

Heuristic Values for A* Search: These are the heuristic values to use in this example. For each city they represent an estimate of the distance from that city to city G. The heuristic values are used only in A* search.

city s:	A	B	C	D	E	F	G	K	H	J
heuristic $h(s)$:	8	6	8	6	3	5	0	3	3	0

(a) **Uniform Cost Search (UCS)**: Apply UCS to find a lowest-distance path from city A to city G.

- (i) Show how the OPEN list (a.k.a. the "fringe") evolves during this search below, starting with the version before the beginning of the first iteration of the UCS main loop that is shown already with starting city A and its cumulative distance 0. Show the new version of the OPEN list at the end of each iteration. Be sure to include each city's cumulative distance (g-value). Within each list, show these pairs in order of increasing distance (even though a real priority queue doesn't have to use a sorted list in its implementation). Breaking any ties using alphabetical ordering (5 points)

  [(A, 0)]

- (ii) List the nodes in the order they are added to the CLOSED list (2 points)

- (iii) Provide the final lowest-distance path found and its total distance (2 points)

(b) **A* Search**: Apply A* search with the given heuristic values to find a lowest-distance path from city A to city G.

- (iv) Show how the OPEN list (a.k.a. the "fringe") evolves during this search, but use the cities' f values instead of their g values. As with UCS, show each snapshot of the OPEN list with cities ordered by associated values, and break any ties using alphabetical ordering. (5 points)

[(A, 7)]

- (v) List the nodes in the order they are added to the CLOSED list (2 points)

- (vi) Provide the final path and distance (2 points)

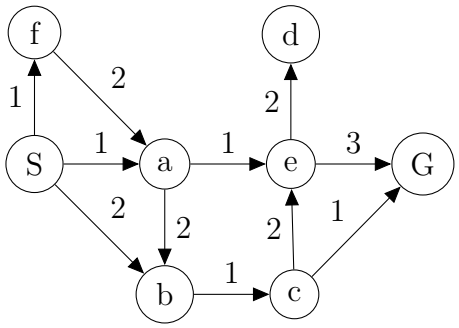
(c) **Node Expansion Comparison**: Count the number of nodes expanded in each of UCS and A* search.

- (vii) How many nodes did UCS expand? (2 points)

- (viii) How many did A* expand? (2 points)

- (ix) Which method expanded fewer nodes, and why? (3 points)

3 Properties of Heuristics (Emilia)



For the following questions, consider three heuristics h_1, h_2, h_3. The table below indicates the estimated cost to goal, G, for each of the heuristics for each node in the search graph.

state (s)	S	a	b	c	d	e	f	G
heuristic $h_1(s)$	4	3	2	2	3	3	6	0
heuristic $h_2(s)$	4	3	2	1	1	1	1	0
heuristic $h_3(s)$	4	3	2	1	4	2	5	0

(a) (2 points) What does it mean for a heuristic to be "admissible"?

(b) (3 points) Which heuristics among $\{h_1, h_2, h_3\}$ shown above are admissible? For heuristics which are not admissible, **identify the nodes where admissibility is violated**.

(c) (2 points) What does it mean for a heuristic to be "consistent"?

(d) (3 points) Which heuristics are consistent? For heuristics which are not consistent, **identify the edges where consistency is violated**.

(e) (2 points) What does it mean for one heuristic to dominate another heuristic?

(f) (4 points) For this question, you need to compare two **different** heuristics that are both admissible and consistent. If you need to, you should modify heuristic values so that you can satisfy these requirements (**note which heuristic(s) you modified, what your modifications were, and what issues your modifications fixed**). Do not modify any heuristic that is already admissible and consistent. Looking at your two admissible and consistent heuristics, which would you consider the best heuristic? Explain your reasoning.

(g) (3 points) Explain why, when using the $A*$ search algorithm, it is important to continue the expansion process until the goal state is removed from the OPEN list and becomes the current state, rather than terminating as soon as the GOAL state is found as a successor of an expanded state.

(h) (3 points) In A* search, if the heuristic overestimates the true distance (i.e., $h(n) \geq$ actual distance to the goal), what is the potential effect on the search? How could this affect the optimality of the solution?

(i) (3 points) Explain why A* search is guaranteed to find the optimal solution when the heuristic is admissible (i.e., $h(s)$ never overestimates the actual distance to the goal). Does this apply to UCS as well?

4 Adversarial Search and Alpha-Beta Pruning (Krish)

Consider the Tic-Tac-Toe game tree shown in the figure.

(a) (8 points) Compute the static values of the leaf nodes in the given tree using the function $v(s)$, where v is defined below, as a weighted sum of three features.

$$v(s) = 1 \cdot f_1(s) + 10 \cdot f_2(s) + 100 \cdot f_3(s)$$

Here $f_1(s)$ counts the number times that there is a line containing an X alone minus the number of times there is a line containing an O alone. Also, $f_2(s)$ counts the number of unblocked instances of two Xs in a row, minus the number of unblocked instances of two Os in a row. Finally, $f_3(s)$ is the number of 3-in-a-row instances of Xs, minus the similar number of Os. (This is equivalent to the static evaluation function used in lecture and a worksheet.)

Show these values underneath states F through Q in the diagram. Hint: $v(F) = 11$.

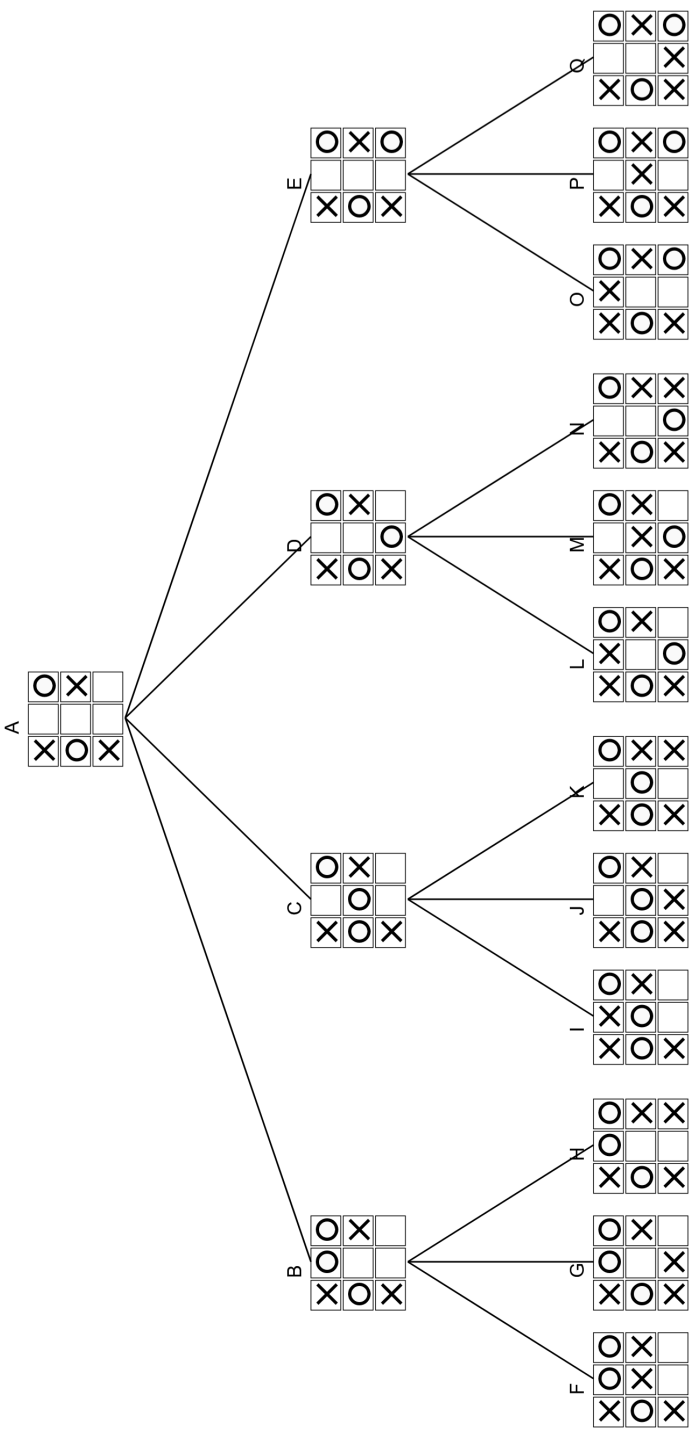
(b) (5 points) Use straight minimax to determine the values of the internal nodes in the tree, and write those values in the table below. The root node is a max node.

node	Minimax value
A	
B	
C	
D	
E	

Table 4.1: Minimax values for internal nodes

(c) (8 points) Next, redo the adversarial search using alpha-beta pruning on the same tree. Fill in the values in Table 4.2 below as you go. For α and β values, write the values that are passed from the state's parent to that state. For example, the value of state F will not be shown in the $\alpha - \beta$ values of state B, but would be reflected in the $\alpha - \beta$ values of state G. If a state does not have to be evaluated, do not write any values for it in the table.

(d) (4 points) Suppose that O knows that X moves randomly choosing among the legal moves with equal probability for each choice. In order for O to choose the best move, O decides to choose the move corresponding to the minimum of the "expected values" of the resulting states. Redo the computation of backed up values of the internal nodes


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

state	value	α	β
A		N/A	N/A
B			
C			
D			
E			
F			
G			
H			
I			
J			
K			
L			
M			
N			
O			
P			
Q			

Table 4.2: State values and parameter values for α and β.

from (b) by using the statistical expectation of the values of states where it is X's turn to move, instead of the maximum value of its children. For each internal node B through E, show how you compute the expected value from its children's values. Assume the uniform probability distribution for three items: $[1/3, 1/3, 1/3]$. Put your expressions and values in Table 4.3.

node	expression	expected value
B		
C		
D		
E		

Table 4.3: Expected values for chance nodes.