# CSE 415–Autumn 2022 — Midterm Exam **Solutions**

by the Staff of CSE 415, Autumn 2022

**INSTRUCTIONS:** Read these instructions carefully. Make sure you have all 8 (??) pages.

Generally put your answers inside the framed rectangular boxes if they are provided in the question. Write legibly and if you use a pencil, make sure that you write darkly enough that a normal scanner will pick up your writing. If you need more space, use the margins. You can go outside the framed rectangles, but the graders will be looking in the rectangles first, so make sure they will see where you continue your answer.

Do all problems. This is a CLOSED-BOOK, CLOSED-NOTES examination. Do not use any books, notes, calculators, or other electronic devices. There is one "administrative problem" below and four content-related problems, worth a total of 100 points. Each problem has multiple parts, and the allocations of points among the parts are as shown on each individual problem. Note that some problems may be significantly easier or more time-consuming than others. However, the overall time to complete the exam is estimated to be 40 minutes for a student familiar with the material.

**POLICY ON QUESTIONS AND CLARIFICATIONS:** Do not ask questions during the exam. Instead, if you find one of the exam problems to be ambiguous or unclear, state your objection here on the cover page, clearly identifying the problem number and part number, as well as describing your issue (and continue in the margin on the page of the problem, if necessary). If your objection is accepted by the grader, then points may be adjusted in your score.

## 0 Administrative (5 points)

(a) (1 point) Put your last name and first name in the blanks at the top of this first page.

(b) (2 points) Put your UW student number and UWNetID in the blanks below.

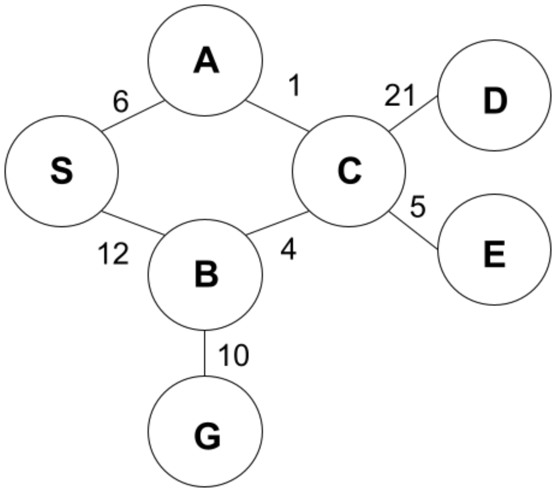
(c) (2 points) Put your last name and first name at the top of each of pages 2 through 7.

| | |
|---|---|
| UW Student Number: | 0123456 |
| UWNetID | johndoe19 |

## 1 Search (25 points)

### Part I.

Perform Breadth-First Search, Depth-First Search, Iterative-Deepening Depth First-Search, and Uniform-Cost Search on the graph below, using S as the starting node and G as the goal. In all cases use graph search, keeping a closed set for duplicate state checking. Break ties alphabetically (A will be expanded before B, all other things equal). Note that the edges are bidirectional, so the search can move in either direction between two nodes.



For each search, **write the order of node expansions (the order in which nodes are selected as the current node)**.

(a) (3 points) Depth-First Search (DFS)

S A C B G

(b) (3 points) Breadth-First Search (BFS)

S A B C G

(c) (3 points) Iterative-Deepening Depth-First Search (IDDFS)

S - S A B - S A C B G

(d) (3 points) Uniform-cost Search (UCS)

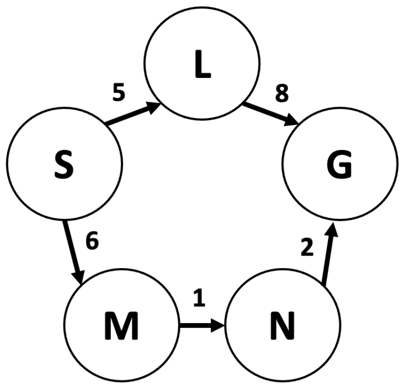
S A C B E G

(e) (3 points) Which, if any, of the above searches finds the cost of the path with the lowest sum of edge weights? Describe what you (or some additional code) would need to do to recover the lowest cost path.

Uniform-cost Search (UCS). Whenever a new node is reached, record with that node not only the G value for the node, but its parent node. Whenever a node is reached again (via some other path), if the new G value is less than the old, update the G value and update the parent. When the goal node is reached, follow the parent references back to the start node, and then reverse that sequence to get a lowest-cost path from the start node to the goal.

## Part II.

The following 5-node graph will be used in A* search. S will be used as the start node and G for the goal node. Note that the edges of this graph are directed. Consider the values provided for the three heuristic functions H1, H2, and H3.



| State | H1 | H2 | H3 |
|---|---|---|---|
| S | 8 | 7 | 6 |
| L | 8 | 6 | 6 |
| M | 2 | 1 | 3 |
| N | 1 | 1 | 3 |
| G | 0 | 0 | 0 |

(f) (6 points) **Admissibility and consistency**
Determine the admissibility and consistency of each heuristic with respect to the graph. Circle yes or no for each question.

(i) **Yes / No** - H1 is admissible. Yes

(ii) **Yes / No** - H2 is admissible. Yes

(iii) **Yes / No** - H3 is admissible. No

(iv) **Yes / No** - H1 is consistent. Yes

(v) **Yes / No** - H2 is consistent. Yes

(vi) **Yes / No** - H3 is consistent. No

(g) (2 points) **Dominance of heuristics**
Recall the definition of *domination* in the context of heuristics. Consider the dominance relationships between the three provided heuristics. Circle yes or no for each question.

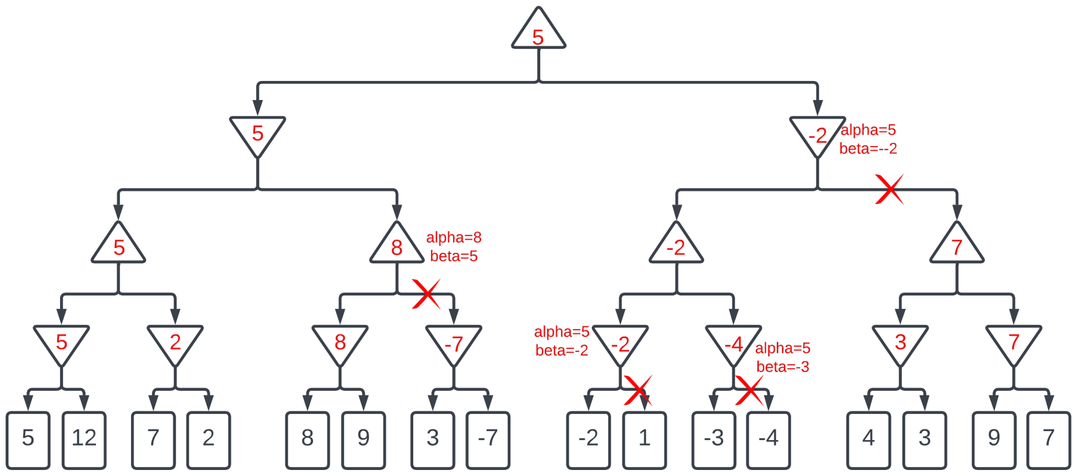
(i) **Yes / No** - Heuristic function H1 dominates H2. Yes

(h) (2 points) **Selection of heuristic**
Considering your responses to parts (f) and (g), which heuristic would you recommend using:

(i) **H1 / H2 / H3** H1, as it is both admissible and consistent and dominates over H2. EDIT: Accepted responses for H1 and/or H2, as both are admissible and consistent and a dominant heuristic could still have drawbacks keeping it from being the "best" choice. In future quarters, could change this question to specify same calculation cost and other considerations or ask students to explain how one might choose between H1 and H2.

## 2 Alpha-Beta Search (25 points)

Consider the tree below in answering the questions.



(a) (5 points) Show the values of all internal nodes, as would be computed by a standard minimax search (without any alpha-beta pruning).

(b) (5 points) Now reconsider the search, applying alpha-beta pruning to determine where subtrees could be cut off. Wherever a subtree can be cut off, draw a double slash across the tree edge where the pruning would occur.

(c) (5 points) Wherever a cutoff occurred in (b), give the corresponding $\alpha$ and $\beta$ values. Recall that a cutoff occurs when $\alpha \geq \beta$ at a node.

(d) (5 points) What advantages in terms of time and space does pruning offer over the original minimax algorithm?

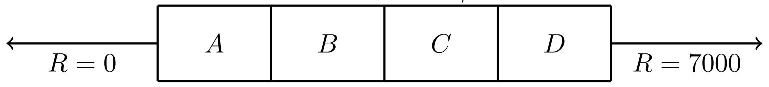
Multiple answers accepted, as long as both time and space considerations were discussed.

(e) (5 points) How many leaf nodes, how many internal nodes, and how many nodes altogether end up being explored at all by alpha-beta search? (Remember to count the nodes needed for pruning decisions.)

8 leaf nodes, 11 internal nodes, 19 total. (Partial credit given for close answers).

## 3 Markov Decision Processes (25 points)

Cristiano finds himself inside the grid world MDP depicted below. Each rectangle represents a possible state. Cristiano has two possible actions, left or right. However, these movement actions only work with probability $y$. With probability $z$ Cristiano trips and moves in the opposite direction. Otherwise Cristiano stays in the same state. I.E. $T(B, right, C) = y$, $T(B, right, A) = z$ and $T(B, right, B) = 1 - y - z$. If Cristiano moves right from state D (with probability $y$ if he chooses right from D, or with probability $z$ if he chooses left from D) he earns a reward of 7000 and enters the terminal state where he can no longer perform actions. Similarly, if Cristiano moves left from A he earns a reward of 0 and enters the terminal state. Note $\gamma$ is the discount factor.



(a) (10 points) Assume $y = 0.5$, $z = 0$, and $\gamma = \frac{2}{3}$ for part (a)

(i) (5 points) What is the value of $V^*(D)$ (the expected value of total discounted rewards Cristiano can get from state D)

$V^*(D) = \boxed{5250}$

$V^*(D) = 0.5 * 7000 + 0.5 * \frac{2}{3} * V^*(D)$
Solve for $V^*(D)$ 0.5 chance of moving and getting a reward 7000,
0.5 chance of staying and getting a reward of $V^*(D) * \gamma$

(ii) (5 points) What is the value of $V^*(C)$ (the expected value of total discounted rewards Cristiano can get from state C)

$V^*(C) = \boxed{2625}$

$V^*(C) = 0.5 * \frac{2}{3} * V^*(D) + 0.5 * \frac{2}{3} * V^*(C)$
Solve for $V^*(C)$ using $V^*(D) = 5250$ from the previous part
0.5 chance of moving and getting a reward $V^*(D) * \gamma$, 0.5 chance of staying and getting a reward of $V^*(C) * \gamma$

(b) (15 points) For each subpart in part (b) you will be given two sets of parameters. Select which set of parameters results in a greater value of $V^*(D)$ (or whether they are the same). Choose only one answer for each sub-question.

(i) (3 points) Set I: $\{y = .2, z = 0, \gamma = .3\}$ Set II: $\{y = .4, z = 0, \gamma = .3\}$

○ Set I's $V^*(D)$ is greater [✓ Set II's $V^*(D)$ is greater] ○ They are equal

Since $y$ is higher in Set II it will reach the reward faster and incur less discounting.

(ii) (3 points) Set I: $\{y = .01, z = 0, \gamma = 1\}$ Set II: $\{y = .99, z = 0, \gamma = 1\}$

○ Set I's $V^*(D)$ is greater ○ Set II's $V^*(D)$ is greater [✓ They are equal]

Even though $y$ is higher in Set II so it will reach the reward faster there is no discounting so they will both get the max reward anyway.

(iii) (3 points) Set I: $\{y = .7,\ z = .2,\ \gamma = .5\}$ Set II: $\{y = .2,\ z = .7,\ \gamma = .5\}$

○ Set I's $V^*(D)$ is greater ○ Set II's $V^*(D)$ is greater [✓ They are equal]

Since $y$ and $z$ are swapped, for whatever policy is optimal for Set I we can choose the opposite actions and get the same performance in Set II.

(iv) (3 points) Set I: $\{y = .1,\ z = 0,\ \gamma = 1\}$ Set II: $\{y = .9,\ z = .1,\ \gamma = 1\}$

[✓ Set I's $V^*(D)$ is greater] ○ Set II's $V^*(D)$ is greater ○ They are equal

In Set I there is no discounting so you will eventually reach and earn the max reward of 3000. In Set II there is a nonzero probability of accidentally taking the reward of 0 so we will have an expectation less than 7000.

(v) (3 points) Set I: $\{y = .5,\ z = 0,\ \gamma = .5\}$ Set II: $\{y = .5,\ z = .1,\ \gamma = .5\}$

[✓ Set I's $V^*(D)$ is greater] ○ Set II's $V^*(D)$ is greater ○ They are equal

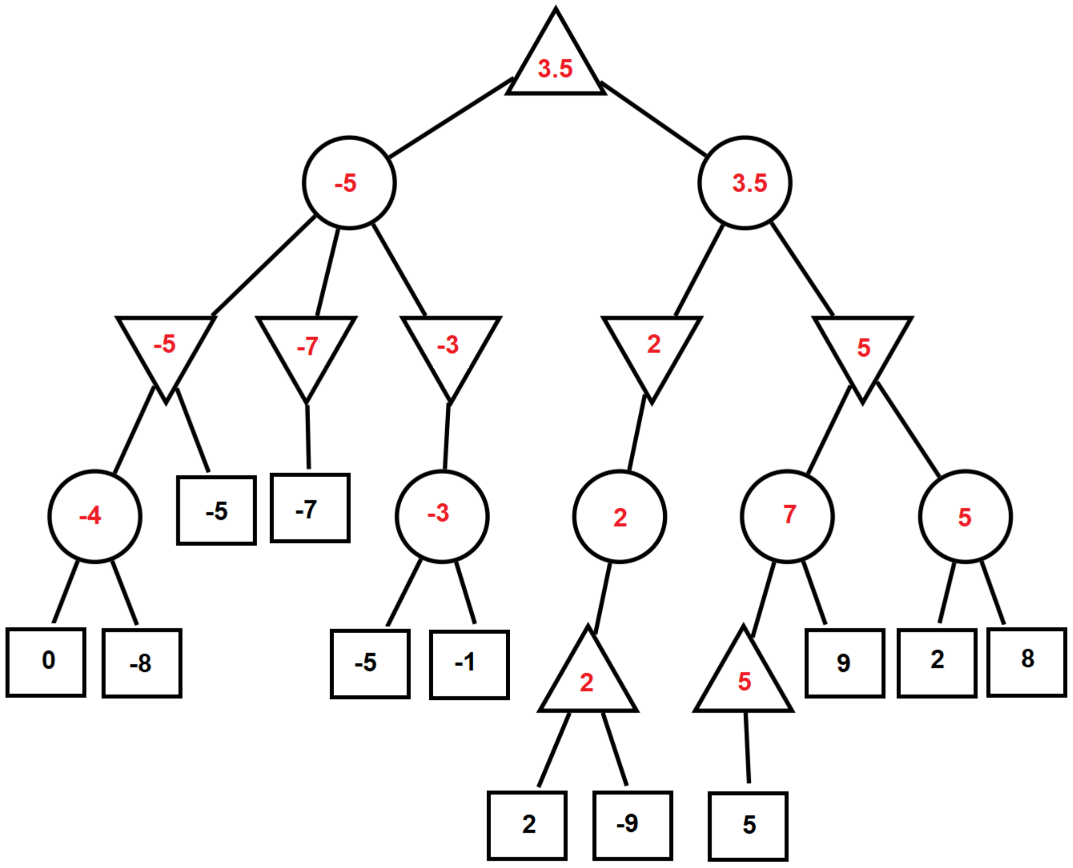
In Set II, the positive Z Value causes you to accidentally go in the opposite direction and earn less than 7000 in expectation.

## 4 Miscellaneous (25 points)

(a) (5 points) Consider the Towers-of-Hanoi Puzzle for different numbers of disks. Give a formula for $f(n)$, the number of states for the version of the puzzle having $n$ disks.

With the Towers-of-Hanoi puzzle having $n$ disks, the number of possible states is easy to describe. Each disk can be on any one of the three pegs. So with $n = 1$, $f(n) = 3$; with $n = 2$, $f(n) = 3 \cdot 3$. In general $f(n) = 3^n$. Thus $f(n)$ is an exponential function of $n$, which is a standard pattern with combinatorial explosions.

(b) (5 points) The tree below represents part of a game in which there is a maximizing player (whose turn it is at the triangular nodes pointing up), a minimizing player (whose turn it is at the triangular nodes pointing down), and there is some randomness in the state transitions (represented by circular nodes). Assume that the the children of a circular node have equal probability. Leaf nodes of this tree are boxes and they have static values given. Fill in the internal nodes of the tree using "Expectiminimax search" to compute those values, using either the expectation, minimum, or maximum of children, according to each node's type.



(c) (5 points) Suppose some Markov Decision Process has 5 states, and 4 actions. Each action is allowed in each state. How many possible policies are there in this MDP?

Since a policy is just one assignment of actions to states, there are three choices of action for each of the 5 states, or $4^5 = 1024$ possible policies.

(d) (5 points) Consider some MDP that has $n$ states, and $m$ actions. Assume that all actions are allowed in each state. Assume $T$ and $R$ (the transition probabilities, and the rewards) are available to a Value Iteration program, and they do not have any special properties like sparseness (lots of 0 values) but are arbitrary (they could be anything legal for an MDP of this size).

(i) How many times will the Bellman update formula (getting $V_{k+1}(s)$ from $V_k(s)$) be evaluated in obtaining $V_1(s)$ from $V_0(s)$ for all $s$?

(ii) What is the big O running time (in terms of $n$ and $m$) for *one* of those Bellman update operations (i.e., for updating the value of one state)?

(i) $n$ times (one update for each of the $n$ states).
(ii) $O(mn)$. In order to max over all $m$ Q-values there will be a loop that runs $O(m)$ times. But within each iteration of that loop there will be a loop over all possible next states, $s'$, and there are $n$ of those. The work to be done inside the body of that inner loop requires only constant time (i.e., getting the reward for the transition, and possibly adding to it the discounted current value of $V_k(s')$, although this would be zero in the case of the very first outer iteration (going from $V_0$ to $V_1$).

(e) (5 points) A domain-specific Turing test involves 3 parties of people (or agents): 1. a human competitor; 2. a computational agent competitor; and 3. a jury of humans. (In addition, there may be a messenger, whose job it is to provide a communication channel among the other parties.)

Assume that the domain is computer programming, so that the human competitor is a professional software developer writing in Python. Explain who should be on the jury, how the test should be conducted, and what each of these parties does. Then describe how the agent passes or fails the test.

The jury should be a panel of human computer programming experts who can judge the quality of a piece of source code in relation to a specification of it. They will be given enough time to inspect code coming from the human and from the computer agent. Out of sight and hearing of the jury, the human competitor, who is a computer programmer, randomly takes the role of either "A" or "B". Also away from the jury, the agent competitor takes the other role. The jury has no way to know which competitor has taken which role. The jury makes questions (which may consist of requests for short pieces of code to be written according to specifications they give, or which may be questions about such code or programming in general, or about the life of a programmer) that are carried (or transmitted electronically) by the messenger, one per turn, to each of the human competitor and the agent competitor, who remain hidden from the jury. Each competitor creates an answer to the question and the messenger takes the answers back to the jury. The messenger first presents A's answer and then presents B's answer. After some number of turns (perhaps 2 or 3, or maybe as many as a dozen or more), the jury is asked to vote on which role has been taken by the human and which by the computational agent. If a majority say A is the human programmer, and A is actually the computational agent, or a majority say B is the human programmer, but B is actually the computational agent, then the agent wins the game and passes the Turing test. It is assumed that the human competitor is trying win, and so it should not be easy for the computational agent to pass the test. The jury can consider the responses of the human and of the computer agent, and do such things as inspecting code, running such code. The jury can also ask questions about the code and consider the answers they receive. (Correct answers are not expected to be this long, but extra detail is given here to be extra clear.)