Homework 1

CSCI 360 Spring 2022

This assignment covers uninformed and informed search, game trees, CSPs, and reinforcement learning. Most questions of this assignment are graded, while some questions and parts will be marked as optional. Optional questions are extra practice for you to complete and won't count towards your homework grade.

Submission Instructions: You will receive an email and a Piazza reminder for enrolling in Gradescope, which is the platform we will be using for HW submissions and grading. To submit your solutions, first enroll in Gradescope and then submit your solution PDF. We recommend you type up your solutions using LaTeX but handwritten solutions are also fine. **Make sure that you correctly assign pages to questions when submitting to Gradescope, otherwise we may not be able to see and grade all of your answers.**

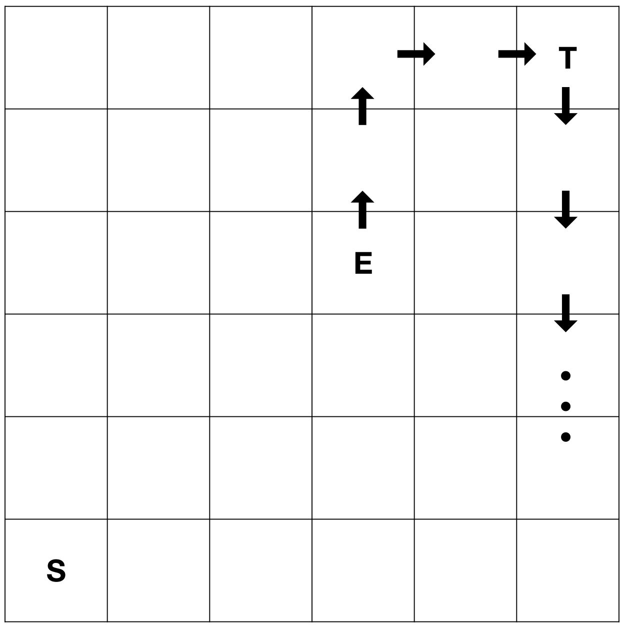
1 Uninformed Search and State Spaces [6 pts]

You are in charge of a new autonomous food delivery service on campus. Being the knowledgable CSCI 360 student that you are, you are going to use your newly acquired knowledge from the class to design state spaces, successor functions, and goal tests to create a solvable search problem to plan a path for your robot delivery vehicles.

There is one problem, however. The constant construction on campus is causing issues for your delivery robots, and you would like to avoid construction zones if possible.

In this problem, denote **S** as your robot's starting location, **E** as the construction vehicle's location, and **T** as Tommy Trojan statue's location. Your goal is to deliver food to some students at **T**ommy Trojan's location, i.e. to get from **S** to **T**. You know the starting location of the construction vehicle **E**.

The construction vehicle **E** puts 1 square under construction during each time step from its current location, going around the campus by cycling in the direction North, East, South, and West as shown in the figure (it's a deterministic path). If all squares adjacent to the vehicle are under construction, then it does not move. Every square on the grid in which **E** has visited is considered under construction, and the vehicle will not ever move into a square already under construction. For example, if the vehicle is traveling North, and at the current square its North, East, and South squares are already under construction or are the edges of the grid, then it will move West instead.



There are costs associated with each move that you make starting from **S**: the cost of entering a square under construction is 3 as the terrain is hard to traverse, and the cost of entering a square not under construction is 1. Assume your grid size is **N** x **N**, and that you can traverse North, East, South, or West, as long as doing so won't take you off of the grid. Finally, also assume a maximum of N^2 timesteps.

Answer each of the following questions with a quick justification of your answer.

(a) (1 pt) What is the **smallest** state space representation possible (in terms of number of states)? What is its size in number of states? What's the successor function (what are the valid actions and their associated costs)? What's the goal test?

(b) (1 pt) What is the cost of the optimal path in the specific instance given in the figure above using **UCS Tree Search** if you were starting at **S** and moving to **T**?

(c) (1 pt) What is the cost of the path in the specific instance given in the figure above using **BFS Graph Search** if you were starting at **S** and moving to **T**? Assume that ties for which states to put onto the fringe are broken in the order North, East, West, South. Hint: Perform this on a 2×2 or 3×3 grid to see a pattern.

(d) (0.5 pt) Would we want to use adversarial game trees for this situation? Why/why not?

(e) (0.5 pt) Now, assume that at each timestep, you have the power to freeze the construction vehicle for 1 timestep, by reporting to the administration that they don't have a valid Trojan Check. You can freeze the vehicle at the same time as you move on the grid. Does the smallest state space representation need to change for guaranteeing that we can find the optimal solution? If so, how would it change?

(f) (Optional) Now, assume that there is no construction going on, and that you need to visit every spot on the grid at least once to deliver food to every spot on the entire campus. However, you have no idea where you are on campus, but you still need to guarantee that you have visited every single square. During the entire search, you will not know where you are. If you attempt to take a step out of bounds of the grid, then you will just stay in the same spot. Assume an unbounded number of timesteps for this task now, and assume that entering every square has the same cost. What is the **smallest** state space representation possible that guarantees we can still find the optimal solution? What is its size in number of states? What is the new goal test?

(g) (1 pt) Now, assume the same setup as in the previous part, except you can only move in any direction (North, South, East, or West) a maximum of three times before needing to change directions. Assume that in the starting state, we have already moved North one time before starting. What is the **smallest** state space representation possible that guarantees we can still find the optimal solution? What is its size in number of states? What is the successor function (when will it let us move in which directions, and what will the states returned look like)? Use constant **K** to denote the state space size from the previous part.

(h) (1 pt) Finally, assume that there are M robots from your delivery company navigating around the campus. Assume the same setup as in part (f) except that you know the positions of all robots. But, you have limited cloud compute (it's expensive!), so each robot must take turns moving (e.g. the first robot moves while all others are frozen, then the next robot moves while all others are frozen, and so on). What is the new state space representation and size? The goal is still to have every location on campus visited by at least one robot.

2 Informed Search and Heuristics [8 pts]

1. (2 pts) You are tasked with delivering packages in an M by N grid space, with each cell corresponding to a location. Suppose you start at a given arbitrary starting location, **S**. You need to deliver K packages. You have to pick up each package from their locations $P_1, P_2, ...P_K$ and deliver to their corresponding locations $D_1, D_2, ...D_K$ i.e, package picked at location P_i should be delivered to location D_i only. Each location P_i or D_i represents one cell location. The cost of moving from one cell to another is 1. Assume that you are allowed to move up, right, down, or left only. You can hold only 1 package at a time, i.e, your capacity is 1. When you enter a square with a package that has not yet been delivered, then you will automatically pick it up as long as you are not already carrying a package.

   (a) What is the size of the minimal state space?

   (b) What is the maximum possible number of actions available at any given state?

   (c) Let d(A, B) be the Manhattan distance between locations A and B. Which of the following heuristics given below is admissible from the starting location? Select all that apply.

      (i) $d(S, P_1) + d(S, P_2) + d(S, P_3) + \ldots + d(S, P_K)$
      (ii) $d(P_1, D_1) + d(P_2, D_2) + d(P_3, D_3) + ... + d(P_K, D_K)$
      (iii) $min\{d(P_1, D_1), d(P_2, D_2), d(P_3, D_3), ..., d(P_K, D_K)\}$
      (iv) $max\{d(P_1, D_1), d(P_2, D_2), d(P_3, D_3), ..., d(P_K, D_K)\}$
      (v) none of the above

   (d) Suppose, you are now allowed to move diagonally as well. Which of the following heuristics given below is admissible from the starting location? Select all that apply.

      i. $d(S, P_1) + d(S, P_2) + d(S, P_3) + \ldots + d(S, P_K)$
      ii. $d(P_1, D_1) + d(P_2, D_2) + d(P_3, D_3) + ... + d(P_K, D_K)$
      iii. $min\{d(P_1, D_1), d(P_2, D_2), d(P_3, D_3), ..., d(P_K, D_K)\}$
      iv. $max\{d(P_1, D_1), d(P_2, D_2), d(P_3, D_3), ..., d(P_K, D_K)\}$
      v. none of the above

   (e) Suggest an admissible heuristic when (d) is allowed.

2. (2 pts) Consider the following problem with a given set of elements A, B, C, D, E. Given a start state say: [B, C, A, D, E], reach the goal state: [A, B, C, D, E] using the following operation: Pick a pivot point and reverse the order from the starting point till the pivot point. For example, if we pick pivot point 2 (indexed starting from 0) in the above start state, performing the operation will change the state from [B, C, A, D, E] to [A, C, B, D, E].

   (a) Given any arbitrary start state, what is the size of minimal state space?

   (b) What is the maximum possible number of actions available at any given state? Consider only those actions that result in a new state different from current state.

   (c) If the cost of performing the operation is equal to 1, is the following heuristic admissible?
   h(s): number of elements not in their desired positions

   (d) If the cost of performing the operation is equal to the number of elements reversed, is the above heuristic admissible?

3. (2 pts) Consider the following graph. Perform A* search with start state as S and goal state as G. In case of ties, choose the nodes in lexicographical order.

   (a) Tree search

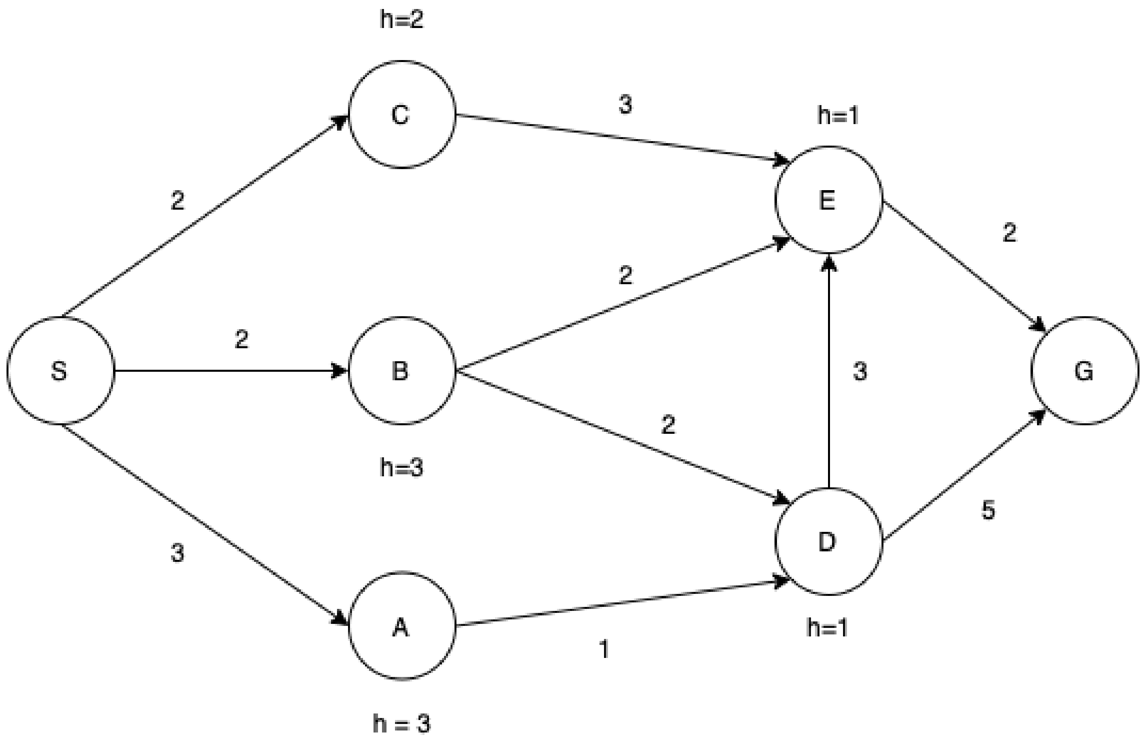
      (1) List the nodes in the order they are expanded from the queue. Each node should be represented with its full path.

      (2) Give the final solution path obtained by the algorithm and its cost.

   (b) Graph Search

      (1) List the nodes in the order they are expanded from the queue. Each node should be represented with its full path.

      (2) Give the final solution path obtained by the algorithm and its cost.



   (c) Is the heuristic admissible? If not, what could be changed to make it admissible?

   (d) Is the heuristic consistent? If not, what could be changed to make it consistent?

4. (2 pts) Suppose we are conducting A* tree search. We already have an **admissible** heuristic h(x), but now we want to test the algorithm with a new heuristic $h'(x)$. Let $h^*(x)$ be the optimal cost of path to reach the goal state from state x. Let C be the cost of path obtained using the A* search with heuristic $h'(x)$. For each of the relation between $h'(x)$ and h(x) listed below, select the option that is correct for the relation between C and $h^*(x)$. You an assume all heuristic values are at least 0.

   (a) $h'(x) = h^*(x) - h(x)$

(i) $C > h^*(S)$ (ii) $C = h^*(S)$ (iii) $C \geq h^*(S)$

(b) $h'(x) = h^*(x)$

(i) $C > h^*(S)$ (ii) $C = h^*(S)$ (iii) $C \geq h^*(S)$

(c) $h'(x) = h(x)^{0.99}$

(i) $C > h^*(S)$ (ii) $C = h^*(S)$ (iii) $C \geq h^*(S)$

(d) $h'(x) = h(x) + h^*(x)$

(i) $C > h^*(S)$ (ii) $C = h^*(S)$ (iii) $C \geq h^*(S)$

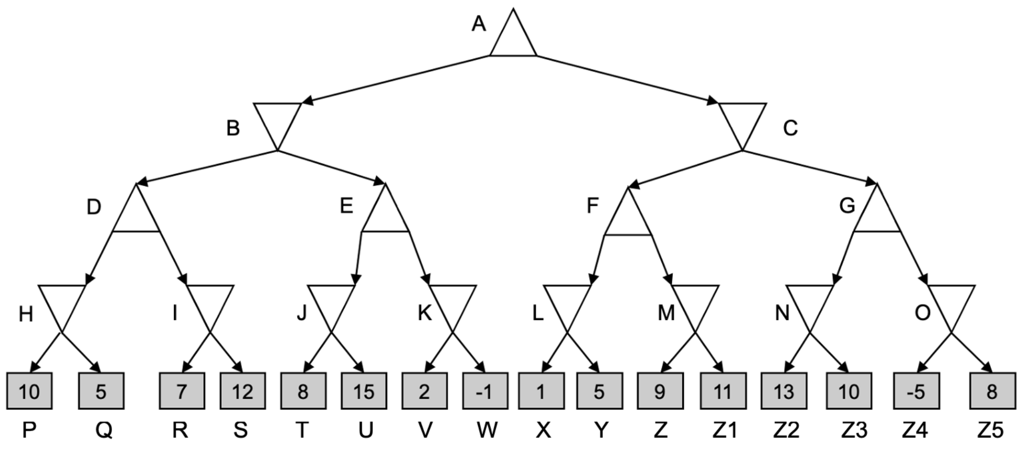
(e) $h'(x) = \sqrt{h(x) * h^*(x)}$

(i) $C > h^*(S)$ (ii) $C = h^*(S)$ (iii) $C \geq h^*(S)$

3 Game Trees [7 pts]

1. (3 pts) Consider a zero-sum game in which there are n coins in a line (assume that n is even here). Two people take turns to take a coin from one of the ends of the line until there are no more coins left. The person with the larger amount of money wins, and there can be negative and positively valued coins with differing values.

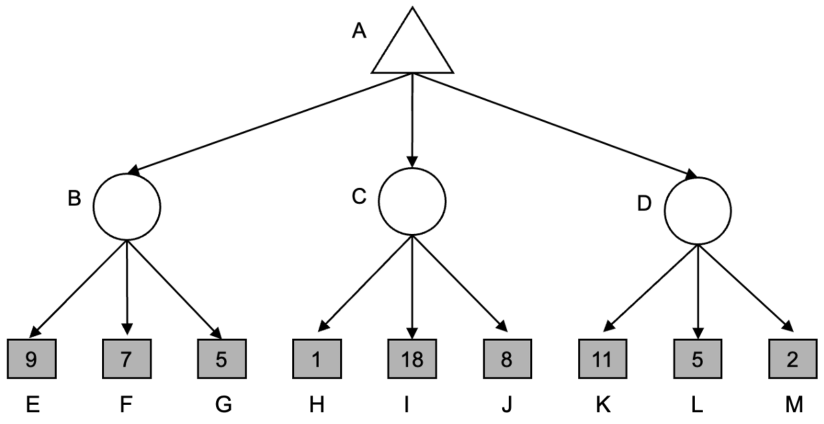
   (a) Consider the game tree for the above game as shown below. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Assuming both players act optimally, write the minimax value of each node (A through O).



   (b) Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. Assume the search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.

2. (4 pts) Let us consider a similar 3-level zero-sum game tree, except that now, instead of a minimizing player, we have a chance node that will select one of the three values uniformly at random.

   (a) Fill in the Expectimax value of each node in the tree. The game tree is shown below for your convenience. Please show the calculation for each node(A through D).

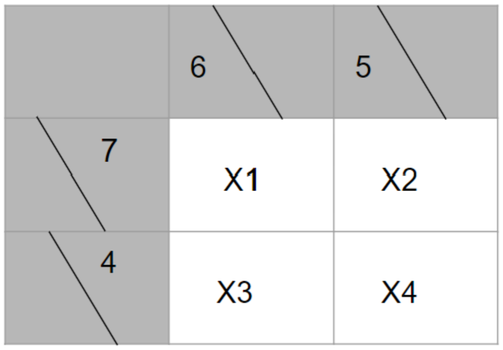


(b) Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not.

(c) Which nodes can be pruned from the game tree above through alpha-beta pruning if there is a bound of +6 on the nodes L and M? Justify your answer.

4 Constraint Satisfaction Problems [6 pts]

Kakuro is a logic puzzle that is often referred to as a mathematical transformation of a crossword puzzle. Just like a crossword it consists of a grid of squares as below:



A set of connected horizontal or vertical blank squares is called a block. The objective is to fill all empty squares using numbers 1 to 9 so the sum of each horizontal block equals the clue on its left, and the sum of each vertical block equals the clue on its top. In addition, no number may be used in the same block more than once.

For instance, in the above puzzle, X1 and X2 form a horizontal block and their sum should equal 7. Also, you cannot use a digit more than once in a block. For example, you cannot use digit 3 twice in the vertical block above leading to 6, to get sum 6.

(a) (1 pt) Formulate the puzzle above as a CSP with variables, domains, and constraints.

(b) (1 pt) Draw the constraint graph representing the above CSP.

(c) (2 pts) Run 2 steps of arc consistency (where one step corresponds to popping one arc off of the arc consistency queue). Pop arcs off of the queue in numerical order of the tail, then head, variables (e.g. $X1 \rightarrow X2$ is popped before $X1 \rightarrow X3$ which would be popped off before $X2 \rightarrow X1$).

1. What arcs are currently on the queue?
2. What are the current domains of all considered variables?

(d) (1 pt) Show the domains of all considered variables after completing arc-consistency.

(e) (1 pt) Let's say the current domains after some steps of arc consistency are as follows: $X1 = \{5, 1, 2\}$, $X2 = \{2, 3, 4\}$, $X3 = \{1, 3\}$, $X4 = \{3, 4, 5\}$. If we were to run search from this point, name what variable would be chosen for assignment with the Minimum Remaining Values ordering strategy:

5 Reinforcement Learning and MDPs [7 pts]

In pseudo-blackjack, you will keep drawing cards from a deck (with replacement). In this simplified version, the only cards in the deck are a 2, 3, or 4. At each timestep, you can choose to either "Hit" (request a card from the dealer) or "Stand" (stop playing for the current round). If the sum of the values of the cards you hold are at least 6, then the round ends and you will receive a reward of 0. When you Stand, your reward will be equal to the sum of your card values and the round comes to an end. When you Hit, you receive no additional reward. Assume no discount, i.e. $\gamma = 1$. Whenever the round comes to an end, assume you enter a special "Done" state from which you cannot take any actions.

(a) (0.5 pt) We will formulate this as a Markov Decision Process (MDP). Enumerate all states in the state space and all possible actions of this MDP.

(b) (1.5 pts) Define the transition function ($T(s, a, s')$) and reward function ($R(s, a, s')$) for this MDP.

$T(s, Stand, Done) =$		
$T(0, Hit, s') =$		if $s' \in \{2, 3, 4\}$
$T(2, Hit, s') =$		if $s' \in \{4, 5, Done\}$
$T(3, Hit, s') =$		if $s' = 5$
$T(3, Hit, s') =$		if $s' = Done$
$T(4, Hit, s') =$		
$T(5, Hit, s') =$		
$T(s, a, s') =$		otherwise
$R(s, Stand, Done) =$		if $s \leq 5$
$R(s, a, s') =$		otherwise

(c) (2 pts) Now we will perform value iteration for a fixed number of iterations. Fill in the following table for the values of each state ($V(s)$) at each iteration.

States	0	2	3	4	5
$V_0(s)$					
$V_1(s)$					
$V_2(s)$					
$V_3(s)$					
$V_4(s)$					

(d) (1 pt) Extract the optimal policy for this MDP (list the actions the policy takes at each state). The optimal policy is given by: $\pi^*(s) = \text{argmax}_a \sum_{s'} T(s, a, s')[R(s, a, s') + V(s')]$

States	0	2	3	4	5
$\pi^*(s)$					

(e) (2 pts) Starting with the given policy below, perform one iteration of policy iteration (ignore values from previous steps). Hint, you will get values of $V^{\pi_0}(s)$ that depend on other $V^{\pi_0}(s')$ but the system of equations can be solved.

States	0	2	3	4	5
$\pi_0(s)$	Hit	Stand	Hit	Stand	Hit
$V^{\pi_0}(s)$					
$\pi_1(s)$					