DEPARTMENT OF
COMPUTING SCIENCE
UNIVERSITY OF ALBERTA

CMPUT 366: Assignment #2

Due at 10pm on October 22, 2019

Abstract

For this assignment use the following consultation model:

- you can discuss assignment questions and exchange ideas with other CMPUT 366 students;
- you must list all members of the discussion in your solution;
- you may **not** share/exchange/discuss written material and/or code;
- you must write up your solutions individually;
- you must fully understand and be able to explain your solution in any amount of detail as requested by the instructor and/or the TAs.

Anything that you use in your work and that is not your own creation must be properly cited by listing the original source. Failing to cite others' work is plagiarism and will be dealt with as an academic offence.

First name: Shaoyang

Last name: Zhou

CCID: Shaoyang@ualberta.ca

Collaborators: _____

Submission

The assignment you downloaded from eClass is a single ZIP archive which includes this document as a PDF file as well as its LaTeX source. Your answers are to be submitted electronically via eClass. Your submission must be a single PDF file with all of your answers. To generate the PDF file you can do any of the following:

1. insert your answers into the provided LaTeX source file between `\begin{answer}` and `\end{answer}`. Then run the source through LaTeX to produce a PDF file;
2. write your answers in the blank spaces under each question. Make sure you write as legibly as possible for we cannot give you any points if we cannot read your hand-writing;
3. use your favourite text processor and type up your answers there. Make sure you number your answers in the same way as the questions are numbered in this assignment.

1 Real-time Heuristic Search

A real-time heuristic search agent is traversing a graph $G = (S, E)$ where S is the set of states/vertices and E is the set of edges connecting the vertices. For each state $s \in S$, $N(s)$ is the set of its immediate neighbors: $N(s) = \{s' \mid s' \in S \ \& \ (s, s') \in E\}$. Both S and E are finite. All edges in E are undirected (i.e., if an edge $(s_1, s_2) \in E$ then the agent can move from s_1 to s_2 and also from s_2 to s_1). The edge cost is a function $c : E \to \mathbb{R}$. All edge costs are above zero and symmetric: $c(s_1, s_2) = c(s_2, s_1)$. The start state is $s_0 \in S$. The goal state is $s_g \in S$. The problem is solvable: there is a path from s_0 to s_g in the graph. The agent's task is to reach s_g from s_0 while minimizing the cost of all edges traversed in the process. The agent has access to a heuristic $h : S \to \mathbb{R}$ which is initially 0 for all states. The agent can modify it in its current position in any way it wants.

1.1

[20 points] Consider the following algorithm:

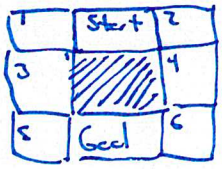
Algorithm 1: Basic LRTA*

input : search problem (G, c, s_0, s_g, h)
output: solution path $(s_0, s_1, \ldots, s_g)$

1 $t \leftarrow 0$
2 $h_t \leftarrow h$
3 **while** $s_t \neq s_g$ **do**
4 $\quad h_{t+1}(s_t) \leftarrow \max\left\{h_t(s_t), \min_{s \in N(s_t)} (c(s_t, s) + h_t(s))\right\}$
5 $\quad s_{t+1} \leftarrow \operatorname{argmin}_{s \in N(s_t)} (c(s_t, s) + h_t(s))$
6 $\quad t \leftarrow t + 1$

Suppose the argmin operator in line 5 of Algorithm 1 breaks ties in a fixed order of preference (e.g., traversing the edge to the left is always preferred to the edge to the right when the $c + h$ values of both neighbors are identical). Give a single example of a small search graph on which the agent finds two different solutions under two different tie-breaking preference orders. The initial heuristic should be 0 for each state.

List each solution (i.e., the sequence of states visited by the agent) and explain why different solutions will be produced.



Path 1: start → ① → ③ → ⑤ → Goal
Path 2: start → ② → ④ → ⑥ → Goal

Supposing we break ties by picking even or odd numbered cells.

Rationale

- Differences in tie breaking, lead to two different paths with the same cost.
- Moving off the start node effectively blocks the search into one channel.
- The search can just greedily process unexplored nodes since they will always have the cheapest $h2$ cost, and will lead to the goal.

Thus different paths are produced because of this channeling effect.

1.2

[2 points] Suppose the set of edges E is modified so that the goal state s_g is no longer reachable from the start state s_0 (i.e., the search problem is not solvable). Will Algorithm 1 stop and declare that no solution exists or will it run forever? Justify your answer.

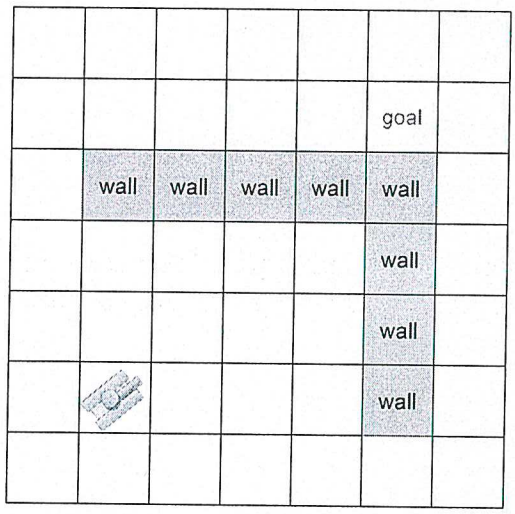
- It would run forever
- By the loop guard of line 3, if the end node is not reachable, then this current node, s', can never be the end node. Hence this while condition will always be true.

1.3

[8 points] Consider a unit in a real-time strategy game (e.g., *StarCraft*) which has to go from its current location to goal on the map in the figure below. The unit cannot go through walls. Assume the unit occupies a single grid cell and can move in cardinal directions. Assume the initial heuristic is the Euclidean distance to the goal state.

Pros

1. Search can now be online, could be interviewed with game updates.
2. Search is now agent centered, can handle world dynamics. For example, maybe the walls are temporary, when the tank reaches the corner, they clear.



Cons

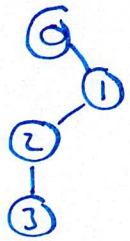
1. Agent will likely crashed some scrubbing when near the corner, ruins immersion of the game.
2. Algorithm will likely not produce the optimal solution.

List two pros and two cons of using Algorithm 1 for pathfinding in this problem.

1.4

[8 points] Consider the following search problem: $S = \{s_0, s_1, s_2, s_3\}$. The start state is s_1, the goal state is s_3. The undirected edges are $E = \{(s_0, s_1), (s_1, s_2), (s_2, s_3)\}$ and each edge costs 1. The initial heuristic is 0 for all states.

Suppose the argmin operator in line 5 of Algorithm 1 breaks ties towards the state with a higher index (e.g., if the agent is in s_1 and $c(s_1, s_0) + h(s_0) = c(s_1, s_2) + h(s_2)$ then the agent moves to s_2). List heuristic values at each time step as the agent traverses the graph until the goal state is reached. Indicate the agent's position at each time step.



T	POS	H(0,1,2,3)
0	1	(0,0,0,0)
1	2	(0,1,0,0)
2	3	(0,1,1,0)
3	end	

1.5

[8 points] Suppose now that the argmin operator in line 5 of Algorithm 1 breaks ties towards the state with a lower index (e.g., if the agent is in s_1 and $c(s_1, s_0) + h(s_0) = c(s_1, s_2) + h(s_2)$ then the agent moves to s_0). List heuristic values at each time step as the agent traverses the graph until the goal state is reached. Indicate the agent's position at each time step.

T	POS	H(0,1,2,3)
0	1	(0,0,0,0)
1	0	(0,1,0,0)
2	1	(2,1,0,0)
3	2	(2,1,0,0)
4	3	(2,1,1,0)
5	end	

2 Markov Decision Processes

Two coins are placed in a tray; each coin is either heads up or tails up with equal probability, independent of the other coin. A robot has a single arm which can be positioned above either of the coins. At each time step, the robot can perform one of the following operations:

- **Randomize** the coin below it: the coin under the arm will be randomly set to either heads up or tails up with equal probability. This operation costs the robot 1 point.
- **Move** to be above the other coin; this operation costs the robot 2 points.
- **Call** the Fairy, who will reward the robot with 20 points if both coins are heads up, reward the robot with 10 points if both are tails up, or fine the robot 5 points if the coins mismatch. After rewarding or fining the robot, the Fairy will then randomize both coins before leaving.

2.1

[10 points] Represent this scenario formally as a Markov Decision Process (MDP), treating points given to or taken away from the robot as the reward signal. Provide the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p)$ as per the lecture slides.

$S = \{$ (Vel.R, Vel.L, Pos) $\}$ $A = \{$ Rand, Move, Call $\}$ $R = \{-1, -2, 20, 10, -5\}$

$\{$ (H H L), (H H R), (H T L), (H T R), (T H L), (T H R), (T T L), (T T R) $\}$

Note: Below, for brevity, I'll list the second case using wild card characters "-" to denote and "x" and "y" for randomization. "x" and "y" as placeholder values

$p = $

S	A	S'	$p(s' \mid s, a)$	r
x ∈ {L,R}	Move	x ∈ {R,L}	1	-2 ie L→R, R→L
HH	Call	x' y' -	$\frac{1}{4}$	20 ie, "-" is wild character for the position
TT -	Call	x' y' -	$\frac{1}{4}$	10
HT -	Call	x' y' -	$\frac{1}{4}$	-5
TH -	Call	x' y' -	$\frac{1}{4}$	-5
x T L	Rand	x' T L	$\frac{1}{2}$	-1
x T R	Rand	x y' R	$\frac{1}{2}$	-1

2.2

[2 points] Is this a continuing or episodic MDP? Justify your answer.

- Continuing since there is no end condition given.

2.3

[3 points] Suppose you have an MDP where each state has two actions available to it: $\mathcal{A} = \{a^*, a^-\}$. Action a^* is optimal. Action a^- is suboptimal. In each state, the policy π selects an action randomly ε percent of the time and selects the optimal action a^* the rest of the time. Express the policy π as a distribution over actions given a state. In other words, come up with a mathematical expression for $\pi(a|s)$: the probability of selecting action $a \in \mathcal{A}$ in state s. Justify your answer.

$\pi(a^*|s) = (1-\varepsilon) + \frac{1}{2}(\varepsilon)$

$\pi(a^-|s) = \frac{1}{2}\varepsilon$

	Non random	Random
A^*	$1-\varepsilon$	$\frac{1}{2}\varepsilon$
A^-	0	$\frac{1}{2}\varepsilon$
Total	$1-\varepsilon$	ε

Reminder

- In non-random cases, we always select A^*, i.e. in $(1-\varepsilon)$ cases, select A^*
- In random cases, we split ε probability over the two actions evenly.
- The sum of the case wise probabilities must be the total chance per action

3 Value Functions

The value function $V^\pi(s)$ introduced in the lecture is the return an agent can expect from state s on, under the policy π. An alternative is a value function defined on state-action pairs: $Q^\pi(s, a)$ is the return an agent can expect by taking action a in state s and then following the policy π thereafter.

3.1

[2 points] Give an equation expressing V^π via Q^π. Your equation should have V^π only on the left side and Q^π only on the right side.

$V^\pi(s) = \sum_{A \in \mathcal{A}} \pi(A|s) \cdot Q^\pi(A|s)$

3.2

[2 points] Now go the other way: give an equation expressing Q^π via V^π. Your equation should have Q^π only on the left side and V^π only on the right side.

$Q^\pi(s,a) = \sum_{s',r \text{ of } s,a} p(s',r|s,a) \cdot [r + V^\pi(s') \cdot d]$ $\quad d$ = discount coeff

3.3

[4 points] The lectures gave the Bellman equation for V^π recursively expressing V^π through itself and other variables:

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[r + \gamma V^\pi(s')\right].$$

Give an analogous Bellman equation for Q^π. The equation should express $Q^\pi(s,a)$ through the environment dynamics $p(s',r|s,a)$, Q^π for other states and actions, the discount factor γ and all the variables involved in those expressions.

$Q^\pi(s,a) = \sum_{s',r \text{ of } s,a} p(s',r|s,a)\left[r + \gamma \max_{a' \text{ of } s'} Q^\pi(s',a')\right]$

3.4

[2 points] $V^*(s)$ is the expectation of the highest possible return collectible from state s (by acting optimally). Similarly, we can define $Q^*(s,a)$ as the expectation of the highest possible return collectible by taking action a in state s and acting optimally thereafter. Give an equation expressing V^* via Q^*. Your equation should have V^* only on the left side and Q^* only on the right side.

$V^*(s) = \max_a Q^*(s,a)$

3.5

[2 points] Now go the other way: give an equation expressing Q^* via V^*. Your equation should have Q^* only on the left side and V^* only on the right side.

$Q^*(s,a) = \sum_{s',r \text{ of } (s,a)} p(s',r|s,a) \cdot [r + \gamma V^*(s')]$

3.6

[2 points] Give the Bellman optimality equation for Q^* expressing Q^* via itself.

$$Q^*(s,a) = \sum_{s' \text{ if } d(s,a)} p(s',r|s,a) \cdot \left[r + \gamma \cdot \max_{a'} Q^*(s',a') \right]$$

4 Policy Improvement

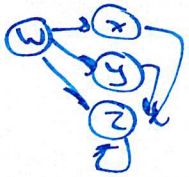
Consider the following MDP with actions $\mathcal{A} = \{a, b\}$, states $\mathcal{S} = \{W, X, Y, Z\}$, and the following dynamics:

$$p(X, 4|W, a) = 0.5$$
$$p(Y, 2|W, a) = 0.5$$
$$p(Z, 3|W, b) = 1$$
$$p(Z, 0|X, a) = 0.8$$
$$p(Z, 25|X, a) = 0.2$$
$$p(Z, 0|X, b) = 1$$
$$p(Z, 1|Y, a) = 1$$
$$p(Z, 5|Y, b) = 1$$
$$p(Z, 0|Z, a) = 1$$
$$p(Z, 0|Z, b) = 1$$

All unspecified transitions have probability 0.

4.1

[8 points] Consider the random policy $\pi(a|s) = \pi(b|s) = 0.5$ for any state $s \in \mathcal{S}$. Under the discount rate of $\gamma = 0.8$, what is the value $V^\pi(s)$ for each state $s \in \mathcal{S}$?



$V^\pi(Z) = 0$

$V^\pi(Y) = 3$

$V^\pi(X) = 2.5$

$V^\pi(W) = \frac{1}{2} \cdot 3$

$\frac{1}{4} \cdot (4 + 0.8 \cdot 2.5)$

$\frac{1}{4} \cdot (2 + 0.8 \cdot 3)$

$= 4.1$

4.2

[4 points] Construct a deterministic policy π' that strictly improves upon π from the previous question. To do so you must show that $V^{\pi'}(s) \geq V^{\pi}(s)$ for all states and $V^{\pi'}(s) > V^{\pi}(s)$ for at least one state.

Tip: to show the inequalities above you can first show that $Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$ for all states and $Q^{\pi}(s, \pi'(s)) > V^{\pi}(s)$ for at least one state. Then you can use the Policy Improvement Theorem.

Policy:

- Pick A in States {w, x, z}
- Pick B in states {y}

Proof:

- In general, I will show that since my policy picks out of the candidate choices of π, dropping always one suboptimal choice, $Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$, thereby improving or keeping up with π.
- Case State z: Since the transition and reward of this state is the same across actions, my policy cannot be worse or better than that of π.
- Case State y: Since both actions lead to the same state, by picking b when action b always has a higher reward than of a, $Q^{\pi}(y, \pi'(y)) \geq V^{\pi}(y)$.
- Case State x: By picking A, we either obtain the same result as picking action b, or gain some upside. Hence $Q^{\pi}(x, \pi'(x)) > V^{\pi}(x)$
- Case state w: Since the expected reward of actions are the same, we only consider the value of future states. Since $V^{\pi}(x)$ and $V^{\pi}(y) > V^{\pi}(z)$, picking action A will improve upon $V^{\pi}(w)$.

- By the above, I prove that $Q^{\pi}(s, \pi'(s)) \geq V^{\pi}(s)$ and in some states $Q^{\pi}(s, \pi'(s)) > \pi^{\pi}(s)$, by the policy improvement theorem, π' improves upon π.

5 Monte Carlo Prediction

5.1

[8 points] Consider an episodic MDP with actions $\mathcal{A} = \{a, b, c\}$, states $\mathcal{S} = \{W, X, Y, Z\}$ (with terminal state Z) and unknown dynamics. Suppose that you have used a policy π to generate 4 episodes with the following trajectories:

$$[S_0 = W, A_0 = a, R_1 = 0, S_1 = X, A_1 = a, R_2 = 10, S_2 = Y, A_2 = b, R_3 = 0, S_3 = Z],$$
$$[S_0 = W, A_0 = a, R_1 = -10, S_1 = X, A_1 = b, R_2 = 0, S_2 = Z],$$
$$[S_0 = W, A_0 = b, R_1 = 2, S_1 = Y, A_1 = c, R_2 = 6, S_2 = Z],$$
$$[S_0 = W, A_0 = b, R_1 = 0, S_1 = Y, A_1 = c, R_2 = 12, S_2 = Z].$$

Use first-visit Monte Carlo prediction to estimate $V^\pi(s)$ for every $s \in \mathcal{S}$. Assume non-discounted rewards (i.e., $\gamma = 1$). Show your work.

Initialization: V(S) = {0,0,0,0} Returns = {[],[],[],[]} // ie, {W,X,Y,Z}

Episode 1

G=0

T=2, S_2=Y, R_3=0, G=0 since Y not in before, V(S) = {0,0,0,0}, Returns = {[], [], [0], []}

T=1, S_1=X, R_2=10, G=10 since X not in before, V(S) = {0,10,0,0}, Returns = {[], [10], [0], []}

T=0, S_0=W, R_1=0, G=10 since no before, V(S) = {10,10,0,0}, Returns = {[10], [0], [0], []}

~~Episode~~

Ep 2

G=0

T=1, S_1=X, R_2=0 Since X not in before, V(S) = {10, 5, 0, 0} Returns = {[10], [10,0], [0], []}

T=0, S_0=W, R_1=-10, G=-10 , V(S) = {0, 5, 0, 0} Returns = {[10,-10], [10,0], [0], []}

Ep 3

G=0

T=1, S_1=Y, R_2=6, G=6 , V(S) = {0, 5, 3, 0} Returns = {[10,-10], [10,0], [6], []}

T=0, S_0=W, R_1=2, G=8 , V(S) = {8/3, 5, 3, 0} Returns = {[10,-10,8], [10,0], [6], []}

Ep 4

G=0

T=1, S_1=Y, R_2=12, G=12 V(S) = {8/3, 5, 6, 0} Returns = {[10,-10,8], [10,0], [6,12], []}

T=0, S_0=W, R_1=0, G=12 V(S) = {5, 5, 6, 0} Returns = {[10,-10,8,12], [10,0], [6,12], []}

Final:

V(S) = {5, 5, 6, 0} // {W, X, Y, Z}