50.021 Artificial Intelligence

Quiz 4

Student Name: **Student ID:**

[Q1]. *Note : Some of the questions have more than one answers.*

1 Which of the following characteristics should a rational agent *always* possess?

A. Able to learn and adapt from past experiences.

B. Follows exactly the actions of an opponent

C. An ability to reason about

D. None of these

2 Which agent deals with happy and unhappy states?

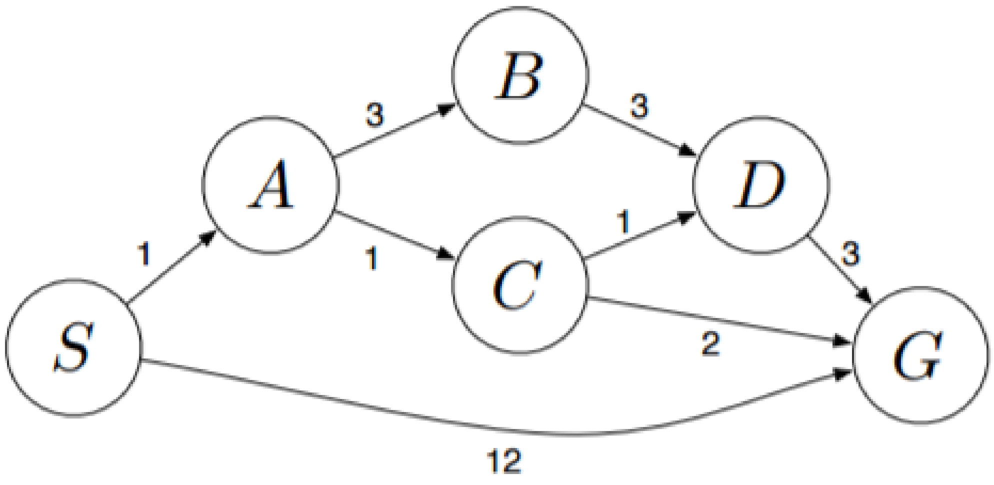
A. Simple reflex agent

B. Model based agent

C. Learning agent

D. Utility based agent

3 Consider the search graph shown below. S is the start state and G is the goal state. All edges are bidirectional. Which of the following statements about the search problem are



true?

A. Breadth-first graph search will return S G, no matter what expansion order

B. Depth-first graph search will return S A B D G, no matter what expansion order

C. Uniform cost graph search will return return S A C G

D. None of these

4 Which of the following statements are true?

A. Let T be a complete binary tree with n nodes. Finding a path from the root of T to a given vertex $v \in T$ using breadth-first search takes $O(lgn)$ time. (note: a complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible.)

B. Consider a graph search problem where for every action, the cost is at least x ,with $x > 0$. Depth-first graph search is guaranteed to return an optimal solution

C. Consider a graph search problem where for every action, the cost is at least x ,with $x > 0$. Uniform-cost graph search is guaranteed to return an optimal solution

D. None of these

5 Which statements about the recurrent neural network are true?

A. The idea behind RNNs is to make use of sequential information.

B. Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm

C. In a recurrent neural network we assume that all inputs (and outputs) are independent of each other.

D. None of these

Solution: 1. D 2. D 3. AC 4. C 5. AB
[Q2]. In class we have learned three uninformed search methods, namely BFS, DFS, and UCS with graph search algorithm. Answer the following questions,

1. In not more than 3 sentences, give one scenario whereby DFS outperforms BFS.
   Solution: On fully connected graph with high branching factor where the goal node is on one of the leaves. BFS will experience exponential blowup in nodes expanded as branching factor increases.

2. In not more than 3 sentences, give one scenario whereby BFS outperforms DFS.
   Solution: On sparse graph with low branching factor, but very long chain down the tree. DFS can get stuck following long and irrelevant chains, if the goal node is located the other shallow branch.

3. In not more than 3 sentences, give one scenario whereby DFS takes up the same memory space as BFS. You can draw a graph with not more than 5 nodes to illustrate this scenario.
   Solution: Consider a graph with just two nodes, a start node pointing to a goal node. Obviously in this case, DFS takes up the same memory cost as BFS.

4. Consider a UCS without cycle pruning. Draw a scenario using not more than 5 nodes in total, where UCS will be *incomplete*, meaning that it will be unable to find the goal node. Give a maximum of 2 sentences explanation.
   Solution: Since there's no cycle pruning, UCS will infinitely loop between node A and B.

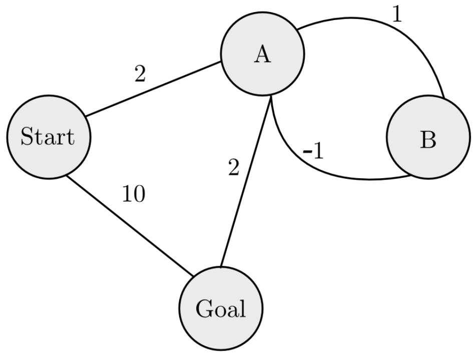


Figure 1: Answer to UCS without cycle pruning

5. Refer to Figure 2. Can UCS with graph search algorithm find the optimal (least-cost) path to the Goal node? What will be the solution after running UCS from the start node?

   Solution: No. The solution from running UCS with graph search algorithm will be Start $\rightarrow$ Goal.

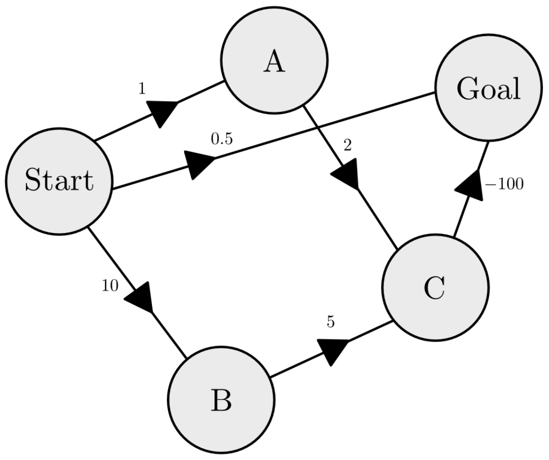


Figure 2: A Graph

6. Now we add +101 cost to all edges in the graph in figure 2 to make them all positive and run UCS until Goal node is found. What will be the solution after running UCS from the start node? Is this path the optimal path from the original graph in figure 2?

   Solution: The solution in the modified graph is still the same, Start → Goal. And no, it is not the optimal path in the original graph. It is however, the optimal path in the *modified* graph.

7. State whether the following statement is true: *UCS with graph search algorithm* ***cannot*** *find optimal path on* ***any*** *graphs that have negative weights.* If Yes, briefly state why, if No, draw a counter-example with no more than 5 nodes.
   Solution: False. This is not true for **all** graphs with negative weight. Consider a directed connected graph where it has negative edges **only** on the **outgoing** edges from the start node. In this graph, UCS with graph search algorithm can find the optimal path to a designated goal node.

[Q3]. Refer to figure 3. We start at location $E1$ (read off the column first and then the row). The order when we expand the cells when searching from s is in lexicographic order. For example (arbitrarily), $A1$ is expanded first before $A2$, and $A2$ is expanded first before $B1$. The black tiles represent "walls", which means that these tiles cannot be expanded. The each tile can expand to any other 8 tiles around it (if any), which means not only the adjacent but also including the diagonals. Run iterative deepening search with graph search algorithm from $d = 1, 2$, and 3. Write down the frontier how it looks like *after* every expansion. The elements of the frontier should be written as a path from the root to the last node of that path, e.g. E1-D1-C2 . Paths that have reached the depth limit are simply removed from the frontier when they are expanded. By the time $d = 3$, will node g be found?

*Note: *after* expansion means the following: if you expand a node.state into 3 new states, then write how the frontier looks like after the expanded node is removed and the 3 new paths (that have the new states as ends of their paths) are added.*

Solution:
At $d = 1$, (assume the frontier is on the left side)

$$E1$$
$$(E1, D1), (E1, E2)$$

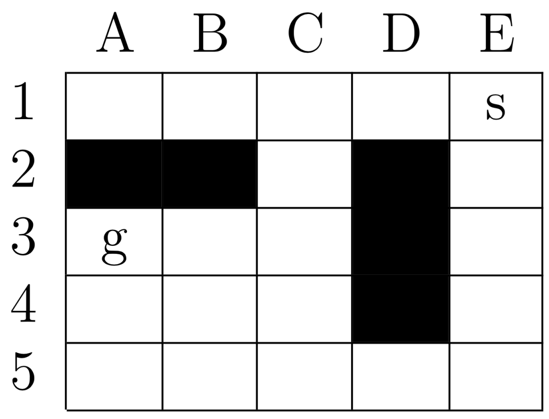


Figure 3: A 5×5 Grid

At $d = 2$,

$$\begin{aligned}
&E1\\
&(E1, D1), (E1, E2)\\
&(E1, D1, C1), (E1, D1, C2), (E1, E2)\\
&(E1, E2, E3)
\end{aligned}$$

At $d = 3$,

$$\begin{aligned}
&E1\\
&(E1, D1), (E1, E2)\\
&(E1, D1, C1), (E1, D1, C2), (E1, E2)\\
&(E1, D1, C1, B1), (E1, D1, C2), (E1, E2)\\
&(E1, D1, C2, C3), (E1, D1, C2, B3), (E1, E2)\\
&(E1, E2, E3)\\
&(E1, E2, E3, E4)
\end{aligned}$$

No, node g is not found by the time $d = 3$.

[Q4]. Consider the RNN structure in figure 4. It has a scalar input at the first time step, and make a scalar prediction at the last time step. The activation function at each node is a nonlinear function $g(z)$.
In other words, the output of each node is multiplied by weights w, and put into the activation function of the next node. For example, $z^2 = g(z^1 \cdot w)$.

Answer the following questions,

1. Express y in terms $z^i, i = 1, ..., T$, weights w and input x. You don't need to express everything in one equation, i.e. you can write more than one equation to express y.
   Solution:

$$\begin{aligned}
y &= z^T\\
z^i &= g(z^{i-1} \cdot w), i = 2, ..., T\\
z^1 &= x
\end{aligned}$$

2. During backpropagation, although all weights w are the same throughout the network, the backpropagated error to each of the weights $w_{i,j}$ between neuron i and j is not the same. Assume an arbitrary error function $E(y)$. The backpropagation rule is, for weight $w_{i,j}$ between neuron i, j,

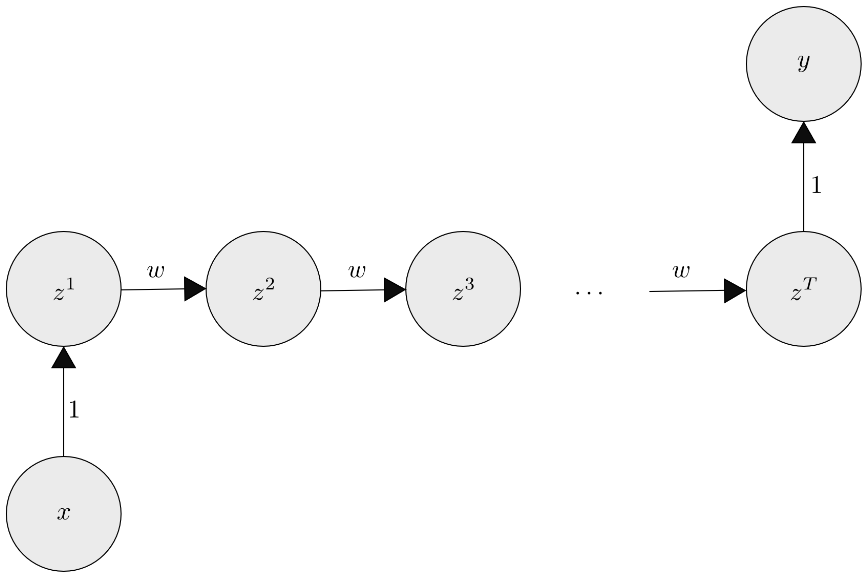


Figure 4: RNN for Q3

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z^T}\frac{\partial z^T}{\partial z^{T-1}}\frac{\partial z^{T-1}}{\partial z^{T-2}}\cdots\frac{\partial z^j}{\partial w_{i,j}} \tag{1}$$

$$\frac{\partial z^j}{\partial z^{j-1}} = (wg'(z^{j-1}\cdot w)) \tag{2}$$

And then since all weights have to be updated the same way, the total derivative is,

$$\frac{\partial E}{\partial w} = \sum_{i=1,j=2}^{i=T-1,j=T} \frac{\partial E}{\partial w_{i,j}}$$

Assume that the input to the network is $x = 0$. Answer the following True/False question with max of 3 sentences of explanation.

(a) The network will suffer only from vanishing gradient problem [T/F]
Solution: False, if $(wg'(z^{T-1}\cdot w)) > 1$ then the gradient explodes.

(b) The network will suffer only from exploding gradient problem [T/F]
Solution: False, if $(wg'(z^{T-1}\cdot w)) < 1$ then the gradient vanishes.

(c) Assume the activation $g(x) = \text{ReLU}(x)$. Recall ReLU function is $g(x) = \max(0, x)$. If negative value is inputted to the ReLU function, it will completely block (meaning prevent, turned them into zero) back-propagation to the earlier layers [T/F]
Solution: True, as long as the input is negative, then ReLu function will always be zero.

(d) Consider a leaky ReLU function as an activation function instead, $\text{h}(x) = \begin{cases} x \text{ if } x > 0 \\ \epsilon x \text{ otherwise} \end{cases}$, where ϵ is a small positive number such as 0.01. Unlike normal ReLU function in the previous part, using leaky ReLU as activation function no longer blocks back-propagation error to the earlier layers [T/F]
Solution: True, because it will no longer be 0 when input is negative so backpropagation still continues down the chain.