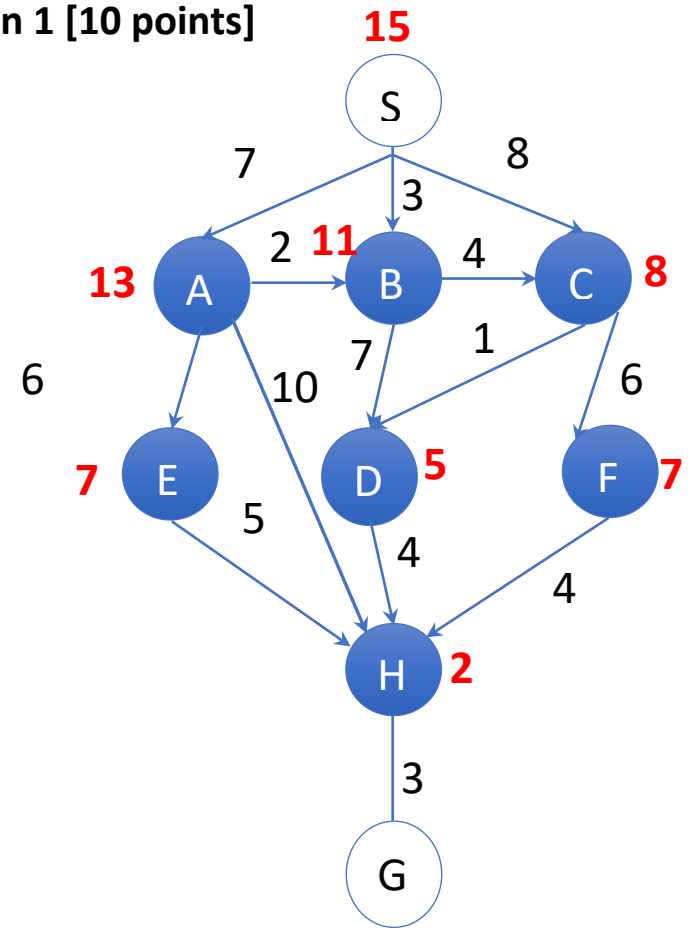
Assignment 2

Note: Your solution for this assignment should have two parts---a pdf document and code files.

- Have a **single pdf** document (**name it only using your full name**) that shows your solution for different questions (show either numerical values if the question asks for it, and/or theoretical justification as required). **Include in this pdf, the code you wrote for the solution for the respective question (if coding is required).**
- Upload your real code files that you used to solve the particular question. Make sure your code is neatly organized per question, runs correctly, and has comments that highlight the part you implemented so that your TA can easily understand it
- Combine your solution pdf and code files in a single zip folder and upload it on the eLearn assignment folder
- Solution should be typeset using a profession software (word, keynote, latex etc). No handwritten solutions are allowed.
- **VERY IMPORTANT: This is an individual assignment. You SHOULD NOT collaborate with others. Supporting or indulging in plagiarism is a violation of academic policy and any such attempts will lead to strict disciplinary action.**

Question 1 [10 points]



For the above graph (heuristic values are provided in red color and actual costs are in black color), please provide answers to the following answers. Also note that S is start state and G is goal state.

(a) Is the heuristic admissible? Provide justification.
You may want to use the intermediate table below to answer this question. Feel free to add rows/columns.

Node n	Minimum cost to reach goal from n	h(n)
S		
A		
B		
C		
E		
D		
F		
H		

(b) Is the heuristic consistent? Provide justification.

(c) Provide the search steps (as discussed in class) with DFS, BFS (with and without priority queue), Best First Search and A* search. Specify for each algorithm if the open list is queue, stack, or priority queue. Feel free to add rows/columns, or other details in the table below. Open list contains nodes that are to be explored, and "Nodes to add" are the successors of the node that is recently popped.

Algorithm: DFS/ BFS (without priority queue)/ BFS-Optimal (with priority queue)/ Best First Search/ A* Search

Step #	Open List	POP	Nodes to add
1	S		

Path: , cost :

Question 2 [10 points]: Build an automated car in a simulator

The goal of this assignment is to guide your car to move in a multi-lane straight road (represented using a rectangular grid) simulator (details explained later). In addition to your car, there are other cars also moving along the road. You need to design an algorithm which can take your car from its start position to the end of the road as soon as possible while avoiding other cars. Other cars are moving randomly. At any step, your car can view up to 4 cells in front, left and right, i.e., it will not be able to look at the other cars which are beyond its visibility range of 4 cells. (This visibility range of 4 is configurable and your algorithm should work irrespective of visibility range).
We describe the state, operators and goal state for this problem below.
State: State of the car is its location, i.e., the grid cell it is currently present in.
Operators/Actions: Following are the available operators or actions for the car

(Forward, Forward-2x, Forward-3x, Left, Right, None)

1. Forward – Moves the car one step ahead in the same lane.
2. Forward-2x – Moves the car two steps ahead in the same lane.
3. Forward-3x – Moves the car three steps ahead in the same lane.
4. Left – Moves the car in the left lane and one step forward
5. Right – Moves the car in the right lane and one step forward.
6. None – Stays at the same place.

If the action is unsuccessful then the car will stay at the same place. Action will be unsuccessful if there is a wall or another car present in the intended direction of move. For example, a forward will be unsuccessful if there is another car present in front of your car. As the left action moves a car to the left and one step forward, a left action taken in grid cell (5,3) will be unsuccessful if there is a car in either of (4,3) or (4,4).
Goal State: There is one goal state, where your car needs to reach.

Given the observed state, you have to:

(a) Write the function that will generate the operator/action to be taken at the current time step (based on search algorithms discussed in class), so that time taken to reach goal is as minimal as possible. **[You may refer "pseudo code.pdf" for implementation details and pseudo code for A* search.]**
(b) Submit the function code in pdf file. The code has to compile and run as part of the simulator without issues.

The code for the Simulator is provided in the zip file SelfDrivingCar.zip. The details about the installation/running of simulator and code details are provided in the file SelfDrivingCar.pdf (present in the zip file).

Question 3 [10 Points]

Taxi drivers in Singapore can pick up customers from any location that is not on highways. However, they require guidance on where to pick up customers when they do not have customers on board and there are no bookings. In this question, we address this guidance problem.

There are four locations: L1, L2, L3 and L4 from where taxi drivers can pick up and drop off customers. At any decision epoch, the chances of the taxi driver picking up a customer from different locations are provided in Table 1 below:

Location	Chance of finding customer
L1	0.3
L2	0.8
L3	0.1
L4	0.6

Table 1

Once the taxi driver picks up a customer, the customer determines the destination. Observed probabilities (from past data) of a customer starting from a source location and going to a destination location are given below:

Source→ Destination	**Probability**
L1 → L2	0.4
L1 → L3	0.35
L1 → L4	0.25
L2 → L1	0.4
L2 → L3	0.6
L3 → L1	0.6
L3 → L4	0.4
L4 → L1	0.65
L4 → L2	0.35

Table 2

Just to avoid any confusion, first row of Table 2 below the heading row indicates that a customer picked up from L1 gets dropped off at L2 with 0.4 probability.

Travel fare between different locations are as follows:

Source→ Destination	**Fare**
L1 → L2	8$
L1 → L3	13$
L1 → L4	15$
L2 → L1	10$
L2 → L3	9$
L3 → L1	13$
L3 → L4	10$
L4 → L1	9$
L4 → L2	7$

If a source destination pair does not appear in the above table, it indicates the fare is 0.

Cost of travelling between locations is as follows:

Source→ Destination	**Cost**
L1 → L2	1$
L1 → L3	1.5$
L1 → L4	1.25$
L2 → L1	1$
L2 → L3	0.75$
L3 → L1	1.5$
L3 → L4	0.8$
L4 → L1	1.25$
L4 → L2	1.00$

If a source destination pair does not appear in the above table, it indicates the cost is infinity. If the taxi does not move (i.e., source and destination are same), then the cost is zero.

The taxi driver can either pickup from the current location or move to another location. Pickup corresponds to picking up a customer (if one is found) and dropping them of at their destination. Pickup action succeeds with probabilities specified in *Table 1* and if the taxi picks up a customer, destination location is determined by the probabilities in *Table 2*. When Pickup action fails, the taxi remains in its current location. Move to another location is always successful and taxi moves to the desired location with probability 1.

Both pickup and move actions take one-time step each. Please provide the following:

(a) You need to provide an MDP model that guides the taxi driver on "move and pickup customers". MDP is the tuple <S, A, P, R>, i.e., the states, actions, transition probability matrices (for different actions) and reward functions. You may use the below table to show different components of the MDP model (you can modify rows and columns as needed).

You can make one table as shown below for each possible current state.

s (current state)	a (action)	s' (new state)	$Pr(s'\|s,a)$	$R(s,a,s')$
State 1				

(b) After providing the MDP, show three iterations of value iteration. Initialize $\forall s, V^0(s) = 0$ and calculate

(i) $\forall s, V^1(s),\ V^2(s)$ **and** $V^3(s)$

(ii) $\forall s, \pi^1(s),\ \pi^2(s)$ **and** $\pi^3(s)$

You can use below format for your solution. You may modify it as needed. You will need additional blocks of rows for writing the value function, Q-function and other things for different states. Replace state 1, state 2 etc with the actual names of states you plan to use.

s	a	$V^0(s)$	$Q^1(s,a)$	$V^1(s)$	$\pi^1(s)$	$Q^2(s,a)$	$V^2(s)$	$\pi^2(s)$	$Q^3(s,a)$	$V^3(s)$	$\pi^3(s)$
State 1		<Value of state here>		<Value of state here>			<Value of state here>			<Value of state here>	

Question 4 [10 Points]

In this question, you need to install OpenAI gym (http://gym.openai.com/docs/). Once the gym is installed, you have to implement Q-learning for the FrozenLake environment (https://gym.openai.com/envs/FrozenLake-v0/) in python using the gym library and show the rewards obtained. You may use a discount factor of 0.95. Please provide the following two things in your solution for this question:

(a) Code that implements Q-learning for Frozenlake example in gym? Copy and paste the code in the solution pdf, and provide the actual code file also.
(b) For each episode, compute the total accumulated reward (also called episode return). Plot the average return (over the last 100 episodes) while your agent is learning (x-axis will be the episode number, y-axis will be the average return over the last 100 episodes). Make sure that you train for sufficiently many episodes so that convergence occurs.

The goal in this question is to ensure you familiarize yourself with OpenAI and understand how to implement Q-learning in OpenAI. There are many resources available online on implementing Q-learning in OpenAI and the right value of learning rate for the frozen lake example. You are free to refer to them, but please write your own code. Tabular Q-learning should work in this question.

Question 5 [10 Points]

In this question, you will employ Singular Value Decomposition to obtain word embeddings and compare the generated word embeddings with the word embeddings generated using word2vec. The corpus (or dataset) to be considered is “200Reviews.csv”. You need to do the following:

(a) Parse the reviews in “200Reviews.csv”, i.e., divide reviews into sentences and sentences into words and remove the stop words. You can employ the “NLP-pipeline-example.ipynb” example we talked about in class.

(b) Create the co-occurrence matrix for all the remaining words (after stop words are eliminated), where the window of co-occurrence is 5 on either side of the word.

(c) Apply SVD and obtain word embeddings of size 100.

Then, please generate word embeddings of size 100 using Word2Vec.pynb (from the NLP lecture) on the same “200Reviews.csv” dataset. Please show comparison on few examples to understand which method works better. Note your observations in your solution.