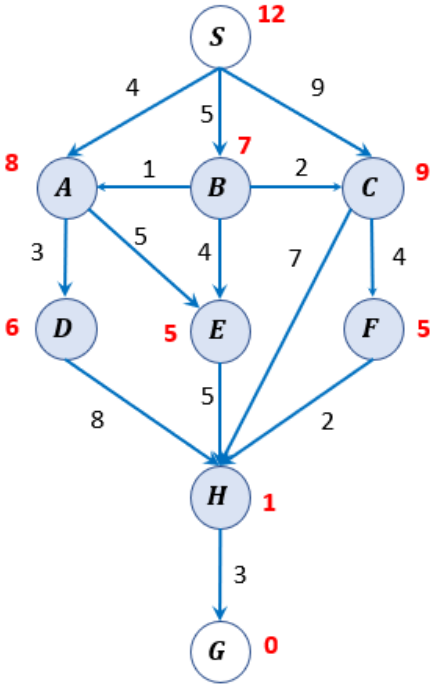
Assignment 2

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:

Please note that S is start state and G is goal state.

(a) Is the heuristic admissible? Provide justification.

Please fill the following intermediate table below to answer this question.

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

(b) Is the heuristic consistent? Provide justification.

(c) Provide the search steps (as discussed in class) with vanilla **BFS**, and **A* search**. Show the **final solution path** and the **cost of that solution** for each algorithm.

Specify for each algorithm if the open list is queue, stack, or priority queue. As a simplifying assumption, let index zero (i.e. the first element) in the open list be the top of the stack or front of the (priority) queue, as appropriate for the corresponding algorithm. Break ties by alphabetical order.

Please use the following tables for your working. Open list contains nodes that are to be explored, and "Nodes to add" are the successors of the node that is recently popped or dequeued.

Algorithm: BFS

Step #	Open List (Queue or stack or priority queue).	POP or DEQUEUE	Nodes to add
1	S		

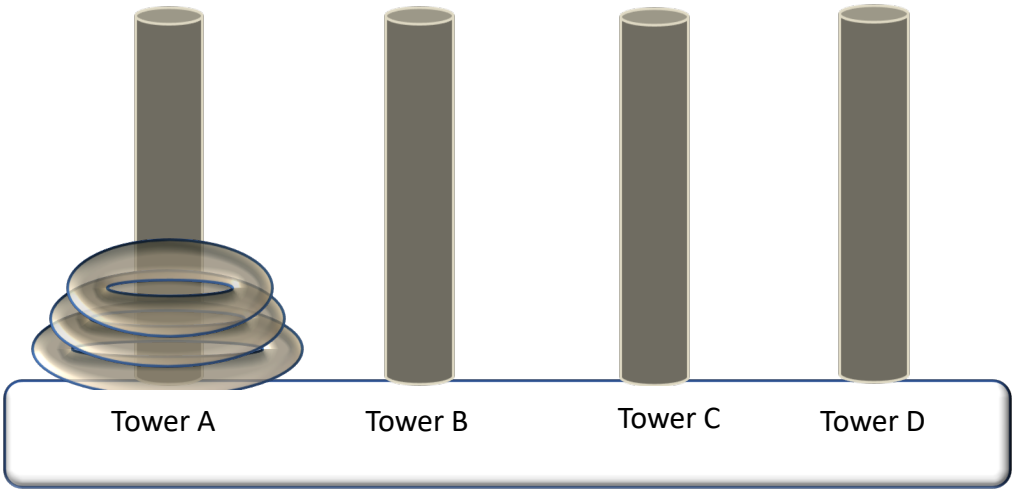
Path: , cost :

Algorithm: A* Search

Step #	Open List (Queue or stack or priority queue).	POP or DEQUEUE	Nodes to add
1	S		

Path: , cost :

Question 2 [10 points]: Towers of Hanoi



Towers of Hanoi is a famous problem where discs of varying sizes must be moved from one tower to another. Here we have a variant to move discs from tower A to tower D with the help of intermediate towers, tower B and C, in the least number of moves. The key constraint is that a bigger disc should never be put on top of a smaller disc on any tower. In case of 3 discs on tower A as shown in figure above, the sequence of moves will thus be:

(1) Move top disc from tower A to tower B
(2) Move top disc from tower A to tower C
(3) Move top disc from tower A to tower D
(4) Move top disc from tower C to tower D
(5) Move top disc from tower B to tower D

Please answer the following questions:

- Given "N" discs (stacked appropriately in tower A with the smallest disc on the top), formulate towers of Hanoi problem as a search problem, i.e.,
    - indicate the states (describe specifically what is a state in this problem, how you would store it in a computer using a data structure, and justify the correctness of your state representation).
    - actions,
    - cost of different actions,
    - successor state (for each action) and
    - the objective.

    Please pay special attention to what **data structure** you would use to store any given state in this problem. We are looking for a state representation that is easily stored in a computer, and it is easy to generate its successor for any valid action. For example, in 8-puzzle problem, a 3x3 numpy array can represent any state. ***Just an intuitive definition for what is a state will not suffice for this question.***

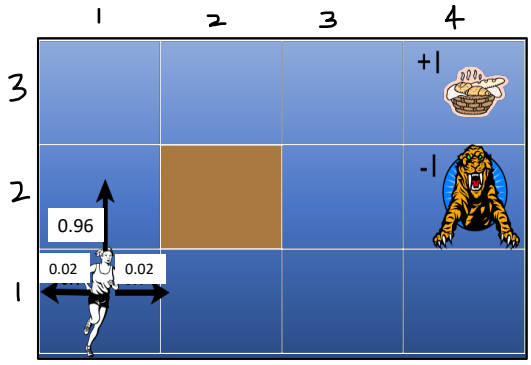
- For using heuristic search method such as A*, provide an admissible heuristic and justify why it is admissible.

  Note: Make sure that heuristic is easy to compute, informative (i.e., do not assume heuristic value as zero for all the states) and intuitive enough so that you can justify that it is admissible.

- Show first three steps of **DFS search**. You can use the table structure as in question 1c to show different steps. Assume the start state is as shown in the figure with three rings placed on tower A.

Question 3 [10 Points]

Please recall the robot runner in the grid world example with tiger and food cells on right last column.



The states are grid cells, i.e., (1,1), (1,2), …. First component of a state is the row number, second component is the column number. The actions available are North, East, West, South.

Rewards are given as follows:

- R(*, *, *) = -0.0025
- R(*, *, (3,4)) = +2 (Moving to Food cell)
- R(*, *, (2,4)) = -1 (Moving to Tiger cell)

Transition probability is 0.96 to the state in the direction of the action and 0.02 in states perpendicular to the direction of the action. If agent hits a wall, the agent moves back to its original location. Terminating states are when agent is in the food cell or in the tiger cell. Discount factor is 0.95.

For value iteration method, the values of different states $V^t(.)$ at an iteration **t** are given by:

3	1.783	1.886	1.992	**Food**
2	1.688		1.829	**Tiger**
1	1.598	1.634	1.727	1.582
	1	**2**	**3**	**4**

(a) Compute the values $V^{t+1}(.)$ for different states at iteration **t+1.** Write numerical answers below for each state:

3				**Food**
2		blocked		**Tiger**
1				
	1	**2**	**3**	**4**

Please show analytical expressions denoting all the computations (without using any python code).

(b) What is the policy for each state as per the Q- function $Q^{t+1}(..)$. Note it down below for each state:

3				**Food**
2		blocked		**Tiger**
1				
	1	**2**	**3**	**4**

Please provide clear analytical justification for extracting the policy as per Q^{t+1}.

Question 4 [10 Points]

In this question, you need to install OpenAI gym:

- https://github.com/openai/gym (installation instructions)
- https://www.gymlibrary.dev/content/basic_usage/ (documentation)

Once the gym is installed, you have to implement Q-learning for the **FrozenLake-v1** environment (https://www.gymlibrary.dev/environments/toy_text/frozen_lake/) in python using the gym library and show the rewards obtained. Use option **is_slippery=True**. Use map size 4x4 (check option `map_name').

You may use a discount factor of 0.90. Please provide the following things in your solution for this question:

(a) Code that implements Q-learning for the FrozenLake-v1 example in the gym? Copy and paste the code in the solution pdf, and provide the actual code file also. A key thing to determine first is what is the size of the state space and action space in this environment to make the appropriate data structures such as the Q-table [**6 points]**

(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. **[3 points]**

(c) What is the learning rate you used for this problem? How did you select this learning rate? Please explain your observations from changing the learning rate (i.e., what effects learning rate had on performance if it was low versus high). **[1 point]**

The goals of this question are to ensure you familiarize yourself with OpenAI and 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 FrozenLake example. You are free to refer to them, but please write your own code. Please use the standard tabular Q-learning for this question (without using any neural nets or any other function approximator.).

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 a set of tweets posted to airline companies on Twitter, a real-life dataset for customer service-related tasks. You need to do the following (you are free to use helper functions if required):

(a) Update the load_data function from the Q5_template_tweets.ipynb to preprocess words: remove non-letters, convert words into the lower case, and remove the stop words. You can employ some functions from the "NLP-pipeline-example.ipynb" example and may use regular expressions from "Word2Vec.ipynb". [2 marks]

(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. What is the size of your vocabulary (i.e., how many unique words you end up with)? [4 marks]

(c) Apply SVD and obtain word embeddings of size 50. [2 marks]

(d) Then, please generate word embeddings of size 50 using Word2Vec.pynb (uploaded in class lecture material) on the same dataset. Please show comparison on few examples to understand which method works better. Note your observations in your solution. You can use words like "delay", "flight" etc to compare the two models. [2 marks]