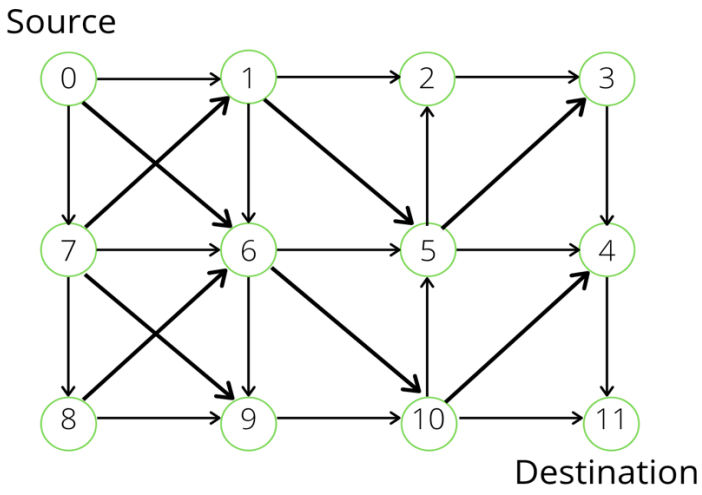
18-462/662 Assignment 3 {First draft}

Carnegie Mellon University
Department of Electrical and Computer Engineering

1. Alice wants to travel from city 0 to city 11 by bus. Bus routes between different cities along the way are as shown using arrows below.



Implement BFS or DFS using Python that :

- Gives the number of paths between source and destination with exactly 5 cities in between.
- Gives the number of paths between source and destination with at most 4 cities in between.
- Gives the number of paths between source and destination with at least 6 cities in between.

2. Implement A* algorithm to solve the 8-puzzle problem shown below in Python. Irrespective of the initial state of the puzzle, the final state has to be constant as shown below. Only one occupied cell can be moved to the place of an empty cell at a time. Make use of operations such as move left / right / up/ down for each cell.

	1	2
3	4	5
6	7	8

Final State

Evaluate and show your code for the initial states below:

5	7	8
2	3	1
4		6

Initial State 1

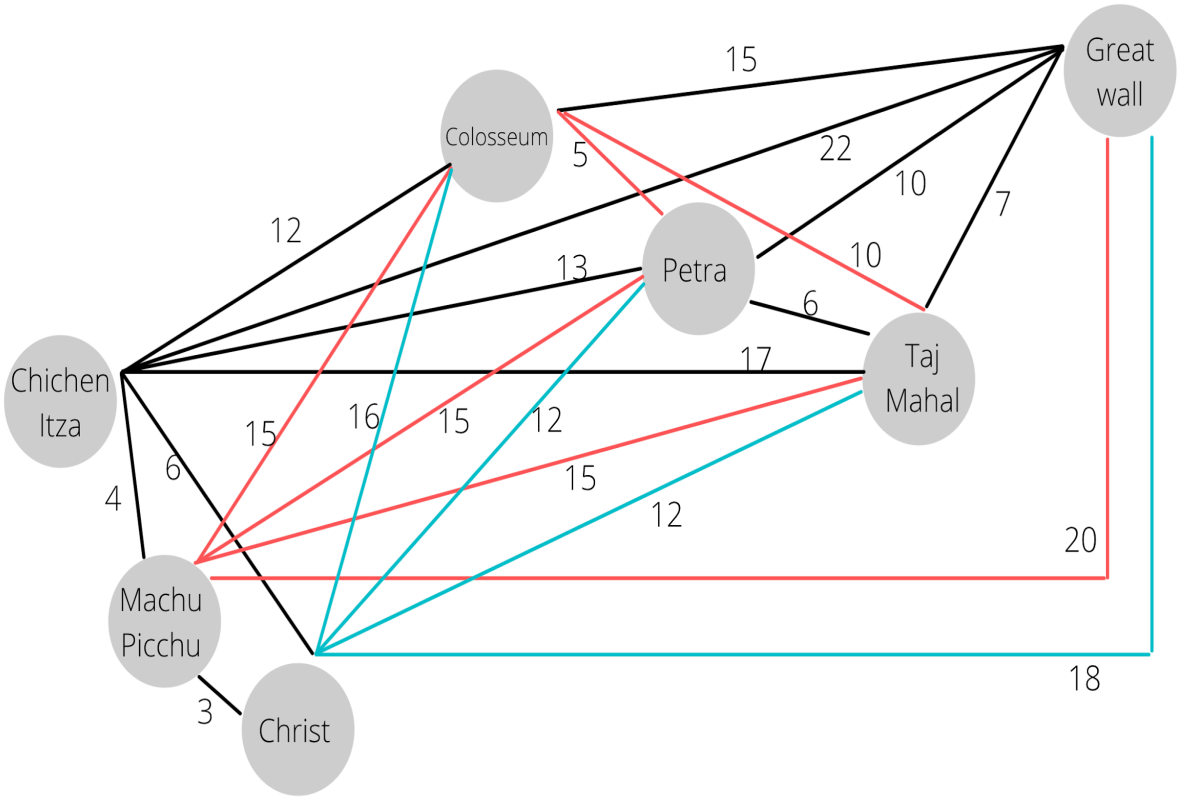
2	3	8
	4	6
1	7	5

Initial State 2

1	6	5
8	7	
3	2	4

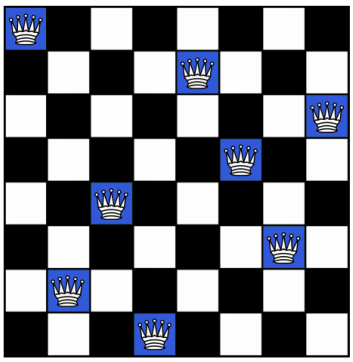
Initial State 3

3. Sheldon wants to visit each of the seven wonders of the world once. Assume that he starts his journey from the Great Wall of China, visits each wonder once along the way and returns back to his origin; i.e., the Great Wall of China.



The weights on edges represent the number of days needed to travel.
Implement Traveling Salesman algorithm using Python to find the path that Sheldon has to use in order to complete his trip in a minimal number of days.

4. Design a Goal-based Intelligent agent for the following example using Python



You are given an 8x8 chessboard, find a way to place 8 queens such that no queen can attack any other queen on the chessboard. A queen can only be attacked if it lies on the same row, or same column, or the same diagonal of any other queen.

For the sake of programming, represent the squares occupied by Queens as 1 and remaining squares as 0 in an 8x8 matrix. Finally, plot the matrix in the form of chess board with queens' locations marked distinctly.

[Hint: You can make use of sns heatmap within seaborn library for plotting]. In writing your Python code, the following information can be used:

States: all arrangements of 0 to 8 queens on the board.
Initial state: No queen on the board
Actions: Add a queen to any empty square
Transition model: updated board
Goal test: 8 queens on the board with none attacked

5. Write a Python code to solve the following Sudoku puzzle by applying the Graph Coloring Algorithm covered in class.

			4					
4		9			6	8	7	
			9			1		
5		4		2				9
	7		8		4		6	
6				3		5		2
		1			7			
	4	3	2			6		5
					5			

Sudoku is a single player logic-based puzzle. A Sudoku puzzle is a grid of 81 cells, which is divided into 9 rows, columns and regions (or blocks). The goal is to place the numbers from 1- 9 into empty cells in such a way, that in every row, every column and every region (3 x 3 block) each number appears only once.

Filling the table with the numbers must follow these rules:

- Numbers in rows are not repeated
- Numbers in columns are not repeated
- Numbers in 3 × 3 blocks are not repeated
- Order of the numbers when filling is not important

Your Python code should display completed Sudoku with empty cells in the above puzzle filled according the mentioned rules.

[Hint: Consider each cell in the Sudoku as a node of the graph. Each node will have an edge to every other node (cell) in its respective row, column, and 3 x 3 block]

6. The table below shows 4 teams: Royal Challengers Bangalore (RCB), Mumbai Indians (MI), Delhi Capitals (DC), Chennai Super Kings (CSK) participating in a Cricket tournament.

Teams	Wins	Losses	Games left	Remaining games Aganinst			
				RCB	MI	DC	CSK
RCB	83	71	8	NA	1	6	1
MI	88	79	3	1	NA	0	2
DC	78	78	6	6	0	NA	0
CSK	77	82	3	1	2	0	NA

The table summarizes the win/loss record for each team, number of games remaining for each team and the opponents to be played against in those games.

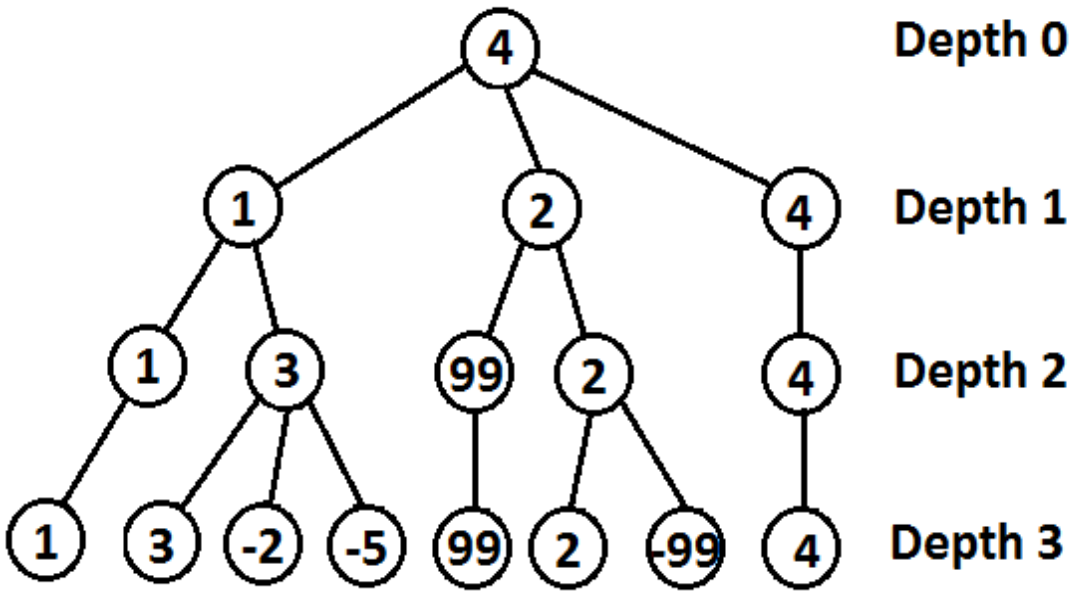
Team with the maximum wins at the end of all matches in the tournament will be the winner. The remaining other teams are considered to be eliminated.
(For example: if Mumbai Indians (MI) win their remaining 3 games, then they would have 91 wins in total. If no other team wins 91 or more matches, then MI will be the winner of tournament).

Write a Python Code to apply Maximum Flow algorithm to determine which of the teams will be eliminated in the race to finish first and which team will win the tournament.

7. Write Python code to implement the Minimax Algorithm covered in class as follows:

The game is fully-observed for both agents (opponent and our intelligent agent). Our intelligent agent wants to maximize our profit, while the opponent wants to maximize his/her profit (minimize ours). Two players choose the next node one by one in order and our agent goes first. The value shown in the node is the final profit (positive value means we win this profit from our opponent, while negative value means we lost that profit to our opponent).

Take a tree object (lists of lists of lists …. to represent this) as an input, and output the index of choice and expected profit. For example,
If the input is [[[1], [3, -2, -5]], [[99], [2, -99]], [4]]



The output should be tuple: (2, 4) which means we should choose node 2 with an expected value of 4. We will test your function by importing it.
Hint: Use Recursion.

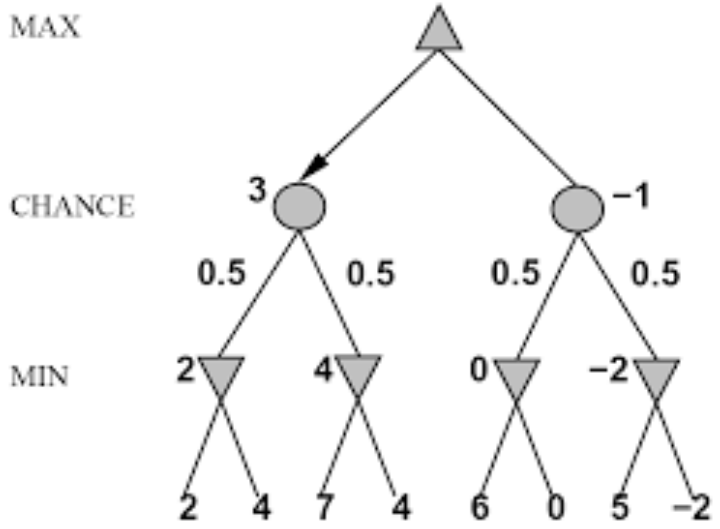
8. Write a Python code to implement the Expectimax (Expectation Maximization) Algorithm covered in lectures.

The game is fully-observed for both agents (opponent and our intelligent agent). The difference between problem 7 and this problem is that after our intelligent agent chooses a node, there is a chance for the game to enter different nodes. Our intelligent agent wants to maximize our profit (the expectation), while the opponent wants to maximize its profit (minimize our expectation). Two players choose the next node one by one in order and our agent goes first. The value shown in the node is the final profit (a positive value means we win this profit from our opponent, while a negative value means we lost that profit to our opponent).

Take a tree object (lists of lists of lists …. to represent this) as an input, and output the index of choice and expected profit. At max node, each element in the list is in the format (p, min_node) where p is the chance to go to this min node by selecting this max node. At min node, each element in the list will be the max_node, like problem 7.

For example,

If the input is [[(0.5, [2,4]), (0.5, [7, 4])], [(0.5, [6, 0]), (0.5, [5, -2])]]



The output should be tuple: (0, 3) which means we should choose node 0 with an expected value of 3. We will test your function by importing it.

Hint: Use Recursion.