AI ASSIGNMENT 2
Date of Submission : 31/10/2023

Total Marks : 25 Marks
Theory section: 10 Marks
Computational section: 15 marks

Instructions:

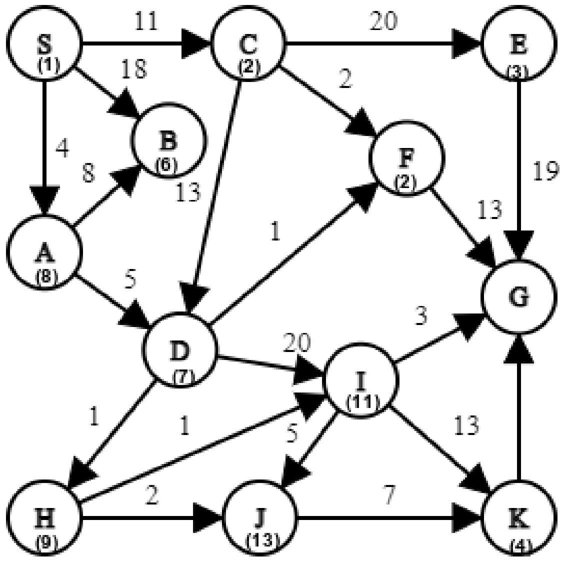
1. Assignments are to be attempted individually.
2. Submit a .pdf for theory question and .py file for programming question.
3. Submit the assignment in a zip folder with name AI2_Rollnumber
4. **Programming Language :** Python
5. Demonstrate the program to your assigned TAs.
6. **Extension and Penalty clause:**
   - Even a 1 minute late submission on google classroom will be considered as late. Please turn-in your submissions atleast 5 minutes before the deadline.
   - Not explaining the answers properly will lead to zero marks.

Theory

1. Suppose you decide not to keep the explored set (i.e., the place where the explored nodes are saved) in the A* algorithm.

   a. If the heuristic used is admissible, will the new algorithm still be guaranteed to find the optimal solution? Explain why (or why not). [**2 marks**]
   b. Will the new algorithm be complete? Explain. [**1 marks**]
   c. Will the new algorithm be faster or not? Explain. [**1 marks**]

2. Given the graph, find the cost-effective path from S to G. S is the source node, and G is the goal node using A* [**2 marks**], BFS [**2 marks**], and Dijkstra algorithm. [**2 marks**] Please note: cost written on edges represents the distance between nodes. The cost written on nodes represents the heuristic value. Heuristic for Goal Node G is 0. Distance from K to G is 6.



Computational

3. (a) Use the city "Road Distance" data given. Assume that only these roads between the cities exist.

   (b) Write a Python program to search a road route from any city to any other city using this data. It should work for both cities that are directly connected (say, Ahmedabad to Indore) as well as for two cities that are not directly connected (say, Agartala to Hubli).

   (c) Show Uniform Cost Search and A* search on this data.

   (d) You should use Python features such as Lists, Input/ Output, Recursion, Backtracking etc.

   (e) Your code should work for different inputs, i.e., the user should be able to input any pair of cities as the origin-destination pair from the command line and the program should return the best found path. This means you *must* not hard-code your solution for a set of pairs.

   (f) Design two different, non-trivial (e.g., a constant $h(n) = c$ is a trivial heuristic) heuristics for A*. (1) The first one should be admissible. Empirically verify that the admissible heuristic yields the optimal solution. (2) Come up with an inadmissible heuristic function that will expand fewer nodes than your admissible heuristic in (1). Show this property for *at least* two origin-destination pairs in your code.

   (g) Python program should work as marks will be based on the working demo of the program and your explanation of the program.

   (h) Marks will be awarded for the assignment as follows:

      - Working program [**8 marks**]

- Compare both the algorithms in terms of performance and explain.[**2 marks**]
- Kind of Python features used[**2 marks**]
- Explain the heuristics chosen for admissible and non-admissible.[**3 marks**]