UNIVERSITY OF WATERLOO

Department of Electrical and Computer Engineering

ECE 457A Adaptive and Cooperative Algorithms

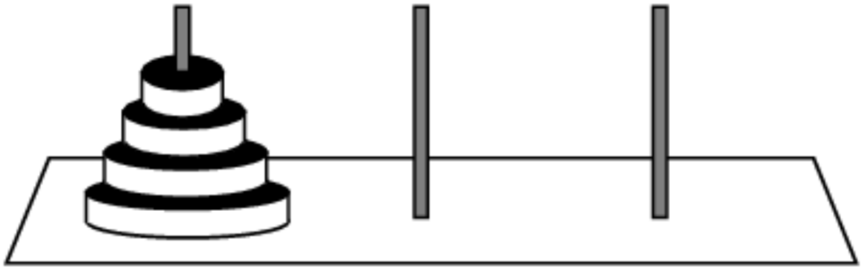
Instructor: Dr. Otman A. Basir

Assignment 1
Due Date: September 27, 2024 11:59PM

- **Instructions**
  - The assignment should be done individually
  - You should upload your answers as a PDF file on learn before 11:59pm of the deadline date
  - Attach any codes used for the assignment separately as a compressed file to the same dropbox
  - You can use any programming language (Matlab and Python are preferred)
  - Works with more than 30% similarity will not be marked
  - Communicate any issues or concerns with the TAs.

Note: For Problems 1, 2 and 3, determine a problem formulation:
- Define the initial and the goal states, and the restrictions imposed by the problem.
- Define the states
- Define all possible actions
- Draw the state space diagram
- Show the sequence of actions which takes us from the initial state to the goal state (Problem formulation)

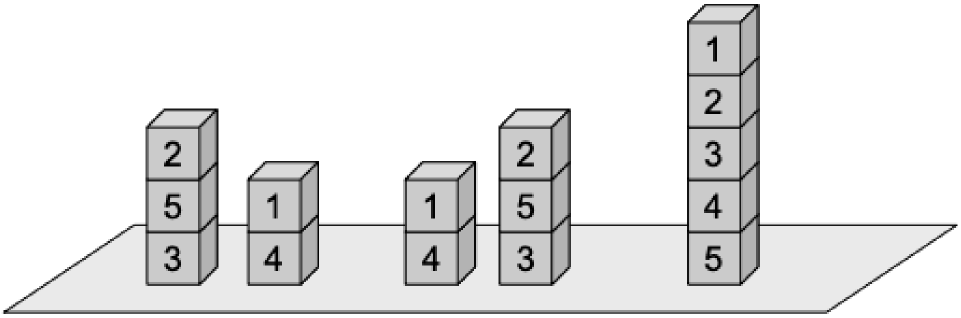
Problem 1. (**1 point**) Assume you have 3 poles and 64 discs of decreasing diameter. A disc may be stacked only on top of a larger disc or onto an empty pole. Initially, all discs are stacked correctly on one of the poles. The objective is to correctly stack all of the discs on a different pole.



Problem 2. (**1 point**) You are given two jugs- a 4 gallon jug and a 3 gallon jug . Neither jug has any measuring marks on it . There is a pump that can be used to completely fill one or both jugs . You can pour water from one jug into the other or onto the ground . The problem is to get exactly 2 gallons of water into the 3 gallon jug .

Problem 3. (**3 points**) The following problem is a well-known problem in AI, specifically in planning problems related to robots. It consists of a set of blocks of various size, shape, and color, possibly identified by a letter or by a number, and placed on a surface (e.g., on a table); the blocks can be stacked into one or more piles, possibly with some constraints (depending, e.g., on their size); the goal is to form a target set of piles starting from a given initial configuration, by moving one block at a time, either to the table or atop another pile of blocks, and only if it is not under another block.

Consider a simple version of this problem, in which there are five blocks with identical shape and size, numbered from 1 to 5. The figure below shows two possible configurations of such blocks; note that the relative position of the piles does not matter, thus the configuration on the left is equivalent to the one in the middle. Every block can be either on the table (there are no constraints on the number of piles) or atop any another block. The goal is to stack the blocks in a single pile, in the order shown in the rightmost configuration in the figure, starting from any given set of piles, by moving the smallest possible number of blocks. Only blocks at the top of the current piles can be moved, and can be placed only on the table or atop another pile.

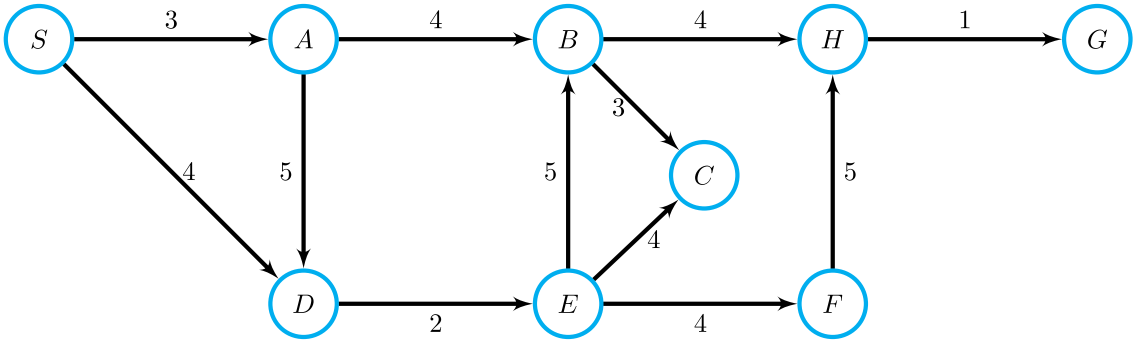


A State this problem as a search problem formulation (make sure to clearly list all the components of the formulation as we discussed in the class).

B Demonstrate, step-by-step, how the search tree can be built by a breadth-first search strategy.

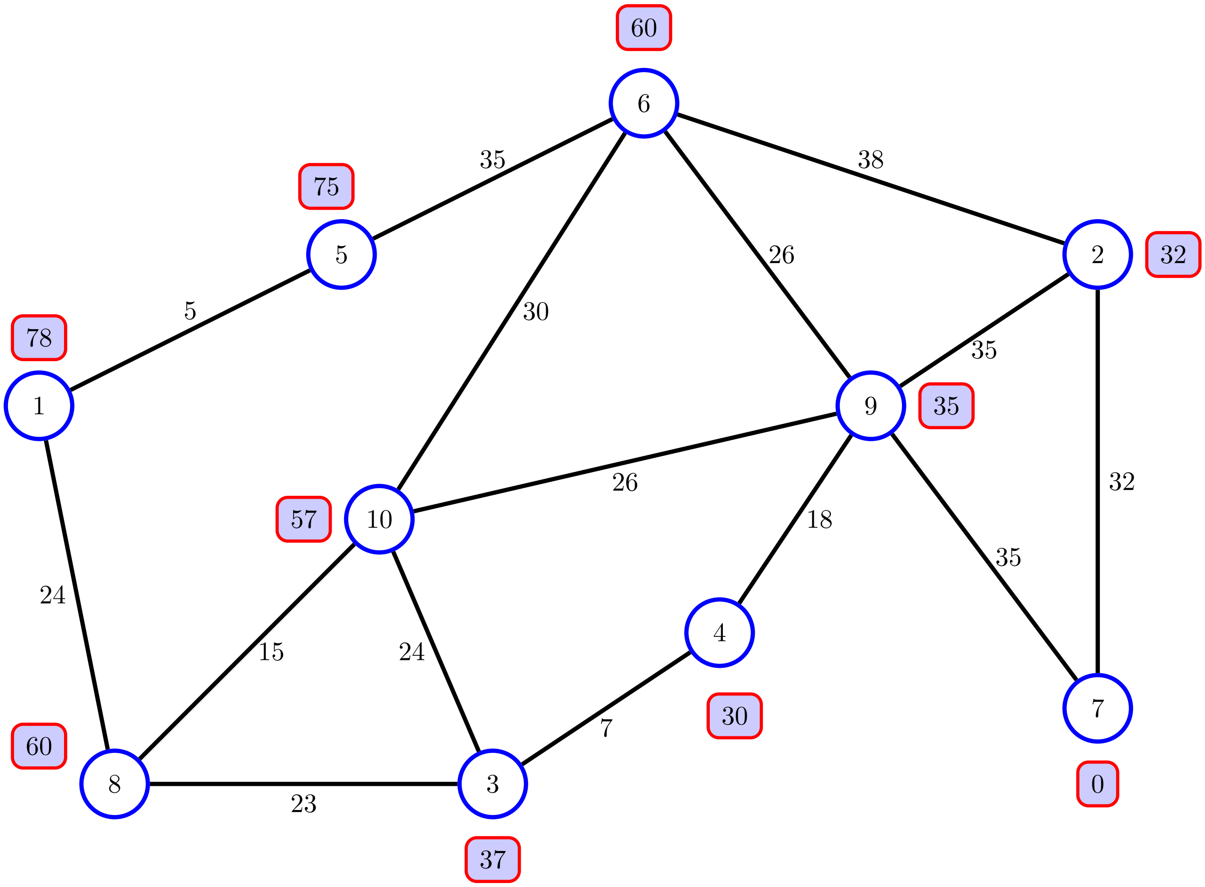
C Demonstrate, step-by-step, how the search tree can be built by a depth-first search strategy.

Problem 4. (**4 points**) **Uninformed Search:** consider the state space graph below in which **S** represents the initial state and **G** represents the goal state. The edges in this graph represent possible actions one can take to traverse the graph towards the goal state. The label on each edge signify the cost of the action represented by the edge.



Assuming alphabetical order is used to break ties during the generation of children, list the sequence of nodes visited (i.e., illustrate the contents of the open and closed queues) and the final paths returned using BFS, DFS and UCS.

Problem 5. (**4 points**) **Informed Search:** Consider the city map below. The goal is to determine the shortest distance from city 1 to city 7.



Edges (which represent valid paths of travel between cities) are shown between cities and are labeled with the actual travel distance. Numbers shown in boxes beside each city represent an estimate of the distance from that city to city 7. Apply

- (i) UCS
- (ii) Greedy Best First Search, and
- (iii) A* Search

to determine a path from city 1 to city 7. Show the contents of the open and closed queues during each step of each algorithm. Break any ties according to the numerical vlaue of the city (smaller cities take precedence). *Ignore repeated states..*

Problem 6. (**7 points**) Assume a 25 × 25 two-dimensional maze. Each square in the maze has an (x,y) coordinate, with the bottom-left corner being (0,0) and the top-right corner being (24,24). Each position in the maze can be either empty or blocked. In addition, there are two "special" positions, the starting position and the exit position, both of which are selected randomly.

The agent can only move up, down, left or right, but never diagonally. It also cannot enter blocked positions or move outside the maze. Its objective is to find a path from its starting position to the exit position, preferably the cheapest one. The cost of a path is the number of positions the agent has to move through, including the starting and exit position. An example of a maze is given in Figure 1.

• **Requirements**
Implement code to find a path from the starting position to the exit position using

1. Breath-First Search,
2. Depth-First Search, and
3. A* Search.

For the A* Search, you must define an appropriate heuristic function, and justify your choice. Your implementation should output information about the search, including the complete path, its cost, and the number of nodes explored (or squares checked) before finding it.
The above three search heuristics represent the minimum requirements for this question.

• **Deliverables**
You are expected to turn in a short paper (the shorter the better) which should include the following:

1. A short description of your implementations of the search methods. In particular, for the A* Search, explain and justify your chosen heuristic function.
2. Sample output of each search method you implemented on the maze of 1.
3. You will have to test each search technique three times:
   (a) With the agent starting at S and ending at E1
   (b) With the agent starting at S and ending at E2
   (c) With the agent starting at (0,0) and ending (24,24)

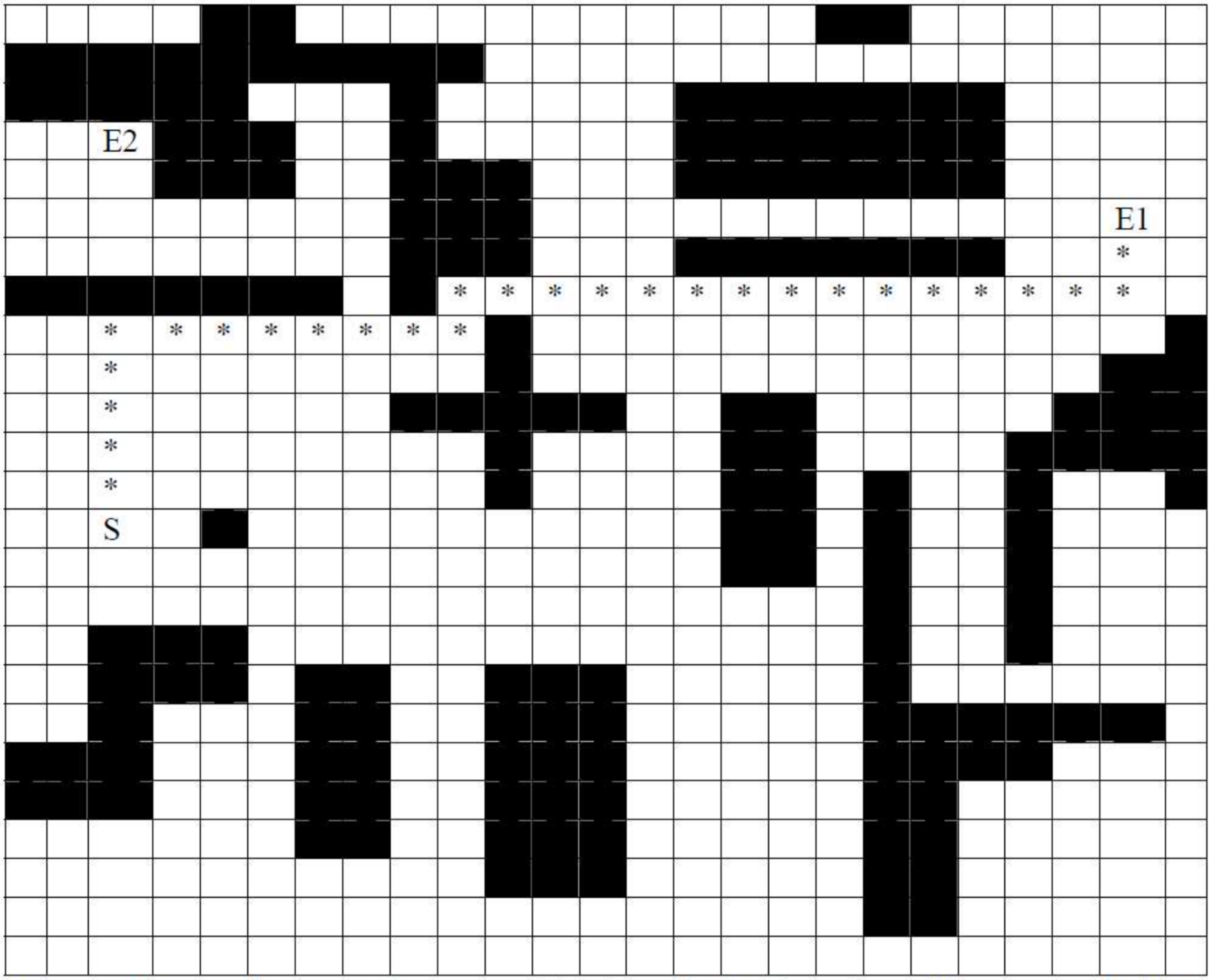


Figure 1: Example maze.The start position is marked S, the exit positions are marked E1 and E2, empty positions are blank and blocked positions are black. The path, marked by stars, has a cost of 30 units