University of Waterloo

Department of Electrical and Computer Engineering

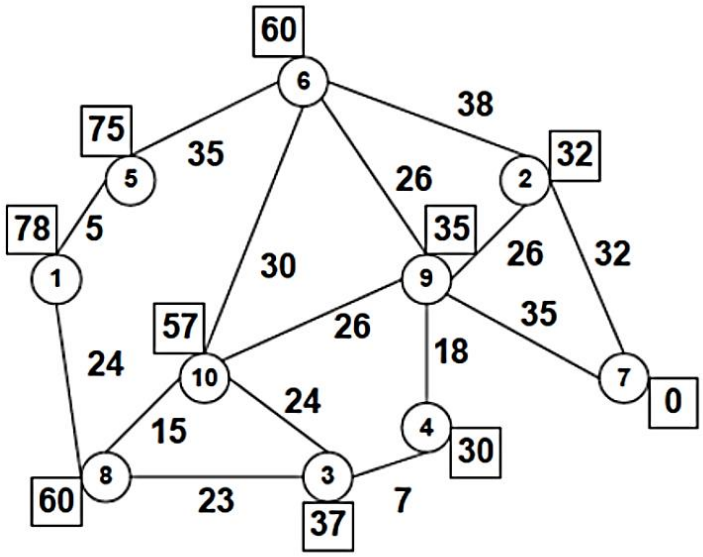
ECE 457A Cooperative and Adaptive Algorithms

Assignment 2: Due Date – Oct 18th, 2021

- **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 be checked for originality**
- **Communicate any issues or concerns with the TAs**

Question1 (3 Points)

(Informed search) Consider the city map below. The objective 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) Uniform Cost Search, (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 the numerical value of the city (smaller cities take precedence).

Ignore repeated states

Question2 (7 Points) (requires coding)

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.
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. The map of the maze is given in the figure below.

Requirements

- Implement a 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.

Deliverables

Your answer should include the following:

1. Upload the code separately as a zip file and name it "YOUR NAME_Maze source code".
2. In the PDF that you submit for this assignment, for this question provide:
   a. A short description of your implementations of the search methods. In particular, for the A* Search, explain and justify your chosen heuristic function.
   b. Sample output of each search method you implemented on the maze of Figure 1.
   c. You will have to test each search technique three times:
      c.1. With the agent starting at S and ending at E1
      c.2 With the agent starting at S and ending at E2
      c.3 With the agent starting at (0,0) and ending (24,24)

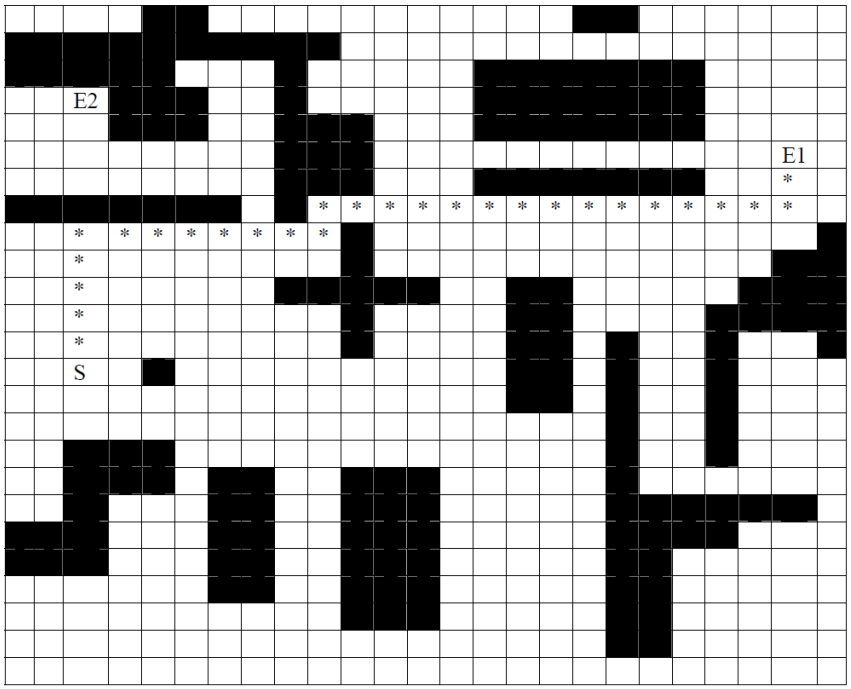


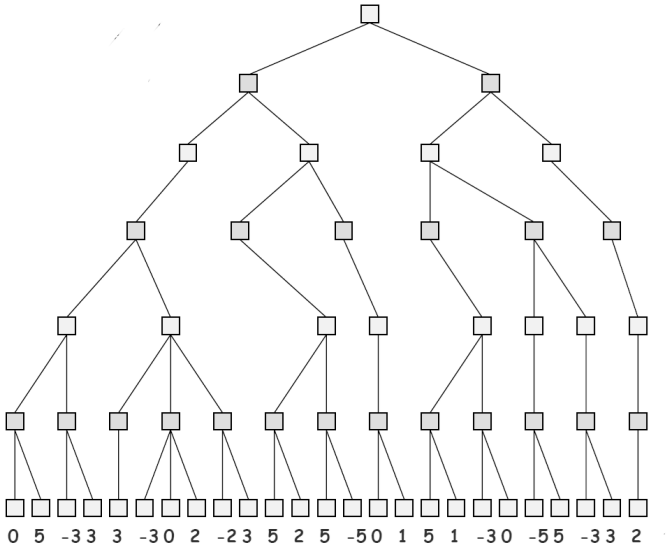
Figure 1: Example maze. The start position is marked S, the exit positions are marked E1 and E2, empty positions are blank and blocked positions are black. The path, marked by stars, has a cost of 30.

Question3 (3 Points)

a. Consider the following game tree in which the evaluation function values are shown below each leaf node. Assume that the root node corresponds to the maximizing player. Assume that the search always visits children left-to-right. Compute the backed-up values computed by the Minimax algorithm. Show your answer by writing values at the appropriate nodes in the tree.

b. Draw the game tree again and show the nodes that would be pruned by the alpha-beta pruning algorithm. Show all the calculations needed to find those nodes.

c. Show at least one change in the tree that can increase the number of pruned branches. The change should not alter the total number of states but can restructure the tree or the order of visiting nodes.



Question4 (7 Points) (requires coding)

We want to minimize the Easom function of two variables:

$$\min_{x} f(x) = -\cos x_1 \cos x_2 \exp\left(-(x_1-\pi)^2-(x_2-\pi)^2\right), \quad x \in [-100, 100]^2$$

The Easom function is plotted in Figure 2 for x∈ [−10,10]. The global minimum value is −1 at $x = (\pi, \pi)^T$. This problem is hard since it has wide search space and the function rapidly decays to values very close to zero, and the function has numerous local minima with function value close to zero. This function is similar to a needle-in-a-hay function. The global optimum is restricted in a very small region.

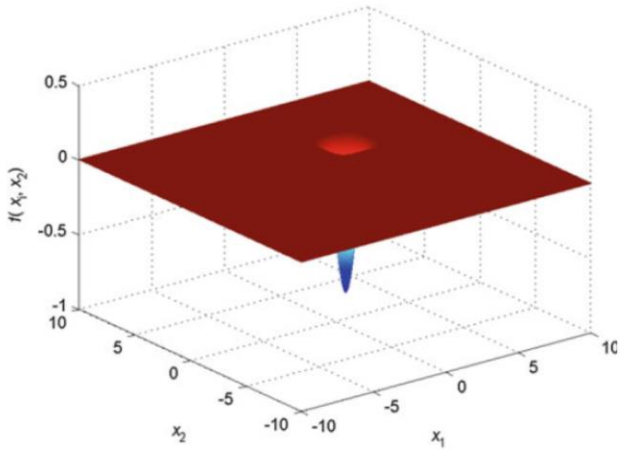


Figure 1: Easom function for x∈ [−10,10]

a. Find a proper problem formulation, neighborhood function and cost function for this problem to apply SA algorithm to solve it.
b. **Report your observations** on SA performance for solving this problem by:
    i. Selecting 10 different initial points randomly in [-100,100] → use the random function, do not generate yourself!
    ii. Selecting 10 different initial temperatures in a reasonable range → provide the range, justify your choice, and use the random function (choose the best initial point that you found in i)
    iii. Selecting 9 different annealing schedules → 3 linear, 3 exponential, and 3 slow using different alpha values (choose the best initial temperature from ii and best initial point from i)
c. What was the best solution you could find? What was the setting of SA to achieve that solution? Why was that setting performing better than other settings?

* You may use SA solver from MATLAB Global Optimization toolbox (or something similar in python or other languages)

** Upload the code separately as a zip file and name it “YOUR NAME_Easom source code”.