Homework #1: Search Problem

MESIIN476024 - AI algorithms

Due: Sunday, October 20, 2024, by 11:59 p.m.

Instructions

Read all the instructions below carefully before starting the assignment and making your submission.

- The homework must be completed individually. Don't forget to write your name and your lab group.
- **Submission:** Your submission should include a **zip file** containing a PDF that summarizes your written answers to the exercises and a notebook file (.ipynb) containing the Python code for Exercise 3. We highly recommend typing your homework, although handwritten and scanned versions are also acceptable. The zip file should follow this naming format: HW1_YourLabGroup_YourName (*e.g.*, HW1_DIA3_PierreFEUILLE).
- Keep your answers concise but detailed enough to demonstrate your understanding of the topic.
- **The assignment is due on October 20, 2024.** Any submission made after this date will incur a penalty of -2 points per day late.
- You must upload your assignments through the corresponding submission space in DeVinci Learning by the due date and time. (Homework → Homework#1 →Deposit homework 1 → DIA_).
- All external sources of material must be cited. The assignments are your individual responsibility, and plagiarism will not be tolerated. Software will be used to check for similarities in writing and code!

Topics: Points

- **Exercise 1**: 4 Points
- **Exercise 2**: 7 Points
- **Exercise 3**: 9 Points

Exercice 1: Reading [4 Points]

As discussed in class, in 1950, Turing published a seminal paper called "Computing Machinery and Intelligence".

Read Turing's original paper: TuringArticle.pdf.

In the paper, Turing discusses several potential objections to his test for intelligence. Which of these objections still hold some significance today? Are his refutations valid? Can you think of new objections that have arisen from developments since the paper was written? Additionally, Turing predicts that by the year 2000, a computer would have a 30% chance of passing a five-minute Turing Test with an unskilled interrogator. What do you think a computer's chances would be today? Provide justification for your answers. (Approx. 1/2 page).

Exercise 2: Search Problem [7 Points]

You are given a challenge called the "Sliding Magic Square" puzzle. It is played on a 3×3 grid containing the digits from 1 to 9. The number 9 is the "sliding number," which can be moved in four possible directions: up, down, left, or right. When moved, the 9 swaps places with the number in the chosen direction.

The puzzle starts in the initial configuration shown in Figure 1. Your goal is to move the number 9 and rearrange the grid so that the sum of the numbers in each row, each column, and both diagonals **equals 15**. There are multiple possible solutions that satisfy this condition. You may stop once you find any configuration that meets this requirement.

(*Hint:* You may want to organize the moves in the following order: up, down, left, and right to minimize the number of steps needed to solve the puzzle.)

6	9	8
7	1	3
2	5	4

Figure 1: Initial configuration

A) Formulate this puzzle as a search problem. What are the states, actions, initial state, and goal condition?

B) Determine whether the state space is represented as a graph or a tree.

C) How large is the state space?

D) What is the maximum branching factor for this problem? Provide justification.

E) Draw a portion of the search graph resulting from Breadth-First Search (BFS) algorithm. Label the nodes in the order in which they are expanded.

F) Draw a portion of the graph search generated by the Depth-First Search (DFS) algorithm and label the states in the order they are expanded.

G) Draw a portion of the graph search generated by the Iterative Deepening Search (IDS) algorithm and label the order in which each state is expanded.

H) What are the advantages and disadvantages of Breadth-First Search and Iterative Deepening Search for this problem? Would Depth-First Search with no limit be a better or worse approach? Why?

I) Propose a non-trivial admissible heuristic for solving this problem.

Exercise 3: Programming the "8-Puzzle Problem" [9 Points]

The 8-puzzle is a simplified version of the well-known 15-puzzle. It consists of a 3x3 grid where 8 numbered tiles (from 1 to 8) are placed, leaving one square empty. The empty square allows adjacent tiles to slide into it, one at a time. The goal of the puzzle is to rearrange the tiles from a given initial configuration to a predefined target configuration by sliding the tiles across the grid.

In this exercise, we will work with the board positions shown in the figure below. The left side illustrates an initial configuration, while the right side shows the target configuration that we aim to achieve.

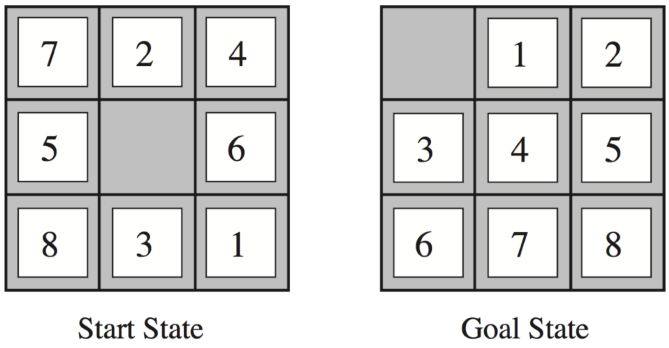

Figure 2: Configurations of the 8-Puzzle: initial state (left) and goal state (right)

Your task is to solve the 8-puzzle problem by implementing various search algorithms, including Uniform Cost Search and A* Search with different heuristics. Afterward, you will compare and analyze the performance of these algorithms based on their effectiveness.

Task Breakdown:

A) **Problem Formulation:** Implement the 8-puzzle as a class `EightPuzzle`. Define the problem's states, possible actions, initial state, and goal state within the class.

B) **Uniform Cost Search:** Implement the Uniform Cost Search algorithm to solve the puzzle efficiently. Your implementation should output both the order in which nodes are expanded and the resulting path to the solution.

C) **Heuristics for A* Search:** Implement two heuristics for A* Search:

- *Misplaced Tiles* heuristic (h_1): Counts the number of tiles that are not in their goal position.
- *Manhattan Distance* heuristic (h_2): Measures the total distance each tile needs to move to reach its goal position.

D) **Best-First Search:** Implement the Best-First Search algorithm using both the h_1 and h_2 heuristics, and compare their performance.

E) **Admissible Heuristic:** Define a third heuristic h_3:

$$h_3 = \text{number of tiles not in the correct row} + \text{number of tiles not in the correct column}.$$

Implement Best-First Search with this heuristic. Is this heuristic admissible? Why or why not? Is it better than the heuristics above?

F) **A* Search:** Implement A* Search using heuristics h_1 and h_2. Compare their performances.

G) **Comparison and Analysis:** Compare Uniform Cost Search and A* Search in terms of performance and solution optimality.

Submission Requirements:: Your solution must be implemented in a notebook file. This file should contain: your `EightPuzzle` class and all search algorithms, clear documentation for each function, explaining its purpose and how it fits into the solution, and a detailed justification of the design choices you made.

.