Assignment 3

Subject

Topics of this session :

1. Informed Search.
2. Formalise a problem as a graph search problem.
3. Define relevant heuristics to guide search.

This assignement is graded and must be submitted (individually) on Moodle before next week's class.

For each exercise, detail your reflexion steps :
- We are mostly interested in your actual thinking process.
- Even if you are unable to solve an exercise, write out what were you reflexion steps.
- For each attempted exercise, a written feedback will be provided.

Reference material : Artificial Intelligence, A Modern Approach, Chapter 3.5.

For coding :
- Noto (Online Jupyter NoteBook).
- Any other python coding environment you prefer using.

Exercise 1

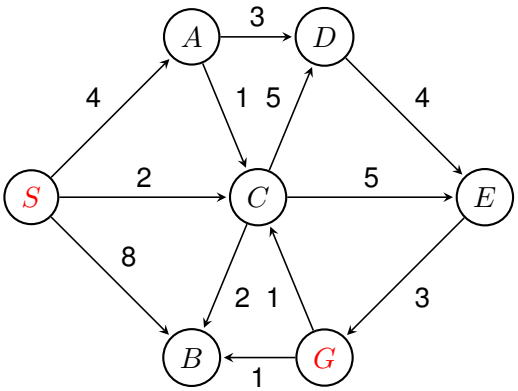

FIGURE 1 – Another weighted directed graph.

We want to determine a path between the starting node **S** and the goal node **G**, using the graph shown in Figure 1. Each edge has an weight representing its cost.

1. For each of the following search algorithms, list :
   - The order in which nodes are visited.

— The final path found from **S** to **G**.
— The total cost of this path.
Apply the following algorithms :

(a) **Depth First Search (DFS)**

(b) **Breadth First Search (BFS)**

(c) **Uniform Cost Search (UCS)**

(d) **A*** with the heuristic h.
- $h(S) = 5$
- $h(A) = 5$
- $h(B) = 25$
- $h(C) = 4$
- $h(D) = 5$
- $h(E) = 2$
- $h(G) = 0$

Important : If there is a need to break ties between nodes, always expand them in **alphabetical order**. For example, if the algorithm reaches nodes **A**, **B**, and **C**, it should expand **A** first.

2. Analyse the heursitic function :
   — Is h admissible ?
   — Is h consistent ?
3. Run Dijkstra's algorithm to find a shortest path from all nodes to G. Use a table such as the one below and provide the order in which nodes were visited (we start with G) :

TABLE 1 – Initial State of Dijkstra's Algorithm

Node	Distance	Parent
A	∞	None
B	∞	None
C	∞	None
D	∞	None
E	∞	None
G	0	-

4. What are the main differences between UCS, Dijkstra's and A* ?

Exercise 2

Using last week implementations, implement A* as a python function accepting any heuristic function.

We will define a heuristic function as a python dictionary, such as :

```
h_1 = {
    "S": 5,
    "A": 4,
    ...
    "G": 0,
    }
```

Project – Part 1

This whole section must be done in groups of 2-3 people.

Starting this week, we will begin working on projects. When submitting on Moodle, please make sure to **include the names of all group members**.

Project assignments consist of two parts :

1. **Guided Project :** A mandatory project for all groups. Serves as an example.
2. **Personal Project :** A problem of your own choice.

1 Guided Project

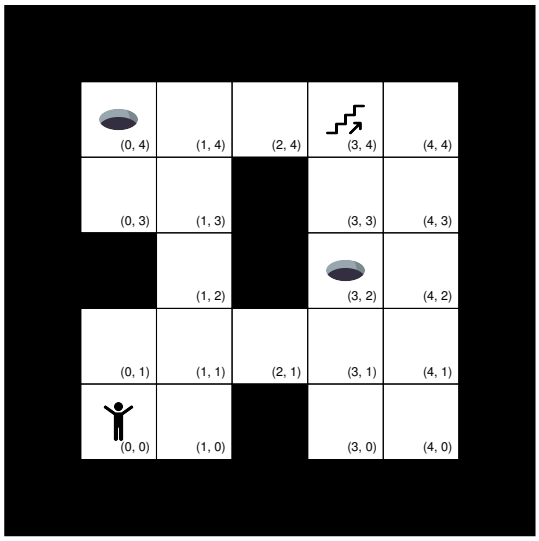

FIGURE 2 – A simple dungeon room with holes. The agent starts at position (0, 0).

Dungeon Gridworld In this environment (see Figure 1), the agent must find the stairs to exit the dungeon without falling into the holes.

Assumptions :

— The agent **has full visibility** of the grid (observation space = state space).
— The goal is to reach the stairs **as quickly as possible**.
— Falling into a hole causes the game to end immediately.

Tasks :

1. **Formalise the problem :** Define actions, states, costs/rewards, start, and goal states.
2. **Construct a graph** representing the problem.
3. **Implement in Python.** Example for a 2x2 grid :

```
grid_graph = {
    (0, 0): [((0, 1), 3), ((1, 0), 2)],
    (1, 0): [((0, 0), 0), ((1, 1), 1)],
    (0, 1): [((0, 0), 2), ((1, 1), 3)],
    (1, 1): [((0, 1), 4), ((1, 0), 2)]
}
```

4. **Compare** an algorithm of your choice (e.g., DFS, BFS) with A* using a relevant, **consistent** and **admissible** heuristic. For example, you can compare in computation time, number of nodes visited, optimality of the solution found...

2 Personal Project

1. **Choose a new problem :** Your problem should be general enough to apply similar methods as in the guided project.

   Note : You will have to keep this problem across assignments.

2. **Repeat the steps 1–4** from the guided project, applying them to your new problem.