**Benchmark – Project 1**

Stuart Spackman

Grand Canyon University

CST-580: Artificial Intelligence

Dr. Jeremy Pretty

July 2, 2025

**It is important that computer scientists learn to identify effective solutions while using AI search techniques to replace human decision making.**

**Choose an aspect of a game or simulation in which *search* is an essential component. Adapt the concept of *hidden treasure*.**

**1. A brief description of the game or simulation (one paragraph).**

The title of the game/simulation I will be developing a search algorithm for could be called *Treasure Hunt in a Maze*. This will be a grid-based simulation in which an AI-controlled adventurer uses search techniques to find a hidden treasure. The environment will be a 2D maze composed of open cells and impassable walls. The adventurer will start from a known position and must explore the maze to discover the shortest or most efficient path to the treasure without any human intervention. The AI present in the code will intelligently explore the maze using search algorithms to find the most efficient path to the treasure while avoiding dead ends. In the end, this simulation will demonstrate how AI search techniques can be used to mimic or optimize human decision-making in exploration scenarios.

**2. A brief description of the search methods that will be used (one paragraph and bullet points outline).**

This project will make use of uninformed and informed search algorithms to allow the adventurer to locate a hidden treasure within a 2D maze. In this context uninformed (i.e., blind) searches are those that explore the space systematically but don’t use any types of shortcuts to reach the treasure. Meanwhile, informed (i.e., heuristic) searches are those that make use of “rules of thumb” to find solutions more quickly (Subramaniam, 2025). This will be happening in a grid-based environment where each cell will represent a wall or a passable space. The searches used are given in the following list:

* **Breadth-First Search (BFS)**
  + Explores all nodes at a given depth before going deeper.
  + Guarantees the shortest path in terms of steps (provided all step costs are equal).
  + Uses a queue (FIFO) to manage the adventurer’s steps.
  + Can be memory intensive.
* **Depth-First Search (DFS)**
  + Explores as deep as possible along each branch before backtracking.
  + Is not guaranteed to find the shortest path.
  + Uses a stack (LIFO) to manage the adventurer’s steps.
  + Expected to be fast in small/shallow mazes but might get stuck in deep paths or loops in larger mazes.
* **A\* Search**
  + Combines cost-so-far and estimated cost-to-go for decision-making.
  + Uses a priority queue to select the most promising of the nearest nodes.
  + Uses the Manhattan distance as a heuristic (Omni Calculator, 2024).

**A diagram of a graph

AI-generated content may be incorrect.**

**3. A flowchart detailing the logic of the algorithm(s) implemented.**

A flowchart accounting for all three algorithms is shown on the proceeding page. Note that in this context a frontier is any data structure that keeps track of nodes the algorithm has discovered but not yet explored (i.e., the algorithm’s to-do list).

A screenshot of a computer program

AI-generated content may be incorrect.

**4. The entire code implemented for this project. If this is not part of a bigger project, state this explicitly. If it is part of a bigger project and the entire parent project is needed to run it, include the parent project. The project may be too large to upload to the digital platform. If this is the case, upload it to your selected repository only, and share the link with the instructor. Note: You will need to submit the code to your selected repository anyway (to include all source files with well commented code).**

**5. A screencast demonstrating execution of the code. If the screencast file is too large to upload to the digital platform, upload it to only your selected repository (video file).**

**6. A list of references you used in your implementation, including code libraries and code snippets you adapted and incorporated. Acknowledge that you are familiar with copyright laws and regulations and that all external code use is permissible (to include a list of sources with links to their usage agreements).**

I have included all references at the end of this document. I am familiar with copyright laws and regulations and that all external code use is permissible.

**7. A bulleted list of the platform and software tools used.**

* **Python 3.x:** The core programming language used for implementing the search algorithms, maze logic, and visualization.
* **Matplotlib:** Plotting library used to visualize the maze and the algorithm’s progress dynamically.
* **Standard Library Modules:** time, collections, and heapq.
* **Custom Modules:** 
  + mazy.py – contains functions for generating the maze along with a Node class.
  + search.py – implements the three search algorithms.
  + utils.py – contains a helper function for calculating the Manhattan distance heuristic for A\*.
* **Platform:** Developed and tested on macOS.
* **IDE:** Visual Studio Code.

**References**

Russell, S., and Norvig, P. (2020). *Artificial intelligence: A modern Approach* (4th ed.). Prentice Hall. ISBN-13: 9780134610993.

Subramaniam, A. (2025). *AI Search Algorithms: Uninformed Vs Informed Search Explained with Real World Examples*. Medium. <https://medium.com/ai-simplified-in-plain-english/ai-search-algorithms-uninformed-vs-informed-search-explained-with-real-world-examples-1e73ea9d5905>

*Manhattan Distance Calculator.* (2024). Omni Calculator. <https://www.omnicalculator.com/math/manhattan-distance>

Gold, M. (2023). *Python’s Path Through Mazes: A Journey of Creation and Solution*. Medium. <https://medium.com/@msgold/using-python-to-create-and-solve-mazes-672285723c96>

Nojek, L. (2025). *Recursive Backtracker maze – more features and animation with Matplotlib and Python*. Lukasz Nojek – Blog. <https://lukasznojek.com/blog/2019/06/recursive-backtracker-maze-more-features-and-animation-with-matplotlib-and-python/>