Project 1. Search

Let's chase Pac-Man!

1 Introduction

Pac-Man is a classic arcade game where the player controls Pac-Man character to navigate a maze, collecting dots while avoiding ghosts. The goal of the game is to survive as long as possible and maximizing the number of collected food points. The ghosts are programmed to chase Pac-Man, creating a challenging and engaging experience.

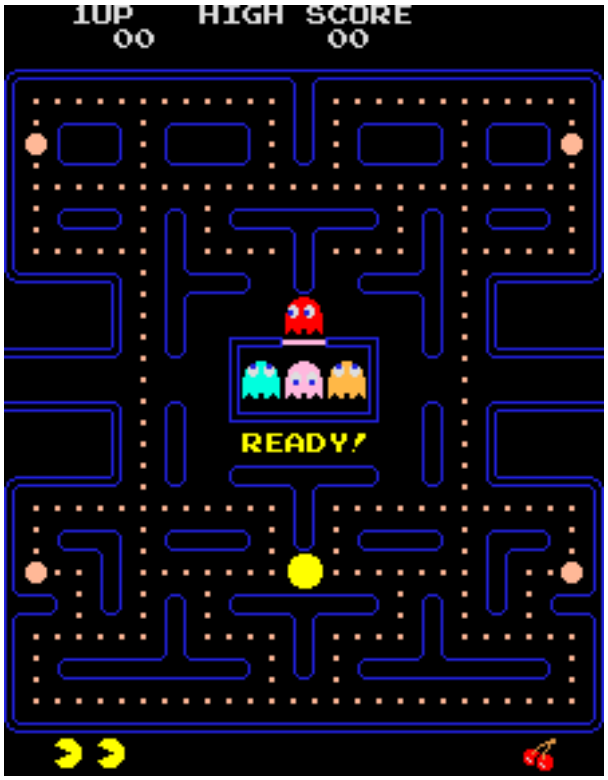

Figure 1: In-game screenshot: Pac-Man below the ghosts in the center. (Source: Wikipedia).

In this project, students will program the ghosts to chase Pac-Man using search algorithms. The objective is to implement different path finding strategies and then analyze their effectiveness. Each ghost is programmed with a search algorithm, including: (1) **BFS** (Breadth-First Search), (2) **DFS** (Depth-First Search), (3) **UCS** (Uniform-Cost Search), and (4) **A*** (A-Star Search); then evaluated based on performance metrics such as (1) **search time**, (2) **memory usage**, and (3) **number of expanded nodes**.

2 Implementation (60 pts)

Each level of implementation introduces new challenges and complexity of the search algorithms. Here are some important things to consider:

- The source code must be written in **Python**.
- You can use supporting libraries, but you must implement the search algorithms **yourself**.
- There is **no restriction** on how to organize your code. However, it should be structured professionally and clearly.

2.1 Level 1 (10 pts): Blue Ghost using BFS

Assumption: To simplify the problem, we **keep Pac-Man's position fixed** and **unchanging**.
Coding requirements: Implement the **Blue Ghost** using **Breadth-First Search** (BFS) algorithm to chase Pac-Man.
Report requirements: Include the following in the **report**:

- Detailed explanation of the algorithm. Illustrative images and diagram are encouraged.
- Test the implemented Ghost and Pac-Man in different starting positions (at least 5 test cases). Record **search time**, **memory usage**, and **number of expanded nodes**.

Level 2, 3, and 4 follow the above assumption and report requirements.

2.2 Level 2 (10 pts): Pink Ghost using DFS

Implement the **Pink Ghost** using the **Depth-First Search** (DFS) algorithm to chase Pac-Man. The Iterative Deepening Search (IDS) algorithm can be used as an alternative if you want ;-).

2.3 Level 3 (10 pts): Orange Ghost using UCS

Implement the **Orange Ghost** using the **Uniform-Cost Search** algorithm to chase Pac-Man. In the report, also explain **how you calculate the cost**.

2.4 Level 4 (10 pts): Red Ghost using A*

Implement the **Red Ghost** using the **A* Search** (A*) algorithm to chase Pac-Man. In the report, also explain **how you calculate the cost** and which **heuristic function** you use.

2.5 Level 5 (10 pts): Parallel Execution

Implement all ghosts (Blue, Pink, Orange, and Red) **moving simultaneously** in the same maze, each ghost follows its respective search algorithm to chase Pac-Man and executes independently. Ensure that no two ghosts occupy the same position at the same time.

2.6 Level 6 (10 pts): User-Controlled Pac-Man

Enable interactive game-play by **allowing the player to control Pac-Man's movement** while the ghosts actively chase him. Ensure real-time updates by making each ghost recalculate its path continuously based on Pac-Man's changing position.

3 Report (30 pts)

Submit a well-formatted PDF report with:

- **Project Planning and Task Distribution (5 pts)**: Document team responsibilities, which includes information on each task assigned to team members and the completion rate. E.g., Student A has percentage of completion 90% and the group work has total score of 9.0, then A receives a score of 9.0 * 90% = 8.1.
- **Algorithm Description (10 pts)**: Provide a detailed explanation of each search algorithm.
- **Experiments (15 pts)**: Assess search performance via Search Time, Memory Usage, and Expanded Nodes; with measurements, visualizations and insights.
- **References**: if any.

4 Video (10 pts)

Upload a demo video to YouTube or Google Drive and provide the **public link** in your report. The video content includes:

- **Graphical Interface (5 pts)**: Demonstrate the GUI, highlighting key visual elements.
- **Feature Presentation (5 pts)**: Demonstrate your implemented functionalities concisely. If a higher level is implemented, lower levels do not necessarily need to be shown separately. The video should have subtitles or narration to make it easy to follow.

5 Submission

Please adhere to the following submission guidelines:

- Your source code and report must be contributed in the form of a compressed file (.zip, .rar) and named according to the format `StudentID1_StudentID2_...`
- If the compressed file is larger than 25MB, upload it to Google Drive and share it via a link. Absolutely no modifications are allowed after the deadline.

Example details of the directory organization:

```
StudentID1_StudentID2_...
├── Source
│   ├── main.py
│   ├── README.txt (how to run source code)
│   ├── requirements.txt (libraries to be installed)
│   └── ...
└── Report.pdf (included demo video URLs)
```

6 Notices

Please pay attention to the following notices:

- This is a **GROUP** assignment. Each group has 3 - 4 members.

  Groups with fewer or more members than the specified limit require lab instructor approval.
- Duration: about 3 weeks.
- AI tools are **not restricted**; however, students should use them wisely. Lab instructors have the right to conduct additional oral interviews with random groups to assess their knowledge of the project.

- Any plagiarism, any tricks, or any lie will have a 0 point for the course grade.

The end.