**Artificial Intelligence**

**Project 1**

| Topic |

**Searching**

by

**PHẠM VŨ DUY - 18127092**

**MAI ĐĂNG KHÁNH – 18127118**

**NGUYỄN PHÚC THỊNH - 18127223**

**HUỲNH NHẬT NAM - 18127014**

1. **Overview:**
2. **Environment:**

* Python 3.8, with graphical library:

+ *pygame* library

* Divide the problem into small object:

+ Ghost

+ Pacman

* Using Object-Oriented-Programing

1. **The degree of completion:**

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Specifications** | **Scores** | **Degree of completion** |
| **1** | Finish level 1 successfully. | **15%** | **100%** |
| **2** | Finish level 2 successfully. | **15%** | **100%** |
| **3** | Finish level 3 successfully. | **10%** | **100%** |
| **4** | Finish level 4 successfully. | **10%** | **100%** |
| **5** | Graphical demonstration of each step of the running process. You can demo in console screen or use any other graphical library. | **10%** | **100%** |
| **6** | Generate at least 5 maps with difference in number and structure of walls, monsters, and food | **10%** | **100%** |
| **7** | Report your algorithm, experiment with some reflection or comments | **30%** | **100%** |
| **Total** | | | **100%** |

1. **Assignment plant:**

|  |  |  |
| --- | --- | --- |
| **Student** | **Job** | **Scores** |
| Mai Đăng Khánh  18127118 | Graphical demonstration | 10% |
| Pacman in level 4 | 5% |
| Demo and testing | 15% |
| **Total** | **30%** |
| Huỳnh Nhật Nam  18127014 | Level 3 | 10% |
| Writing the report | 15% |
| **Total** | **25%** |
| Nguyễn Phúc Thịnh  18127223 | Ghost in level 4 | 5% |
| Create maps | 10% |
| Fix bug in level 2 | 5% |
| **Total** | **20%** |
| Phạm Vũ Duy  18127092 | Level 1 | 15% |
| Level 2 | 10% |
| **Total** | **25%** |

1. **Algorithm description:**
2. Level 1 and 2:

* Use *Breadth First Search* to find food:

+ *BFS* always find solution to a problem if it exist

+ BFS is very useful especially when finding the shortest and most optimal path to one edge

+ In level 1 and 2, the map is not close. With *BFS*, pacman can take advantage on this characteristic of the map and reduce the path to the minimum number of move

+ In level 1 and 2, pacman know the food location, so *A\** can also find the shortest and the most optimal path to food. But, without a admissible heuristic which can support not closed map, *A\** can’t do the search better than *BFS*

1. Level 3:

- Using local search because the pacman can only see his 3 nearest steps

- Because the goal is eating all the foods, so for every steps, pacman chooses the direction with highest heuristic (not exactly a hill climbing search. Pacman just chooses better states if possible)

- If there are no ghosts or coins in pacman’s sight, pacman will randomly move forward or to 2 sides, but he will not move backward since it’s useless. Go backward then forward or forward then backward will make the total point decreases by 2 but pacman is still in the same position that there are no coins around, so the heuristic for going backward is -2. If pacman meets dead end, directions which are walls will have heuristics -100, so he can move back to get out of the dead end.

- Again our ultimate goal is to eat all coins so pacman will move around the map until it sees coin(s).

- Heuristic will be calculated with BFS to search for pacman’s possible nearest 3 steps:

+ Coin closer to pacman will have higher heuristic.

+ Only ghosts within 2 steps will be considered threats: h = -1000 (because obviously ghost outside pacman’s next 2 steps can’t harm him in next move)

+ if pacman neighbour is path: h = -1 or wall: h = -100 (when using BFS to calculate heuristic, walls will have h = 0, which means they don’t affect the calculation. Wall heuristic only = -100 when it’s pacman’s neighbour)

+ There is a case when pacman can go backward. There are coins in backward direction which are in pacman’s sight, but the last time pacman move from position, there is no ghost that can harm pacman. Because of that he try to contest for that coins. Else pacman will not move backward unless he has to.

- Stop when there is no coin left or pacman is dead.

1. Level 4:
2. Pacman:

* At every step, the agent uses Breadth-First Search algorithm to find the paths to the foods, and random choice to avoid ghosts
* With so many foods, it is irrational to find path to all the foods and choose the shortest path. And breadth-first deepening search algorithm can accomplish both tasks in one run. BFS algorithm can find the nearest food to pacman from a set of foods and the optimal path to that nearest food with only one iteration
* Pacman will try and maintain a 2-cells distance to any and all ghosts. Each ghost position and there adjacent cells are marked dangerous. Pacman will treat those cells same as walls. Therefore, the path finding algorithms will not find the path that pass or come too close to ghosts
* If pacman happens to be in a dangerous cell, check if the path to the nearest food pass through any dangerous cell:

+ If no, then proceed along the found path

+ If yes, move to an adjacent cell that is not a dangerous cell

1. Ghost

* At every step, the agent use A\* algorithm with Manhattan distance heuristic to find the optimal path to the Pacman
* The A\* algorithm is both complete and optimal. Thus, the path they found are the shortest path to Pacman. The runtime of this algorithm generally lower than most other search algorithms.