**HO CHI MINH UNIVERSITY OF SCIENCE**

**FACULTY OF INFORMATION TECHNOLOGY**



**REPORT: Project 01:**

**Hide and Seek**

By:

22127057 – Đỗ Phan Tuấn Đạt

22127064 – Phạm Thành Đạt

22127123 – Lê Hồ Phi Hoàng

22127131 – Trần Nguyễn Minh Hoàng

TABLE OF CONTENTS

[A. INTRODUCTION 3](#_Toc163322424)

[B. WORKING ENVIRONMENT 4](#_Toc163322425)

[I. Programing language 4](#_Toc163322426)

[II. Code editor 4](#_Toc163322427)

[III. Code management 4](#_Toc163322428)

[C. GENERAL CODE ANALYSIS 5](#_Toc163322429)

[I. Seeker Agent 5](#_Toc163322430)

[1. Code representation 5](#_Toc163322431)

[2. Seeker’s vision 5](#_Toc163322432)

[3. Path-finding using search algorithms 6](#_Toc163322433)

[II. Graphical demonstration 7](#_Toc163322434)

[1. Main menu 7](#_Toc163322435)

[2. Demonstrating the running process 7](#_Toc163322436)

[3. Score display 7](#_Toc163322437)

[D. LEVEL-SPECIFIC ANALYSIS 8](#_Toc163322438)

[I. Level 1 8](#_Toc163322439)

[II. Level 2 8](#_Toc163322440)

[E. EVALUATION & COMMENTS 8](#_Toc163322441)

[I. Level 1 8](#_Toc163322442)

[1. Map 1 8](#_Toc163322443)

[2. Map 2 9](#_Toc163322444)

[3. Map 3 9](#_Toc163322445)

[4. Map 4 10](#_Toc163322446)

[5. Map 5 11](#_Toc163322447)

[II. Level 2 12](#_Toc163322448)

[1. Map 1 12](#_Toc163322449)

[2. Map 2 13](#_Toc163322450)

[3. Map 3 13](#_Toc163322451)

[4. Map 4 14](#_Toc163322452)

[5. Map 5 15](#_Toc163322453)

[III. Level 3 16](#_Toc163322454)

[F. REFERENCES 17](#_Toc163322455)

# INTRODUCTION

This project was initialized as part of the course CSC14003 – Introduction to Artificial Intelligence. Its purpose is to help the students learn the basics of creating and implementing artificially intelligent agents through the simple childhood game hide-and-seek.

Note that the agents in this project are in no way capable of learning since all utilized algorithms are search algorithms, not neural networks.

# WORKING ENVIRONMENT

## Programing language

100% Python

## Code editor

Visual Studio Code

## Code management

GitHub

# GENERAL CODE ANALYSIS

## Seeker Agent

### Code representation

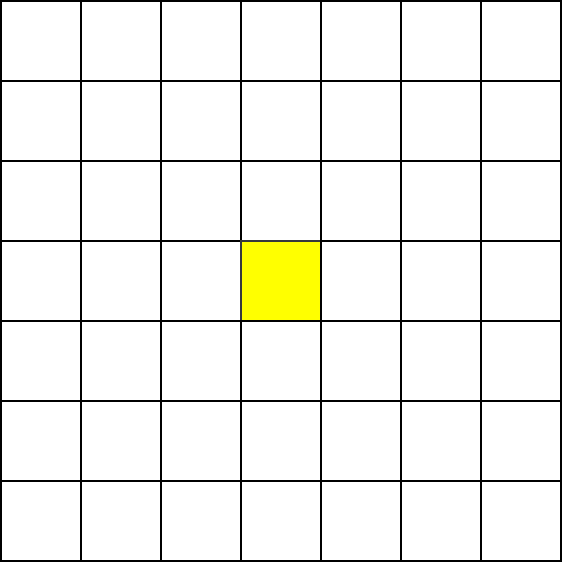
* The seeker agent is represented as the Seeker class, stored in the seeker.py file of each level.
* A Seeker object is constructed as follow:



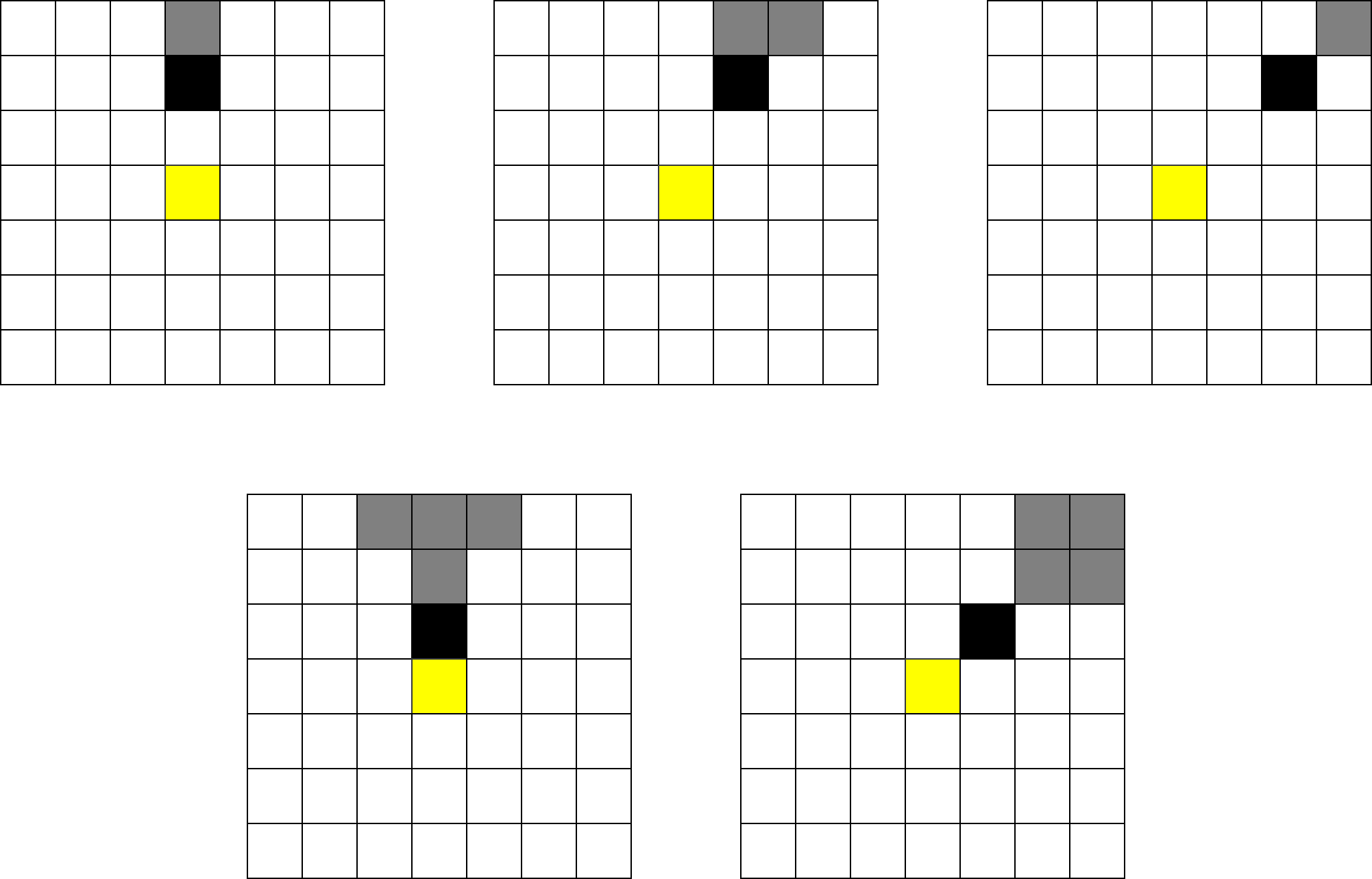
* In case you haven’t realized, this Seeker class also serves a dual purpose as a representation of the puzzle state, hence the *path\_cost*, *heuristic\_cost* and *parent* paremeters used in initialization, as well as the *\_\_lt\_\_* operator implemented further down the source code. This may look a bit odd, but it will come in handy later.

### Seeker’s vision

* As required, the seeker has a vision that ranges 3 tiles away from it in all 8 directions, as illustrated below:



* When there is an obstacle/wall within its range of vision, observable tiles are updated as follow:



(**Black tiles** represents obstacles/walls; **grey tiles** are those that are now unobservable. The same logic applies for other quarters within the range of vision)

* In cases of long walls or multiple obstacles, the tiles that become unobservable are simply combinations of the above conditions.

### Path-finding using search algorithms

* Within the Seeker class, 3 search algorithms are implemented:
  + Breath-first search (BFS)
  + Hill climbing search, in which the seeker tries to move towards positions with the highest possible number of observable tiles.
  + A\* search using Manhattan distance between current position and target position as the heuristic function.
* The priorities of each search strategies are as follow:
  + When neither hider or signal is present within observable tiles, traverse the map using Hill climbing search.
  + Upon encountering a local maximum (when all possible moves lead to a reduce in observable ties), switch to BFS to search for another unexplored portion of the map.
  + At any point during traversal, if a signal or hider appears in sight, immediately switch to A\* search to reach the position of said hider/signal.
* The seeker takes 1 step per time unit, and also remembers tiles it has already observed/walked through so as to reduce redundant steps.

## Graphical demonstration

* This process is greatly assisted by the **pygame** module. So make sure to have the module installed before running the program.

### Main menu

* Upon starting the program, users will be met with the main menu containing the title and subtitle of the game, along with some buttons.
* The buttons are objects of the Button class, which is designed specifically around the **pygame** module.
  + Each button can change size when the cursor hovers on it.
  + Upon clicking a button, it initiates a function, which opens another screen.

### Demonstrating the running process

* The seeker’s path will be illustrated in a “game runner” screen. This screen will initialize after all calculation and processing are done (meaning only after the seeker finishes the game) so it might take a while to show up.
* Since the whole path taken by the seeker is saved, the GUI will only have to display every step in a slideshow.

### Score display

* The bottom of the game runner will have a bit of space reserved for score display.
* Score starts at 0, and decreases by 1 per time unit that passes. For each hider the seeker manages to catch, the score will increase by 20.
* When terminating the game runner, said score will also be printed out to console for debugging.

# LEVEL-SPECIFIC ANALYSIS

## Level 1

* In this level, things are at their most simple form: 1 hider which cannot move, 1 seeker and no time limit; the game ends when the seeker successfully catch the hider (i.e. they occupy the same tile).
* The seeker’s strategy is also at its base form that I have described in section C above.

## Level 2

* There are multiple stationary hiders in this level, the number of which the seeker knows.
* The strategy used by the seeker is largely the same, with the only difference being that it will not stop until all hiders are caught.

# EVALUATION & COMMENTS

* This section will focus on evaluating the performance of the program. Each level features 5 different maps. 3 runs will be performed on each map to record the average time and resource consumed by the program.

## Level 1

### Map 1

A screenshot of a computer game

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -13 | 22.105455 | 1.148438 |
| 2 | -28 | 95.970869 | 3.605469 |
| 3 | -13 | 9.648561 | 1.164062 |
| Avr | -18 | 42.57496167 | 1.972656333 |

### Map 2

A black and white maze with a blue square

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -422 | 2164.680 | 30.094 |
| 2 | -47 | 13.485 | 1.484 |
| 3 | -333 | 1294.753 | 17.547 |
| Average | -267.3 | 1157.64 | 16.375 |

### Map 3

A maze with many different colored squares

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -209 | 2418.203 | 26.703 |
| 2 | -165 | 1460.073 | 16.918 |
| 3 | -192 | 1903.081 | 26.387 |
| Average | -188.67 | 1927.110 | 23.336 |

### Map 4

A maze with a square pattern

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -91 | 642.522 | 14.047 |
| 2 | -63 | 174.661 | 10.535 |
| 3 | -132 | 3776.902 | 18.035 |
| Average | -95.33 | 1531.36 | 114.205 |

### Map 5

A maze with many different colored squares

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -10 | 26.582 | 3.504 |
| 2 | -10 | 14.787 | 3.547 |
| 3 | -11 | 85.644 | 9.652 |
| Average | -10.33 | 42.447 | 5.5677 |

## Level 2

### Map 1

A screenshot of a crossword puzzle

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | 27 | 297.400 | 3.254 |
| 2 | 35 | 70.783 | 1.465 |
| 3 | 35 | 230.805 | 2.758 |
| Average | 32.33 | 199.66 | 2.492 |

### Map 2

A black and white maze

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -443 | 9585.526 | 36.570 |
| 2 | -442 | 10845.965 | 38.926 |
| 3 | -356 | 8593.667 | 29.031 |
| Average | -413.66 | 9675.052 | 34.842 |

### Map 3

A maze with different colored squares

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -184 | 10921.771 | 29.594 |
| 2 | -85 | 8108.374 | 34.168 |
| 3 | -217 | 11707.010 | 26.641 |
| Average | -162 | 10245.718 | 30.134 |

### Map 4

A maze with a square pattern

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -172 | 15674.640 | 48.75 |
| 2 | -215 | 16618.833 | 52.109 |
| 3 | -338 | 53120.585 | 118.676 |
| Average | -241.6 | 28471.352 | 73.178 |

### Map 5

A maze with many black lines and blue squares

Description automatically generated with medium confidence

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -470 | 118617.489 | 160.703 |
| 2 | -529 | 115374.011 | 112.895 |
| 3 | -355 | 66782.402 | 109.594 |
| Average | -451.33 | 100257..9673 | 127.73 |

## Level 3

* Level 3 uses the same set of maps as level 2, so only the result tables will be listed
  + Map 1

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | 40 | 34.73 | 1.25 |
| 2 | 30 | 38.686 | 1.051 |
| 3 | 35 | 69.499 | 1.547 |
| Average | 35 | 47.638 | 1.283 |

* + Map 2

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -168 | 4028.924 | 24.477 |
| 2 | -278 | 6556.450 | 30.910 |
| 3 | -299 | 10329.924 | 32.977 |
| Average | -248.33 | 6971.766 | 29.454 |

* + Map 3

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -186 | 13528.262 | 37.375 |
| 2 | -197 | 10723.233 | 25.934 |
| 3 | -193 | 12502.166 | 28.02 |
| Average | -192 | 12251.22 | 30.443 |

* + Map 4

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -157 | 12374.505 | 50.457 |
| 2 | -129 | 15581.947 | 67.988 |
| 3 | -337 | 57083.051 | 159.547 |
| Average | -207.66 | 28346.501 | 92.664 |

* + Map 5

|  |  |  |  |
| --- | --- | --- | --- |
| Run | Score | Time (miliseconds) | Space (megabytes) |
| 1 | -565 | 181661.581 | 297.09 |
| 2 | -479 | 96586.793 | 166.895 |
| 3 | -546 | 131795.551 | 123.34 |
| Average | -530 | 136681.308 | 199.775 |

# REFERENCES

* <https://www.youtube.com/watch?v=GMBqjxcKogA>