

Artificial Intelligence (A.I.) Project #1

Miika Malin

Biomimetics and Intelligent Systems Group (BISG)
University of Oulu

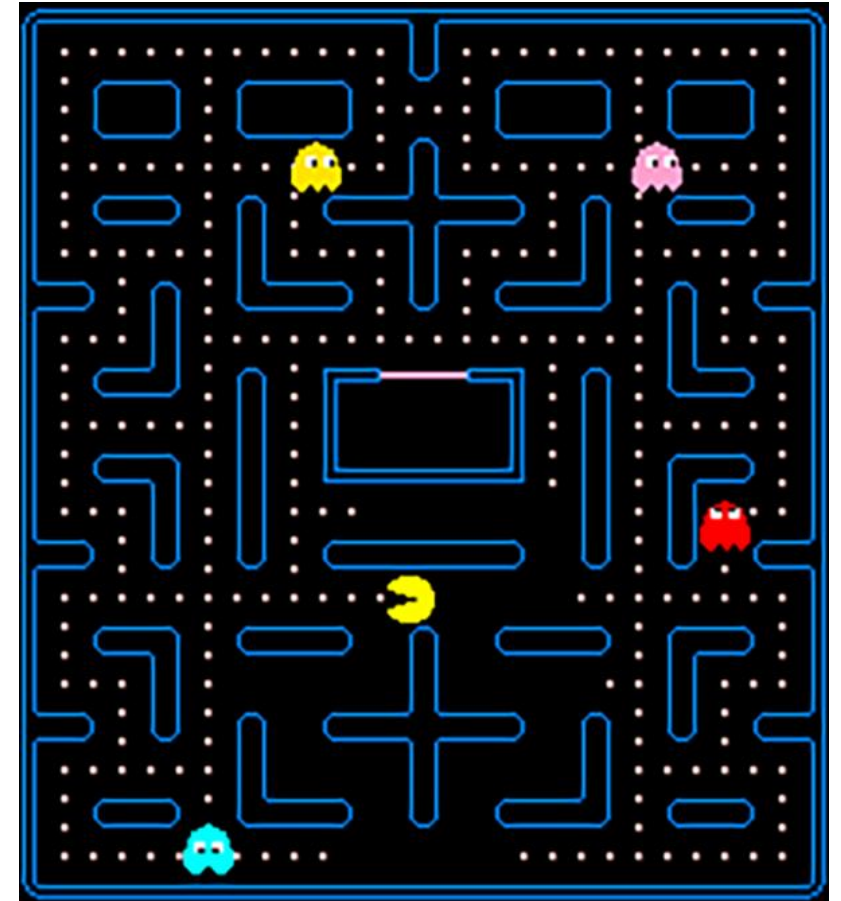
Let's use A.I. to
win a game!



PacMan World

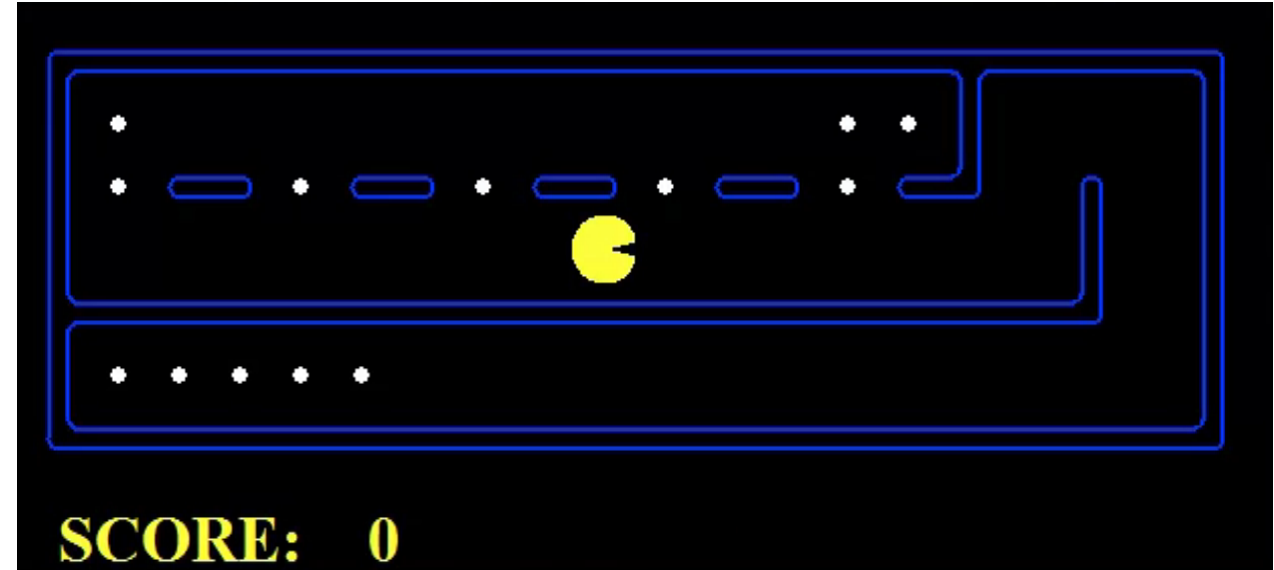
The player navigates Pac-Man through a maze containing dots, known as Pac-Dots, and four multi-colored ghosts: Blinky, Pinky, Inky and Clyde.

The goal of the game is to accumulate as many points as possible by collecting the dots and eating ghosts. When all of the dots in a stage is eaten, that stage is completed.



Project 1: Search methods in Pacman

In this project, your Pacman agent will find paths through his maze world, both to reach a particular location and to collect food efficiently. You will build general search algorithms and apply them to Pacman scenarios.

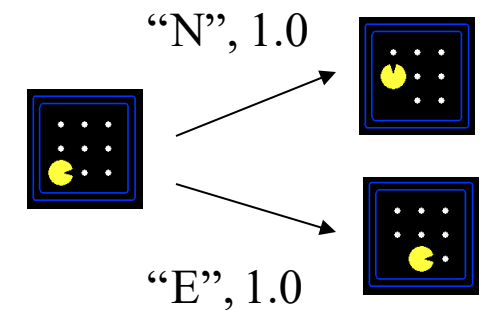


Help the Pacman agent to navigate effectively in the Pacman world by implementing depth-first, breadth-first, uniform cost, and A* search algorithms.

Project 1: Search methods in Pacman

The search problem is defined by:

1. A state representation, *i.e.* {Pacman Position, Food Positions, Ghosts Positions}
2. A successor function including actions and cost
3. A final goal



How to solve the problem?

- ✓ A solution includes a series of sequential actions that transform the start state (state 0) to the final goal.



Project 1: Search in Pacman

search.py

Q1: Finding a Fixed Food Dot using Depth First Search (3 points)

Q2: Breadth First Search (3 points)

Q3: Varying the Cost Function (3 points)

Q4: A* Search (3 points)

searchAgent.py

Q5: Finding All the Corners (3 points)

Q6: Corners Problem: Heuristic (3 points)

Q7: Eating All the Dots (4 points)

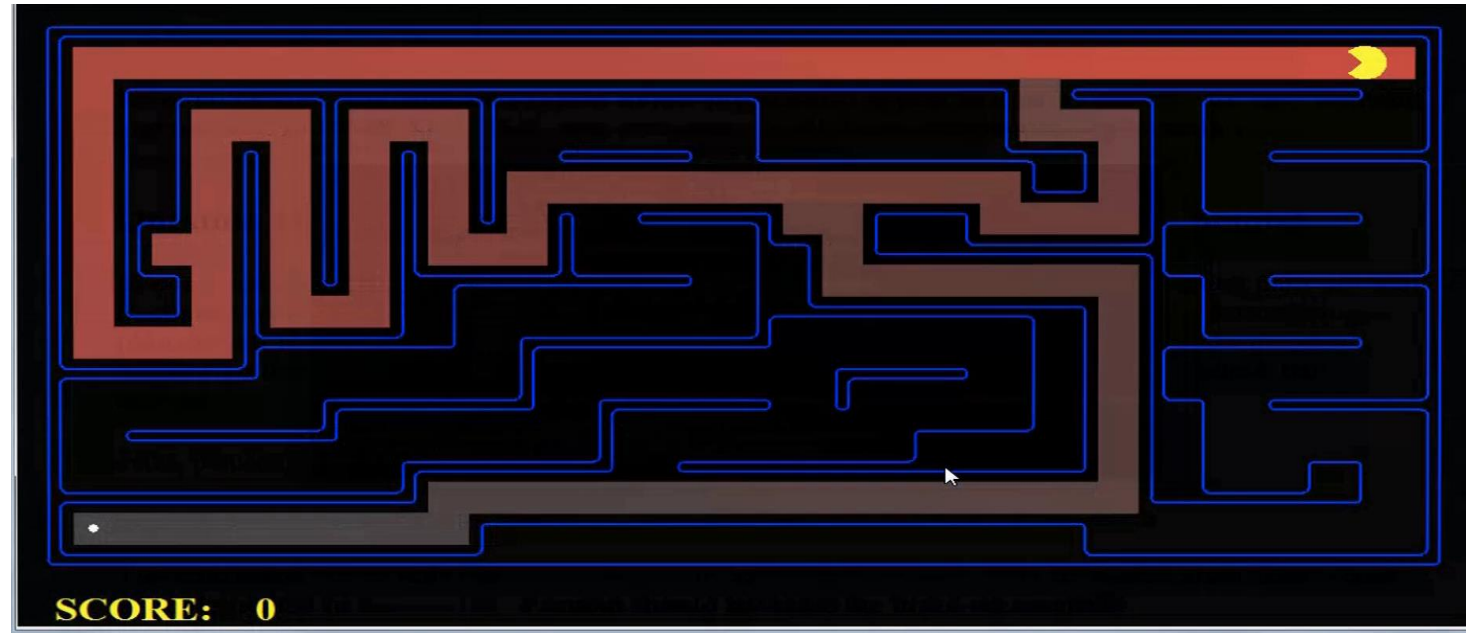
Q8: Suboptimal Search (3 points)

25 Points

Project 1: Search in Pacman

Questions 1-4

- **Problem: Navigation**
 - **States:** The current location (x,y) .
 - **Successor:** update the location of Pacman by taking appropriate actions.
 - **Goal test:** Navigate Pacman agent to the food position, *i.e.* $(x,y) == \text{Food Position}$



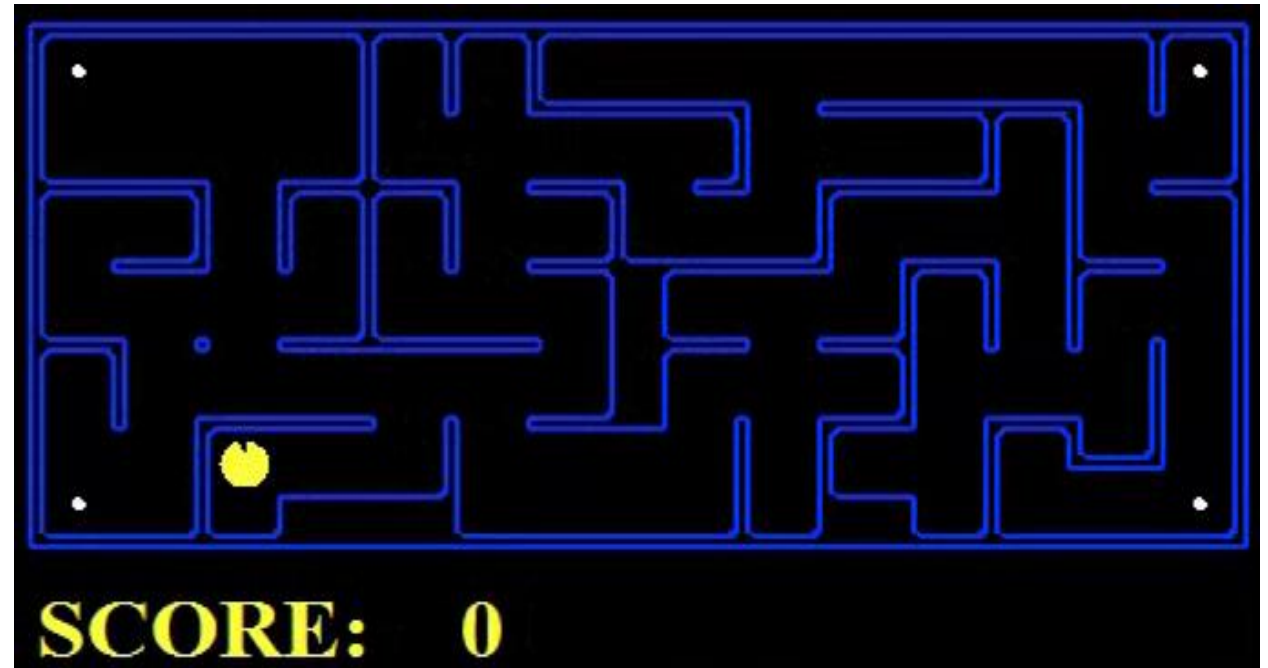
A sample solution for the mandatory questions.

Project 1: Search in Pacman

Questions 5 & 6

In corner mazes, there are four dots, one in each corner. Our new search problem is to find the shortest path through the maze that touches all four corners (whether the maze actually has food there or not).

- Problem: Finding All the Corners
 - **States:** $\{(x,y), \text{corners booleans}\}$
 - **Successor:** update location and possibly a corner boolean
 - **Goal test:** all corners are true



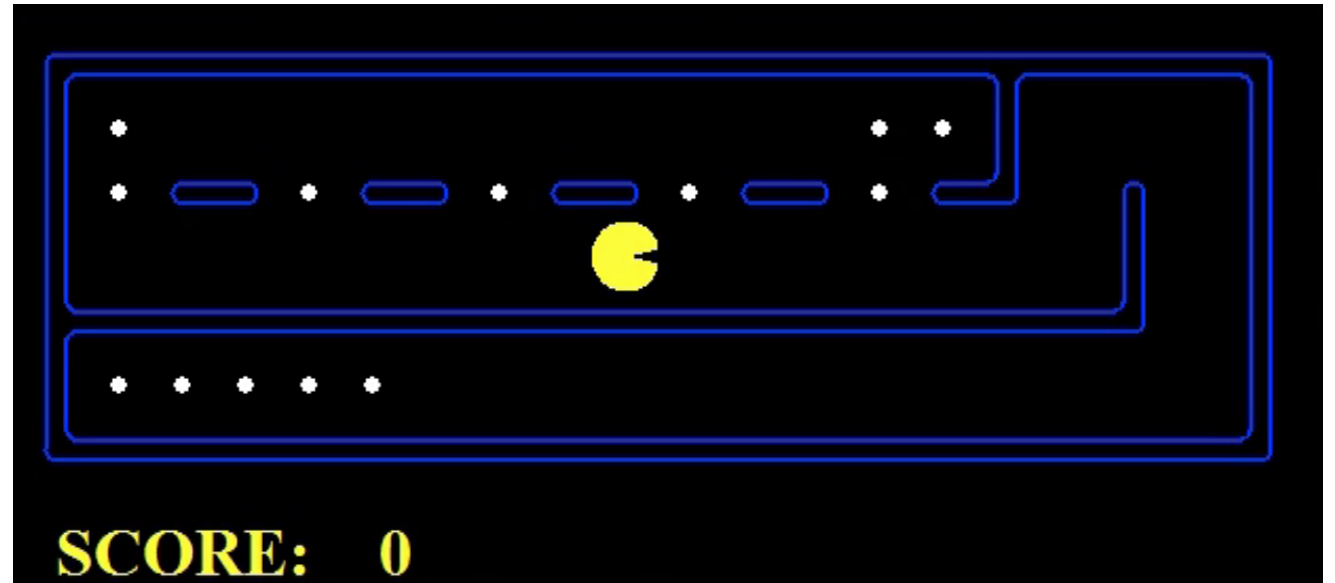
A sample solution for the questions 5 and 6.

Project 1: Search in Pacman

Questions 7 & 8

Eating all the Pacman food in as few steps as possible. For this, we'll need a new search problem definition which formalizes the food-clearing problem. A solution is defined to be a path that collects all of the food in the Pacman world.

- **Problem:** Eating All The Dots
 - **States:** $\{(x,y), \text{dots booleans}\}$
 - **Successor:** update location and possibly a dot boolean
 - **Goal test:** all dots are true



A sample solution for the questions 7 and 8.

Project 1: Search in Pacman

Implementation

- The projects for this class assume you use Python 3.6 (at least)



- Python Tutorial can be found in <https://docs.python.org/3.6/tutorial/>
- A crash course for beginners <https://youtu.be/JJmcL1N2KQs>
- No need for any extra libraries or packages!

Project 1: Search in Pacman

Implementation

✓Files you'll edit:

search.py: Where all of your search algorithms will reside.

searchAgents.py: Where all of your search-based agents will reside.
(successor, goal function).

✓Files you might want to look at:

pacman.py: The main file that runs Pacman games. This file describes a Pacman GameState type, which you use in this project.

game.py: The logic behind how the Pacman world works. This file describes several supporting types like AgentState, Agent, Direction, and Grid.

util.py: Useful data structures for implementing search algorithms.

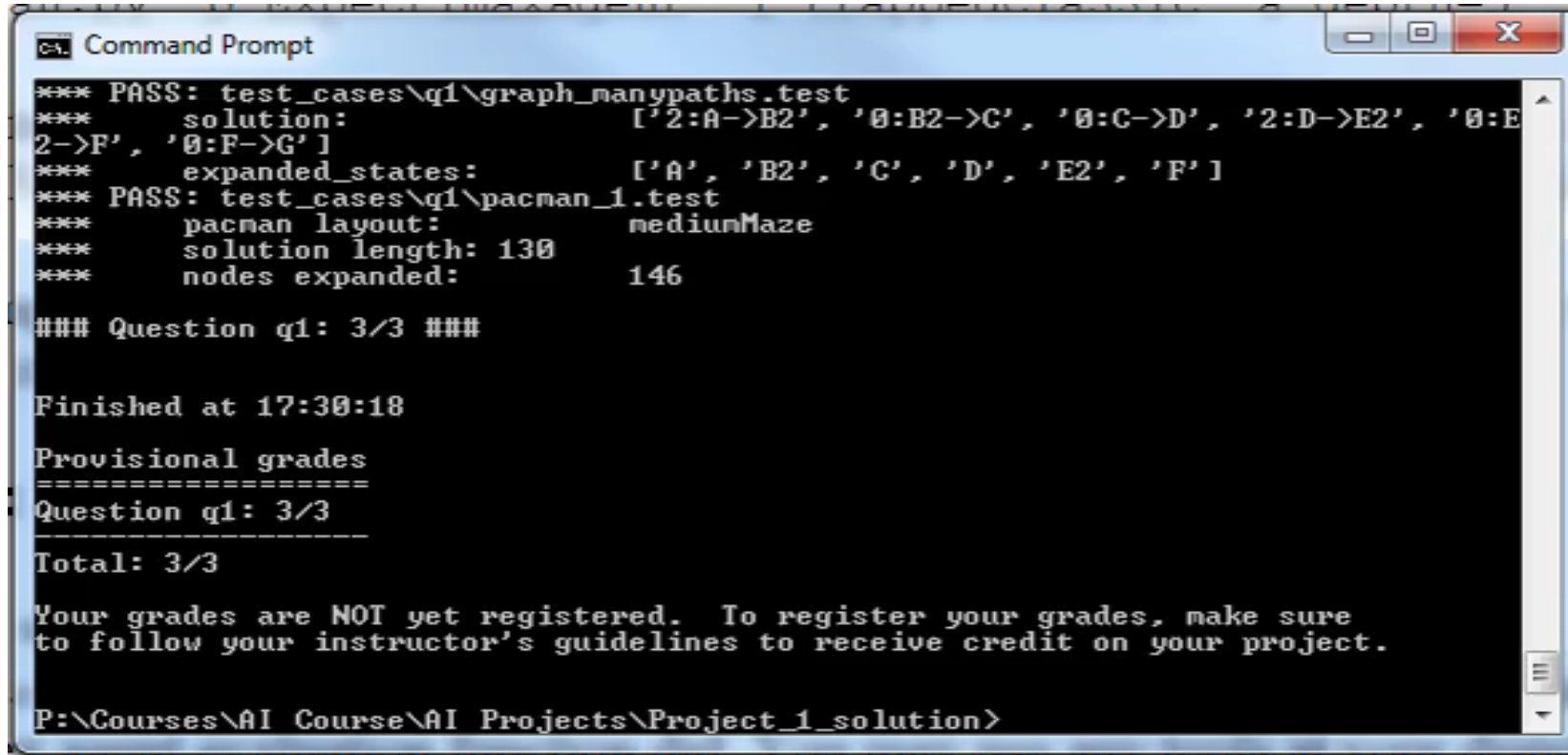
Project 1: Search in Pacman

Implementation (Library)

- `start=problem.getStartState()`: get the starting position
- `succ=problem.getSuccessors(state)`: get successors of state (**state**)
- `stack=util.Stack()`: Define a Stack structure
- `queue=util.Queue()`: Define a Queue
- `Pqueue=util.PriorityQueue ()`: Define a PriorityQueue **Pqueue**.
- `util.Stack.push(stack, state)`/ `util.Queue.push(queue, state)`: push the state (**state**) into the Stack (**sack**) or Queue (**queue**)
- `state=util.Stack.pop(stack)`: pop the last element from a Stack (**stack**)
- `state=util.Queue.pop(queue)`: pop the last element from the Queue (**queue**)
- `problem.isGoalState(state)`: check if the state (**state**) is a final state

Project 1: Search in Pacman

- ✓ You can check your implementation using the provided autograder function.



```
C:\ Command Prompt
*** PASS: test_cases\q1\graph_manypaths.test
*** solution: ['2:A->B2', '0:B2->C', '0:C->D', '2:D->E2', '0:E2->F', '0:F->G']
*** expanded_states: ['A', 'B2', 'C', 'D', 'E2', 'F']
*** PASS: test_cases\q1\pacman_1.test
*** pacman layout: mediumMaze
*** solution length: 130
*** nodes expanded: 146

### Question q1: 3/3 ###

Finished at 17:30:18

Provisional grades
=====
Question q1: 3/3
-----
Total: 3/3

Your grades are NOT yet registered. To register your grades, make sure
to follow your instructor's guidelines to receive credit on your project.

P:\Courses\AI Course\AI Projects\Project_1_solution>
```

python autograder.py

(for checking the whole project)

python autograder.py -q q1

(for a specific question, e.g. Q1)

Project 1 Points

- The submission deadline: **February 11, 2022 23:59**
- Wrap all the codes into a ZIP file for submission in Moodle.
- The project may be done in a group of two people at most.
- This project assignment has 2 points. You should get at least 18 and 23 score from autograder to get 1 and 2 points, respectively. At least 1 point is required to pass the course.

Minimum score from autograder	Project points
18	1
23	2

- You may reach me by email: miika.malin@oulu.fi
- Tutoring session every Monday 16:15 – 17:00 in Zoom (starting 24.1.).



GAME  OVER