FACULTY OF
INFORMATION TECHNOLOGY
AND ELECTRICAL ENGINEERING

UNIVERSITY
OF OULU

Artificial Intelligence (A.I.)

Project #1

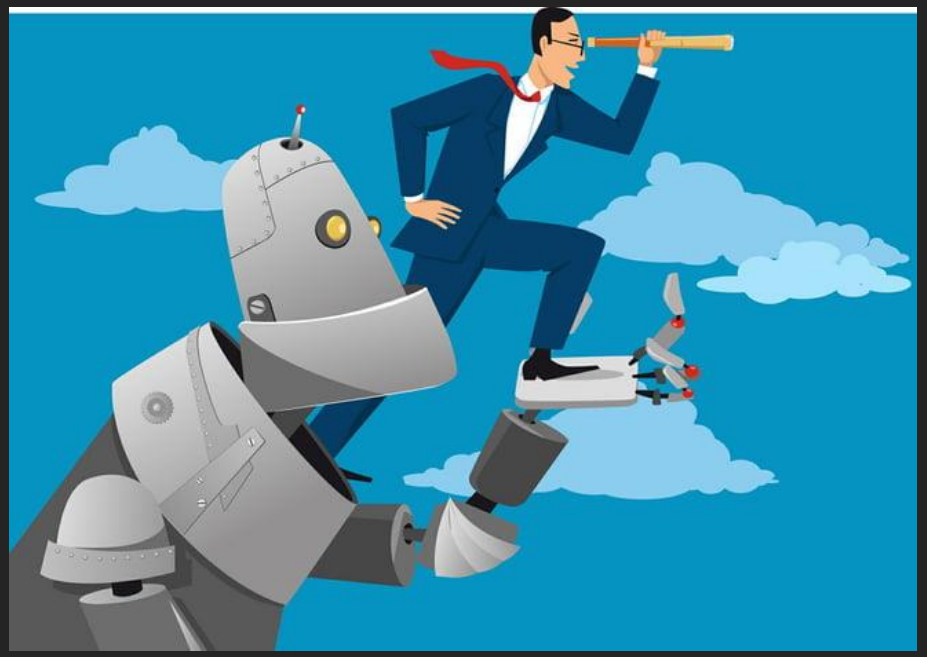
Miika Malin

Biomimetics and Intelligent Systems Group (BISG)
University of Oulu

This project is based on http://ai.berkeley.edu/

Original slides made by Mohammad Tavakolian (CMVS 2021)

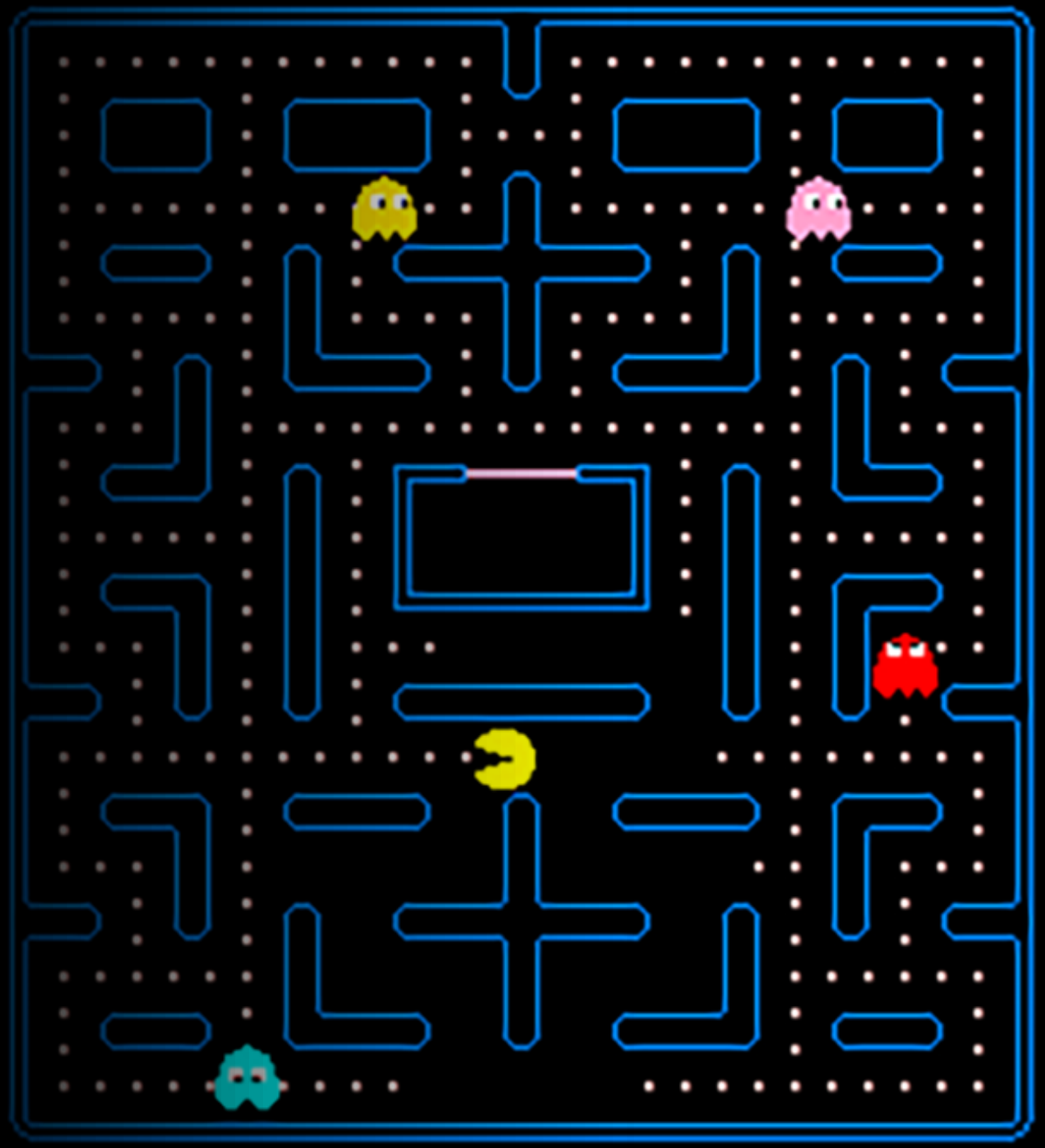
Let’s use AI to win a game!



PacMan World

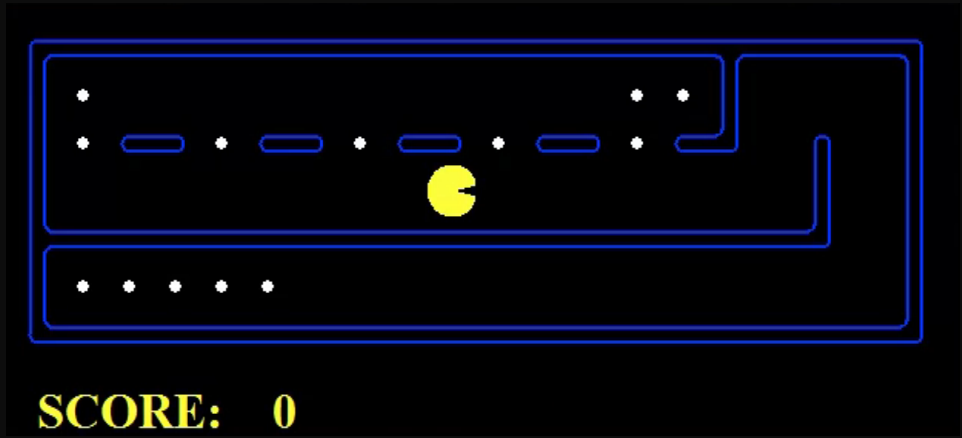
The player navigates Pac-Man through a maze containing dots, known as Pac-Dots, and four multi-colored ghosts: Blinky, Pinky, Inky and Clyde.

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.

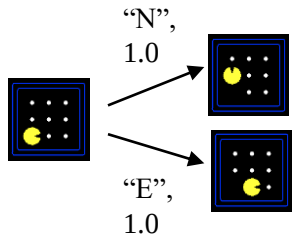


Help the Pacman agent to navigate effectively in the Pacman world by implementing depth-first, breadth-first, uniform cost, and A* search algorithms.

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 Funtion (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)

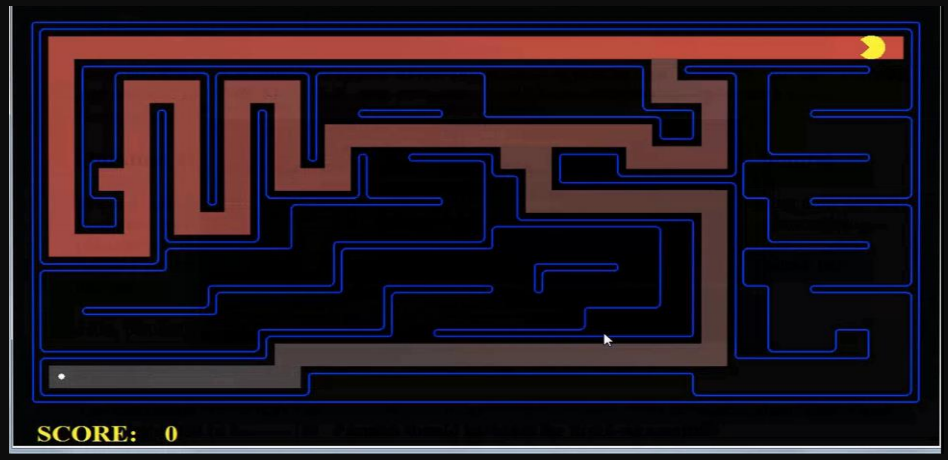
Q8: Suboptimal Search (3 points)

25 Points

Project 1: Search in Pacman

Questions 1-4

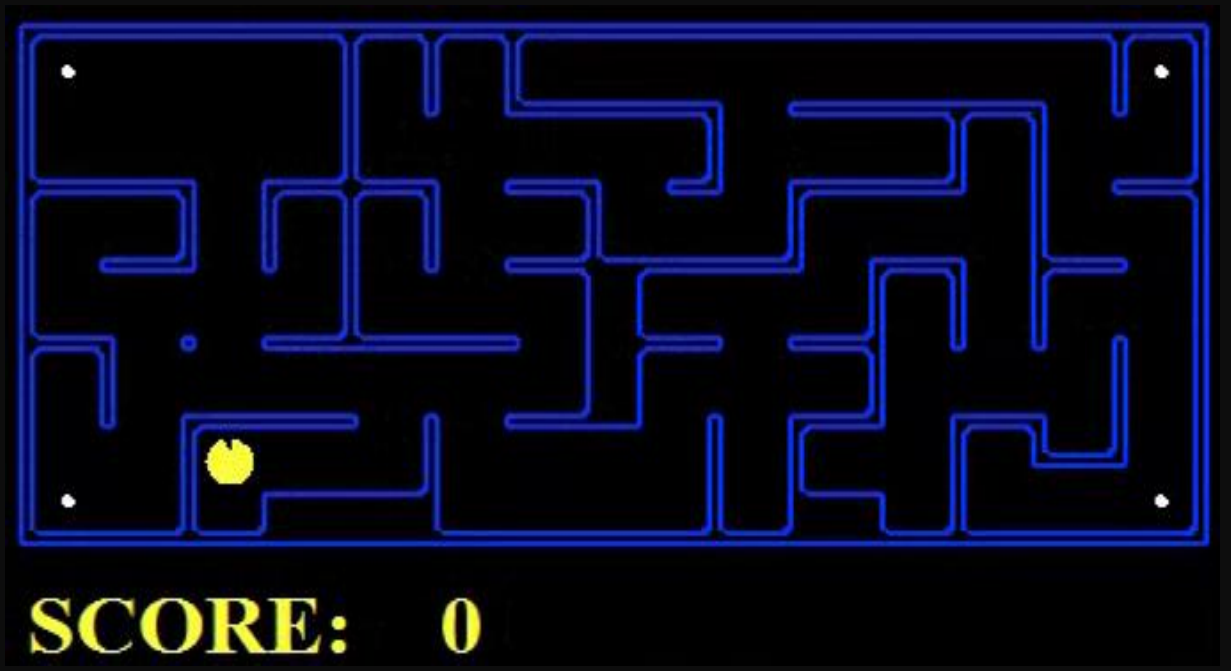
- Problem: Navigation
  - **States:** The current location (x, y).
  - **Successor:** update the location of Pacman by taking appropriate actions.
  - **Goal test:** Navigate Pacman agent to the food position, *i.e.* (x, y) == Food Position



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).

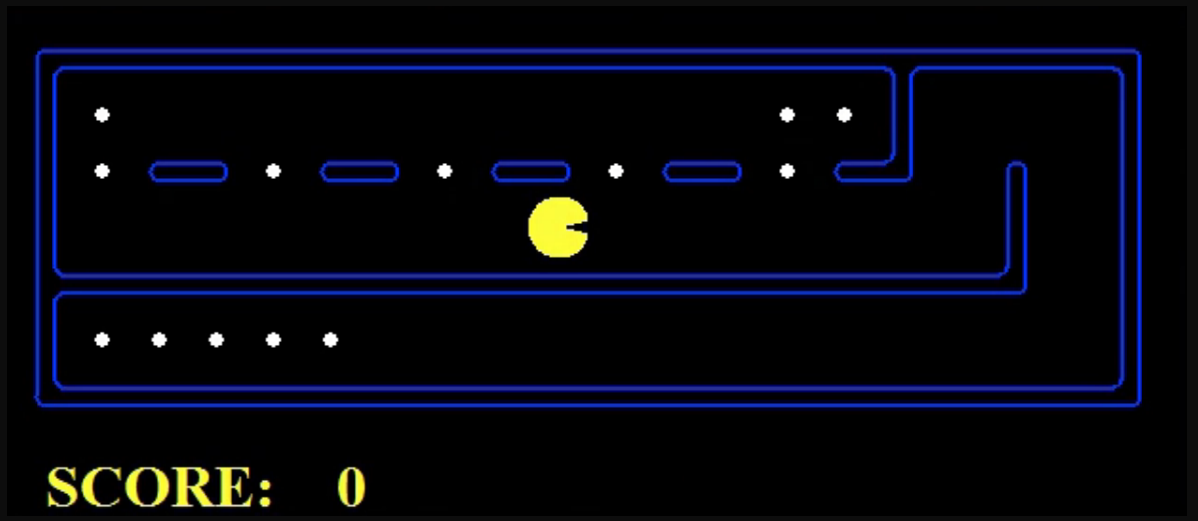


- Problem: Finding All the Corners
  - **States:** {(x,y), corners booleans}
  - **Successor:** update location and possibly a corner boolean
  - **Goal test:** all corners are true

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), dots booleans}
  - **Successor:** update location and possibly a dot boolean
  - **Goal test:** all dots are true

Project 1: Search in Pacman

General information

The projects for this class assume you use Python 3.8.1

Python Tutorial can be found in https://docs.python.org/3.8/tutorial/

A crash course for beginners https://youtu.be/JJmcL1N2KQs

No need for any extra libraries or packages!

Project 1: Search in Pacman

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.

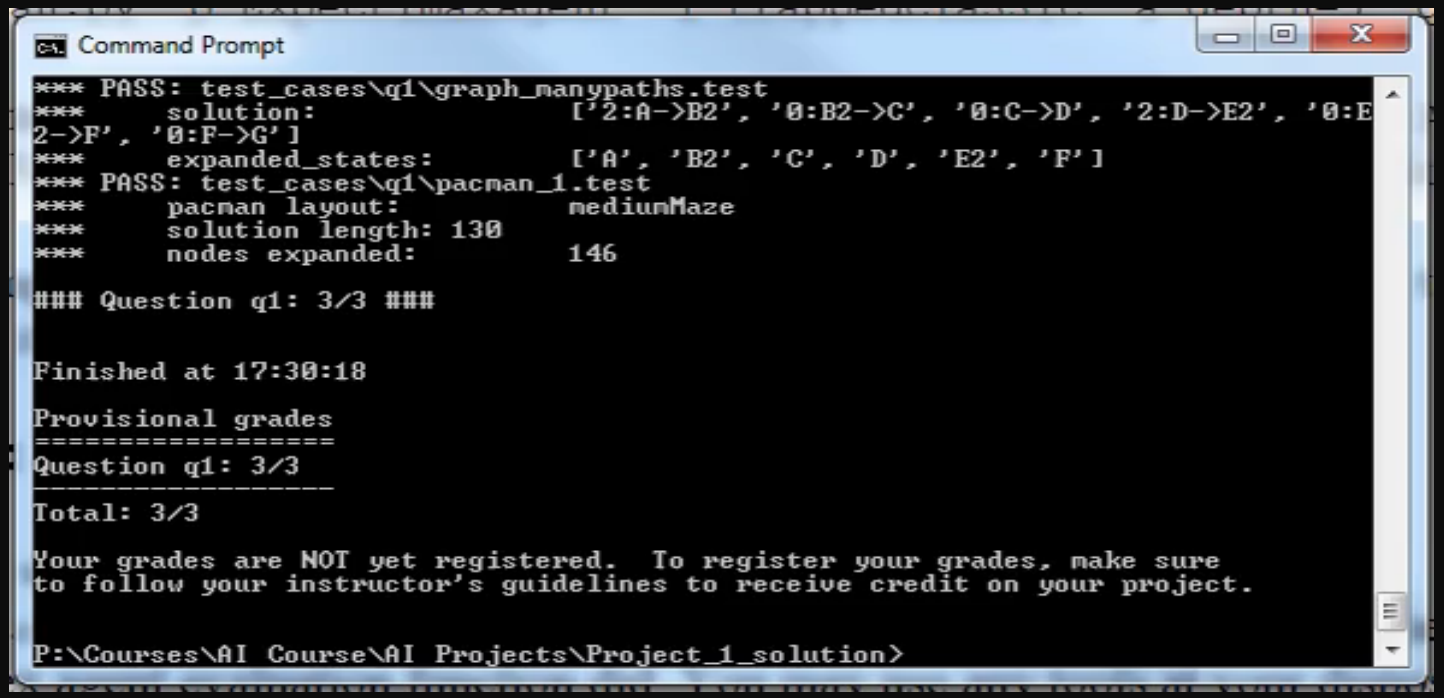
util.py: Useful data structures for implementing search algorithms.

Project 1: Search in Pacman

Get the starting position	start = problem.getStartState()
Get successors of state	succ = problem.getSuccessors(state)
Define a Stack structure	stack = util.Stack()
Define a Queue	queue = util.Queue()
Define a PriorityQueue pqueue	pqueue = util.PriorityQueue ()
Push the state (state) into the Stack (stack) or Queue (queue)	util.Stack.push(stack, state)) / util.Queue.push(queue, state))
Pop the last element from a Stack (stack)	state = util.Stack.pop(stack)
pop the last element from the Queue (queue)	state = util.Queue.pop(queue)
Check if the state (state) is a final state	problem.isGoalState(state))

Project 1: Search in Pacman

You can check your implementation using the provided autograder function.



python autograder.py (for checking the whole project)

python autograder.py –q q1 (for a specific question, e.i. Q1)

Project 1 Practicalities

- Submission deadline is 10.2. 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.
- You should get at least score of 18 to pass the project, which is mandatory part of the course
- One of the questions in first exam will be related to some of the Project 1 programming tasks

How to get help

- Exercise sessions
  - Schedule is in Moodle (every Tuesday 14:15-16:00, TS101, Starting next week)
  - Main way to get help with projects / weekly exercises
  - We will check some of the problems from handouts together, after that there is time to get help with project or exercise
  - Bring your laptop if you are going to work with the project
- You may also reach TAs via email


GAME OVER