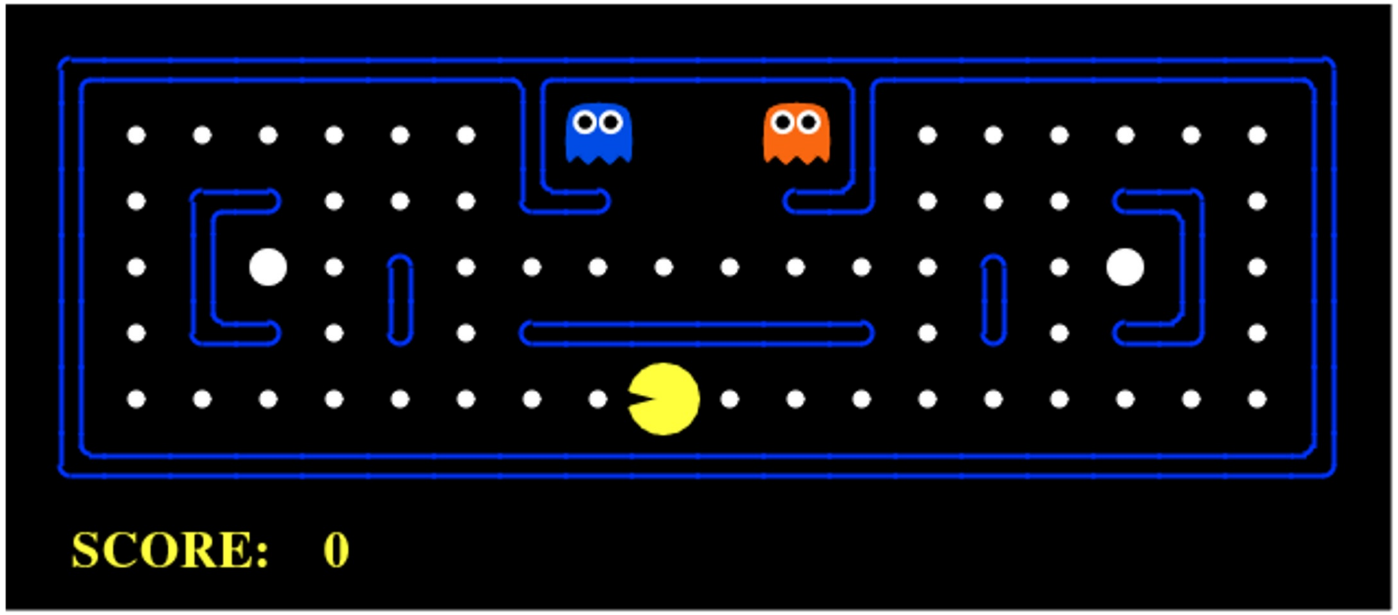
Artificial Intelligence

Homework #2
Multi-Agent Search

Wen-Huang Cheng (鄭文皇)
National Taiwan University
wenhuang@csie.ntu.edu.tw

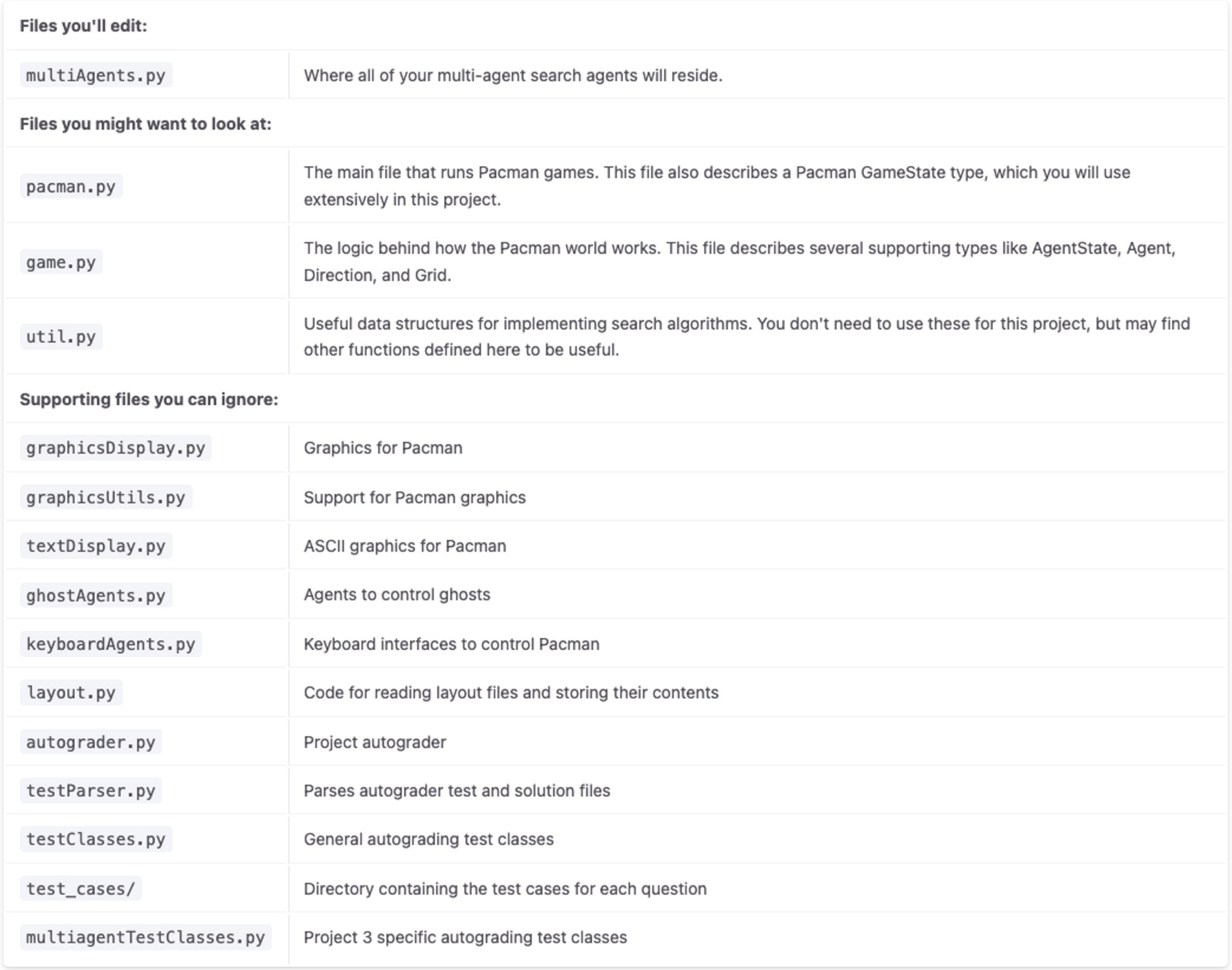
Objective

❖ **Multi-Agent Pacman:**
- ➢ Q1. Reflex Agent
- ➢ Q2. Minimax
- ➢ Q3. Alpha-Beta Pruning



Important note:
- ❖ In this homework, you will design agents for the classic version of Pacman, including ghosts. Along the way, you will implement both minimax and Alpha-Beta Pruning.
- ❖ The code base has not changed much from the previous HW1, but please start with a fresh installation, rather than intermingling files from HW1.

Files in this Homework



Files you'll edit:	
`multiAgents.py`	Where all of your multi-agent search agents will reside.
Files you might want to look at:	
`pacman.py`	The main file that runs Pacman games. This file also describes a Pacman GameState type, which you will use extensively 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. You don't need to use these for this project, but may find other functions defined here to be useful.
Supporting files you can ignore:	
`graphicsDisplay.py`	Graphics for Pacman
`graphicsUtils.py`	Support for Pacman graphics
`textDisplay.py`	ASCII graphics for Pacman
`ghostAgents.py`	Agents to control ghosts
`keyboardAgents.py`	Keyboard interfaces to control Pacman
`layout.py`	Code for reading layout files and storing their contents
`autograder.py`	Project autograder
`testParser.py`	Parses autograder test and solution files
`testClasses.py`	General autograding test classes
`test_cases/`	Directory containing the test cases for each question
`multiagentTestClasses.py`	Project 3 specific autograding test classes

- The code for this project consists of several Python files, **some of which you will need to read and understand in order to complete the assignment**, and some of which you can ignore.
- You will fill in portions of **multiAgents.py** during the assignment.
- Your code will be autograded for technical correctness. Please do not change the names of any provided functions or classes within the code, or you will wreak havoc on the autograder.

Multi-Agent Pacman

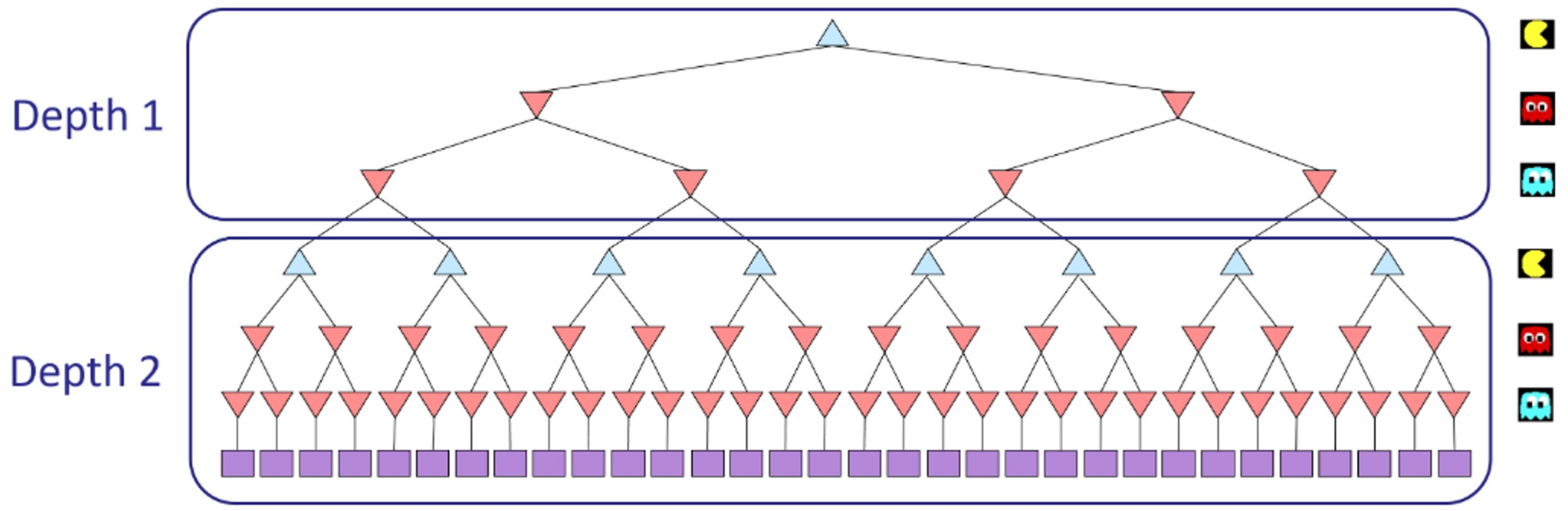
- Q1. Reflex Agent
  - Improve the **ReflexAgent** in **multiAgents.py** to play respectably. The provided reflex agent code provides some helpful examples of methods that query the **GameState** for information. A capable reflex agent will have to consider both food locations and ghost locations to perform well.
  - Grading:
    - $ python autograder.py –q q1
  - Grading w/o graphics:
    - $ python autograder.py –q q1 --no-graphics

Detailed information is listed in README.md inside AI2024-hw2.zip.

Multi-Agent Pacman (Cont.)

Q2. Minimax

- Now you will write an adversarial search agent in the provided **MinimaxAgent** class stub in **multiAgents.py**. Your minimax agent should work with any number of ghosts, so you'll have to write an algorithm that is slightly more general than what you've previously seen in homework. In particular, your minimax tree will have multiple min layers (one for each ghost) for every max layer.
- Grading: $ python autograder.py –q q2

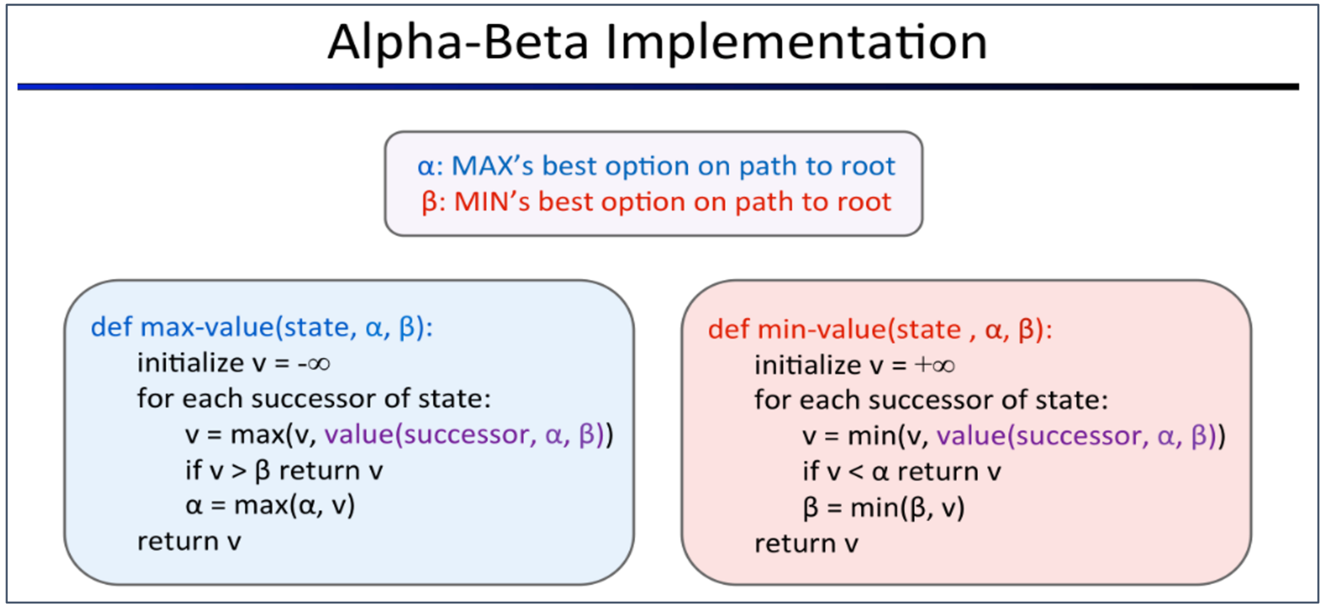


Detailed information is listed in README.md inside AI2024-hw2.zip.

Multi-Agent Pacman (Cont.)

Q3. Alpha-Beta Pruning

- Make a new agent that uses alpha-beta pruning to more efficiently explore the minimax tree, in **AlphaBetaAgent**. Again, your algorithm will be slightly more general than the pseudocode from homework, so part of the challenge is to extend the alpha-beta pruning logic appropriately to multiple minimizer agents.
- Grading: $ python autograder.py –q q3



Alpha-Beta Implementation

α: MAX's best option on path to root
β: MIN's best option on path to root

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v > β return v
        α = max(α, v)
    return v
```

```
def min-value(state , α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v < α return v
        β = min(β, v)
    return v
```

Detailed information is listed in README.md inside AI2024-hw2.zip.

Report

- **Show your autograder results and describe each algorithm:**
  - Q1. Reflex Agent (2%)
  - Q2. Minimax (2%)
  - Q3. Alpha-Beta Pruning (2%)
- **Describe the idea of your design about evaluation function in Q1. (2%)**
- **Demonstrate the speed up after the implementation of pruning. (2%)**

Grading

- **Multi-Agent Pacman (90%)**
  - Q1. Reflex Agent (30%)
  - Q2. Minimax (30%)
  - Q3. Alpha-Beta Pruning (30%)
- **Report (10%)**
  - PDF format only

Submission

- Deadline: 2024/04/10 (Wed.) 23:59
- Zip all files as **hw2_<student_id>.zip**
- Submit to NTU COOL
- Your submission should include the following files:
  - hw2_<student_id>.pdf
  - AI2024-hw2
- Do not put report.pdf into AI2024-hw2 folder

Any Question

ai.ta.2024.spring@gmail.com