P01 Pacman

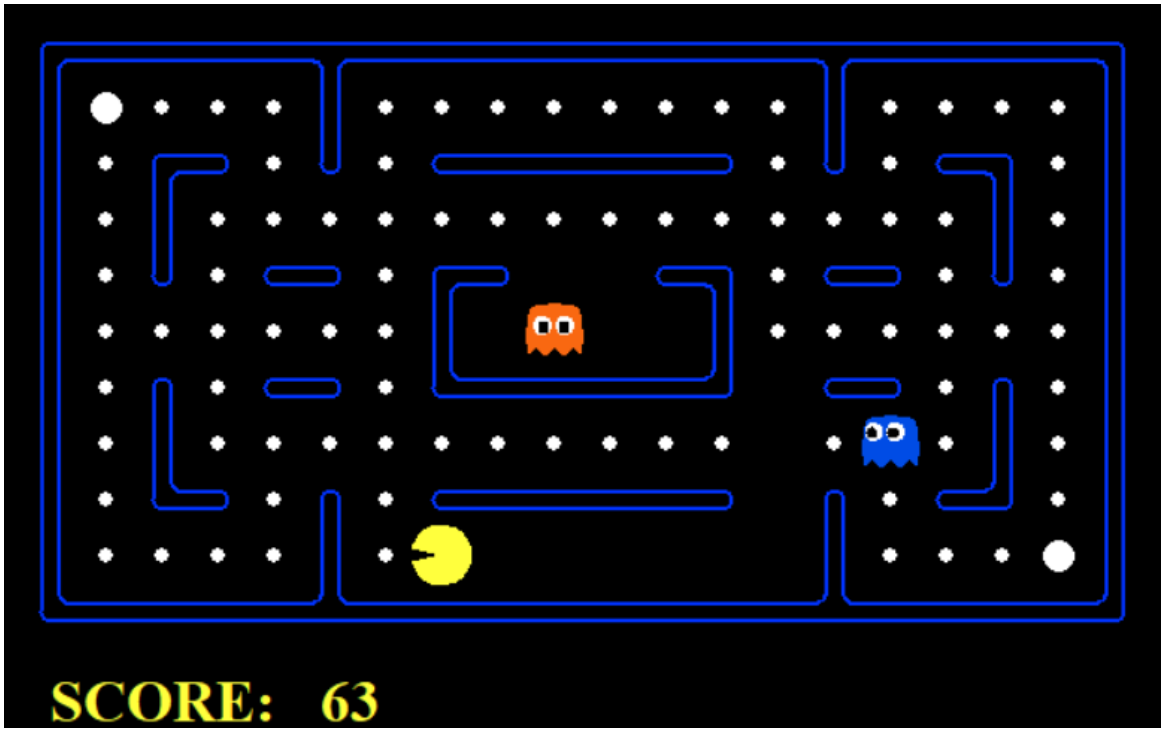
Suixin Ou

School of Computer Science
Sun Yat-sen University

September 14, 2021

Pacman

Pacman is a maze action game developed and released by Namco for arcades in 1980. The player controls the pacman through an enclosed maze. The objective of the game is to eat all of the dots placed in the maze while avoiding four colored ghosts (You can run "python pacman.py" to play the game). You need to design a Pacman agent to play the game automatically in this project.



Part 1

Problem

In this part, your Pacman agent will find paths through its 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.

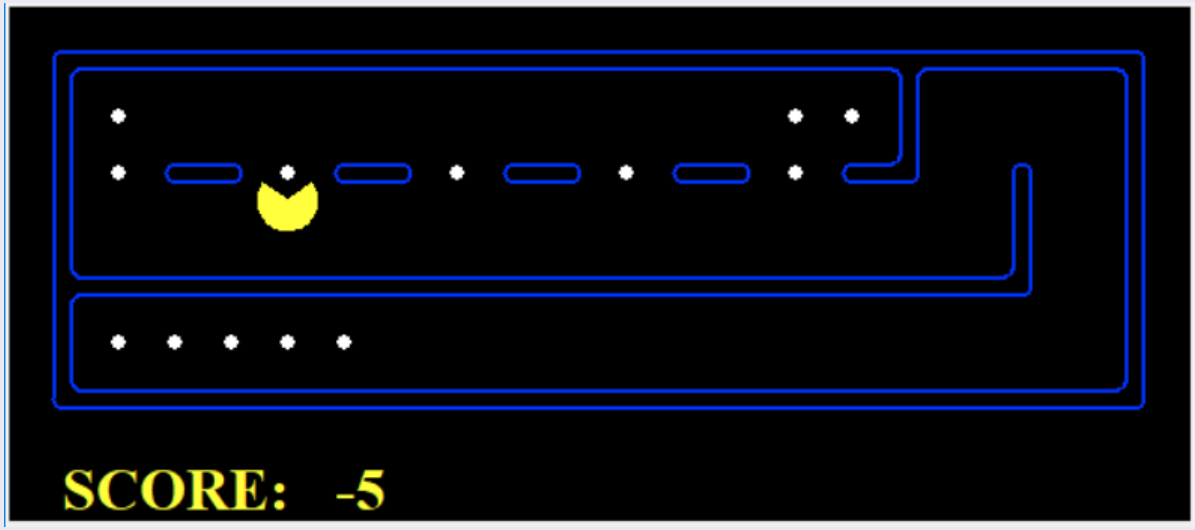


Figure 1: Searching by A*

Question 1 (3 points)

Find a path through a maze to a fixed position using the Manhattan distance heuristic. Run `python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic` for a test.

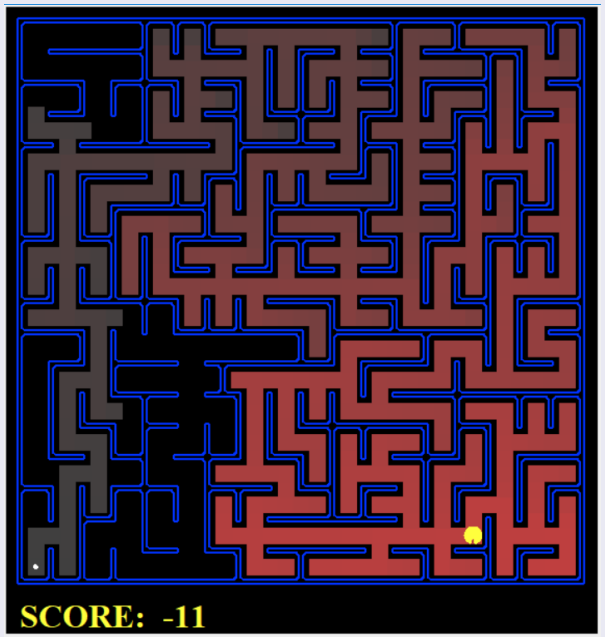


Figure 2: Finding a path by A*

Part 1

Given the class Position Problem

searchAgents.py × search.py

```
135
136  class PositionSearchProblem(search.SearchProblem):
```

Get initial state by problem.getStartState()

```
167      def getStartState(self):
168          return self.startState
```

Get successors by problem.getSuccessors(), each successor is a tuple (nextState, action, cost)

```
183      def getSuccessors(self, state):
184          """
```

Judge that the terminate state has been reached by problem.isGoalState()

```
170      def isGoalState(self, state):
171          isGoal = state == self.goal
172
173          # For display purposes only
174          if isGoal and self.visualize
175              self._visitedlist.append
176              import __main__
177              if '_display' in dir(__ma
178                  if 'drawExpandedCells
179                      __main__._displa
180
181          return isGoal
```

Given Manhattan Heuristic

searchAgents.py | search.py

```
253  def manhattanHeuristic(position, problem, info={}):
254      "The Manhattan distance heuristic for a PositionSearchProblem"
255      xy1 = position
256      xy2 = problem.goal
257      return abs(xy1[0] - xy2[0]) + abs(xy1[1] - xy2[1])
```

You should finish the function "aStarSearch" in search.py

searchAgents.py | search.py

```
109  def aStarSearch(problem, heuristic=nullHeuristic):
110      """Search the node that has the lowest combined cost and heuristic first."""
111      "*** YOUR CODE HERE ***"
```

Question 2 (3 points)

Implement a non-trivial, consistent heuristic for the CornersProblem in cornersHeuristic. Run `python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5` for a test.

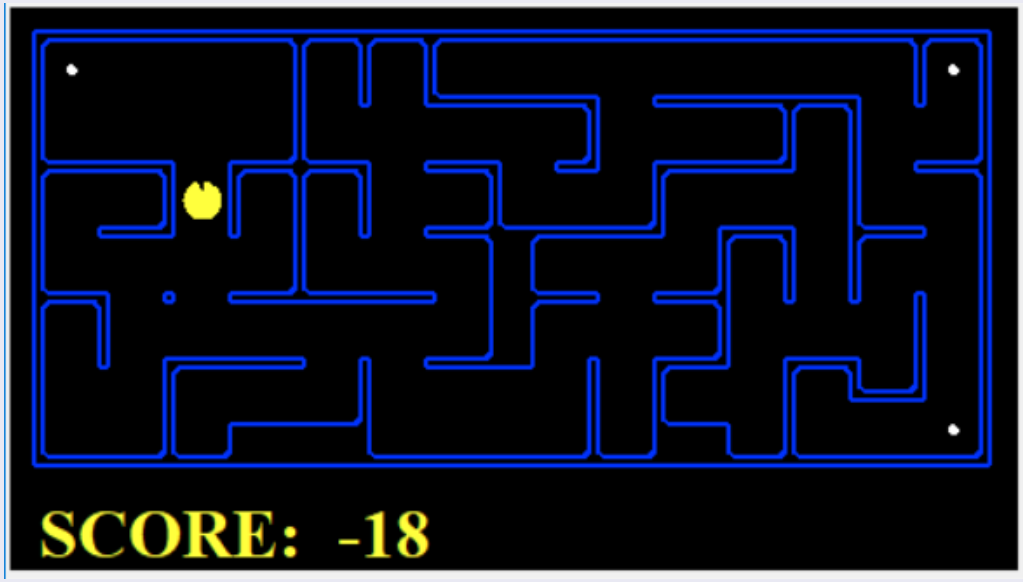


Figure 3: Testing Corners Heuristic

You should finish the class "CornersProblem" and the function "cornersHeuristic" in searchAgent.py

```
269 class CornersProblem(search.SearchProblem):
270     """
271     This search problem finds paths through all four corners of a layout.
272
273     You must select a suitable state space and successor function
274     """
```

Compared with the Position Problem, the status of CornersProblem should also include the four corners.

```
346 def cornersHeuristic(state, problem):
347     """
348     A heuristic for the CornersProblem that you defined.
```

The heuristic value can be the farthest distance from the current position to the four corners.

Grading for Question 2

Depending on how few nodes your heuristic expands, you'll be graded:

Number of nodes expanded	Grade
more than 2000	0/3
at most 2000	1/3
at most 1600	2/3
at most 1200	3/3

Question 3 (4 points)

Now we'll solve a hard search problem: eating all the Pacman food in as few steps as possible. Run `python pacman.py -l trickySearch -p AStarFoodSearchAgent` for a test.

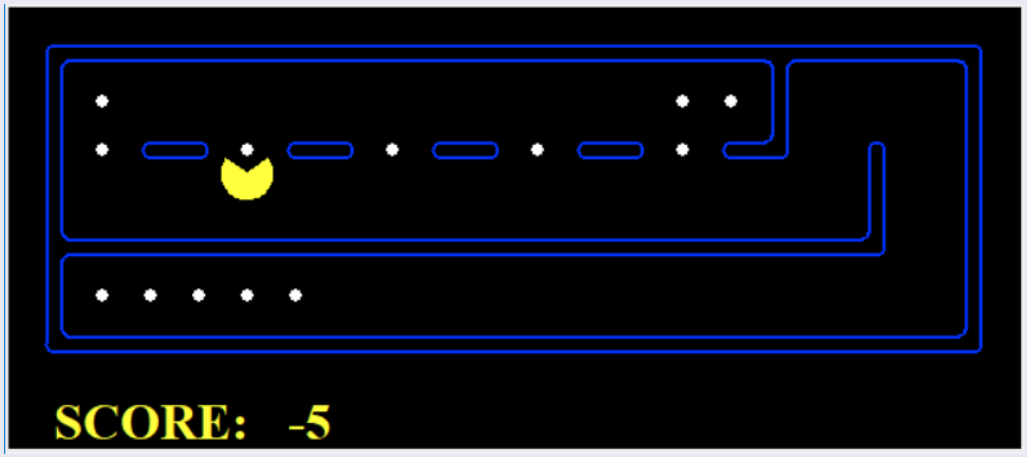


Figure 4: Testing Food Heuristic

Part 1

You should finish the function "foodHeuristic" in searchAgent.py

```
447  def foodHeuristic(state, problem):
448      """
449      Your heuristic for the FoodSearchProblem goes here.
```

Grading for Question 3

Depending on how few nodes your heuristic expands, you'll get additional points:

Number of nodes expanded	Grade
more than 15000	1/4
at most 15000	2/4
at most 12000	3/4
at most 9000	4/4 (full credit; medium)
at most 7000	5/4 (optional extra credit; hard)

Problem

In this section, you will design agents for the classic version of Pacman, including ghosts. Along the way, you will implement both minimax and $\alpha - \beta$ pruning.

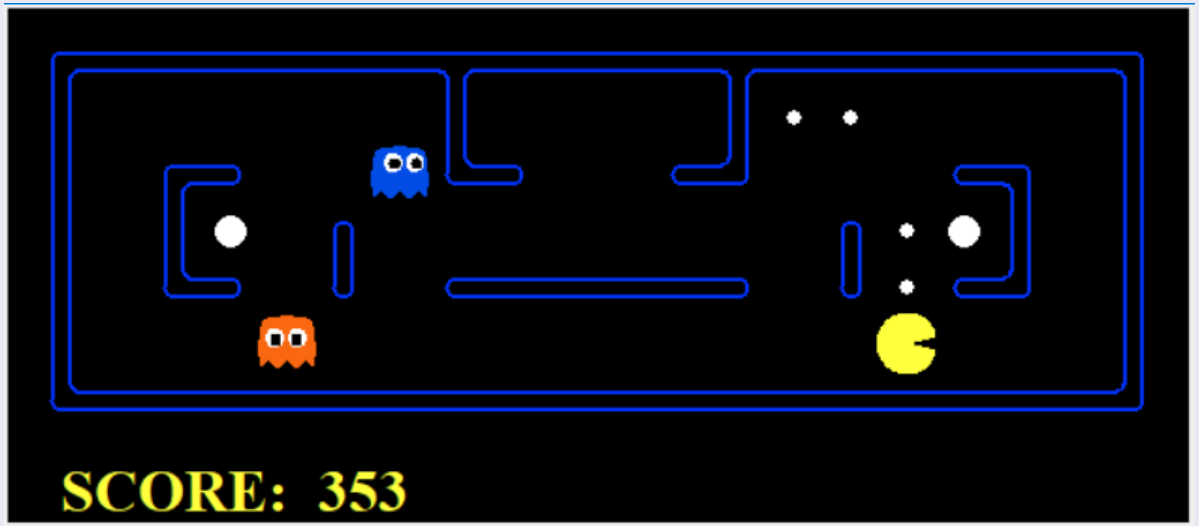


Figure 5: Searching by $\alpha - \beta$ pruning

Pacman

Submission

Pack your report `report.pdf` and source code into zip file `E02_YourNumber.zip`, then send it to `ai_course2021@163.com`.

The End