Artificial Intelligence
AIC_4301C
TP2
2022-2023

Exercise 1: A^* Serach

1. Implement A* graph search in the empty function **aStarSearch** in **search.py**.

   A* takes a heuristic ($h(n)$ function) as an argument. The **nullHeuristic** heuristic function in search.py is a trivial example.

   Test your code:

   python3 pacman_AIC.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=nullHeuristic

2. You can test your A* implementation on the original problem of finding a path through a maze to a fixed position using the Manhattan distance heuristic (implemented already as **manhattanHeuristic** in **searchAgents.py**).

   Test your code:

   python3 pacman_AIC.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

Exercise 2: Corners Problem: Heuristic

The trivial heuristics are the ones that return zero everywhere (UCS) and the heuristic which computes the true completion cost.

1. Implement a non-trivial, consistent heuristic for the **CornersProblem** in **cornersHeuristic** (in searchAgents.py). If UCS and A* return paths of different lengths, your heuristic is inconsistent.

2. Test your code:

   python3 pacman_AIC.py -l mediumCorners -p AStarCornersAgent -z 0.5

   AStarCornersAgent is a shortcut for:

   -p SearchAgent -a fn=aStarSearch,prob=CornersProblem,heuristic=cornersHeuristic

Exercise 3: Eating All The Dots

Eating all the Pacman food in as few steps as possible problem is implemented in: **FoodSearchProblem** in **searchAgents.py**.

A solution is defined to be a path that collects all of the food in the Pacman world. If you have written your general search methods correctly, A* with a null heuristic (equivalent to uniform-cost search) should quickly find an optimal solution to **testSearch** with no code modifications (total cost of 7):

python3 pacman_AIC.py -l testSearch -p AStarFoodSearchAgent
AStarFoodSearchAgent is a shortcut for:
-p SearchAgent -a fn=astar,prob=FoodSearchProblem,heuristic=foodHeuristic

1. Fill in foodHeuristic in searchAgents.py with a consistent heuristic for the FoodSearchProblem. For this lab, heuristics can depend on the placement of walls, regular food and Pacman.

2. Test your code:

   python3 pacman_AIC.py -l trickySearch -p AStarFoodSearchAgent

Exercise 4: Suboptimal Search

Write an agent that always greedily eats the closest dot. **ClosestDotSearchAgent** is implemented for you in **searchAgents.py**, but a function that finds a path to the closest dot is missing.

1. Implement the function **findPathToClosestDot** in **searchAgents.py**.
2. Test your code:

   python3 pacman_AIC.py -l bigSearch -p ClosestDotSearchAgent -z .5