Project 1 Part 2 (Pacman)

David Hoang, 913611912

CSC 665 Spring 2018

Overview

The objective of this assignment was to do questions 5-7, filling in the CornersProblem, cornersHeuristic, and foodHeuristic functions so that they would work in accordance to the provided auto-grader, and within a reasonable amount of expanded nodes.

CornersProblem

The goal of this part was to fill in the getStartState, isGoalState, and getSuccessors functions in the CornersProblem. These were very similar to the given PositionSearchProblem functions, except for some additions. One of these additions was to combine a state with a list of explored corners as a tuple when expanding successors. Another was to add some extra conditionals in the checking of the goal state, seeing if all corners have been explored using that list.
This implementation completes mediumCorners with a total cost of 106 in 0.5 seconds with 2448 search nodes expanded, and a score of 434. It passes the autograder's tests.

CornersHeuristic

This function uses the manhattanDistance function in util.py to calculate the closest corner, then repeats that process zero to three more times depending on how many corners have been explored. It returns the sum of the calculations to each corner, or 0 if all corners have already been explored.
This implementation completes mediumCorners with a total cost of 106 in 0.4 seconds with 901 search nodes expanded, and a score of 434. This is the same optimal solution as the previous CornersProblem, but with less than half of the amount of search nodes expanded. It passes the autograder's tests.

FoodHeuristic

The calculation used in this heuristic finds the furthest food dot away from the current position with the given mazeDistance function, and returns its distance. This works because it expands nodes first headed towards that dot regardless of other nearby dots, but as the exploration expands towards the furthest dot, at some point the original furthest dot is no longer the furthest dot from the current position and it bounces in that space increasing cost and lowering that path’s priority in the queue. The logic at the end of it is something along the lines of “how can it collect the furthest initial dot without having to go back and collect the other dots.”

This implementation completes trickySearch with a total cost of 60 in 75.5 seconds with 4137 search nodes expanded, and a score of 570. It passes the autograder’s tests.

Thoughts

The CornersProblem was straightforward and didn’t require much tinkering on my part, as explained earlier, so it was relatively easy.

The CornersHeuristic took some figuring out, but as it used the same idea of storing corners in a list, all that was needed was to find a way to change explored corners in to a value. I decided to use manhattanDistance, but I think there are other ways to do it.

The FoodHeuristic was the most difficult problem. It was possible to write it as a single line, “return len(foodGrid.asList())” which simply returns the number of food dots left. However, this solution only gave 2/4 on the autograder. I tried a number of different ways, such as reusing the idea of CornersHeuristic, adding either mazeDistance or manhattanDistance between dots as a heuristic, but it only passed tests 1, 10-17, 2-5, then got up to test 6 after which it failed the consistency test. I also tried adding the mazeDistance/manhattanDistance of the closest food dot along with the number of food dots left then returning the sum, but that failed the 10th test. After which, I removed the food dot part of the previous implementation and just tried the distance of the closest dot, but that didn’t work either. I actually found the current solution partly by accident as I was trying to change things up here and there in an attempt to pass the tests, and just altered the min() to max() in my implementation. After passing the tests, I did not know why it worked, because to me it was not inherently intuitive to use the furthest dot as a heuristic, as evidenced by the fact that I kept on using the closest dot in my previous attempts. But I thought about it and eventually vaguely understood why it worked.