Programming Assignment Report

Assignment 02

Pacman problem

Problem

The problem given Is to design an algorithm to successfully find the single food piece on the given layout. We were tasked to design a breadth first search, depth first search, AStar, and a hill climb algorithm. For AStar and Hill climb we were tasked to use two differing heuristics, Manhattan and Euclidean.

Execution

Included in the zip is a list of every command executed that I used for this assignment. Each algorithm was used with each of the required layouts. Included are the commands to run each of the algorithms that use heuristics with both Euclidian and Manhattan.  
they are located in the ***commands.txt*** file.

Screenshots

(there are too many commands to screenshot every command line output) Each were ran without the Gui visible but it does not change the algorithmic output. Towards the end are the outputs showing how my function work visually.

Hillclimb partial command line outputs

A screenshot of a computer program

AI-generated content may be incorrect.

DFS partial command line outputs  
A screenshot of a computer program

AI-generated content may be incorrect.

BFS partial command line outputs  
A screenshot of a computer program

AI-generated content may be incorrect.

ASTAR partial command line outputs  
A screenshot of a computer program

AI-generated content may be incorrect.

DFS GUI

A screenshot of a video game

AI-generated content may be incorrect.

In this example it searches the Open maze layout for the furthest point to the left and ends up getting stuck and then having to find a differing path that instead goes down at some point. It gets stuck again on the right side so it again has to find a path that does not go right into the second catch.

BFS GUI

A screenshot of a video game

AI-generated content may be incorrect.

This is using the Open maze layout and is instead searching outward in a diamond pattern to get the best result. The visualization shows it got stuck on the right catch. This caused it to find a path that cut left to get to the food piece.

A\* Euclidean GUI  
A screenshot of a computer screen

AI-generated content may be incorrect.

Using the Euclidian heuristic this algorithm was able to find the optimal solution to the problem. It also did not explore the right lower block.

A\* Manhattan GUI

A screenshot of a computer

AI-generated content may be incorrect.

Using the Euclidian heuristic this algorithm was able to find the optimal solution to the problem. It also did not explore the right lower block. It also seemed to be better than the Euclidian heuristic overall but still ending up with the same score.

Hill Climb Euclidean and Manhattan GUI  
A screenshot of a video game

AI-generated content may be incorrect.

I could not get this algorithm to work on any other layout. It seemed to get stuck and never find the end. But this maze does not show the algorithms effectiveness.

HillClimb Manhatten and Euclidean GUI

A screenshot of a computer

AI-generated content may be incorrect.

This shows how effective this algorithm is at running when using this heuristic. It goes form point a to b while not exploring any other nodes. This is by far the best algorithm when trying to got from a to b.

HillClimb Error GUI

A maze with a yellow ball in the middle

AI-generated content may be incorrect.  
  
This one will forever hang on this point. I did not have time to implement the functionality to fully understand where the walls were. So it gets stuck on any wall in its path to success.