**P102**

**Sergi Olives 193196**

**Óscar Subirats**

**Lab Project 1**

**Artificial Intelligence**

**1. What works?**

Exercises from 1 to 6 and exercise 8 work correctly passing all tests.

**2. What does not work?**

Exercise 7 does not pass the last test (tricky maze) and expands more than 7000 nodes so the final grade for that exercise is ¾, we couldn’t achieve to lower it more.

**3. Which problems did you have doing the exercise?**

At the very first moment, we had a bit of trouble using python since we have little experience with it. Also, we lost a bit of time with the initial algorithms of the first exercises because we were not applying them correctly to the problem. The only problem that we couldn’t overcome was the fact of lowering the number of expanded nodes down to 7,000, we couldn’t reach less than 9,000 expanded nodes.

**4. Specific questions from Berkeley Search Assignment**

**4.1 DFS**

**Is the exploration order what you would have expected?**

**Does Pacman actually go to all the explored squares on his way to the goal?**

**4.2 BFS**

**Does BFS find a least cost solution?**

**4.4 A\* search**

**What happens on openMaze for the various search strategies?**

DFS applied to openMaze is the worst search strategy. It expands 576 nodes that not bad but the path that it founds has a cost of 298, much more than other search algorithms. In this maze BFS and UCS search have the same results, expand 682 nodes to eventually find a path with a cost of 54 that it is the shortest path to the goal while DFS didn’t really find it correctly. Finally, A\* search finds the shortest path to the goal as the last two algorithms but it works better expanding less nodes: 535 nodes.

In conclusion, we can say that A\* is the best search algorithm to find food in a pac-man problem while DFS in some cases cannot even find the shortest path to the goal.

**4.8 Suboptimal Search**

**Your ClosestDotSearchAgent won't always find the shortest possible path through the maze. Make sure you understand why and try to come up with a small example where repeatedly going to the closest dot does not result in finding the shortest path for eating all the dots.**

The algorithm does not always find the shortest path to eat all dots in the maze and this is a clear example that proves this. We have the graphical case on the top and the schematic one. The red lines are one possible path to follow, going first for the closest dot and then going to the left dot it will take 6 steps to eat every dot while going through the green path avoiding the first closest dot will return a cost of 5.



