Project 4- Logical PacMan

muthukumar suresh, Praveen Alam

April 12, 2015

Contents

1 question 1: Logical Pacman Depth First Search

1.1 Design

We use the recursive structure of prolog itself to implement the DFS search, we expand the current node in a greedy fashion until we reach a goal state. By the definition of DFS, the search is not optimal and does not give optimal solution

1.2 Results

1.2.1 tinymaze

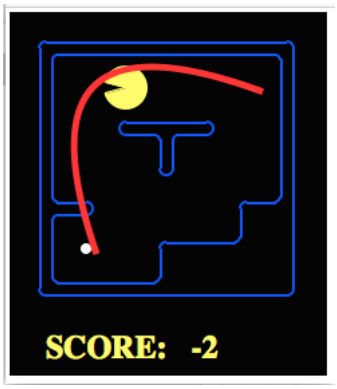

Figure 1: DFS on TinyMaze

```
['West', 'West', 'West', 'West', 'South', 'South', 'East', 'South', 'South', 'West']
Path found with total cost of 10 in 0.6 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 500
Average Score: 500.0
Scores:        500
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: As we can see from the figure, pacman takes the longer path to reach the goal.

1.2.2 Medium Maze

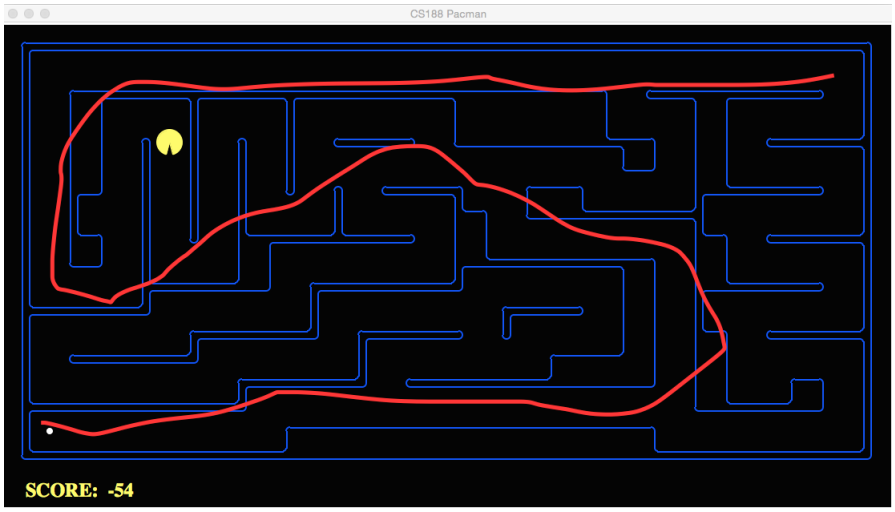

Figure 2: DFS on Medium Maze

```
Path found with total cost of 130 in 0.8 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 380
Average Score: 380.0
Scores:        380
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: DFS again takes the longer path here, rather than taking the shorter one. This is the probelm with DFS, it is not optimal.

1.3 Big Maze

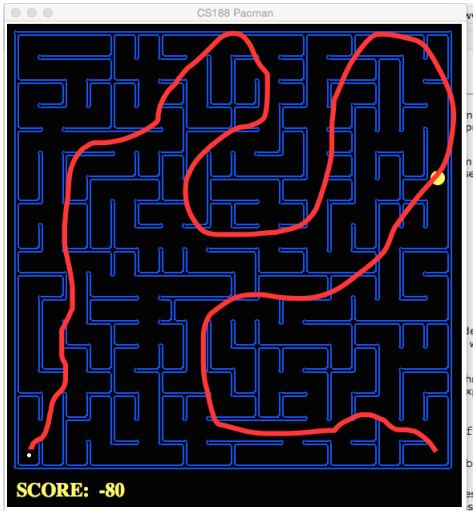

Figure 3: DFS on Big Maze

```
Path found with total cost of 212 in 0.7 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 298
Average Score: 298.0
Scores:        298
Win Rate:      1/1 (1.00)
Record:        Win
```

ANANYSIS: From the traces we can see random good behavior of the dfs for this problem. For big maze, it performs quite well for a big maze. But the optimality is not a guarantee.

2 question 2: Logical Pacman Breath First Search

2.1 Design

For implementing BFS in prolog, we keep a queue as a fringe that we add nodes to. we also keep a visited list to prevent visiting the same nodes again. Also, since the fringe is in a FIFO order, the solution is always optimal even though it takes more time.

2.2 Results

2.2.1 tinymaze BFS

```
Path found with total cost of 8 in 5.6 seconds
#Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502
Win Rate:      1/1 (1.00)
Record:        Win
```

Figure 4: BFS on TinyMaze

ANALYSIS: As we can see that the BFS finds the most optimal path and therefore the most optimal since the fringe is expanded step by step FIFO.

2.2.2 Medium Maze BFS

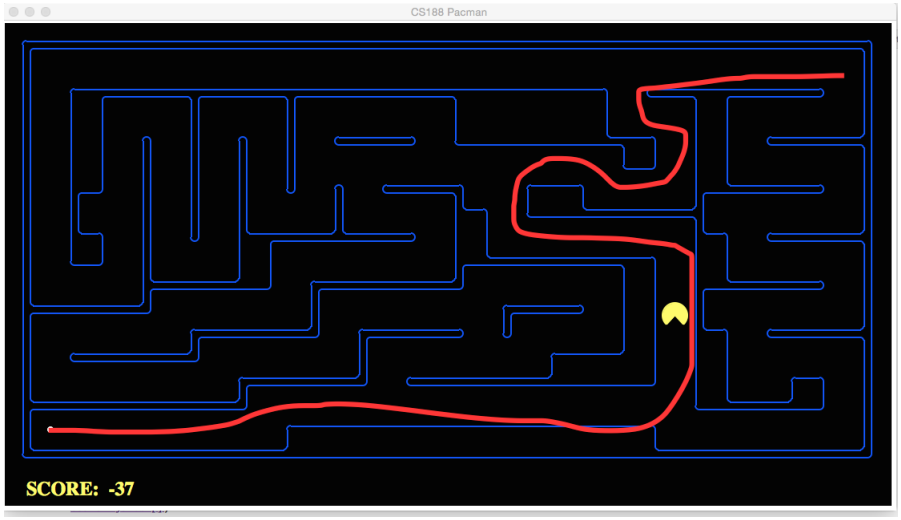

Figure 5: BFS on Medium Maze

```
Path found with total cost of 68 in 0.7 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: We see the optimality of BFS here and also time is lesser for BFS because it might expand lesser number of useless paths.

2.3 Big Maze BFS

```
Path found with total cost of 210 in 0.7 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 300
Average Score: 300.0
Scores:        300
Win Rate:      1/1 (1.00)
Record:        Win
```

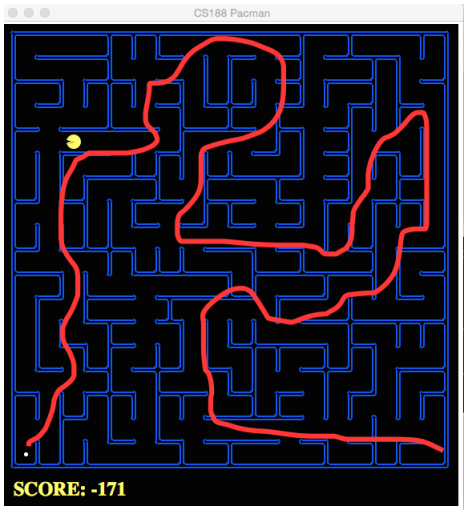

Figure 6: BFS on Big Maze

ANALYSIS: For big Maze, we see that the path is similar to the one taken by DFS. Now that is because Bigmaze does not have a lot of path choices.

3 question 3: Logical Pacman A* Search

3.1 Design

For implementing A*, we maintain the same logic as BFS but we expand the fringe in the order of $f(n) = g(n) + h(n)$. Since this is a knowledge base problem, we store the heuristic value as facts. So for each node we maintain the priority queue strucure as a list, we take out the element with the lowest cost. We expand that node, we add the cost till now and the heuristic value from that node to result which is stored as a fact. Rest of the algorithm is similar with a few implementation changes. We use the manhattan heuristic, but we can use anyone, since we take the algotithm from the python side and we add it as a fact to prolog. This maintains the generality of the algorithm.

3.2 Results

3.2.1 tinymaze A*

Figure 7: A* on TinyMaze

```
['South', 'South', 'West', 'South', 'West', 'West', 'South', 'West']
Path found with total cost of 8 in 5.9 seconds
Search nodes expanded: 0
```

```
Pacman emerges victorious! Score: 502
Average Score: 502.0
Scores:        502
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: We can see A* gives the same optimal path as BFS. That is the advantage of BFS without expanding as many nodes.

3.2.2 Medium Maze A*

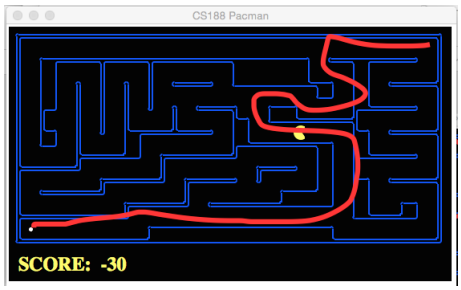

Figure 8: A* on Medium Maze

```
Path found with total cost of 68 in 0.8 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 442
Average Score: 442.0
Scores:        442
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: We see that A*, we are taking the same path as what we took in BFS but we dont expand as many nodes as BFS.

3.3 Big Maze A*

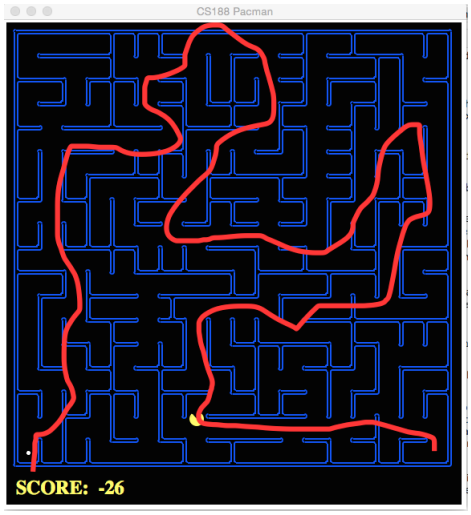

Figure 9: A* on Big Maze

```
Path found with total cost of 210 in 1.1 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 300
Average Score: 300.0
```

```
Scores:        300
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: We again see that for big maze all the algorithms perform the same. So bigmaze is a bad judge of the quality of the algorithm.

4 question 4: Logical Pacman 4 corners

4.1 Design

We keep the implentation the same as BFS but we modify it to make it handle any number of goals. In this case, the goal is all 4 corners. So until all 4 corners are traversed the BFS continues, so since we are integrating the goal into the BFS design, the underlying algorithm is still same, the solution is still optimal.

4.2 Results

4.2.1 tinymaze 4 corners

Figure 10: 4 corners on TinyMaze

```
Path found with total cost of 32 in 0.7 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 508
Average Score: 508.0
Scores:        508
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: Since our goral state is all 4 nodes beigng traversed, so it will take the shortest path to visit all 4 corners. So it is the most optimal path for visiting all 4 corners. The underlying BFS makes sure of that, only that now the goal state is visiting 4 corners

4.3 Medium Corners - Corners Problem

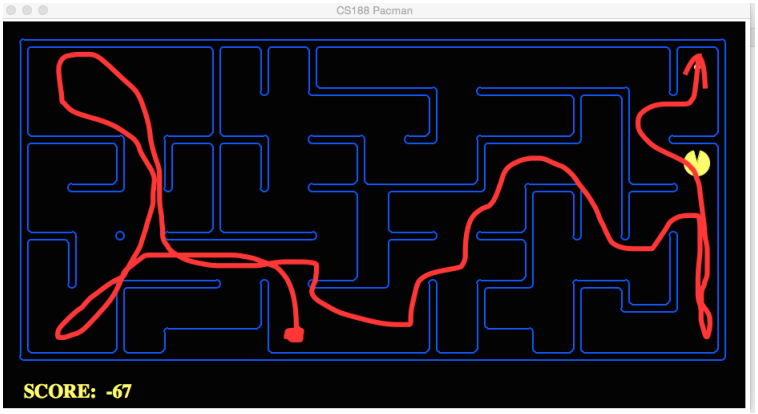

Figure 11: A* on Big Maze

```
Path found with total cost of 106 in 18.4 seconds
Search nodes expanded: 0
Pacman emerges victorious! Score: 434
Average Score: 434.0
Scores:        434
Win Rate:      1/1 (1.00)
Record:        Win
```

ANALYSIS: Again we see that since the underlying algorithm is BFS, it does really well for the 4 corners problem.

5 Conclusion

DFS	10	130	212
BFS	8	68	210
A*	8	68	210
	Tiny Maze	Medium Maze	Big Maze

As we can see from the above table BFS and A* perform the same in terms of the number of nodes expanded but due to selective expansion A* is better.

For corners problem, we can see the results, it is not anything differernt from BFS but with a more complex goal state.