CSE 537 Searching Pac-Man
Project Report 1

Anshul Anshul (anshul.anshul@stonybrook.edu)
Nitish Garg (nitish.garg@stonybrook.edu)
Ravi Chandra Sadineni(ravi.sadineni@stonybrook.edu)

September 25, 2014

Answer 1 Yes, the exploration order is what we have expected. The algorithm iteratively explores the rightmost node in every subtree (because we are inserting the elements into the stack in the order they are received from the successor function. The last element (right most of the sub tree) inserted will be popped first.) and returns to the previous level once it is not able to find any further node to expand. Then it explores the leftmost subtree.
No, the pacman doesn't visit all the explored squares. It only visits the nodes that are part of the final path to the goal state.

(a) **python pacman.py -l tinyMaze -p SearchAgent**
Search nodes Expanded : 16
Total Cost : 10
Time Taken : 0.0 sec
Average Score : 500

(b) **python pacman.py -l tinyMaze -p SearchAgent**
Search nodes Expanded : 147
Total Cost : 130
Time Taken : 0.0 sec
Average Score : 380

(c) **python pacman.py -l bigMaze -z .5 -p SearchAgent**
Search nodes Expanded : 391
Total Cost : 210
Time Taken : 0.0 sec
Average Score : 300

Answer 2 Yes, BFS finds the best cost solution from the starting point to the destination. BFS explores all the nodes in a given level before continuing to the next level. Thus the goal path involving the least cost will be explored first before other paths are expanded.

(a) **python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs**
Search nodes Expanded : 267
Total Cost : 68
Time Taken : 0.0 sec
Average Score : 442

(b) **python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5**
Search nodes Expanded : 617

Total Cost : 210
Time Taken : 0.1 sec
Average Score : 300

Answer 3 Uniform-cost search works normally in the generic *SearchAgent* case, since the cost for every move is same (1). But in the case of *StayEastSearchAgent*, the cost is very low because of the following cost function:-

```
costFn = lambda pos: .5 ** pos[0]
```

Because of cost function 0.5^{pos} :- if the agent stays East, the position will be high and the cost will be low.
The cost function in case of *StayWestSearchAgent* is:-

```
costFn = lambda pos: 2 ** pos[0]
```

Because of cost function 2^{pos} :- if the agent stays East, the position will be high and so will be the cost

(a) **python pacman.py -l mediumMaze -p SearchAgent -a fn=ucs**
Search nodes Expanded : 268
Total Cost : 68
Time Taken : 0.1 sec
Average Score : 442

(b) **python pacman.py -l mediumDottedMaze -p StayEastSearchAgent**
Search nodes Expanded : 186
Total Cost : 1
Time Taken : 0.1 sec
Average Score : 646

(c) **python pacman.py -l mediumScaryMaze -p StayWestSearchAgent**
Search nodes Expanded : 108
Total Cost : 68719479864
Time Taken : 0.1 sec
Average Score : 418

Answer 4 A^* also takes the heuristic along with the total cost into consideration. Thus, if the heuristic is admissible, A^* leads us to the solution.

(a) **python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic**
Search nodes Expanded : 538
Total Cost : 210
Time Taken : 0.2 sec
Average Score : 300

(b) **What happens on *openMaze* for the various search strategies?**
In case of DFS :
It depends on the position of the open end. If the path containing the open end is explored before the path containing the solution, then DFS wont give us a solution.

In case of BFS :
Always find the solution. Since all the nodes in the given level are expanded before expanding

the nodes in the next level. Thus the problem is complete. In fact, it might be effective than the other solutions.

In case of UCS :
Always find the solution. If the pacman gets lost in the open end, he keeps exploring and the cost increases accordingly. So, at some point the path involving the solution should be less than the cost of the path involving open edge. Thus the solution is complete.

In case of A^* :
It is also complete like UCS. But how fast it reaches the goal state depends on the heuristic.

Answer 5 Implement the *CornersProblem* search problem.
Regarding the Goal State : We maintain the visited list and check if all the four corners have been visited or not. So it doesn't use the game state of the problem.

(a) python pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
Search nodes Expanded : 260
Total Cost : 28
Time Taken : 0.0 sec
Average Score : 512

(b) python pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem
Search nodes Expanded : 1943
Total Cost : 106
Time Taken : 0.4 sec
Average Score : 434

Answer 6 Implemented *cornersHeuristic* in *CornersProblem*.
Regarding the *cornersHeuristic* admissibility:- we try to find the manhattan distance from the current position to the nearest corner. We then find the distance of the nearest corner from the current corner. We continue the above process for all the corners yet to be visited. Thus this distance is always less than the actual distance (since heuristic only represents the distance without considering walls) .

(a) python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5
Search nodes Expanded : 490
Total Cost : 106
Time Taken : 0.1 sec
Average Score : 434

(b) python pacman.py -l testSearch -p AStarFoodSearchAgent
Search nodes Expanded : 12
Total Cost : 7
Time Taken : 0.0 sec
Average Score : 513

Answer 7 Fill in *foodHeuristic* for the *FoodSearchProblem*.
This heuristic is also admissible because while finding the Minimum Spanning Tree (MST) via all the nodes, we consider the distance between two nodes without considering the walls in between.

(a) **python pacman.py -l trickySearch -p AStarFoodSearchAgent**
Search nodes Expanded : 7514
Total Cost : 62
Time Taken : 18.2 sec

Average Score : 568

Answer 8 Critical Analysis of search methods.

After analyzing several search algorithms, we understood the importance of the heuristic. Though BFS and DFS seems working fine in normal problems but when the number of nodes to be explored in a given problem grows drastically, then their performance degrades. Its important that we provide some admissible heuristic guiding the search pattern. Consider the question number 6 and 7. A normal search like BFS expanded around 2000 nodes before finding the solution whereas A* with a consistent heuristic is able to find the solution by expanding just 500 nodes. But one needs to consider the fact that path found in the case of A* might not be optimal. There are certain scenarios where only BFS or UCS can help especially when we have to find the best possible solution. Though the choice of the algorithm depend on the specific scenario, in most of the real world problem which might have high branching factor, relying on simple search algorithm without proper heuristic might be a grave mistake.