CS2180: Artificial Intelligence

Lab 1

Name: Modukuri Venkata Nagasurya

Roll Number: 112001027

Q1) Finding a Fixed Food Dot using Depth First Search:

Command for implementing tinyMaze: python2 pacman.py -l tinyMaze -p SearchAgent

Command for implementing mediumMaze: python2 pacman.py -l mediumMaze -p SearchAgent

Command for implementing bigMaze: python2 pacman.py -l bigMaze -z .5 -p SearchAgent

	Tiny Maze Search	Medium Maze Search	Big Maze Search
Path found with total cost of:	8	130	210
Search nodes expanded:	0	146	390

Q2) Breadth First Search:

Command for implementing mediumMaze: python2 pacman.py -l mediumMaze -p SearchAgent -a fn=bfs

Command for implementing bigMaze: python2 pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5

	Medium Maze Search	Big Maze Search
Path found with total cost of:	68	210
Search nodes expanded:	269	620

BFS has done 8 puzzle problem (python2 eightpuzzle.py) and BFS found a path of 7 moves: ['left', 'down', 'right', 'up', 'left', 'left', 'up']

Q3) For Uniform Cost Search:

For implementing the Medium maze using Uniform Cost Search, use the following command in the terminal: python2 pacman.py -l mediumMaze -p SearchAgent -a fn=ucs

For implementing the Medium dotted maze using Uniform Cost Search, use the following command in the terminal: python2 pacman.py -l mediumDottedMaze -p StayEastSearchAgent

For implementing the Medium scary maze using Uniform Cost Search, use the following command in the terminal: python2 pacman.py -l mediumScaryMaze -p StayWestSearchAgent

	Medium Maze	Medium Maze Search	Big Maze Search
Path found with total cost of:	68	1	68719479864
Search nodes expanded:	269	186	108

Q4) Varying the Cost Function:

For implementing the Big maze using A* Search, use the following command in the terminal: python2 pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

	Big Maze Search
Path found with total cost of:	210
Search nodes expanded:	549

Q5) Finding All the Corners

For implementing the tiny corners using BFS, use the following command in the terminal:

python2 pacman.py -l tinyCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

For implementing the medium corners using BFS, use the following command in the terminal:

python2 pacman.py -l mediumCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

	Tiny corners	Medium corners
Path found with total cost of:	28	106
Search nodes expanded:	252	1966

Q6) Corners Problem: Heuristic

Here the heuristic function taken is Manhattan distance between the state and the nearest corner + (Manhattan distance between the nearest unvisited corner chosen from state and nearest unvisited corner from the chosen nearest unvisited corner (do this till all the corners are visited)). As we have considered the Manhattan distance, the heuristic wont be negative.

Admissibility check: The actual heuristic would be the distance through the maze to all the unvisited corners whereas the heuristic defined by us is just the Manhattan distance between them (straight line between them). So, the heuristic defined is always less than or equal to the actual heuristic and hence is admissible.

Consistency check: For consistency check, we need to prove that h(n) <= cost from n to k + h(k) where n and k are state given and its adjacent state respectively. No matter where ever you take the state to be, the h(n) – h(k) will always be less than equal to 1(cost from n to k) here. Example: Let us consider that all the corners are unvisited (namely a, b, c, d). Now, let us take the n as the state and the k as one of n's neighbor (k is a neighbor of n which is on the path from n to a). Now, h(n) would be the Manhattan distance from the n to a (n's nearest corner) + distance from a to b (a's nearest unvisited corner) + distance from b to c (b's nearest unvisited corner) + distance from c to d (c's nearest unvisited corner). h(k) would be Manhattan distance from the k to a (n's nearest corner) + distance from a to b (a's nearest unvisited corner) + distance from b to c (b's nearest unvisited corner) + distance from c to d (c's nearest unvisited corner) -1. So, h(n)-h(k)=1 which satisfies the consistency check. Similarly for all the other states, the consistency check will be satisfied.

By using the above heuristic in corners problem (python pacman.py -l mediumCorners -p AStarCornersAgent -z 0.5):

Path found with total cost of 106

Search nodes expanded: 692

Q7) Eating All The Dots:

Here the heuristic we have taken is the Manhattan distance between the state and the coordinate of the unvisited food which is farthest from the state. As, we have considered the Manhattan distance, the heuristic wont be negative.

Admissibility check: The actual heuristic would be the distance through the maze to all the unvisited food. But the heuristic defined by us is the Manhattan distance between the state and its farthest unvisited food. Now, let us start from the start state, the Manhattan distance between the start state and the farthest food will always be less than or equal to actual heuristic (which is the distance from the start state to the farthest food through). Similarly, if we go to the next state, the Manhattan distance between the state and the farthest food will always be less than or equal to actual heuristic. Similarly, this is applicable for other states too. Hence, we can say that the heuristic defined by us is admissible.

Consistency check: For consistency check, we need to prove that h(n) <= cost from n to k + h(k) where n and k are state given and its adjacent state respectively. No matter where ever you take the state to be, the h(n) – h(k) will always be less than equal to 1(cost from n to k) here. Example: Let n be state and h(n) will be the Manhattan distance from n to farthest unvisited food, k be the neighbor of n and the h(k) is the Manhattan distance from the k to it's farthest unvisited food. Now, the farthest unvisited food from k can be same as that of n's. If that's the case, then h(n)-h(k) is going to be zero which is less than 1. If farthest unvisited food for k is different than that of n, it means that the farthest unvisited food from k is greater than that n's because if it less than n's, then then we can change k's farthest unvisited food same as n's because it larger than k's. Then for this case, h(n)-h(k) will be less than 0. Hence, h(n)-h(k)<=1 which ensures that heuristic defined (h) is consistent.

For implementing the testsearch, use the following command in the terminal: python2 pacman.py -l testSearch -p AStarFoodSearchAgent

For implementing the trickysearch, use the following command in the terminal: python2 pacman.py -l trickySearch -p AStarFoodSearchAgent

	testsearch	trickysearch
Path found with total cost of:	7	60
Search nodes expanded:	12	9551