Artificial Intelligence
CSE 537

Homework 4
Report

Apurva Kumar 109751733
Gaurav Gawde 110008637

Question 1: DFS Algorithm

This function makes a generic implementation of DFS. Following variables are mainly used:

- state_stack - It is the stack class imported from util class. It keeps track of the nodes to be expanded and pops the nodes depth first.
- parents - It is used to backtrack the path of the node after it reaches the goal.
- direction - This is also used to backtrack the path of the node once it reaches goal. It gives the exact direction to trace back.
- visited - This keeps track of visited nodes so as to avoid a deadlock.
- path - This is used to return the path to the main function

Statistics:

- python pacman.py -l tinyMaze -p SearchAgent

  Results:
  Path Found With Total Cost Of 8 In 0.8 Seconds
  Search Nodes Expanded:: 18
  Pacman Emerges Victorious! Score: 502
  game-run-trace 5
  after calling game run()
  Average Score: 502.0
  Scores: 502
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l mediumMaze -p SearchAgent

  Results:
  Path Found With Total Cost Of 68 In 0.8 Seconds
  Search Nodes Expanded:: 282
  Pacman Emerges Victorious! Score: 442
  Average Score: 442.0
  Scores: 442
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l bigMaze -z .5 -p SearchAgent

  Results:
  Path Found With Total Cost Of 210 In 0.8 Seconds:
  Search Nodes Expanded:: 648
  Pacman Emerges Victorious! Score: 300
  Average Score: 300.0
  Scores: 300
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l tinyMaze -p SearchAgent -a fn=dfs

  Results:
  Path Found With Total Cost Of 8 In 0.8 Seconds::
  Search Nodes Expanded:: 18
  Pacman Emerges Victorious! Score: 502
  Average Score: 502.0
  Scores: 502
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l mediumMaze -p SearchAgent -a fn=dfs

  Results:
  Path Found With Total Cost Of 68 In 0.8 Seconds::
  Search Nodes Expanded:: 282
  game-run-trace 5
  Pacman Emerges Victorious! Score: 442
  game-run-trace 5
  after calling game run()
  Average Score: 442.0
  Scores: 442
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l bigMaze -p SearchAgent -a fn=dfs -z .5

  Results:
  Path Found With Total Cost Of 210 In 0.8 Seconds::
  Search Nodes Expanded:: 648
  Pacman Emerges Victorious! Score: 300
  game-run-trace 5
  after calling game run()
  Average Score: 300.0
  Scores: 300
  Win Rate: 1/1 (1.00)

Record: Win

Question 2: BFS Algorithm

This function makes a generic implementation of BFS. Following variables are mainly used:

- state_queue - It is the queue class imported from util class. It keeps track of the nodes to be expanded and pops the nodes breadth first.
- parents - It is used to backtrack the path of the node after it reaches the goal.
- direction - This is also used to backtrack the path of the node once it reaches goal. It gives the exact direction to trace back.
- visited - This keeps track of visited nodes so as to avoid a deadlock.
- path - This is used to return the path to the main function

Statistics:

- python pacman.py -l tinyMaze -p SearchAgent -a fn=bfs

  Results:
  Path Found With Total Cost Of 8 In 0.8 Seconds::
  Search Nodes Expanded:: 18
  Pacman Emerges Victorious! Score: 502
  game-run-trace 5
  after calling game run()
  Average Score: 502.0
  Scores: 502
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l mediumMaze -p SearchAgent -a fn=bfs

  Results:
  Path Found With Total Cost Of 68 In 0.8 Seconds::
  Search Nodes Expanded:: 282
  Pacman Emerges Victorious! Score: 442
  game-run-trace 5
  after calling game run()
  Average Score: 442.0
  Scores: 442
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l bigMaze -p SearchAgent -a fn=bfs -z .5

  Results:
  Path Found With Total Cost Of 210 In 0.8 Seconds::
  Search Nodes Expanded:: 648

Pacman Emerges Victorious! Score: 300
Average Score: 300.0
Scores: 300
Win Rate: 1/1 (1.00)
Record: Win

Question 3: A* Algorithm

This function makes a generic implementation of A*. Manhattan distance is used as the heuristic for distance calculation. Following variables are mainly used:

- state_queue - It is the queue class imported from util class. It keeps track of the nodes to be expanded and pops the nodes breadth first.
- parents - It is used to backtrack the path of the node after it reaches the goal.
- direction - This is also used to backtrack the path of the node once it reaches goal. It gives the exact direction to trace back.
- visited - This keeps track of visited nodes so as to avoid a deadlock.
- path - This is used to return the path to the main function

Statistics:

- python pacman.py -l tinyMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

  Results:
  Path Found With Total Cost Of 8 In 0.6 Seconds::
  Search Nodes Expanded:: 18
  Pacman Emerges Victorious! Score: 502
  Average Score: 502.0
  Scores: 502
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l mediumMaze -z .5 -p SearchAgent -a fn=astar,heuristic=manhattanHeuristic

  Results:
  Path Found With Total Cost Of 68 In 0.6 Seconds::
  Search Nodes Expanded:: 282
  Pacman Emerges Victorious! Score: 442
  game-run-trace 5
  after calling game run()
  Average Score: 442.0
  Scores: 442
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l bigMaze -z .5 -p SearchAgent -a fn=astar, heuristic=manhattanHeuristic

  Results:
  Path Found With Total Cost Of 210 In 0.6 Seconds::
  Search Nodes Expanded:: 648
  Pacman Emerges Victorious! Score: 300
  game-run-trace 5
  after calling game run()
  Average Score: 300.0
  Scores: 300
  Win Rate: 1/1 (1.00)
  Record: Win

Question 4: Corner Problem

We are using BFS algorithm to find the shortest path that goes through all the corners of a maze.From python code we call BFS prolog file which gives the list of directions to be taken by pacman agent.

Statistics:

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

  Results:
  Path found with total cost of 28 in 0.0 seconds
  Search nodes expanded: 253
  Pacman Emerges Victorious! Score: 512
  Average Score: 512.0
  Scores: 512
  Win Rate: 1/1 (1.00)
  Record: Win

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

  Results:
  Path found with total cost of 106 in 0.0 seconds
  Search nodes expanded: 1967
  Pacman Emerges Victorious! Score: 434
  Average Score: 434.0
  Scores: 434
  Win Rate: 1/1 (1.00)
  Record: Win

- python pacman.py -l bigCorners -p SearchAgent -a fn=bfs,prob=CornersProblem

  Results:
  Path found with total cost of 162 in 0.1 seconds
  Search nodes expanded: 7950
  Pacman Emerges Victorious! Score: 378
  Average Score: 378.0
  Scores: 378
  Win Rate: 1/1 (1.00)
  Record: Win

Critical Analysis and Conclusion:

Thus according to the expanded nodes and time taken by the algorithm A* algorithm works better than BFS and DFS algorithm.