INDIAN INSTITUTE OF TECHNOLOGY ROPAR

ARTIFICIAL INTELLIGENCE : CSL512

Pacman Search

Submitted To:
Shashi Shekhar Jha
Computer Science
Department

Submitted By :
Siddharth Nahar
2016CSB1043

Contents

1 DFS Search

Depth First Search algorithm searches the deepest node first.It doesn't gives least cost solution always but it saves lot of space, Space-Complexity $\bigcirc(bm)$ .Time-Complexity $\bigcirc(b^m)$.Used Stack for implementation.

1.1 Is the Exploration order Expected ?

YES, exploration order is what I have Expected(using Stack order.).DFS get's to goal following a different path than optimal because that was the node that got pushed later. So DFS follows that order.As we know Stack is LIFO so order in which nodes got explored was known.Also we have pushed successors in preference order of N,S,E,W.

1.2 Does Pacman goes to all Explored Square ?

NO, It can't go to all explored squares. If Pacman gets to dead end or visits previously visited node it returns back, but it has explored that node but doesn't traverse it to reach goal.

1.3 Does DFS gives least cost solution for mediumMaze ?

NO, DFS doesn't gives least cost solution to mediumMaze.

Suppose there are two paths to reach to goal. DFS will always choose the path whose direction was pushed later.In MediumMaze path to reach to optimal has to be South directional successor at node, but it got pushed before East directional successor. So East directional path reach the goal which was not optimal.DFS highly depends in which order node's successors are pushed.

1.4 Results :

- TinyMaze :- Search Nodes Expanded : 15, Cost : 10
- MediumMaze :- Search Nodes Expanded : 146, Cost : 130
- BigMaze :- Search Nodes Expanded : 390, Cost : 210

2 BFS Search

BFS search algorithm searches all path of d length before d+1 length. Space-Complexity $\bigcirc(b^d)$. Time-Complexity $\bigcirc(b^d)$. Used Queue for implementation.

2.1 Does BFS gives least cost solution ?

YES, In given scenario cost of each step is 1, So (cost α depth(n)) therefore BFS will always Optimal Solution. By definition BFS searches shortest path first So it will always return optimal solution. Number of Nodes Expanded will be more as it tries all path of length 1,2..d.

2.2 Results :

- MediumMaze : Search Nodes Expanded : 269, Cost : 68.
- BigMaze : Search Nodes Expanded : 620, Cost : 210.
- EightPuzzle : Gives minimum steps to solve puzzle.

3 Uniform Cost Search

UCS search algorithm works similar to BFS but instead of selecting shallowest node it selects node with minimum cost. Used Priority Queue for implementation. BFS gives minimum path length solution but UCS gives minimum Cost Solution.

Suppose we have to travel from city A to city B. There are two path A-B and A-C-B. A-B path is all wrecked so travelling it reduces efficiency. BFS will provide A-B path but UCS takes cost in consideration so will give alternate solution.

3.1 Results :

- MediumMaze : Search Nodes Expanded : 269, Cost : 68.
- MediumDottedMaze : Search Nodes Expanded : 186, Cost : 1.
- MediumScaryMaze : Search Nodes Expanded : 108, Cost : 68719479864.

4 A* Search

A* search algorithm is Informed Search Algorithm. UCS algorithm takes consideration of cost of path but doesn't take consideration how far it is from goal State. A* defines heurestic which estimates how long it is from goal State. Used Priority Queue for implementation.

4.1 Results :

- UCS : Search Nodes Expanded : 620, Cost : 210.
  A* : Search Nodes Expanded : 549, Cost : 210. for BigMaze.
- OpenMaze :
  - DFS : Search Nodes Expanded : 806, Cost : 298
  - BFS : Search Nodes Expanded : 682, Cost : 54
  - UCS : Search Nodes Expanded : 682, Cost : 54
  - A* : Search Nodes Expanded : 535, Cost : 54

  We can see that DFS has cost much higher than others Showing it doesn't take cost in consideration. A* has many less nodes expanded with respect to UCS due to manhattan heuristic used.It saves A* to look for other unnecessary path that take them away from goal state.

4.2 Overall Evaluation of all Search Methods :

- DFS may have small space but it is not optimal for finding solution. As we can see for OpenMaze problem DFS has given solution but with a very high cost.
- BFS gives minimum number of nodes path as solution.It may be optimal if (cost α depth(n)). It is complete always end to solution.
- UCS always gives minimum cost solution.It is uninformed search it doesn't take into consideration how far it is from goal state.
- A* gives optimal Solution takes into consideration heuristic and cost. Optimality depends on what heurestic we choose.

5 Finding All Corners Problem

We have to design corners problem, such that Pacman visits to all corners atleast once. Design mainly points towards State Representation and goal State.

- State Representation : (position, visitedCorners)
- Goal State : Whether all corners are visited or not.

5.1 Results :

- TinyCorners :- Search Nodes Expanded : 435, Cost : 28
- MediumCorners :- Search Nodes Expanded : 2448, Cost : 106

6 Corners Heuristic :

I have implemented heuristic as maximum of maze distance from current position to all corners.

$$MazeDistance = BFS(position, corner)$$

$$h(n) = \max(BFS(position, unvisitedcorners))$$

Admissiblity :

$$h(n) \leq h^*(n)$$

We know minimum path travelled from any node to goal state is BFS path else it will be sum from node to corner paths which obviously will be more than node to single goal state. So $h(n)$ is admissible heuristic.

Consistency :

$$h(n) \leq c(n, n^{'}) + h(n^{'})$$

We know :

$$c(n, n^{'}) \geq 1$$

$$h(n^{'}) = (h(n), h(n) + 1, h(n) - 1)$$

Moving from node to it's successor travels a unit length So maximum change possible is 1. So combining above two equation we get :

$$h(n) \leq c(n, n^{'}) + h(n^{'})$$

6.1 Results :

- MediumCorners :- Search nodes Expanded : 978, Cost : 106
- I have tried with using above heurestic with manHattan Distance Expanded nodes comes around 1150. mazeDistance is more accurate than manHattan Distance So gives more optimal cost.

7 Eating All Food Heuristic :

We have create heuristic to optimal to eat all food.I have used similar corners heuristic same as above but optimized algorithm for not doing repetitive tasks.

Algorithm :

- Initially calculate path from starting position to all food points.
- Create dictionary for all food points and stores corresponding position and path.
- Add its all successor for all food point, we know from current to successor either it changes path by if it follows same direction or opposite.So we change current position path accordingly.
- We don't recompute path each time we derive it from previous.So it saves lot of time.

This Heuristic is admissible and Consistent as shown in previous section.

7.1 Results :

- TrickySearch :-
  - Nodes Expanded = 4069
  - Time Taken : 2.0 secs
  - Cost : 60

8 SubOptimal Search :

This search uses Greedy Strategy and eats closest dot.As asked for anyfood problem eat the closest one I have specified goal state for AnyFoodProblem and call relevent search method.

- AnyFoodSearchProblem Goal State : Check if this state is food or not.
- Search Method for ClosestDistance : BFS or UCS both will give correct solution, As problem provides path cost as 1.
- BigSearch :- Cost : 350

8.1 Example for failing Greedy Strategy :

Consider a Graph below to visit all A,B and C vertices starting from S:

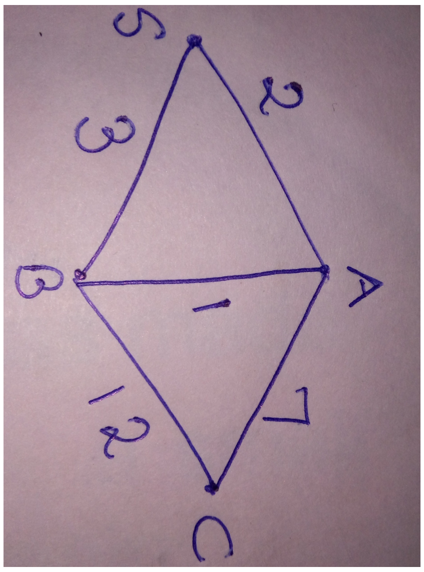

Figure 1: Caption

- Optimal Path S-B-A-C Cost= 11
- Path by algorithm S-A-B-C , Cost = 15
- Greedy Search doesn't always gives minimum Cost Solution.
- Greedy Search doesn't consider what future choices can be So can't be optimal always.