BFS DFS and UCS

(Project1)

Nguyen Quoc Huy

23020082

CS 188

Method

Research Design

Objective: Understand the functionality of three Uninformed Searches including Depth-First Search, Breadth-First Search, and Uniform Cost Search.

Approach: Analyzing completeness, optimality, time complexity, and space complexity.

Results

Depth-First Search (DFS)

Completeness: If there are cycles in the state space graph, the corresponding search tree will be infinite in depth. In such cases, DFS will search for the deepest node in an infinite-sized tree, making it unable to find a solution.

Optimality: DFS finds a solution without considering path costs, so it is not optimal.

Time complexity: In the worst case, DFS may explore all entire nodes in the search tree. Hence, given a tree with maximum depth m, the runtime of DFS is $O(b^m)$.

Space complexity: In the worst-case scenario, DFS maintains b nodes at each level of depth. Once b children exist on the frontier, it implies that some parents are enqueued. Therefore, the space complexity of DFS is O(bm).

Breadth-First Search (BFS)

Completeness: The completeness of BFS is guaranteed by the finite depth of the shallowest node, ensuring the existence of a solution.

Optimality: BFS does not consider costs when determining so it is not optimal.

Time complexity: In the worst case, we go through all nodes at every depth from 1 to s. Hence, the time complexity is $O(b^s)$.

Space complexity: In the worst-case scenario, DFS maintains all the nodes in the level corresponding to the shallowest solution. Hence, there are $O(b^s)$ nodes at this depth.

Uniform Cost Search (UCS)

Completeness: It must have some finite length shortest path to guarantee the solution exists. Hence, UCS is complete.

Optimality: When all edge costs are nonnegative, our approach involves expanding our frontier by removing a node with the lowest cost. Consequently, it ensures that the path to a goal state is guaranteed to have the lowest cost.

Time complexity: Assuming the optimal path cost is C and the minimal cost between two nodes in the state space graph is e, we roughly explore all nodes at depths ranging from 1 to C / e. Hence, the time complexity is $O(b^{C/e})$.

Space complexity: Our frontier contains all nodes at the level of the cheapest solution, so the space complexity of UCS is $O(b^{C/e})$.

Project1

I implemented three search algorithms in file search.py. These pass all tests successfully.

Discussion

About when we should mark a node as visited in code:

- With DFS: it is required to make the path go deep. Hence, we do not instantly mark a node as visited when pushing it to the stack. We mark a node as visited when updating the frontier from it.

- With BDS: it always selects the shallowest frontier node from the start node for expansion. We need to guarantee the path to all children of a parent node represents the same level of depth. Hence, we mark a node as visited instantly when enqueue it.

- With UCS: the strategy for exploration is selecting the lowest cost frontier node from the start code for expansion. We need to update some paths to a node until that node is removed from the frontier. Hence, we mark a node as visited when updating the frontier from it.

References

CS188. *Uninformed Search*. Fall 2018.