Maximillian Hennessey (mh1285)

02/01/22

CS730

1. I feel the most important decision I made in my design is using a hash table for the closed list to check if my searches had made any duplicates. Using a hash table allowed quick (O(1)) look ups for each node. If I had used a different data structure, most basic ones would increase the overall search time exponentially.
2. In this problem the states are the cells (walls not included) that can have dirt or not. They also can have the agent or not. So, cell-count * 2^cell-count
3. Uniform:
    - Time = b^d (b = #branches, in this case 5) (d = solution depth)
    - Space = b^d
    - Admissible? Yes if all action costs are 1.

    DFS:
    - Time = b^m (b = #branches, in this case 5) (m = max depth)
    - Space = b * m
    - Admissible? No.
4. Uniform:
    - Time = b^d. We first expand our root node, then the roots successors are expanded, then their successors are expanded and so on. This leads to b^m, however this search method stops when a solution is reached, so we may stop the search at solution depth (d) b^d.
    - Space = b^d. We continuously search every successor of a successor. We need to keep all these in memory as we don't know where the solution is and need to create a path of pointers back to the root from the solution.
    - Uniform is admissible if costs for actions are 1. We explore the nodes by their depth. First we search the root at depth 1, then its successors at depth 2, and then their successors at depth 3 and so on. This ensures that we are at the lowest depth possible for the first solution as all costs are 1.

    DFS:
    - Time = b^m. DFS searches for the deepest node, then goes back up and finds the next deepest and so on. Therefore the time complexity will be b^m/
    - Space = b * m. DFS space grows linearly. We search a nodes branch till we find the deepest (m) then as we move back up we forget what we already explored.
    - DFS does not return the optimal solution because it returns the first solution, this solution could be from the deepest node.

5. It is made to seem like DFS can't have a closed list, but that implementation works. Maybe I misinterpreted something but I was confused at first as to how implement DFS.