Student Name: Rakishev Sanzhar
ID: 20248016

I. Introduction:

In this project, we implemented both Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms to enable a vacuum cleaner to efficiently clean all dirt from a simulated environment. The main objectives were to ensure that the vacuum cleaner can reach and clean all dirt positions from any starting location within the grid while utilizing BFS and DFS algorithms effectively. The program is designed to be efficient.

II. Task Description:

The vacuum cleaner task involves navigating a grid-based environment to clean dirt from designated squares. The environment consists of a grid with cells, some of which may contain dirt. The vacuum cleaner's goal is to visit each cell with dirt and clean it while minimizing the number of moves.

III. Algorithm Implementation:

Breadth-First Search (BFS):

BFS is an algorithm that explores the neighbor nodes at the present depth prior to moving on to nodes at the next depth level. In the context of the vacuum cleaner task:

- *Traversal Strategy:*
  - BFS explores nodes level by level, starting from the initial node.
  - It explores all the nearest neighbors before moving on to the next level of neighbors.
  - This results in a breadth-first traversal, meaning it explores all the neighboring nodes before going deeper.
- *Implementation:*
  - The BFS_actuators method uses a queue to store nodes to be explored.
  - It dequeues a node from the queue, explores its neighbors, and continues until it finds the solution or exhausts all possibilities.
- *Pros:*
  - Guaranteed to find the shortest path to the goal in an unweighted graph.
  - Less likely to get stuck in infinite loops compared to DFS.
- *Cons:*
  - Requires more memory compared to DFS as it needs to store information about all nodes at the current level.

- May take longer to find a solution if the goal is far away from the start node.

Depth-First Search (DFS):

DFS is an algorithm used for traversing or searching tree or graph data structures. It starts at the root node and explores as far as possible along each branch before backtracking. In the context of the vacuum cleaner task:

- *Traversal Strategy:*
  - DFS explores each branch of the grid until it reaches a dead-end, then it backtracks to the nearest unexplored node and continues.
  - This results in a deep traversal, meaning it explores a path all the way to the end before exploring another path.
- *Implementation:*
  - The DFS_actuators method uses a stack to store nodes to be explored.
  - It pops a node from the stack, explores its neighbors, and continues until it finds the solution or exhausts all possibilities.
- *Pros:*
  - Requires less memory compared to BFS as it only needs to store information about the current path.
  - May find a solution faster if the goal is closer to the start node.
- *Cons:*
  - It may get stuck in infinite loops if there are cycles in the graph.
  - Not guaranteed to find the shortest path to the goal.

IV. Comparison:

- *Completeness:* Both DFS and BFS algorithms are complete for the vacuum cleaner task, ensuring that all squares with dirt are visited and cleaned.
- *Optimality:* BFS is guaranteed to find the shortest path to each square with dirt, making it optimal for this task. DFS, on the other hand, may find longer paths before finding the shortest one, making it non-optimal.
- *Space Complexity:* DFS typically requires less memory compared to BFS because it only needs to store a single path in memory at a time. BFS, however, needs to store all paths at the current depth level, leading to higher memory usage.
- *Time Complexity:* In terms of time complexity, both DFS and BFS have a worst-case time complexity of O(V + E), where V is the number of vertices (cells) and E is the number of edges (connections between cells). However, BFS may be more time-consuming due to its need to explore all possible paths at each depth level.

V. Challenges Faced:

- *Algorithm Selection:* Choosing between BFS and DFS required careful consideration of the task's requirements. BFS guarantees optimality but may consume more memory, while DFS is memory-efficient but not guaranteed to find the shortest path.
- *Integration with Pygame:* Integrating the algorithms with Pygame for visualization posed challenges in synchronizing the cleaner's movements with the graphical representation of the environment.

VI. Conclusion:

The program helps the vacuum cleaner clean the room effectively using both BFS and DFS methods. By implementing Pygame we can see the vacuum cleaner moving around and cleaning. Both DFS and BFS are viable traversal algorithms for the vacuum cleaner task, each with its own advantages and limitations. While BFS guarantees optimality and completeness, DFS may offer better memory efficiency in certain scenarios. The choice between DFS and BFS depends on factors such as the structure of the graph, the location of the goal, and the available computational resources.

Also, I provided the code files *(BFS_20248016.py, DFS_20248016.py and BFS_20248016_fastest.py)*. The code has comments to explain how everything works.