Task 1: Graph Traversal

For this task, I modeled the graph given in the instructions (nodes A through F) and ran the three algorithms to find the path from Start (A) to Goal (F).

Results:

BFS:

Strategy: BFS

Open Nodes: 7

Closed Nodes: 6

A C F

DFS:

Strategy: DFS

Open Nodes: 4

Closed Nodes: 3

A C F

UCS:

Strategy: UCS

Open Nodes: 7

Closed Nodes: 6

A C F

My Observations:

- All three algorithms actually found the same path (A -> C -> F), which has a length of 2.
- DFS was actually the most efficient here. It only looked at 3 nodes, while BFS and UCS looked at 6.

- I think DFS just got lucky. Because of the order the children were added, it went down the 'C' branch first and hit the goal immediately. If the goal had been down the 'B' branch, DFS might have taken longer. BFS and UCS checked everything to make sure they had the shortest path.

---

3. Task 2: Robot Cleaning Trash

The robot starts at 'R' and has to find the trash 'T' while avoiding walls '#'.

I ran the algorithms on the provided grid string.

Results:

BFS:

Strategy: BFS

Open Nodes: 22

Closed Nodes: 20

Actions: right, right, right, right, down, down, right

DFS:

Strategy: DFS

Open Nodes: 16

Closed Nodes: 10

Actions: right, right, right, right, right, down, left, down, right

UCS:

Strategy: UCS

Open Nodes: 24

Closed Nodes: 21

Actions: right, right, right, right, right, down, down

My Observations:

- BFS and UCS found the best path (7 steps). DFS found a path that was 9 steps long.
- DFS tends to just keep walking in one direction until it hits a wall. You can see in my results that DFS went too far right, then had to backtrack (down, left) to get to the trash.
- Even though DFS gave a worse path, it explored fewer nodes (10) than BFS (20). So it was technically "faster" computation-wise, but the result wasn't as good

4. Task 3: Designing Heuristics

For the last task, I had to write functions that *guess* how close we are to the goal without actually running the search.

Heuristic for Task 1 (The Graph)

Since the graph doesn't have X/Y coordinates, I couldn't use a math formula. Instead, I looked at the graph structure and made a simple lookup logic (basically a table) representing the shortest number of hops to node 'F'.

- **Logic:** h(n) = minimum edges to F.
- **Example:** If we are at A, we are 2 hops away, so return 2. If we are at C, return 1.

Heuristic for Task 2 (The Robot)

For the grid, I used the Manhattan Distance. This is basically counting how many grid blocks (rows + cols) are between the robot and the trash.

- **Logic:** `abs(robot_x - trash_x) + abs(robot_y - trash_y)`
- **Handling Multiple Trash:** My function calculates the distance to the **nearest** piece of trash.
- This is fast to calculate. It ignores walls, so it’s just an estimate, but it’s admissible because it never thinks the goal is closer than it actually is.

Example Calculation:

If the robot is at (1,1) and trash is at (1,5):

- |1 - 1| + |1 - 5| = 0 + 4 = 4.
- So the estimated cost is 4 steps.