Comparative Analysis: Time Complexity of BFS, DFS, UCS

1. BFS

- Explores all nodes at the current level before moving deeper.
- **Complexity:** Exponential in the number of moves required to reach the goal. For hard puzzles, the number of open/closed nodes can be extremely high.
- **Example:**
    - Easy 8-puzzle: Open Nodes: 9686, Closed Nodes: 6345, Steps: 14
    - Hard 8-puzzle: Open Nodes: 241,921, Closed Nodes: 241,918, Steps: 31
    - 15-puzzle (almost solved): Open Nodes: 9, Closed Nodes: 4, Steps: 1

2. DFS

- Explores as deep as possible along each branch before backtracking.
- **Complexity:** Can be much lower for easy cases, but may get stuck in deep or infinite branches for hard cases. Not guaranteed to find the shortest path.
- **Example:**
    - Solved 8-puzzle: Open Nodes: 1, Closed Nodes: 1, Steps: 0
    - 15-puzzle (solved): Open Nodes: 1, Closed Nodes: 1, Steps: 0

3. UCS

- Explores the least cost nodes first .
- **Complexity:** Similar to BFS for N-puzzle since all moves have cost 1 but can be more efficient if costs vary.
- **Example:**
    - Medium 8-puzzle: Open Nodes: 6848, Closed Nodes: 4391, Steps: 14
    - Hard 8-puzzle: Open Nodes: 75,672, Closed Nodes: 54,549, Steps: 20
    - 15-puzzle (solved): Open Nodes: 1, Closed Nodes: 1, Steps: 0

Reasons for Complexity Differences

- The number of possible moves from each state increases with puzzle size, causing exponential growth in explored nodes.

- If the start state is close to the goal, all algorithms perform efficiently. If far, BFS/UCS must explore many states.
- DFS may find a solution quickly if lucky, but is not optimal and can get stuck in deep, irrelevant branches.
- BFS and UCS guarantee the shortest solution for unit cost at the cost of high memory and time usage.