**Breadth First Search (BFS):**

BFS explores the search space level by level. It starts from the initial state and explores all possible moves from that state before moving on to the next level of states. BFS guarantees that the shortest path to a solution is found, as it explores all possible paths in order of their distance from the initial state.

**Depth First Search (DFS):**

DFS explores as far as possible along a branch before backtracking. It goes deep into a path and explores all possible moves before backtracking to explore other paths. DFS may find a solution faster if the solution is located deep in the search space, but it doesn't guarantee the shortest path.

In the context of the sliding puzzle problem (such as the grid you provided), where you're trying to move the blank space to reach the target configuration, BFS is likely to be faster and more reliable in finding the shortest solution. This is because BFS systematically explores all possible paths layer by layer, ensuring that the solution with the fewest steps is found.

Example of grid

Initial grid =

3  2  1

4  5  6

8  7  B

Target grid=

1  2  3

4  5  6

7  8  B

In this example, it would take fewer steps to reach the target grid by using BFS. BFS would explore states level by level, and since the target is reachable with a small number of steps, BFS would quickly find the solution. DFS might find a solution as well, but it might need to explore deeper paths before finding the solution, which could result in more steps.

In cases where the solution is located deep within the search space and the search space is large, DFS might perform better in terms of memory usage as it only needs to store the current path. However, for finding the shortest path, BFS is generally more reliable.

In conclusion, BFS is generally faster and more suitable for finding the shortest path to a solution in cases where the solution is reachable within a reasonable number of steps. DFS might be more suitable when memory usage is a concern or when the solution is located deep in the search space, but it might not guarantee the shortest path.