CS571:Artificial Intelligence Lab-1

1)
Breadth-First Search (BFS) consistently finds the shortest path to the goal.
Depth-First Search (DFS) might initially explore longer routes, especially when the goal is closer to a leaf node.
In situations where reaching the **goal isn't feasible**, both methods exhaustively search through **all potential options**.
BFS is adept at efficient paths and managing low-cost actions, while **DFS** excels when the goal is situated nearer to a leaf node and cost considerations are less significant.

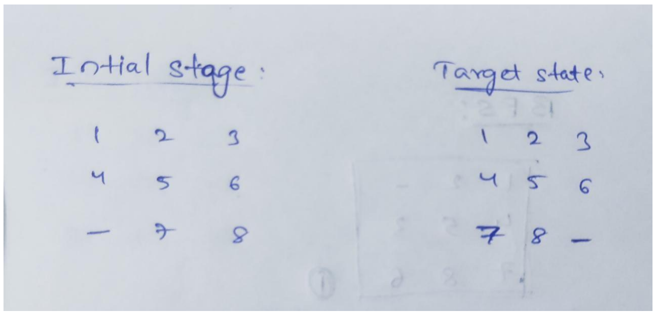
2)
Comparing BFS and DFS: Speed and Scenarios The choice between Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms is not universal; it depends on the specific problem and the relative positions of the source and target nodes in the search space. Both BFS and DFS have their strengths and weaknesses, making them more suitable for different scenarios.

BFS: Faster When Target is Nearby:

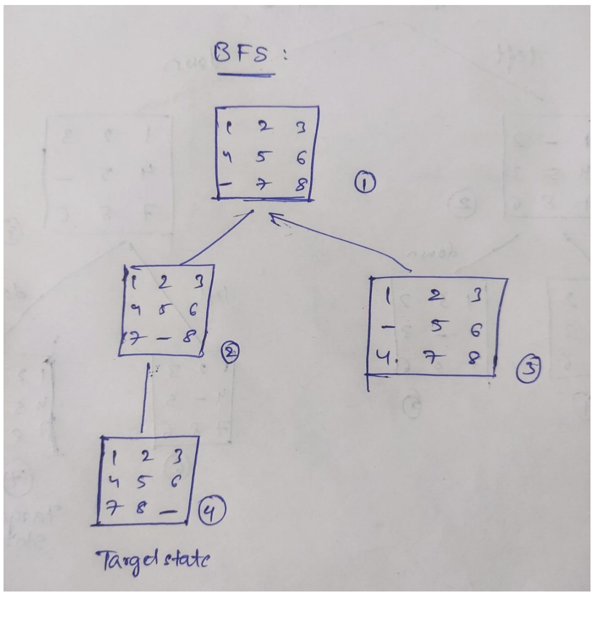
BFS is often faster when the target node is relatively close to the source node. This is because BFS systematically explores neighbouring nodes before moving to nodes farther away. In situations where the minimum number of steps (cost) required to reach the target is relatively low, BFS tends to perform exceptionally well. It guarantees that the first instance of the target node encountered will be the shortest path to it.

Example:

Initial state to target state:



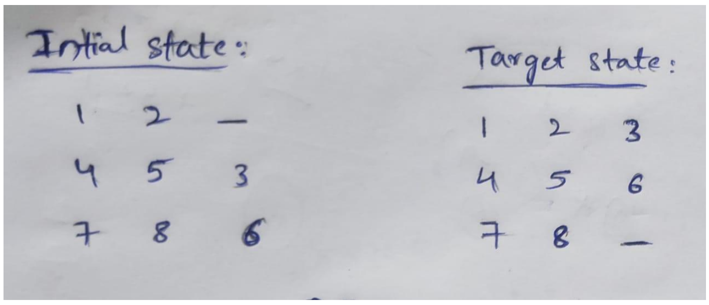
Respective BFS of the search:



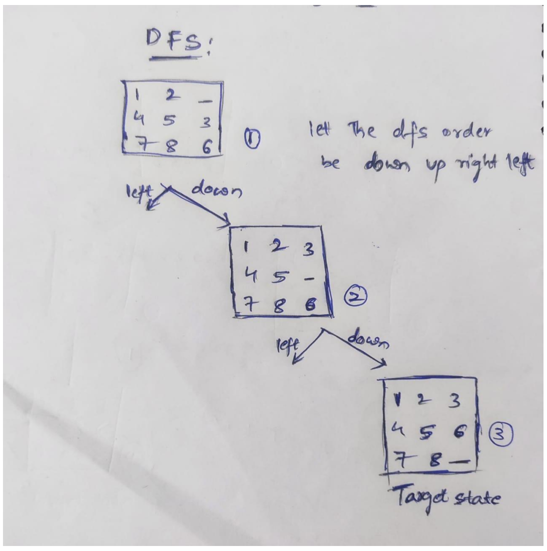
In this example, achieving the goal requires a minimum of 3 steps. Consequently, BFS will traverse 8 iterations, covering various states in its search process.
However, in the identical scenario using DFS, the algorithm will navigate through over a thousand states (the exact count can fluctuate based on DFS traversal sequence).

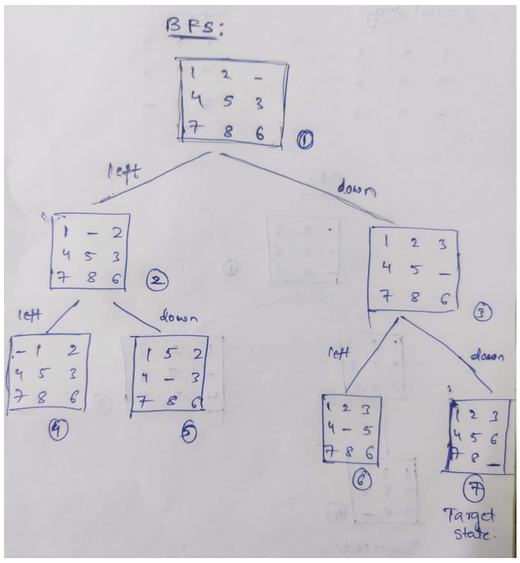
DFS: Faster When Target is Near a Leaf Node:
DFS can be faster when the target node is situated closer to a leaf node in the search tree. DFS explores a single path as deeply as possible before backtracking. If the target node is coincidentally along this path, DFS can find it quickly.

Example:
Initial state to target state:



Respective DFS and BFS of the search:

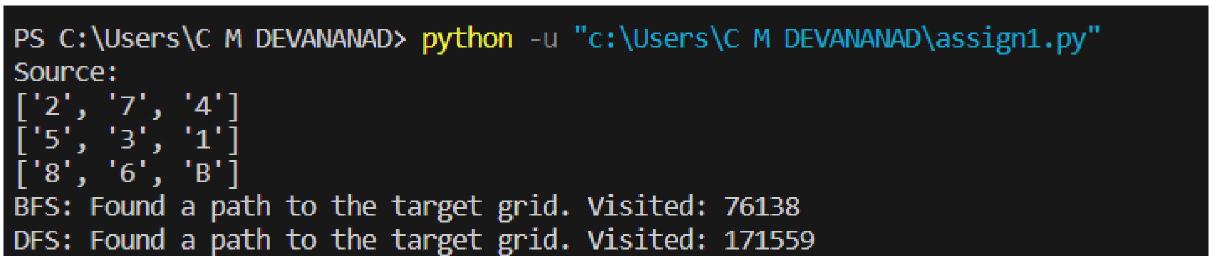




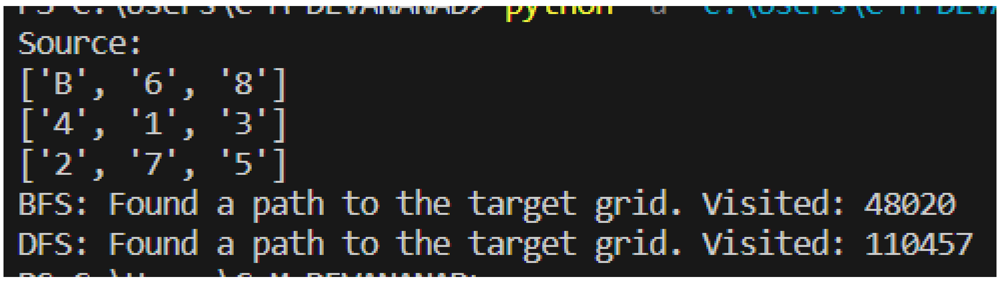
Consider another instance with a path cost of 2.Above example illustrates the DFS traversal for this case. Remarkably, due to the fortunate positioning of our target on the closest leaf node, we attain the goal with minimal effort – merely 2 steps suffice to reach the target layout. In contrast, BFS requires a traversal of 7 states before reaching the desired target arrangement.

Conclusion:
While DFS might occasionally be faster in scenarios where the target is close to a leaf node.BFS excels when the target node is nearby and can be reached with a minimal number of steps.
So,BFS generally outperforms DFS in most cases due to its systematic exploration of all possibilities. The efficiency of both algorithms can also be influenced by factors such as the branching factor of the search space and the order in which nodes are explored.

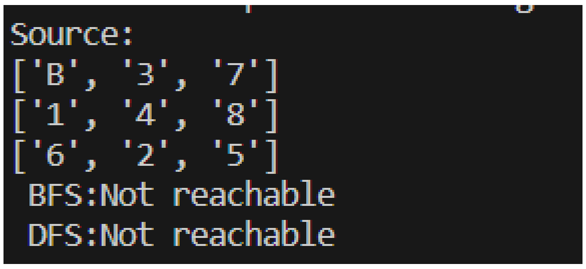
Code Execution:



```
PS C:\Users\C M DEVANANAD> python -u "c:\Users\C M DEVANANAD\assign1.py"
Source:
['2', '7', '4']
['5', '3', '1']
['8', '6', 'B']
BFS: Found a path to the target grid. Visited: 76138
DFS: Found a path to the target grid. Visited: 171559
```



```
Source:
['B', '6', '8']
['4', '1', '3']
['2', '7', '5']
BFS: Found a path to the target grid. Visited: 48020
DFS: Found a path to the target grid. Visited: 110457
```



```
Source:
['B', '3', '7']
['1', '4', '8']
['6', '2', '5']
 BFS:Not reachable
 DFS:Not reachable
```

Team Members:
1.Teja Vardhan 2001cs26
2.Gnaneshwar 2001cs15
3.Devanand 2001cs19