18/10/24

* Vaccum cleaner

```
Function vaccum_world():
    Initialize goal_state as {"A":'0', 'B':'0'}
    initialize cost as 0

    Get location_input from user
    Get status_input for location_input from user
    Get other_location based on location_input
    Get status_input_complement for other location from user

PRINT initial state of goal_state

Function clean(location):
        Update goal_state(location) to '0'
        Increment cost by 1
        Print cleaned status and current cost

for each location in [location_input, other_location]:
    IF location is Dirty:
            print that the location is Dirty
            Call clean(location)
            If moving to the other location:
                    increment cost by 1 for movement
                    print movement cost

Print final goal_state
Print performance measurement(cost)

Call Vaccum_world()
```

18/10/24

Bafna Gold
Date: Page:

* 8 Puzzle game

→ BFS Algorithm

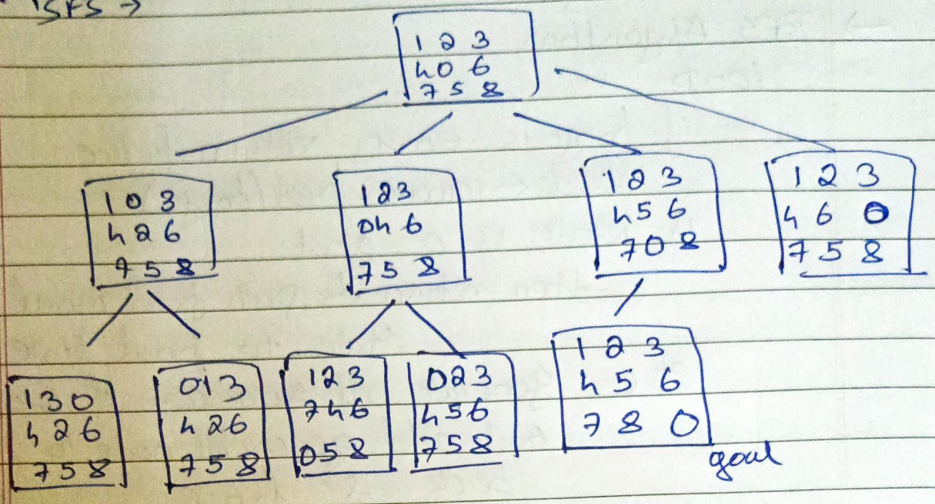
Loop
if fringe is empty return failure
Node ← remove-first(fringe)
if Node is a goal
then return the path from initial state to final state
else generate all successors of Node
And add generated node to the back of fringe
End Loop

→ DFS Algorithm

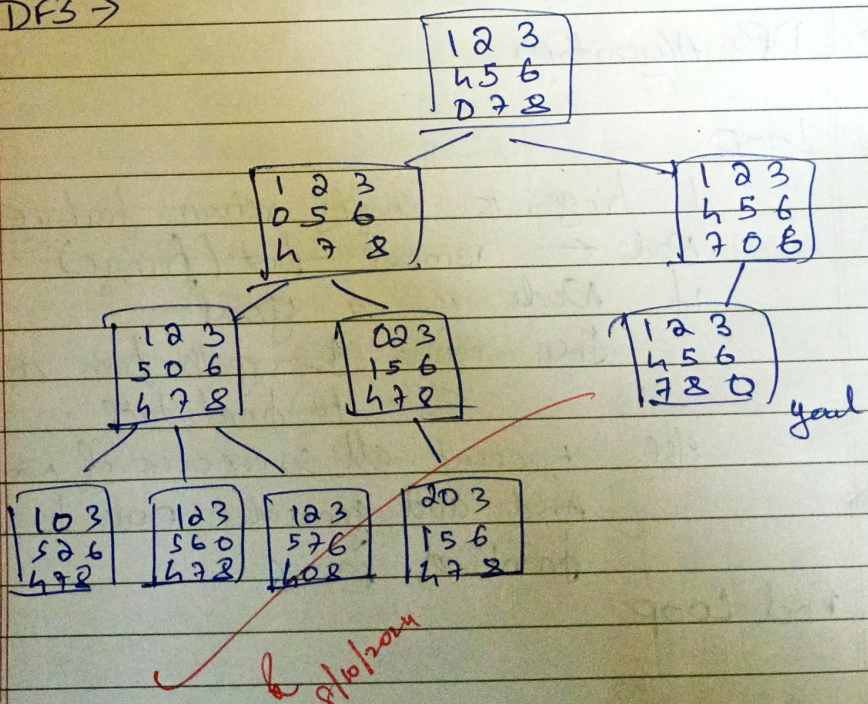
Loop
if fringe is empty return failure
Node ← remove-first(fringe)
if Node is a goal
then return the path from initial state to final state
else generate all successors of Node
And add generated node to the front of fringe
End Loop

State space tree

BFS →



DFS →



18/10/2024