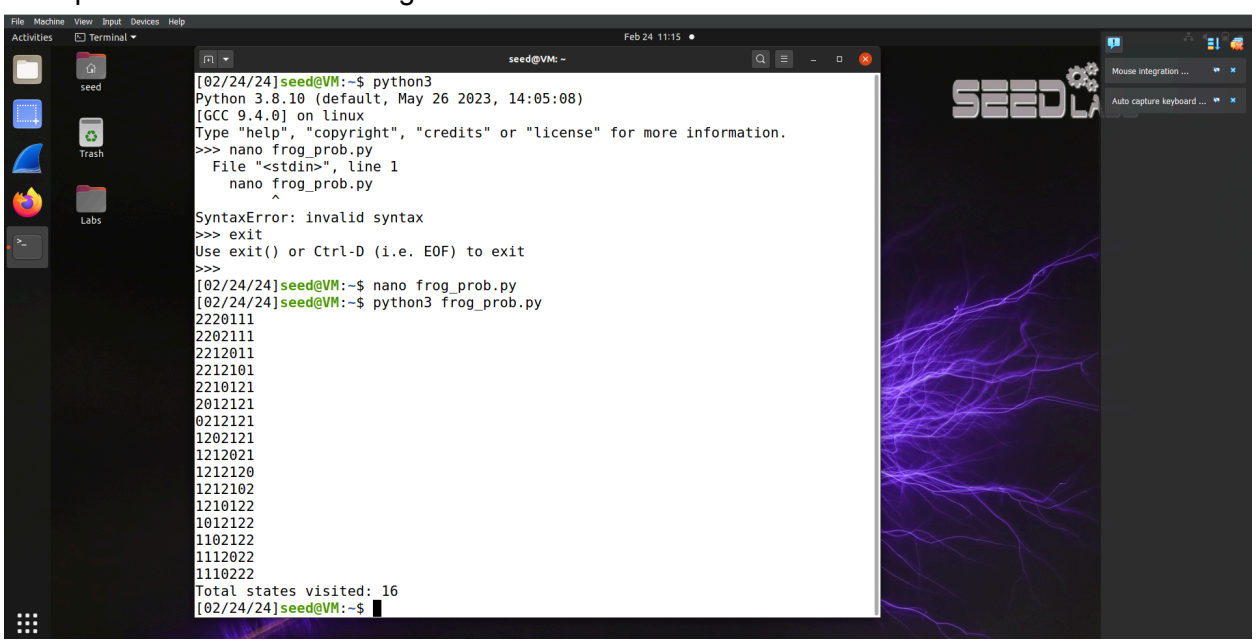
IZHAN SOHAIL

Readme file with code uploaded on the link:
🗀 Frog_problem-23118

Output:

To solve the frog problem, where the goal is to swap positions of green and red frogs on a log, we used a classic algorithmic strategy. The Python code for this task charts each possible hop and leap the frogs can make.

The optimal route to reach the goal state:



Total number of possible moves:

```
seed@VM: ~
Legal move: 2121120 -> 2121102
Legal move: 2121012 -> 2101212
Legal move: 2121012 -> 2121102
Legal move: 1212021 -> 1210221
Legal move: 1212021 -> 1212120
Legal move: 2112120 -> 2112102
Legal move: 0121221 -> 1021221
Legal move: 2101212 -> 0121212
Legal move: 2101212 -> 2110212
Legal move: 1210221 -> 1012221
Legal move: 1212120 -> 1212102
Legal move: 2112102 -> 2110122
Legal move: 1021221 -> 1120221
Legal move: 0121212 -> 1021212
Legal move: 2110212 -> 2111202
Legal move: 1012221 -> 1102221
Legal move: 1212102 -> 1210122
Legal move: 2110122 -> 2111022
Legal move: 1120221 -> 1102221
Legal move: 1021212 -> 1120212
Legal move: 2111202 -> 2111022
Legal move: 1210122 -> 1012122
Legal move: 1210122 -> 1211022
Legal move: 1120212 -> 1102212
Legal move: 1120212 -> 1121202
Legal move: 1012122 -> 1102122
Legal move: 1121202 -> 1121022
Legal move: 1102122 -> 1112022
Legal move: 1121022 -> 1101222
Legal move: 1112022 -> 1110222
Legal move: 1101222 -> 1110222
Total number of states: 72
[02/27/24]seed@VM:~$
```

```
seed@VM: ~
  GNU nano 4.8                    frog_prob.py                    Modified
from collections import deque

def is_valid(state, pos, move):
    if move == 1 and pos < len(state) - 1:
        return state[pos + 1] == '0'
    if move == 2 and pos < len(state) - 2:
        return state[pos + 2] == '0'
    return False

def generate_moves(state):
    moves = []
    for i, frog in enumerate(state):
        if frog == '2' and i < len(state) - 1:  # Green frog moves
            if is_valid(state, i, 1):
                moves.append(state[:i] + '0' + frog + state[i+2:])
            if is_valid(state, i, 2):
                moves.append(state[:i] + '0' + state[i+1:i+2] + frog + state[i+>
        elif frog == '1' and i > 0:  # Red frog moves
            if is_valid(state[::-1], len(state) - 1 - i, 1):

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit       ^R Read File  ^\ Replace    ^U Paste Text ^T To Spell   ^_ Go To Line
```