Gardencity University

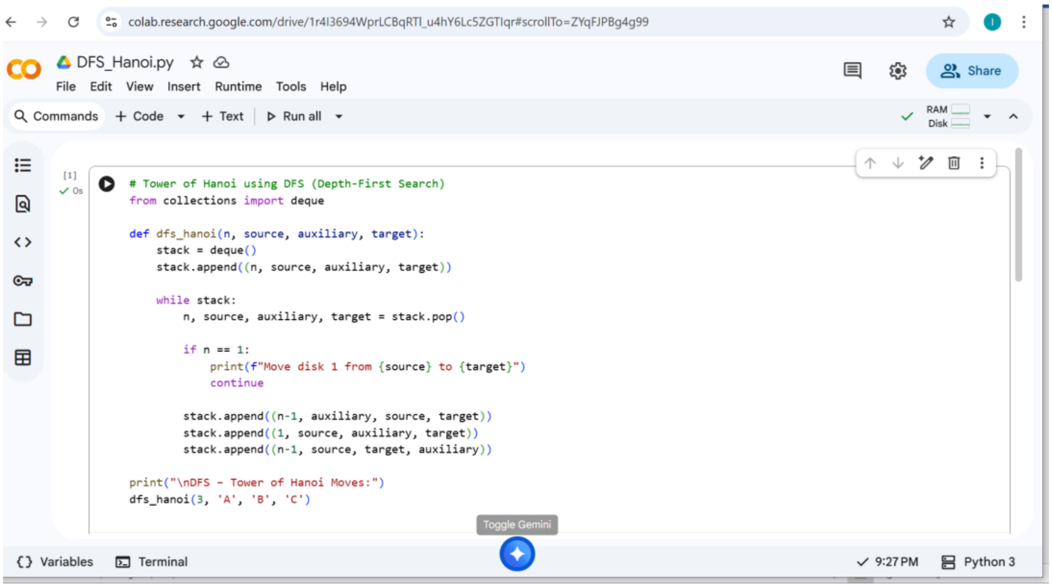
EMPHASIS IN LIFE

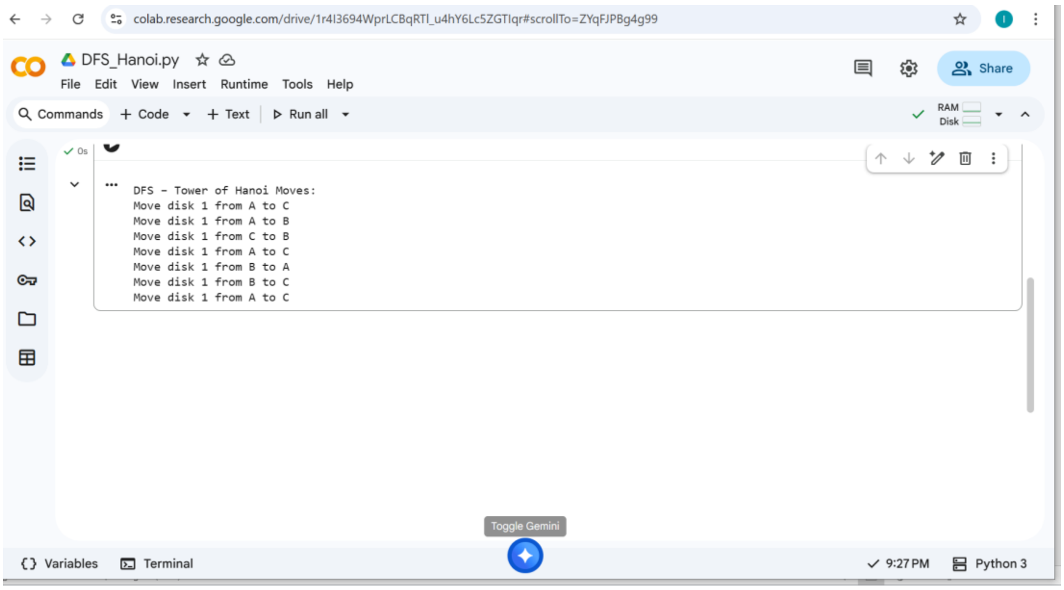
Name: Lalitha.T

Roll No:25MCAR149

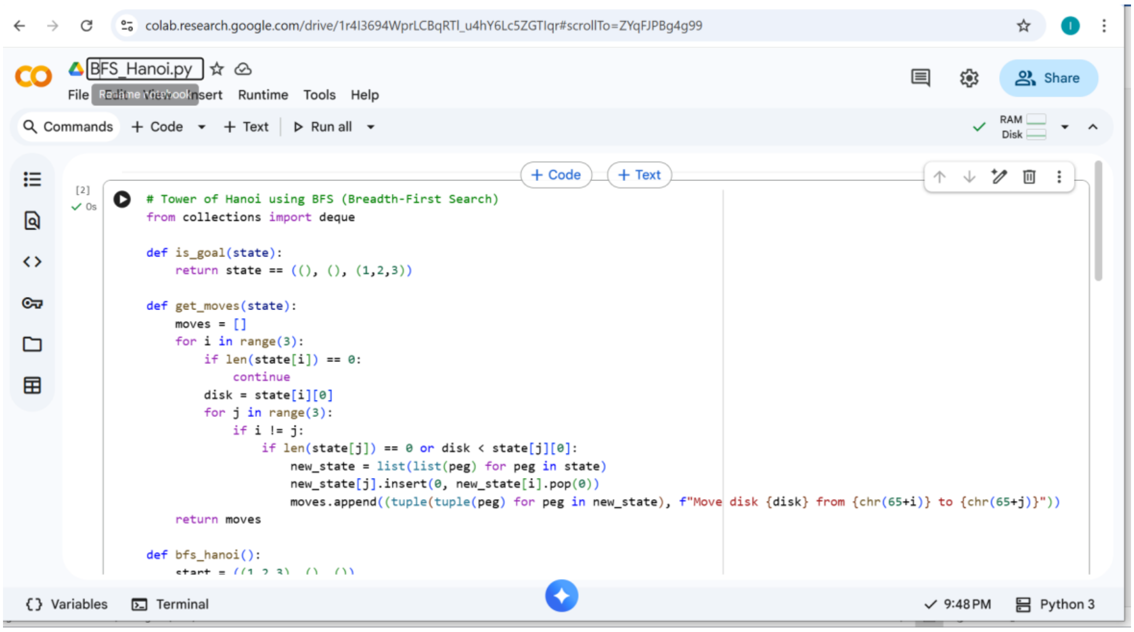
AI Quantitative assignment Output Screenshot

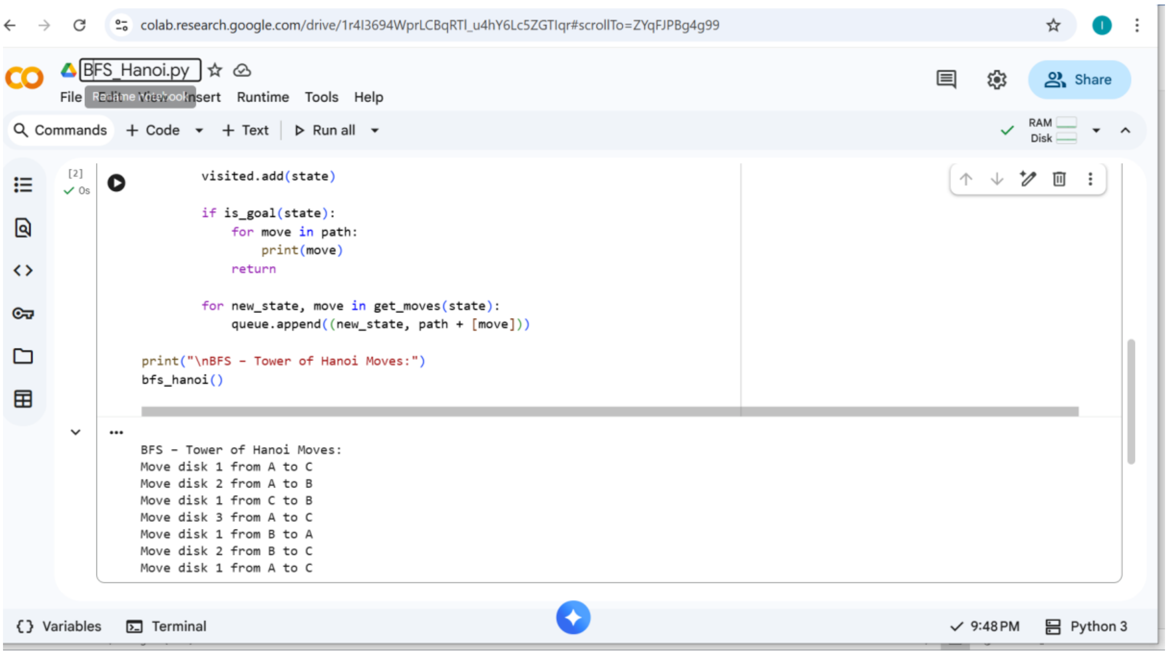
01.DFS



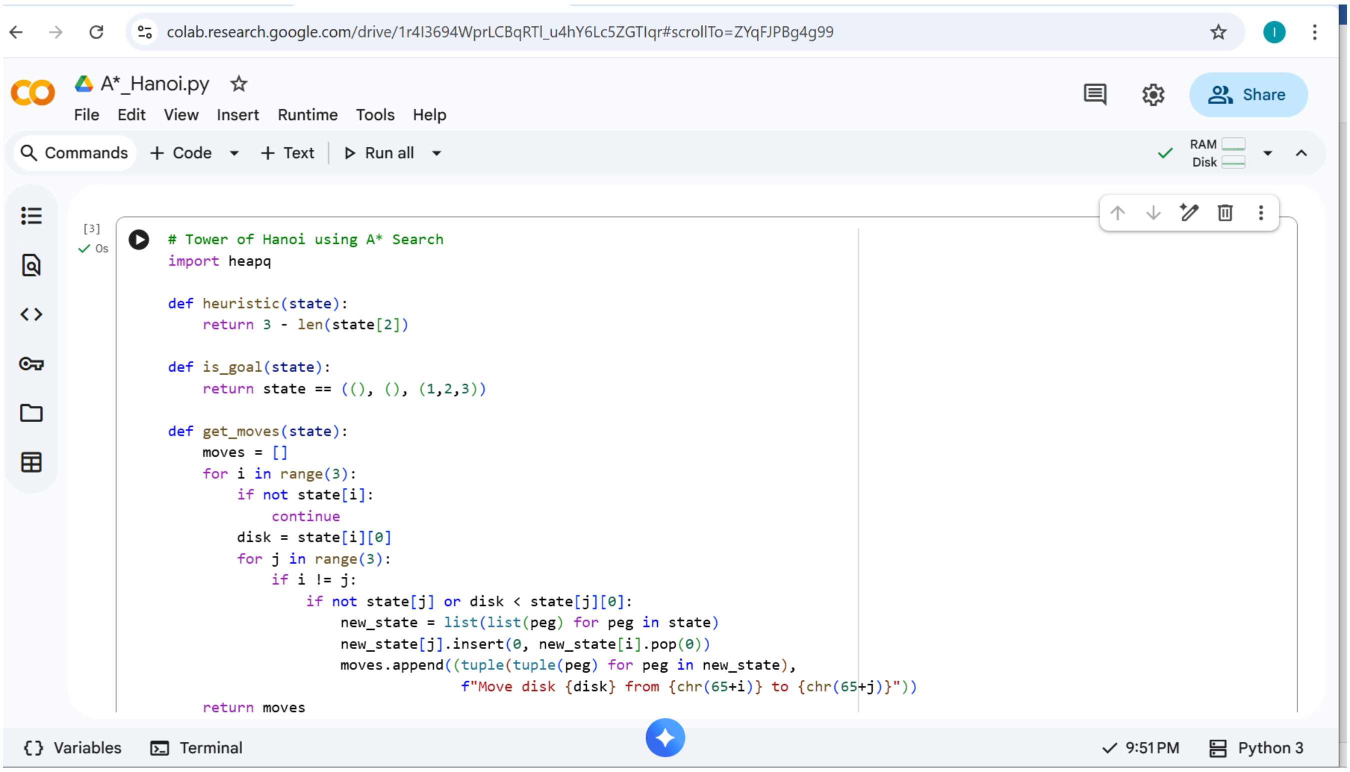


02.BFS


colab.research.google.com/drive/1r4I3694WprLCBqRTI_u4hY6Lc5ZGTIqr#scrollTo=ZYqFJPBg4g99
BFS_Hanoi.py
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
+ Code + Text
[2]
0s
# Tower of Hanoi using BFS (Breadth-First Search)
from collections import deque
def is_goal(state):
return state == ((), (), (1,2,3))
def get_moves(state):
moves = []
for i in range(3):
if len(state[i]) == 0:
continue
disk = state[i][0]
for j in range(3):
if i != j:
if len(state[j]) == 0 or disk < state[j][0]:
new_state = list(list(peg) for peg in state)
new_state[j].insert(0, new_state[i].pop(0))
moves.append((tuple(tuple(peg) for peg in new_state), f"Move disk {disk} from {chr(65+i)} to {chr(65+j)}"))
return moves
def bfs_hanoi():
Variables
Terminal
9:48 PM
Python 3


colab.research.google.com/drive/1r4I3694WprLCBqRTI_u4hY6Lc5ZGTIqr#scrollTo=ZYqFJPBg4g99
BFS_Hanoi.py
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[2]
0s
visited.add(state)
if is_goal(state):
for move in path:
print(move)
return
for new_state, move in get_moves(state):
queue.append((new_state, path + [move]))
print("\nBFS - Tower of Hanoi Moves:")
bfs_hanoi()
BFS - Tower of Hanoi Moves:
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
Variables
Terminal
9:48 PM
Python 3

03.A* Algorithm

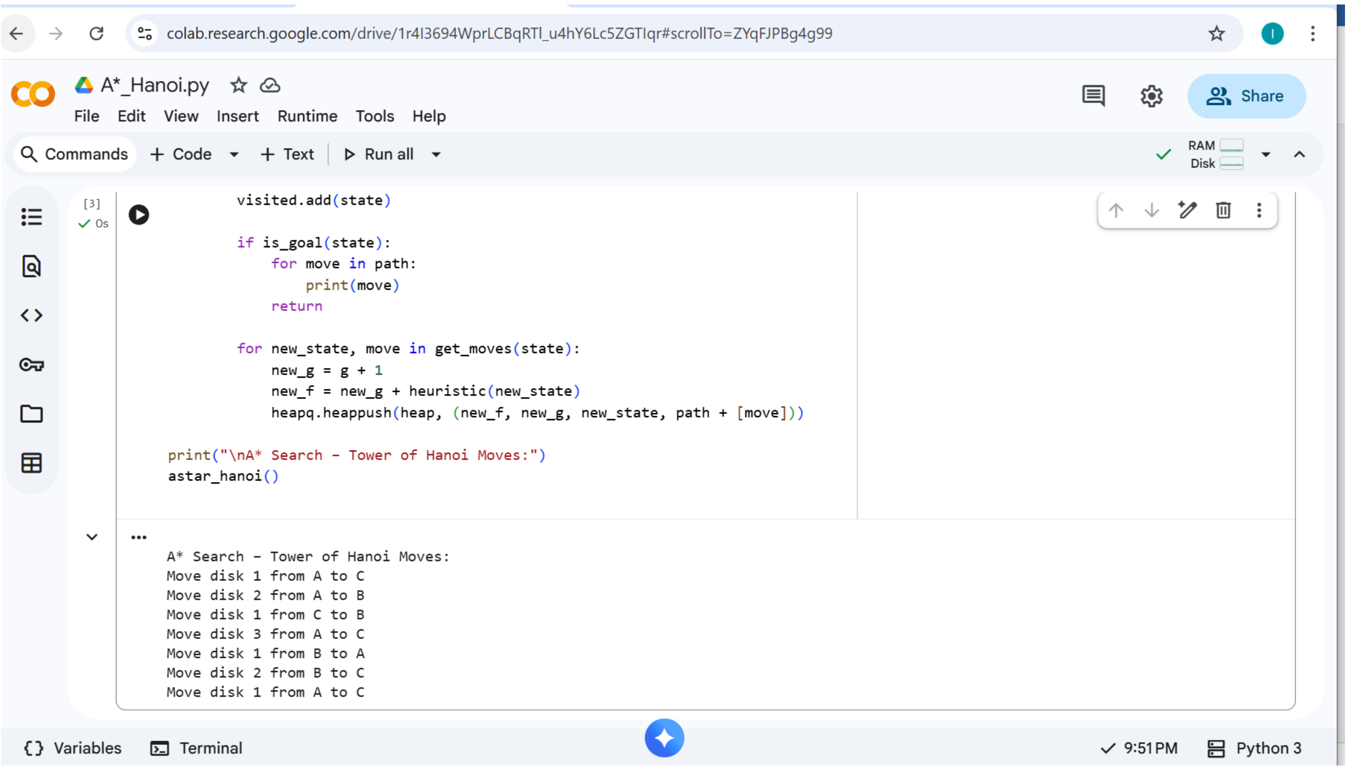


```python
# Tower of Hanoi using A* Search
import heapq

def heuristic(state):
    return 3 - len(state[2])

def is_goal(state):
    return state == ((), (), (1,2,3))

def get_moves(state):
    moves = []
    for i in range(3):
        if not state[i]:
            continue
        disk = state[i][0]
        for j in range(3):
            if i != j:
                if not state[j] or disk < state[j][0]:
                    new_state = list(list(peg) for peg in state)
                    new_state[j].insert(0, new_state[i].pop(0))
                    moves.append((tuple(tuple(peg) for peg in new_state),
                                  f"Move disk {disk} from {chr(65+i)} to {chr(65+j)}"))
    return moves
```



```python
        visited.add(state)

        if is_goal(state):
            for move in path:
                print(move)
            return

        for new_state, move in get_moves(state):
            new_g = g + 1
            new_f = new_g + heuristic(new_state)
            heapq.heappush(heap, (new_f, new_g, new_state, path + [move]))

print("\nA* Search - Tower of Hanoi Moves:")
astar_hanoi()
```

```
A* Search - Tower of Hanoi Moves:
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```