‘’’Name:Rutuja Ashok Jagtap
std:BE(comp)-A

Title: Design and implement Parallel Breadth First Search and Depth First Search based on existing

algorithms using OpenMP. Use a Tree or an undirected graph for BFS and DFS .’’’

```
import multiprocessing
from multiprocessing import Manager
from concurrent.futures import ThreadPoolExecutor

class Graph:
    def __init__(self, V):
        self.V = V  # Number of vertices
        self.adj = {i: [] for i in range(V)}  # Adjacency list

    def add_edge(self, u, v):
        self.adj[u].append(v)
        self.adj[v].append(u)  # Undirected graph

    def bfs_worker(self, queue, visited, lock):
        while not queue.empty():
            node = queue.get()
            print(node, end=" ")

            with lock:
                neighbors = self.adj[node]

            with ThreadPoolExecutor() as executor:
                for neighbor in neighbors:
                    if not visited[neighbor]:
                        visited[neighbor] = True
                        queue.put(neighbor)
```

```
                executor.submit(lambda: None)  # Dummy thread (to simulate parallelism)

def parallel_bfs(self, start):
    manager = Manager()
    visited = manager.list([False] * self.V)
    queue = manager.Queue()
    lock = manager.Lock()

    queue.put(start)
    visited[start] = True

    print("Parallel BFS Traversal: ", end="")
    self.bfs_worker(queue, visited, lock)
    print()

def dfs_worker(self, stack, visited, lock):
    while stack:
        node = stack.pop()
        if not visited[node]:
            visited[node] = True
            print(node, end=" ")

            with lock:
                neighbors = reversed(self.adj[node])  # Reverse for DFS order

            with ThreadPoolExecutor() as executor:
                for neighbor in neighbors:
                    if not visited[neighbor]:
                        stack.append(neighbor)
                        executor.submit(lambda: None)  # Dummy thread
```

```
    def parallel_dfs(self, start):
        manager = Manager()
        visited = manager.list([False] * self.V)
        stack = manager.list()
        lock = manager.Lock()

        stack.append(start)

        print("Parallel DFS Traversal: ", end="")
        self.dfs_worker(stack, visited, lock)
        print()


if __name__ == "__main__":
    g = Graph(6)

    # Creating an undirected graph
    g.add_edge(0, 1)
    g.add_edge(0, 2)
    g.add_edge(1, 3)
    g.add_edge(1, 4)
    g.add_edge(2, 5)

    print("Starting from node 0:")

    g.parallel_bfs(0)
    g.parallel_dfs(0)
```

Output:

```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr  5 2023, 00:38:17) [MSC v.
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informati

============== RESTART: C:/Users/Rutuja/Desktop/hpc/practical1.py ===
Starting from node 0:
Parallel BFS Traversal: 0 1 2 3 4 5
Parallel DFS Traversal: 0 1 3 4 2 5
```