Design and implement parallel BFS and DFS based on existing algorithms using openMP use a tree or an undirected graph for BFS and DFS.

```
import queue
import threading
import time

NUM_THREADS = 5 # no of threads

def bfs(start, adj_list):
    level = [-1] * len(adj_list)
    visited = [False] * len(adj_list)

    q = queue.Queue()
    q.put(start)
    level[start] = 0
    visited[start] = True

    while not q.empty():
        u = q.get()
        for v in adj_list[u]:
            if not visited[v]:
                visited[v] = True
                level[v] = level[u] + 1
                q.put(v)

    for i in range(len(adj_list)):
        if visited[i]:
            print(i," ",end="")

def parallel_bfs(start, adj_list):
    level = [-1] * len(adj_list)
    visited = [False] * len(adj_list)

    q = queue.Queue()
    q.put(start)
    level[start] = 0
    visited[start] = True

    def worker():
        while not q.empty():
            u = q.get()
            for v in adj_list[u]:
                if not visited[v]:
                    visited[v] = True
                    level[v] = level[u] + 1
                    q.put(v)

    threads = []
    for i in range(NUM_THREADS):
        t = threading.Thread(target=worker)
        t.start()
        threads.append(t)

    for t in threads:
        t.join()

    for i in range(len(adj_list)):
        if visited[i]:
            print(i," ",end="")

adj_list = {
    0: [1, 2],
    1: [0, 3],
    2: [0, 4],
    3: [1, 5],
    4: [2],
    5: [3],
}

print("Sequential BFS:")
start_time = time.time()
bfs(0, adj_list)
print("\nElapsed time:", time.time() - start_time)

print("Parallel BFS:")
start_time = time.time()
parallel_bfs(0, adj_list)
print("\nElapsed time:", time.time() - start_time)
```

```
import threading
import time

NUM_THREADS = 5

def dfs(u, visited, adj_list):
    visited[u] = True
    for v in adj_list[u]:
        if not visited[v]:
            dfs(v, visited, adj_list)

def parallel_dfs(start, adj_list):
    visited = [False] * len(adj_list)

    def worker(u):
        dfs(u, visited, adj_list)

    threads = []
    for u in range(len(adj_list)):
        if not visited[u]:
            t = threading.Thread(target=worker, args=(u,))
            t.start()
            threads.append(t)

    for t in threads:
        t.join()

    for i in range(len(adj_list)):
        if visited[i]:
            print(i," ",end="")

adj_list = {
    0: [1, 2],
    1: [0, 3],
    2: [0, 4],
    3: [1, 5],
    4: [2],
    5: [3],
}

print("Sequential DFS:")
start_time = time.time()
dfs(0, [False] * len(adj_list), adj_list)
print("Elapsed time:", time.time() - start_time)

print("Parallel DFS:")
start_time = time.time()
parallel_dfs(0, adj_list)
print("\nElapsed time:", time.time() - start_time)
```

```
Sequential DFS:
Elapsed time: 0.00016307830810546875
Parallel DFS:
0  1  2  3  4  5
Elapsed time: 0.0015635490417480469
```