

```

#include <iostream>
#include <vector>
#include <queue>
#include <omp.h>

using namespace std;

// Graph class representing the adjacency list
class Graph {
    int V; // Number of vertices
    vector<vector<int>> adj; // Adjacency list

public:
    Graph(int V) : V(V), adj(V) {}

    // Add an edge to the graph
    void addEdge(int v, int w) {
        adj[v].push_back(w);
    }

    // Parallel Depth-First Search
    void parallelDFS(int startVertex) {
        vector<bool> visited(V, false);
        parallelDFSUtil(startVertex, visited);
    }

    // Parallel DFS utility function
    void parallelDFSUtil(int v, vector<bool>& visited) {
        visited[v] = true;
        cout << v << " ";

        #pragma omp parallel for
        for (int i = 0; i < adj[v].size(); ++i) {
            int n = adj[v][i];
            if (!visited[n])
                parallelDFSUtil(n, visited);
        }
    }

    // Parallel Breadth-First Search
    void parallelBFS(int startVertex) {
        vector<bool> visited(V, false);
        queue<int> q;

        visited[startVertex] = true;
        q.push(startVertex);

        while (!q.empty()) {
            int v = q.front();
            q.pop();
            cout << v << " ";

            #pragma omp parallel for
            for (int i = 0; i < adj[v].size(); ++i) {
                int n = adj[v][i];
                if (!visited[n]) {
                    visited[n] = true;
                    q.push(n);
                }
            }
        }
    }
};

```

```

    }
}
};

```

```

int main() {
    // Create a graph
    Graph g(7);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 3);
    g.addEdge(1, 4);
    g.addEdge(2, 5);
    g.addEdge(2, 6);

```

```

/*
    0 ----->1
    |         / \
    |        /   \
    |       v     v     v
    2 -----> 3       4
    |         |
    |         |
    v         v
    5         6
*/

```

```

cout << "Depth-First Search (DFS): ";
g.parallelDFS(0);
cout << endl;

cout << "Breadth-First Search (BFS): ";
g.parallelBFS(0);
cout << endl;

return 0;
}

```