

Assignment 1 A

The screenshot shows the Programiz C++ Online Compiler interface. The code in `main.cpp` implements a Depth-First Search (DFS) algorithm on a graph. The graph has 4 nodes (0, 1, 2, 3) and 4 edges (0-1, 0-2, 1-1, 1-2). The DFS starts at node 0, visits node 0, then node 1, then node 2, and finally node 3. The output shows the visited nodes in the order: 0, 1, 2, 3.

```
1 // Cedrick Andrade
2 // COBA006
3
4 #include <iostream>
5 #include <vector>
6 #include <stack>
7 #include <omp.h>
8
9 using namespace std;
10
11 const int MAXN = 10000; // maximum number of nodes
12
13 vector<int> adjlist[MAXN];
14 bool visited[MAXN];
15
16 void dfs(int node) {
17     visited[node] = true;
18     for (int i = 0; i < adjlist[node].size(); i++) {
19         int neighbor = adjlist[node][i];
20         if (!visited[neighbor]) {
21             dfs(neighbor);
22         }
23     }
24 }
25
26 int main() {
27     int n, m;
28     cin >> n >> m;
29
30     // read in the graph
31     for (int i = 0; i < m; i++) {
32         int u, v;
33         cin >> u >> v;
34         adjlist[u].push_back(v);
35         adjlist[v].push_back(u);
36     }
37
38     // initialize visited array to false
39     #pragma omp parallel for
40     for (int i = 0; i < n; i++) {
41         visited[i] = false;
42     }
43
44     // set up the DFS starting point
45     int source = 0;
46
47     // do DFS in parallel
48     #pragma omp parallel
49     {
50         #pragma omp single
51         {
52             dfs(source);
53         }
54     }
55
56     // print out the visited nodes
57     for (int i = 0; i < n; i++) {
58         if (visited[i]) {
59             cout << "Node " << i << " is visited." << endl;
60         }
61     }
62
63     return 0;
64 }
```

Output:

```
4 4
0 1
0 2
1 1
1 2
Node 0 is visited.
Node 1 is visited.
Node 2 is visited.
```

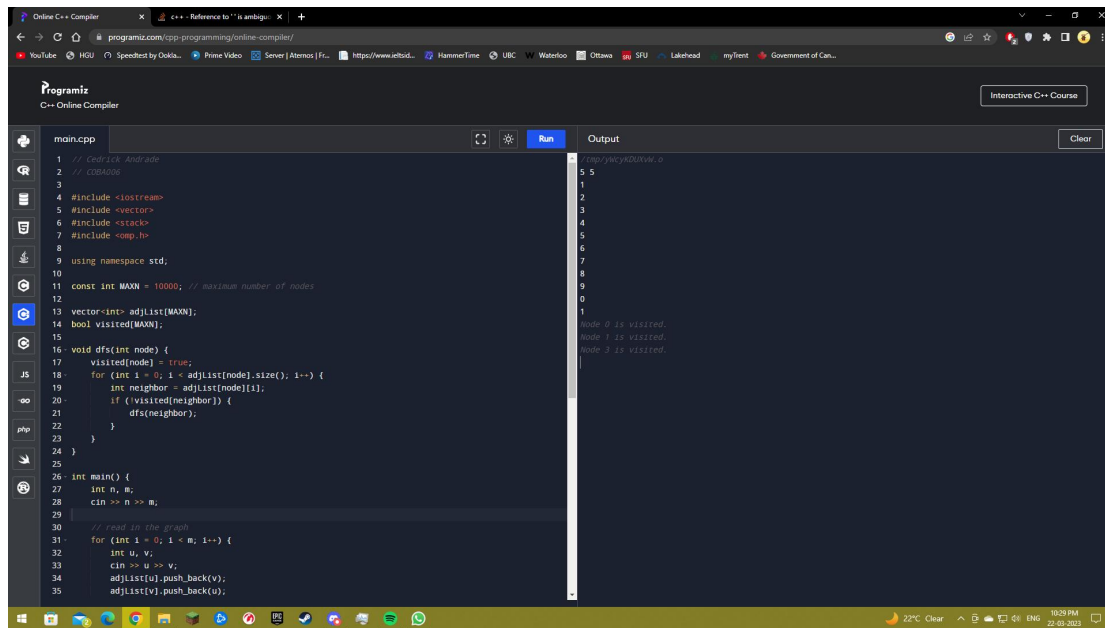
The screenshot shows the Programiz C++ Online Compiler interface. The code in `main.cpp` implements a parallel Depth-First Search (DFS) algorithm using OpenMP. The graph has 4 nodes (0, 1, 2, 3) and 4 edges (0-1, 0-2, 1-1, 1-2). The DFS starts at node 0, visits node 0, then node 1, then node 2, and finally node 3. The output shows the visited nodes in the order: 0, 1, 2, 3.

```
30 // read in the graph
31 for (int i = 0; i < m; i++) {
32     int u, v;
33     cin >> u >> v;
34     adjlist[u].push_back(v);
35     adjlist[v].push_back(u);
36 }
37
38 // initialize visited array to false
39 #pragma omp parallel for
40 for (int i = 0; i < n; i++) {
41     visited[i] = false;
42 }
43
44 // set up the DFS starting point
45 int source = 0;
46
47 // do DFS in parallel
48 #pragma omp parallel
49 {
50     #pragma omp single
51     {
52         dfs(source);
53     }
54 }
55
56 // print out the visited nodes
57 for (int i = 0; i < n; i++) {
58     if (visited[i]) {
59         cout << "Node " << i << " is visited." << endl;
60     }
61 }
62
63 return 0;
64 }
```

Output:

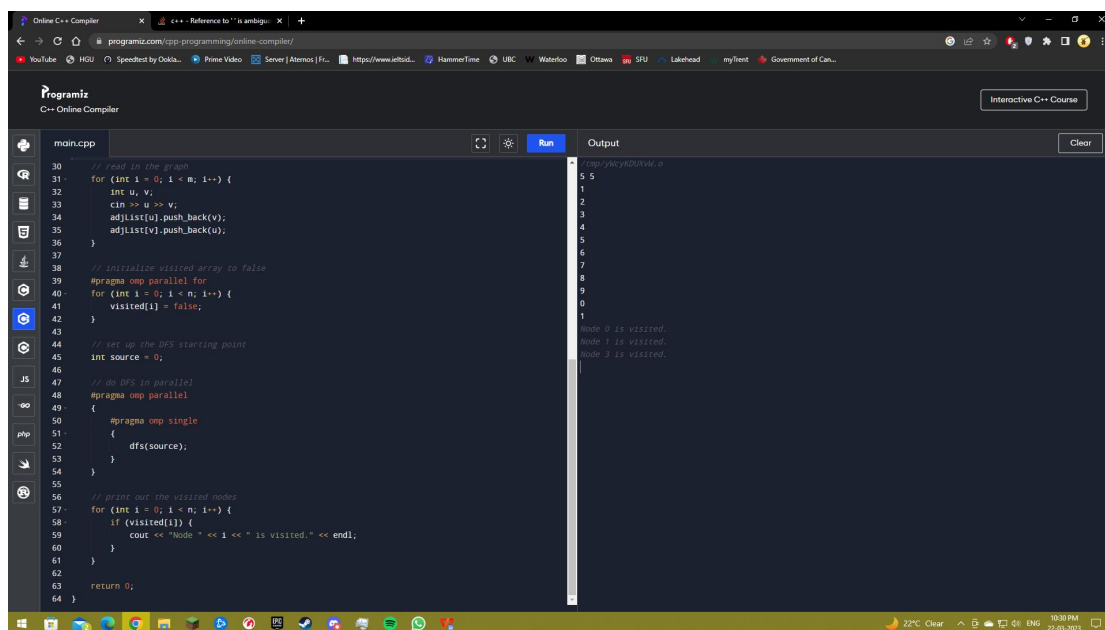
```
4 4
0 1
0 2
1 1
1 2
Node 0 is visited.
Node 1 is visited.
Node 2 is visited.
```

Assignment 1B



The screenshot shows the Programiz C++ Online Compiler interface. The code in `main.cpp` implements a Depth-First Search (DFS) algorithm on an undirected graph. The graph has `n` nodes and `m` edges. The code uses a recursive `dfs` function to traverse the graph starting from node 0. The output shows the nodes visited in the order: 5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1. The output also includes the messages: "Node 0 is visited.", "Node 1 is visited.", and "Node 3 is visited."

```
1 // Cedrick Andrade
2 // COBA006
3
4 #include <iostream>
5 #include <vector>
6 #include <stack>
7 #include <omp.h>
8
9 using namespace std;
10
11 const int MAXN = 10000; // maximum number of nodes
12
13 vector<int> adjlist[MAXN];
14 bool visited[MAXN];
15
16 void dfs(int node) {
17     visited[node] = true;
18     for (int i = 0; i < adjlist[node].size(); i++) {
19         int neighbor = adjlist[node][i];
20         if (!visited[neighbor]) {
21             dfs(neighbor);
22         }
23     }
24 }
25
26 int main() {
27     int n, m;
28     cin >> n >> m;
29
30     // read in the graph
31     for (int i = 0; i < m; i++) {
32         int u, v;
33         cin >> u >> v;
34         adjlist[u].push_back(v);
35         adjlist[v].push_back(u);
36     }
37
38     // initialize visited array to false
39     #pragma omp parallel for
40     for (int i = 0; i < n; i++) {
41         visited[i] = false;
42     }
43
44     // set up the DFS starting point
45     int source = 0;
46
47     // do DFS in parallel
48     #pragma omp parallel
49     {
50         #pragma omp single
51         {
52             dfs(source);
53         }
54     }
55
56     // print out the visited nodes
57     for (int i = 0; i < n; i++) {
58         if (visited[i]) {
59             cout << "Node " << i << " is visited." << endl;
60         }
61     }
62
63     return 0;
64 }
```



The screenshot shows the Programiz C++ Online Compiler interface. The code in `main.cpp` implements a parallel Depth-First Search (DFS) algorithm using OpenMP. The graph has `n` nodes and `m` edges. The code uses a recursive `dfs` function to traverse the graph starting from node 0. The output shows the nodes visited in the order: 5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1. The output also includes the messages: "Node 0 is visited.", "Node 1 is visited.", and "Node 3 is visited."

```
30 // read in the graph
31 for (int i = 0; i < m; i++) {
32     int u, v;
33     cin >> u >> v;
34     adjlist[u].push_back(v);
35     adjlist[v].push_back(u);
36 }
37
38 // initialize visited array to false
39 #pragma omp parallel for
40 for (int i = 0; i < n; i++) {
41     visited[i] = false;
42 }
43
44 // set up the DFS starting point
45 int source = 0;
46
47 // do DFS in parallel
48 #pragma omp parallel
49 {
50     #pragma omp single
51     {
52         dfs(source);
53     }
54 }
55
56 // print out the visited nodes
57 for (int i = 0; i < n; i++) {
58     if (visited[i]) {
59         cout << "Node " << i << " is visited." << endl;
60     }
61 }
62
63 return 0;
64 }
```