

# Lab Report – Week 10

CS2023 Data Structures and Algorithms

Dept. of Computer Science and Engineering, University of Moratuwa

Name: Tharindu Perera

Index Number: 210472J

---

## Section1: Implementing Graph ADT

1.

1: 2, 3, 4, 5  
2: 1, 3, 6  
3: 1, 2  
4: 1, 6, 7, 8  
5: 1, 6, 7, 8  
6: 2, 4, 5  
7: 4, 5  
8: 4, 5

4.

```
PS C:\Users\thari\UoM-DSA-S2-Labs\Lab10> g++ -o bin\app.exe .\graph_lab.cpp
PS C:\Users\thari\UoM-DSA-S2-Labs\Lab10> ./bin/app.exe
1: 2 3 4 5
2: 1 3 6
3: 1 2
4: 1 6 7 8
5: 1 6 7 8
6: 2 4 5
7: 4 5
8: 4 5
PS C:\Users\thari\UoM-DSA-S2-Labs\Lab10> █
```

```

19 void initializeNodes()
20 //iterate through the nodes and assign labels
21 for(int i=0;i<n;i++){
22     nodes[i].label=i+1;
23 }
24
25
26 void addEdge(int u, int v){
27     //select node u and push v into u's neighbour
28     nodes[u-1].neighbours.push_back(v-1);
29
30     //select node v and push u into v's neighbour
31     nodes[v-1].neighbours.push_back(u-1);
32 }
33
34 void addEdge(int from, int to){
35     //select node from and push to into from's neighbour
36     nodes[from-1].neighbours.push_back(to-1);
37 }
38
39 void print(){
40     //lets iterate through each node and print its neighbours
41     for (int i = 0; i < n; i++)
42     {
43         cout << nodes[i].label << " ";
44         for (list<int>::iterator it = nodes[i].neighbours.begin(); it != nodes[i].neighbours.end(); ++it)
45             cout << " " << nodes[*it].label;
46         cout << "\n";
47     }
48 }
49
50
51
52
53

```

```

PS C:\Users\thari\UoM-DSA-S2-Labs\Lab10> g++ -o bin/app.exe -lgph_lab.cpp
PS C:\Users\thari\UoM-DSA-S2-Labs\Lab10> ./bin/app.exe
1: 2 3 4 5
2: 1 3 6
3: 1 2
4: 1 6 7 8
5: 1 6 7 8
6: 2 4 5
7: 4 5
8: 4 5
PS C:\Users\thari\UoM-DSA-S2-Labs\Lab10>

```

5.

```

void addEdge(int from, int to){
    //select node from and push to into from's neighbour
    nodes[from-1].neighbours.push_back(to-1);
}

```

\*Index is decremented to fit the zero based indexing in the list.

## Section2: Working out link prediction, no coding required

Traverse through each of the neighbours of node 1 and and for each of them travers through their neighbours and save it to a set data structure. Next traverse through the neighbours of node 4 and save it to a nother set data structure. Save bothe of those to another set structure to get all naighbours of both and get the count of it (n\_all) get the counts of the previous two sets n\_1 nd n\_4 we can get the count of neighbours that both 1 and 4 have in common by  $n_{com} = n_1 + n_4 - n_{all}$  then find similarity of each of 1s neighbours and 4 by  $similarity = n_{com} / n_{all}$ . Suggest 1s neigh bour with the highest similarity with 4.

GitHub Link : [Tharindu6516/UoM-DSA-S2-Labs \(github.com\)](https://github.com/Tharindu6516/UoM-DSA-S2-Labs)

