CSCI 310 – 02 (Fall 2019)

Programming Foundations

Lab #24: Calculating the degree of vertices in a graph

DUE: Wed, Dec 4, 11:59pm (`turnin` time)

Specifications

Recall that given a digraph $G = (V, E)$

1. the *indegree* of a vertex $v \in V$, denoted $\mathsf{deg}^-(v)$, is the number of vertices $u \in V$ where $(u, v) \in E$;
2. the *outdegree* of a vertex $v \in V$, denoted $\mathsf{deg}^+(v)$, is the number of vertices $u \in V$ where $(v, u) \in E$; and
3. the *degree* of a vertex $v \in V$, denoted $\mathsf{deg}(v)$, is the sum of its *indegree* and *outdegree*, so $\mathsf{deg}(v) = \mathsf{deg}^-(v) + \mathsf{deg}^+(v)$.

For undirected graphs, only the *degree* of a vertex, denoted $\mathsf{deg}(v)$, is defined as the number incident edges to it (no double counts, so that if $(u, v) \in E$ then $(v, u) \in E$ is not counted).

Modify your `Graph` class to have the functionality to

1. have a constructor that allows the client to specify whether a graph is a directed graph or an undirected graph (default);
2. have accessors to calculate and return the *indegree*, *outdegree*, and *degree* of a vertex – return values should be zero if the operation is

The driver for the program, provided in **turnin**, will be the following:

```
#include <iostream>
#include <string>
using namespace std;

#include "Graph.h" // user-defined template class
#include "lab24.h" // implementation of showDegrees()

int main()
{
    Graph< unsigned > G1( "G1" , true ), //   directed graph
                      G2( "G2" );        // undirected graph
    unsigned E,u,v;

    cin >> E; // number of edges in G1
    for( unsigned i=0 ; i<E ; i++ )
    {
            cin >> u >> v;
            G1.add( u , v );
    }

    cin >> E; // number of edges in G2
    for( unsigned i=0 ; i<E ; i++ )
    {
            cin >> u >> v;
            G2.add( u , v );
    }

    // Display graphs G1 and G2
    cout << G1 << endl;
    showDegrees( G1 );
    cout << G2 << endl;
    showDegrees( G2 );

    return 0;
}
```

Input

All input comes from standard input. Assume the edge information for one *directed graph* `G1` (note the `Graph` constructor use) and one *undirected graph* `G2` (again, note the `Graph` constructor use) will be provided. The edge information starts with an `unsigned` value, say E, that indicates the number of edges in the graph. This is followed by E pairs of `unsigned` values representing edges in the graph. In these pairs, each `unsigned` value references a vertex in the graph.

Sample Input

Be sure to draw the graphs so you can manually check and confirm the output of your program.

```
13
3 5
4 6
5 7
6 8
7 9
8 9
9 2
8 3
7 4
6 5
6 4
8 4
6 2

14
1 2
2 3
3 4
4 6
4 7
5 6
3 5
7 8
6 10
5 9
10 11
11 12
11 13
12 13
```

Output

Based on the driver program provided, the output will consist of the graph information (from overloaded `operator<<`) followed by the degree of every vertex in the graph. The *indegree* and *outdegree* information should be generated and displayed if the graph is a directed graph; otherwise, only the *degree* information should be generated and displayed.

Note that the the `showDegrees()` function displays the vertex degrees in tabular form (where each column has a fixed width of 10). This is the tabular form of the *degree sequence* of a graph.

Sample Output

Here is the output for the Sample Input provided above:

```
digraph G1 has 8 vertices and 13 edges:
  V={2,3,4,5,6,7,8,9}
  E={(3,5),(4,6),(5,7),(6,2),(6,4),(6,5),(6,8),(7,4),(7,9),(8,3),(8,4),(8,9),(9,2)}
    Vertex  inDegree outDegree
         2         2         0
         3         1         1
         4         3         1
         5         2         1
         6         1         4
         7         1         2
         8         1         3
         9         2         1
regular graph G2 has 13 vertices and 28 edges:
  V={1,2,3,4,5,6,7,8,9,10,11,12,13}
  E={(1,2),(2,1),(2,3),(3,2),(3,4),(3,5),(4,3),(4,6),(4,7),(5,3),(5,6),(5,9),(6,4),(6,5),(6,10),
     (7,4),(7,8),(8,7),(9,5),(10,6),(10,11),(11,10),(11,12),(11,13),(12,11),(12,13),(13,11),(13,12)}
    Vertex    Degree
         1         1
         2         2
         3         3
         4         3
         5         3
         6         3
         7         2
         8         1
         9         1
        10         2
        11         3
        12         2
        13         2
```

Submission

Your submission will consist of the following file(s), submitted using the **turnin** facility.

- `lab24.h` – implementation of the `showDegrees()` function called in lines #30 and #32 above
- `Graph.h` – header file for your `Graph` class
- `Graph.cpp` – implementation file for your `Graph` class