CSCI 310 – 02 (Fall 2019)

Programming Foundations

Lab #21: Union and Difference with Graphs

DUE: Fri, Nov 8, 11:59pm (`turnin` time)

Specifications

Having successfully implemented a `struct` approach in representing graphs using a `set` of vertices and a `multimap` of edges, you are now ready to implement some operations on graphs.

Recall that for sets, say $A = \{1, 2, 3\}$ and $B = \{2, 4, 6\}$, the following operations are defined:

1. the *union* (denoted by $\cup$) of two sets is the set of all elements in both A or B. Hence, $A \cup B = \{1, 2, 3, 4, 6\}$.
2. the *difference* (also known as the *relative complement*) of A and B (denoted $A \setminus B$) is the set of elements in A but not in B. Hence, $A \setminus B = \{1, 3\}$.

Note that we can extend the definition of *union* and *difference* to graphs. Basically, given two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$

1. the *union* $G_1 + G_2$ is the graph $G = (V_1 \cup V_2, E_1 \cup E_2)$.
2. the *difference* $G_1 - G_2$ is the graph $G = (V_1 \setminus V_2, E_1 \setminus E_2)$.

You are to overload `operator+` and `operator-` to represent the *union* and *difference* operations on graphs.

The driver for the program will be the following:

```
#include "Graph.h"

int main()
{
    Graph G1, G2;
    unsigned E,u,v;

    // Add edges to graph G1
    cin >> E; // number of edges in G1
    for( unsigned i=0 ; i<E ; i++ )
    {
        cin >> u >> v;
        addEdge( G1 , u , v );
    }
    // Display graph G1
    cout << "1st " << G1 << endl;

    // Add edges to graph G2
    cin >> E; // number of edges in G2
    for( unsigned i=0 ; i<E ; i++ )
    {
        cin >> u >> v;
        addEdge( G2 , u , v );
    }
    // Display graph G2
    cout << "2nd "<< G2 << endl;

    // Display the union and difference of G1 and G2
    cout << "Union " << G1+G2 << endl;
    cout << "Difference " << G1-G2 << endl;

return 0;
}
```

Input

All input comes from standard input. Assume the edge information for two *undirected graphs* will be provided. For each graph, the first `unsigned` value provided, say E, will indicate 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 each graph so you can determine what the union and difference should be.

```
3
1 2
2 3
4 5
3
2 4
4 6
8 10
```

Notice that vertex names do not necessarily start with 0 and are not necessarily evenly spaced, so make sure you have accounted for this in your implementation of the `visited` container.

Output

Based on the driver program provided, the output will consist of displaying each of the two input graphs provided, followed by the union of, and then the difference between, these two graphs.

Sample Output

Here is the output for the Sample Input provided above:

```
1st Graph has 5 vertices and 6 edges.
  V={1,2,3,4,5}
  E={(1,2),(2,1),(2,3),(3,2),(4,5),(5,4)}
2nd Graph has 5 vertices and 6 edges.
  V={2,4,6,8,10}
  E={(2,4),(4,2),(4,6),(6,4),(8,10),(10,8)}
Union Graph has 8 vertices and 12 edges.
  V={1,2,3,4,5,6,8,10}
  E={(1,2),(2,1),(2,3),(2,4),(3,2),(4,2),(4,5),(4,6),(5,4),(6,4),(8,10),(10,8)}
Difference Graph has 3 vertices and 0 edges.
  V={1,3,5}
  E={}
```

If the graphs were switched in order, the output would be:

```
1st Graph has 5 vertices and 6 edges.
  V={2,4,6,8,10}
  E={(2,4),(4,2),(4,6),(6,4),(8,10),(10,8)}
2nd Graph has 5 vertices and 6 edges.
  V={1,2,3,4,5}
  E={(1,2),(2,1),(2,3),(3,2),(4,5),(5,4)}
Union Graph has 8 vertices and 12 edges.
  V={1,2,3,4,5,6,8,10}
  E={(1,2),(2,1),(2,3),(2,4),(3,2),(4,2),(4,5),(4,6),(5,4),(6,4),(8,10),(10,8)}
Difference Graph has 3 vertices and 2 edges.
  V={6,8,10}
  E={(8,10),(10,8)}
```

Submission

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

- `Graph.h` – implementation of the three items listed in the Specifications section