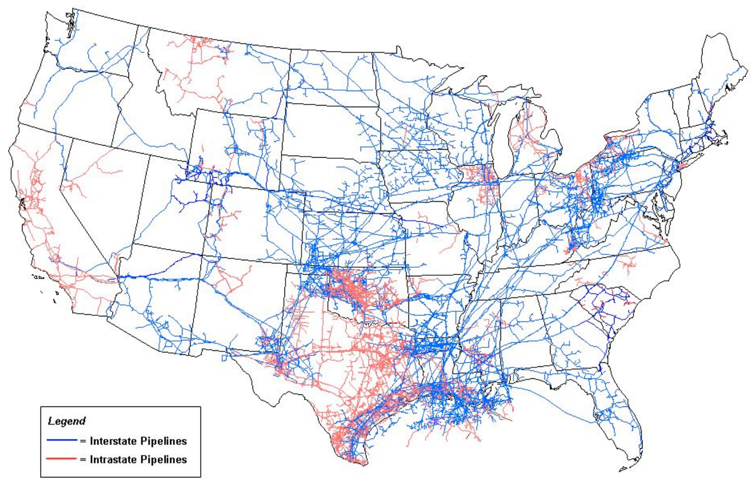
CS 302 – Assignment #11, Final Project

Purpose: Learn concepts regarding graph algorithms and develop application specific data structures. Learn how graphs apply to real-world problems.
Due: Part A → Tuesday (11/21), Part B → Thursday (11/30)
Points: Part A → 75 pts, Part B → 225 pts

Assignment:

Given a large network of natural gas pipelines, a company might be interested in how much gas can be sent from a source location to destination location.



This problem, referred to as a maximum flow problem[1], can be viewed as a graph problem where the various locations are vertices's and the pipelines are directed graph edges.

Design a set of C++ classes to read and store a large, directed graph[2] of locations (vertices's) and connections (edges). Your design, at a minimum, must provide functionality to:

- Read and store the location data.
- Read and store the connections data.
- Find and show the maximum flow from a source location to a destination location.
- Graph statistics (for reference).

To find the maximum flow, the program should implement the Edmonds-Karp algorithm[3] which is a specific implementation of the Ford-Fulkerson algorithm[4] using a BFS search.

Note, some of the graph files will have > 50,000 vertices's and > 100,000 edges.

Part A:

Perform the basic design. Your design should address the class hierarchy and applicable data structures. The data structures should be customized and optimized for this problem.

For part A, create and submit a brief write-up including the following:

- Name, Assignment, Section.
- UML diagram of the classes.
  - Please ensure the functions have appropriate names. Provide simple explanations as necessary. Include the applicable constructor's and destructor's.
  - Additional features/functions, as needed.
- Detailed description of data structure for the graph (node type).
- Summary of other data structures proposed with an explanation.

1 For more information, refer to: https://en.wikipedia.org/wiki/Maximum_flow_problem
2 For more information, refer to: https://en.wikipedia.org/wiki/Graph_(abstract_data_type)
3 For more information, refer to: https://en.wikipedia.org/wiki/Edmonds-Karp_algorithm
4 For more information, refer to: https://en.wikipedia.org/wiki/Ford-Fulkerson_algorithm

The class information format should be similar to past assignments, showing a UML diagram table with the class variables/functions and the function descriptions (1-2 sentences). In addition, the write-up should include a graph showing the class hierarchy.

The basic approach should use Edmonds-Karp implementation of the Ford-Fulkerson algorithm. As such, the choices for the data structures are very important for efficiency. There are many options for the data structures and your choices should be as efficient as possible. A set of data files will be provided. All code must be your own. You should ***not*** use the standard template library. The goal is to provide an overall solution as efficient as possible and demonstrate effective coding techniques.

Part B:

Implement the objects designed in Part A. You do ***not*** need to wait until Part A is scored to start on part B. A simple main is provided. As needed, you may update/correct/alter the original design. Any major changes should be coordinated with the instructor. The graph statistics should be displayed. Refer to the output formatting section for additional information.

Edmonds-Karp Algorithm

Use the following algorithm to find the maximum flow.

```
inputs:                                     outputs:
    G = (V, E)                                  FG (flow graph) = (V, E)
    source node, s                              maximum flow, f
    sink node, t

edmunds-karp
    f = 0
    FG = G                                      // FG is a copy of G
    pred[|V|] = -1                              // set all to -1
    repeat
        sinkStatus = BFS(G, FG, s, t, pred[])
        if (sinkStatus == not visited)
            break
        currFlow = ∞

        v = t
        while (v ≠ s)
            u = pred[v]
            currFlow = min(currFlow , weightFG[u,v])
            v = pred[v]

        v = t
        while (v ≠ s)
            u = pred[v]
            FG[u,v]  =  FG[u,v]  -  currFlow
            FG[v,u]  =  FG[v,u]  +  currFlow
            v = pred[v]

        maxFlow += currFlow

    return maxFlow
```

Breadth First Search

```
BFS(G, FG, s, t, pred[])
    create/initialize queue Q to be empty
    create/initialize visited[|V|] and mark all as unvisited

    mark visited[s] as visited
    Q.enqueue(s)
    pred[s] = -1

    while Q is not empty
        u = Q.dequeue()
        for all vertices, v
            if (!visited[v]  and  weightFG[u,v] > 0.0)
                Q.enqueue(v)
                pred[v] = u
                mark visited[v] as visited

    sinkStatus = visited[t]
    delete  visited
    return  sinkStatus
```

You must use this algorithm.

Required Functions

Based on the provided main, your implementation must include the following functions.

- printGraph() → print the formatted graph. See examples for formatting.
- printFlowGraph() → print the final formatted flow graph. See examples for formatting.
- findMaxFlow() → find the maximum flow for the given graph.
- readGraph() → read a formatted graph file.
- getVertexCount() → return vertex count for the current graph.
- showGraphStats() → show the graph statistics for the current graph. See examples for formatting.

Submission:

Part A (11/21)

- Submit a copy of the write-up (PDF format).
  - You are welcome to come to my office to discuss your approach before submission.

Part B (11/30)

- Submit a compressed zip file of the program source files, header files, and makefile via the on-line submission by 23:50.
- All necessary files must be included in the ZIP file. The grader will download, uncompress, and type **make**. You must have a valid, working *makefile*.
- Do ***not*** submit the data files (we have them).

Input File Format

The input files will be edge lists. The first four lines will contain a graph title, node count, edge count, source node name, and sink node name. For example, below is *graph0.txt*, a small test file.

```
# very simple test graph, maxflow = 7
# Nodes: 6
# Edges: 11
# Source: Kwik-E-Mart
# Sink: Springfield Town Hall
Kwik-E-Mart;Springfield Nuclear Power Plant;3.0
Kwik-E-Mart;Barney's Bowlarama;4.0
Springfield Nuclear Power Plant;Duff Brewery;6.0
Springfield Nuclear Power Plant;Barney's Bowlarama;4.0
Duff Brewery;Barney's Bowlarama;5.0
Duff Brewery;Spinster City Apartments;2.0
Duff Brewery;Springfield Town Hall;7.0
Barney's Bowlarama;Springfield Nuclear Power Plant;2.0
Barney's Bowlarama;Spinster City Apartments;5.0
Spinster City Apartments;Duff Brewery;5.0
Spinster City Apartments;Springfield Town Hall;1.0
```

The program should provide an error for self links (and not allow). For multiple edges between the same vertices's, use the latest edge (thus, ignoring previous edges).

Output Formatting

To accommodate the testing, the program output must follow a specific format. The output should include the graph statistics in the format shown. The final output should include the graph statistics, and optionally, the graph and flow graph.

The following are a series of example program executions;

```
ed-vm% ./maxFlow graph0.txt
*****************************************************************
CS 302 - Assignment #11
Maximum Flow Program

-----------------------------------------------------------------
Graph Adjacency List:
   Title: # very simple test graph, maxflow = 7

Vertex    Verextes...
------    ----------------------------------------------
Kwik-E-Mart
          Barney's Bowlarama (4)
          Springfield Nuclear Power Plant (3)

Springfield Nuclear Power Plant
          Barney's Bowlarama (4)
          Duff Brewery (6)

Barney's Bowlarama
          Spinster City Apartments (5)
          Springfield Nuclear Power Plant (2)

Duff Brewery
          Springfield Town Hall (7)
```

```
          Spinster City Apartments (2)
          Barney's Bowlarama (5)

Spinster City Apartments
          Springfield Town Hall (1)
          Duff Brewery (5)


-----------------------------------------------------------------
Graph Statistics:
   Title:  # very simple test graph, maxflow = 7
   Nodes:  6
   Edges:  11
   Source: Kwik-E-Mart
   Sink:   Springfield Town Hall

Max Flow: 7.00

Flow Graph:
-----------------------------------------------------------------
Flow Graph Adjacency List:
   Title: # very simple test graph, maxflow = 7

Vertex    Verextes...
------    ----------------------------------------------
Kwik-E-Mart
          Barney's Bowlarama (4.00)
          Springfield Nuclear Power Plant (3.00)

Springfield Nuclear Power Plant
          Barney's Bowlarama (0.00)
          Duff Brewery (5.00)

Barney's Bowlarama
          Spinster City Apartments (2.00)
          Springfield Nuclear Power Plant (2.00)

Duff Brewery
          Springfield Town Hall (6.00)
          Spinster City Apartments (0.00)
          Barney's Bowlarama (0.00)

Spinster City Apartments
          Springfield Town Hall (1.00)
          Duff Brewery (1.00)


-----------------------------------------------------------------

*****************************************************************
Game over, thanks for playing.
ed-vm%
```

```
ed-vm% ./maxFlow graph1.txt
*****************************************************************
CS 302 - Assignment #11
Maximum Flow Program

-----------------------------------------------------------------
Graph Adjacency List:
   Title: # very simple test graph 1, maxflow = 23

Vertex    Verextes...
------    ----------------------------------------------
Planet Express
          Bachelor Chow Corp Headquarters (13)
          Benderbräu Brewery (16)

Benderbräu Brewery
          Pabst Blue Robot (12)
          Bachelor Chow Corp Headquarters (10)

Bachelor Chow Corp Headquarters
          eyePhone Factory (14)
          Benderbräu Brewery (4)

Pabst Blue Robot
          Slurm Bottle Corporation (20)
          Bachelor Chow Corp Headquarters (9)

eyePhone Factory
          Slurm Bottle Corporation (4)
          Pabst Blue Robot (7)


-----------------------------------------------------------------
Graph Statistics:
   Title:  # very simple test graph 1, maxflow = 23
   Nodes:  6
   Edges:  11
   Source: Planet Express
   Sink:   Slurm Bottle Corporation

Max Flow: 23.00

Flow Graph:
-----------------------------------------------------------------
Flow Graph Adjacency List:
   Title: # very simple test graph 1, maxflow = 23

Vertex    Verextes...
------    ----------------------------------------------
Planet Express
          Bachelor Chow Corp Headquarters (11.00)
          Benderbräu Brewery (12.00)

Benderbräu Brewery
          Pabst Blue Robot (12.00)
          Bachelor Chow Corp Headquarters (0.00)

Bachelor Chow Corp Headquarters
          eyePhone Factory (11.00)
```

```
          Benderbräu Brewery (0.00)

Pabst Blue Robot
          Slurm Bottle Corporation (19.00)
          Bachelor Chow Corp Headquarters (0.00)

eyePhone Factory
          Slurm Bottle Corporation (4.00)
          Pabst Blue Robot (7.00)


-----------------------------------------------------------------

*****************************************************************
Game over, thanks for playing.
ed-vm%
```