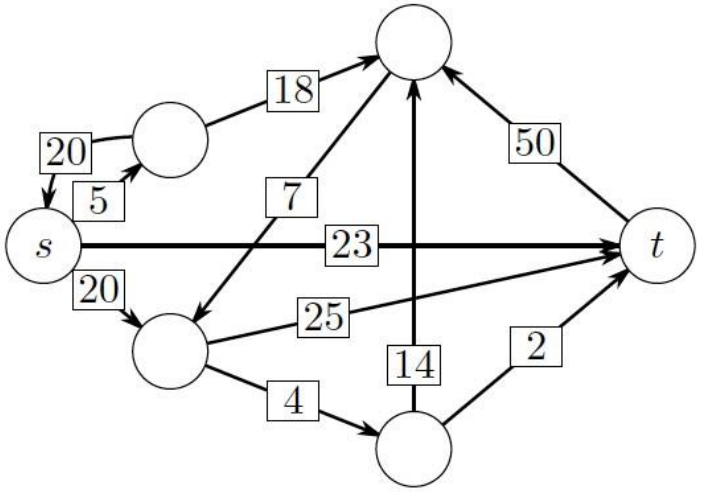
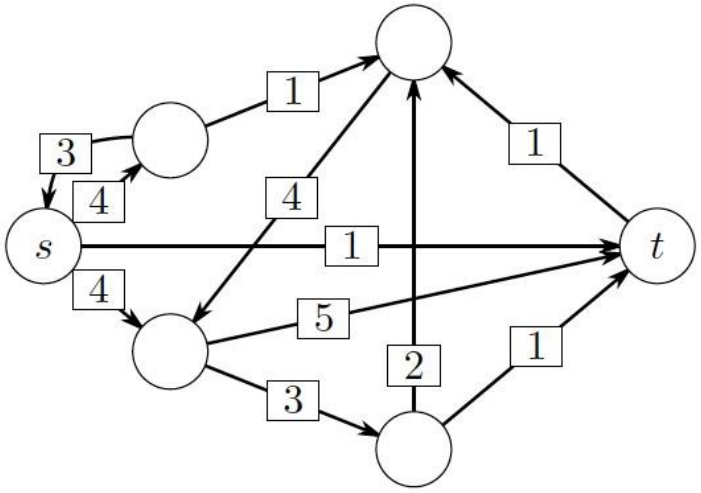
CSC 226 Fall 2015
ALGORITHMS AND DATA STRUCTURES II
Homework 4 – Programming Part

1 Programming Assignment

A *flow network* is a directed graph containing a source node s, a sink node t and a numerical capacity for each arc, as in the example below.



A *flow* in a network is an assignment of flow values to edges such that no edge carries more flow than its capacity value, and for all vertices besides s and t, the amount of flow entering the vertex is equal to the amount of flow leaving it. The diagram below gives a valid flow on the network pictured above.



The total value of a flow is equal to the net amount of flow leaving the source s (or, equivalently, the net amount of flow entering the sink t). The total value of the flow above is 6, since although there are 9 units of flow leaving vertex s, 3 units are sent back to s. Generally, it is desirable to maximize the total value of a flow. The flow above is not maximum.

In this assignment, you will implement an algorithm which takes a flow network G and returns a feasible flow with the maximum possible total value. A Java template has been provided containing an empty method `MaxFlow`, which takes an adjacency matrix for G as its only argument (see the Data Format section below for information on how the flow network is specified). The expected behavior of the method is as follows:

Input: An $n \times n$ adjacency matrix representing a flow network G with n vertices.

Output: An $n \times n$ matrix containing a feasible flow on G from source vertex 0 to sink vertex 1 with the maximum possible value.

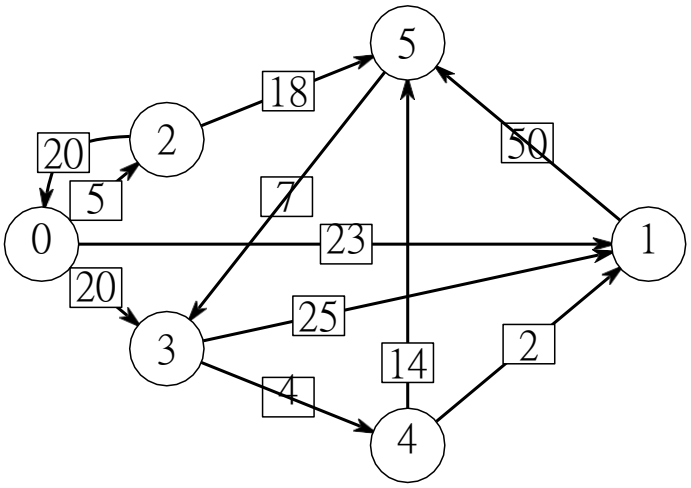
Your task is to write the body of the `MaxFlow` method. You must use the provided Java template as the basis of your submission, and put your implmentation inside the `MaxFlow` method in the template. You may not change the name, return type or parameters of the `MaxFlow` method. You may add additional methods as needed. The `main` method in the template contains code to help you test your implementation by entering test data or reading it from a file. You may modify the `main` method, but only the contents of the `MaxFlow` method (and any methods you have added) will be marked, since the `main` function will be deleted before marking begins. Please read through the comments in the template file before starting.

2 Data Format

Each input flow network is represented by an adjacency matrix, which also contains the capacities of each edge. The source vertex will always be vertex 0 and the sink vertex will always be vertex 1. Flow networks can be represented with a weighted adjacency matrix A, where entry (i, j) is the capacity of the edge from vertex i to vertex j, or 0 if no edge exists. For example, the matrix

$$A = \begin{bmatrix} 0 & 23 & 5 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \\ 20 & 0 & 0 & 0 & 0 & 18 \\ 0 & 25 & 0 & 0 & 4 & 0 \\ 0 & 2 & 0 & 0 & 0 & 14 \\ 0 & 0 & 0 & 7 & 0 & 0 \end{bmatrix}$$

corresponds to the flow network below.

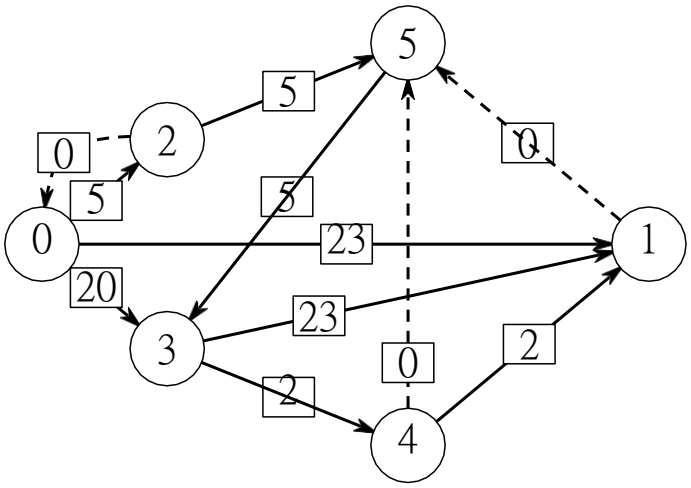


The parameter G to the `MaxFlow` function will be a matrix in the format described above. The size of the matrix can be used to determine the number of vertices in the network.

The testing code in the `main` function of the template reads a sequence of adjacency matrices and runs the `MaxFlow` function on each one. Each adjacency matrix in the sequence will be preceded by the number of vertices in the graph. For example, the network above would be encoded as follows:

```
6
  0  23   5  20   0   0
  0   0   0   0   0  50
 20   0   0   0   0  18
  0  25   0   0   4   0
  0   2   0   0   0  14
  0   0   0   7   0   0
```

The maximum flow over the network above has value 48. One possible maximum flow is given in the diagram below.



For an input network G on n vertices, the return value of the `MaxFlow` function will be an $n \times n$ matrix with entry (i, j) giving the total amount of flow assigned to the edge from vertex i to vertex j. The flow above would correspond to the following matrix:

$$F = \begin{bmatrix} 0 & 23 & 5 & 20 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 23 & 0 & 0 & 2 & 0 \\ 0 & 2 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 5 & 0 & 0 \end{bmatrix}$$

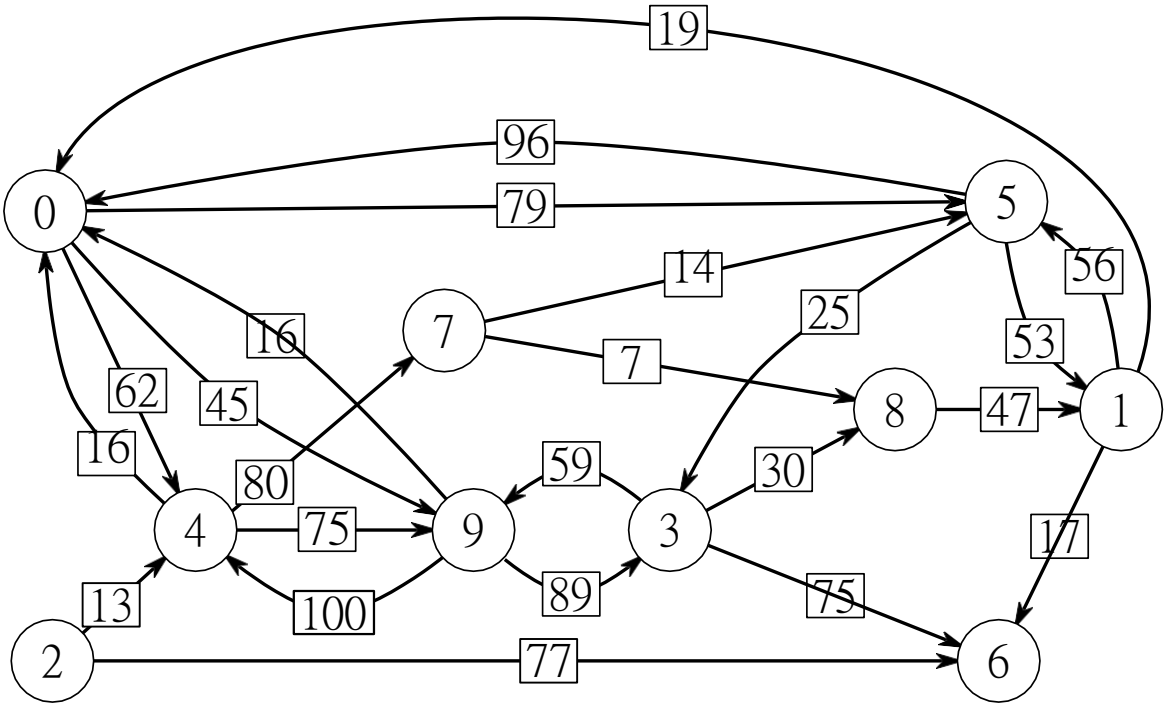
The template contains functions which will verify that the flow returned by the `MaxFlow` function is valid and print the total value of the flow if it is valid. A sample run of a model

solution on the network above is shown below. Console input is shown in blue.

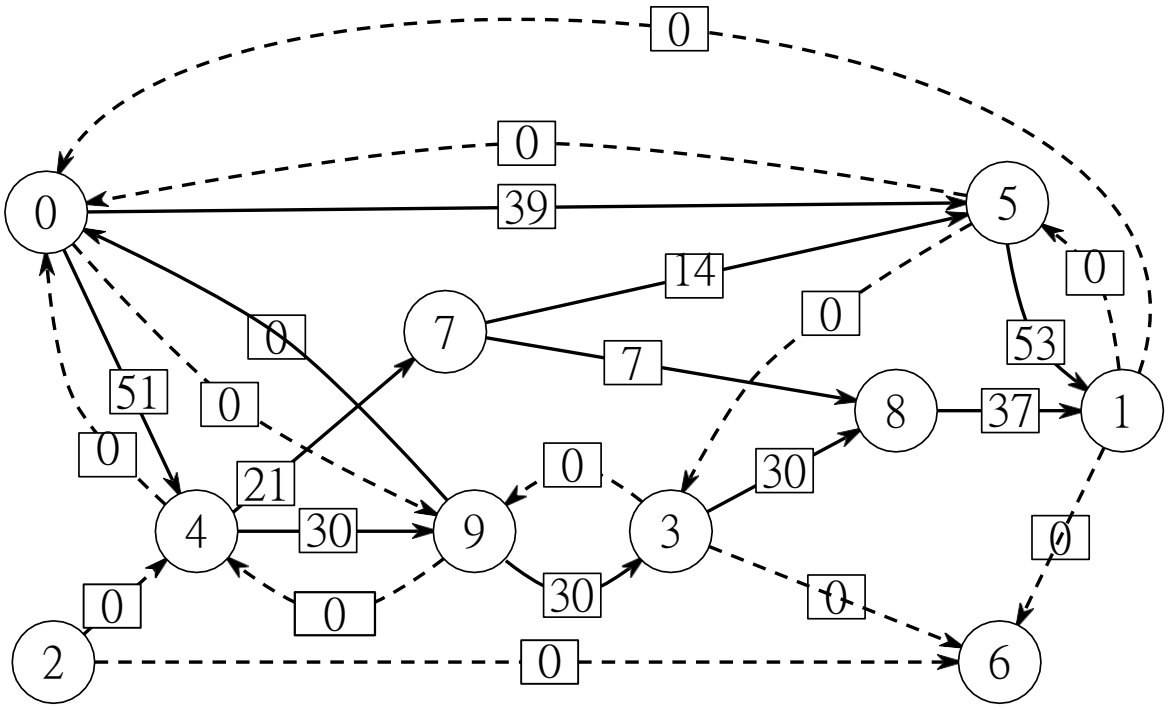
```
Reading input values from
stdin. Reading graph 1
6
 0  23   5  20   0   0
 0   0   0   0   0  50
20   0   0   0   0  18
 0  25   0   0   4   0
 0   2   0   0   0  14
 0   0   0   7   0   0
Graph 1: Max Flow Value is 48
Processed 1 graph.
Average Time (seconds): 0.00
```

3 Test Datasets

Three collections of randomly generated flow networks have been uploaded to conneX. The maximum flow values on each graph in each collection are also posted. The diagram below shows the first graph in the 10 vertex collection:



The diagram below shows a maximum flow on the graph above (the total value is 90):



4. Evaluation Criteria

The programming assignment will be marked out of 50, based on a combination of automated testing (using large test networks similar to the ones posted on conneX) and human inspection.

Score (/50)	Description
0 – 5	Submission does not compile or does not conform to the provided template.
6 – 25	The implemented algorithm is inaccurate on a significant number of tested inputs.
26 – 35	The implemented algorithm is substantially accurate but does not find a valid flow on all tested inputs.
36 – 50	The implemented algorithm finds a feasible maximum flow on all tested inputs and runs in time $O(f(n+m))$ on a graph with n vertices, m edges and maximum flow value f.

To be properly tested, every submission must compile correctly as submitted, and must be based on the provided template. You may only submit one source file. **If your submission does not compile for any reason (even trivial mistakes like typos), or was not based on the template, it will receive at most 5 out of 50**. The best way to make sure your submission is correct is to download it from conneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. The conneX site will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, conneX will automatically send you a confirmation email. If you do not receive such an email, your submission was not received. If you have problems with the submission process, send an email to the instructor **before** the due date.