IT251 Lab Assignment 11 – Network Flows

Note: Read in the input using a text file, and NOT by typing it in the console. The input file should be given as an argument while running your code. For e.g. for a file solution.cpp, compile it by 'g++ test.cpp' and run it by './a.out input.txt' , where 'input.txt' contains the input to the problem.

In this lab assignment we will implement Ford Fulkerson's algorithm for computing maximum flow in networks. Given a directed graph, with a special source and sink vertex, and edge weights representing flow capacitites, you will need to compute the maximum flow that can be sent from the source vertex s to the sink vertex t. The steps in the algorithm are as follows:

while there is a s-t path 'p' in the graph:

1. Compute the maximum flow f that can be sent along path p. (This corresponds to the least weighted edge in the path)
2. Compute the *residual graph* by subtracting the flow f from the edge weights of all the edges in path p. Add reverse edges for the flow f.

The algorithm terminates when t is no longer reachable from s. The maximum flow is the sum of the flows sent in each of the iterations of the while loop.

Input: The input text file encodes the directed graph. The first line contains two numbers, the number of nodes n and the number of edges m. Each of the m next lines represents an edge in the graph. Each edge is encoded by a triple u v c which denotes the following: a directed edge from vertex u to vertex v with edge capacity c. (The source vertex in the graph will be vertex 0 and the sink vertex will be vertex n-1)

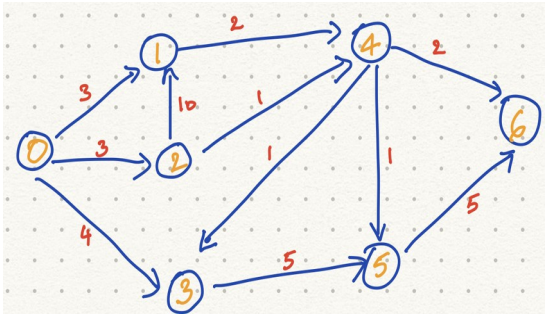
Output: The maximum flow that can be sent across this network. Simply return the maximum units of material that can be sent across, and not the graph with the flows.

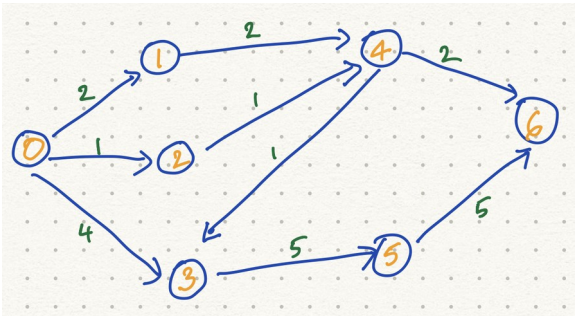
Constraints: 0 < n,m < 10,000. Capacity of each edge c < 10,000.

Sample Input/Output:
Input:
7 11
0 1 3
0 2 3
0 3 4
1 4 2
2 1 10
2 4 1
3 5 5
4 3 1
4 5 1
4 6 2
5 6 5

Output:
7

Explanation: The input corresponds to the graph on the left. The maximum flow of 7 is shown on the right.





Optional Problem : Maximum Bipartite Matching.
Use the algorithm of the previous problem to solve the maximum bipartite matching problem. The input is a set of boys and girls and a set of pairs denoting the couples that like each other.