CS 330 – Ford-Fulkerson for Maximum Flow

deadline: Friday, December 6 at 11:59 pm

In this exercise you will implement the Ford-Fulkerson algorithm for solving the maximum flow problem. Given a directed graph with positive capacities on the edges, as well as source and sink vertices, find the maximum flow. The algorithm is described in Section 7.1 of *Algorithm Design*, with the main pseudocode on page 344. Your code should run in at most $\mathcal{O}(mC)$ time, where m is the number of edges and C is an upper bound on the value of the flow, the sum over the capacities of edges exiting the source.

Tips

- The algorithm itself is simple: while you can, find an augmenting path in the residual graph and augment it. Before starting, take some time to think about how you will keep track of the necessary information. For instance, you'll need to be able to keep track of which edges in the residual graph are "forward" and which are "backward."
- You need to be able to find simple s-t paths in the residual graph. BFS will work great here - you can adapt your code from that exercise or write it from scratch.

Submission Format

On Gradescope submit `ford_fulkerson.py` or `ford_fulkerson.java` (names must match exactly) reading a text file `input` and creating a text file `output` containing your solution. You may submit other files containing additional functions and classes. The autograder will put your code in the same directory as `input`, call it from the command line, and look for the file `output` in the same directory.

Input

The input format is similar to that of the other directed graph problems we've seen (Dijkstra, Bellman-Ford). The file `input` will have $m + 4$ lines specifying a directed, weighted graph. The first line will be n, the number of vertices. The second line will be m, the number of edges. The third line will be the index s, specifying the source. The third line will be the index t, specifying the sink. Every row after that will specify an edge by giving the starting node, the ending node, and the capacity. The capacities will all be positive integers. As in the book, you can assume that s will have no edges coming into it, and t will have no edges leaving it. The vertices will be indexed 0 to $n - 1$.

We have uploaded files corresponding to the visible test cases on Gradescope. They may help you test and debug locally. Note: passing these test cases does not guarantee your code is correct!

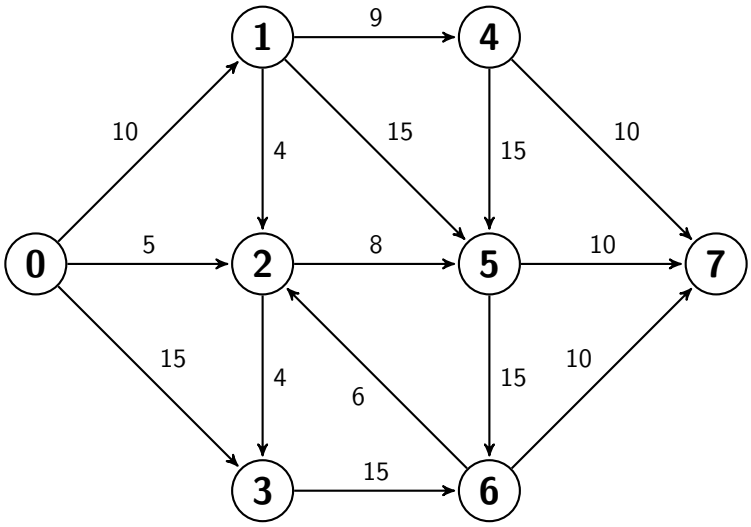
Output

Your algorithm must create a text file `output` specifying the maximum flow, edge by edge. On each line write the starting vertex, the ending vertex, and the value of the flow across that edge, all separated by commas. The value of the flow on an edge may be zero. Your output file should have m lines; they can be written in any order.

Example begins on the next page.

Example

Consider the following directed, weighted graph. It is the first example in the lecture slides `lect_11_19_FF.pdf`, available on Piazza. The source is vertex 0 and the sink is vertex 7. The maximum flow has a value of 28.




```
input:

8
15
0
7
0,1,10
0,2,5
0,3,15
1,2,4
2,3,4
1,4,9
1,5,15
2,5,8
6,2,6
3,6,15
4,5,15
5,6,15
4,7,10
5,7,10
6,7,10
```

The output file is on the next page.

output:

```
0,1,10
0,2,5
0,3,13
1,2,0
2,3,0
1,4,9
1,5,1
2,5,8
6,2,3
3,6,13
4,5,0
5,6,0
4,7,9
5,7,9
6,7,10
```