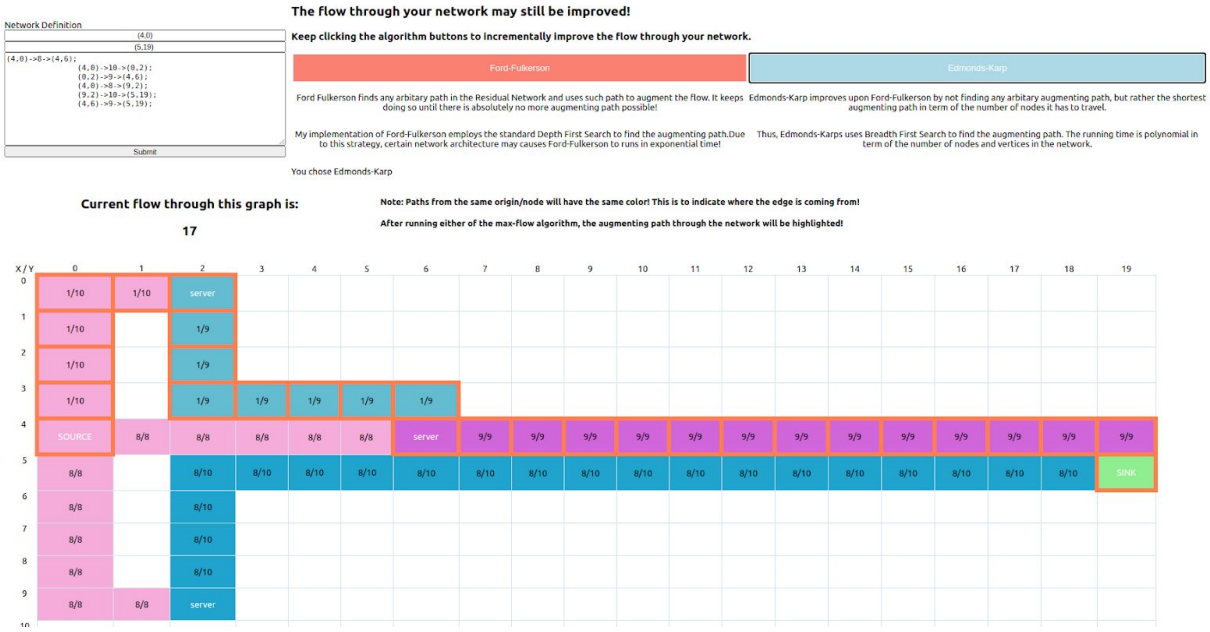
COMP 4702
Theory of Algorithms
Final Project Writeup

Khai Lai's Max Flow Visualizer



For my final project, I decided to implement a web-app to visualize both Ford Fulkerson and Edmonds-Karp algorithms. Here is the link to the deployed website (I'm currently using a free server from Heroku so that's why the URL is weird), and the link to my GitHub repository with the source code.

Max Flow Visualizer: https://infinite-fortress-56541.herokuapp.com/
GitHub Repository: https://github.com/treelover28/MaxFlowNetworkVisualizer

Implementation details

For this project, I used HTML, CSS and React.js framework. I originally planned to use the D3.js data visualization library to implement the visualizer, since it comes with a visualization module with functions specialized in drawing directed graphs. However, I just thought this approach would be like taking the easy way out, since you're just passing in an adjacency matrix into a blackbox function that "magically" does everything for you. **Thus, I implemented everything from scratch using an HTML table as a Grid and each square in the grid as a node in the network.**

A node can be both a vertex of the network, or a "path" node representative of an edge in the network. To allow users to create their own network to play with, I used a TextArea where users

could enter in the details of the network (using a simple pseudo-"programming-language" I came up with - see the Network Definition section). Essentially, users would be specifying the edges in the network and explicitly pointing out your source node and sink node. This network definition will then be parsed into an appropriate adjacency list to be used by the algorithm.

All edges coming from the same origin/node share the same color. This allows users to identify where the edge is coming from and which node it is connecting to! Furthermore, the edge's flow and capacity are also displayed as the text content of the node (which is essentially an HTML table's record).

To allow easier visualization and understanding of what the algorithms are doing, I implemented the **algorithm to run incrementally**. Instead of displaying the final network with the maximum flow immediately, only the resultant network of one augmenting path is shown at a time. This resultant network may not have the maximum flow yet, thus **users will have to keep clicking on the button to keep improving the network**. Once maximum flow has been achieved, the visualizer will alert the user! Furthermore, the augmenting path each time will be highlighted to show which edges in the network have been augmented!

Strengths of the implementation

I decided to use a grid as the environment to visualize the network. The grid-system allows easier visualization of the augmenting path while also simplifies the overall implementation of the visualizer a little bit.

The incremental visualization allows users to see which augmenting path has been chosen, and understand that Ford Fulkerson and Edmonds-Karp does not "magically" produce the maximum flow in one go, but rather through a series of iterative improvements.

The network definition allows users to create their own network and play around on their own!

Weaknesses of the implementation

Using the grid system means each node AT MOST can only have 4 edges coming out of it, since a square only has 4 sides.

Furthermore, in a real network graph, edges may intersect one another. However, under the Grid constraint, edges intersecting would create quite a convoluted network which does not aid the purpose of visualization. Thus, my implementation **strictly enforces DISJOINT** paths. Because of this, the visualizer may fail to draw an edge between two nodes, if there is already another

edge running through the space between those two nodes! Thus, users are limited to very small/ simple networks!

Users will also need to learn a special syntax to create the network, which may take a while to get used to!

Network Definition

Here is how you define a network using my program.

1. Type in where you SOURCE node is using the format (x,y)
2. Type in where your SINK node is using the format (x,y)
3. Define your network architecture, by specifying the edges connecting the nodes in your network! Here are the steps:
    a. Type in the coordinate of your first node using the same (x,y) format.Make sure there is no space between the characters
    b. Follow by typing in a right arrow ->, then type in the edge capacity
    c. Type in another right arrow ->
    d. Type in the coordinate of the node to which the edge is connecting to, using (x,y) format
    e. Finally terminate the current edge specification using a semi-colon ;
    f. The format is like this: (x,y)->edge capacity->(x2,y2);
4. You only need to specify which edge connects which two nodes. The visualizers will handle drawing the path representative of suchedge in the grid-world using Breadth-First-Search.

Here's are some sample programs

Program Name	*Small Graph*	*Big Graph*
Source	(4,0)	(5,0)
Sink	(5,19)	(6,19)
Edge Specification	(4,0)->8->(4,6); (4,0)->10->(0,2); (0,2)->9->(4,6); (4,0)->8->(9,2); (9,2)->10->(5,19); (4,6)->9->(5,19);	(5,0)->10->(1,4); (5,0)->9->(5,4); (5,0)->8->(9,4); (1,4)->5->(0,6); (1,4)->8->(2,6); (5,4)->4->(3,7); (5,4)->8->(5,7); (7,7)->10->(5,7); (5,4)->9->(7,7); (5,7)->6->(5,10); (3,7)->5->(5,10); (5,10)->6->(7,10); (12,7)->5->(7,7); (9,4)->5->(12,7); (0,6)->8->(0,13); (2,6)->5->(0,6); (7,10)->8->(6,19); (0,13)->10->(6,19); (12,7)->6->(7,10);

Conclusion

Overall, I had a lot of fun with this project! Hopefully, this will help professors and other students to have an easier time teaching and understanding these maximum network flow algorithms!