Documentation

The Ford–Fulkerson algorithm is a greedy algorithm that computes the total flow in a flow network. As long as there is a path from the source to the sink, with available capacity on all edges in the path, we send flow along one of the paths. Then we find another path, and so on. A path with available capacity is called an augmenting path.

For this project I have divided the code into four different functions. The first function is the read_file() method, which basically reads the test cases to do the breadth first search. Using this method, we can figure out the number of vertices, source and the sink, weight of each edge. The breadth_first_search() method returns true if there is a path found going from the source to the sink. First we make everything false as none of the vertices are visited. We create an array FIFO_list[] that stores all the digits from the test case. I have made a while loop that checks each path if it has been visited or not. If a node has been visited for the first time it marks it as visited. We find the adjacent vertices and go to it. We mark that as visited too. This is how we find the augmenting paths. Once we reach the sink, the function returns true.

The third function is run(), which returns the maximum flow of the algorithm. We find the bottleneck capacity and make each edge in the path. I am calling 2 different functions in the run() function: update() and print_path(). The update just keeps updating the capacity of each edge after each flow. And the print_path() is just the output function with the output statements.

All the test cases pass and give the same total flow. But I have changed the priority so the paths I take are different.

To run the code:

Through terminal

- $ python src.py <input_file>

Deliverable file output:

Augmenting Path found: 0 2 7 14 with flow: 2

Augmenting Path found: 0 5 13 14 with flow: 3

Augmenting Path found: 0 1 6 9 14 with flow: 2

Augmenting Path found: 0 2 7 9 14 with flow: 1

Augmenting Path found: 0 3 8 10 14 with flow: 2

Augmenting Path found: 0 4 11 12 14 with flow: 5

Augmenting Path found: 0 3 8 11 12 14 with flow: 2

Total flow: 17