1. (7 points) Implement and evaluate Ford-Fulkerson algorithm for maximum flow. The input is a directed graph with edge capacities, a source node and a sink node. The output is the maximum flow and an assignment of flows to edges. You will be comparing the two approaches "*Fat Pipes*" and "*Short Pipes*" to find augmented paths on Erdos-Renyi graphs.

Graph generation: You will generate graphs using the same G(N.p) model you used in Homework 6, which takes two paramaters N, the number of nodes, and p the probability of an edge being present. Set p = 2 ln N/N, which makes the graph connected with a very high probability. You can use the gnp_random_graph function in the networkx package.

https://networkx.org/documentation/latest/reference/generators.html#module-networkx.generators.random_graphs (Links to an external site.)

Assign each edge a random integer capacity in the range of (1, 100). Also assign a direction to each edge with probability 0.5. Arbitrarily choose some node to be s. Do a random walk of length 10000 starting from s (i.e., for 10000 steps, pick one of the neighbors of each node at random) and call the final node t. The random walk ensures that there is a directed path from s to t.

Data generation: Generate 10 graphs each of sizes 1100, 1200, ..., 2000 nodes using the above approach. For each size graph, run both the algorithms and check that the maximum flows are the same. Report the fraction of the times the actual flows along each edge are also the same for both approaches for graphs of each size. Finally, report the average running times of the algorithms as a function of number of nodes in a plot/table. Comment on their relative performance. Is there a clear winner?

Both capacities and flows are integers. You will compare the two approaches "Fat Pipes" and "Short Pipes" to find augmented flows in the residual network.

Fat pipes: In this approach, the augmented flows are computed by finding a path from source to sink (an st-path) in the residual graph that has a maximum capacity. The capacity of the path is the minimum (remaining) capacity of any edge (the 'bottleneck') in that path. The algorithm works like Dijkstra's algorithm except that it tries to build maximum capacity path (rather than shortest path) from the source to sink.

Short pipes : A breadth first search in the residual graph which returns the shortest path (i.e., the path with the least number of edges) with non-zero flow.

You can use the following examples to verify that your programs are working.

Examples 1: Input: (0, 3, [(0,1,1), (0,2,5), (1,2,1), (2,3,2), (1,3,6)])

Explanation: The source node is 0, the sink node is 3; the capacity of edge (0->1) is 1; the capacity of edge (0->2) is 5; etc.

Output: (3, [(0, 1, 1), (0, 2, 2), (1, 3, 1), (2, 3, 2)])

Explanation: "3" is the maximum flow; [(0, 1, 1), (0, 2, 2), (1, 3, 1), (2, 3, 2)] is an assignment of flows to edges.

Example 2: Input: (0, 4, [(0, 1, 2), (0, 3, 6), (1, 2, 3), (1, 3, 8), (1, 4, 5), (2, 4, 7), (3, 4, 9)])

Output: (8, [(0, 1, 2), (0, 3, 6), (1, 4, 2), (3, 4, 6)])

2. (1 point) Suppose you are running a conference and want to assign 40 papers to 12 reviewers. Each reviewer bids for 20 papers. You want each paper to be reviewed by 3 reviewers. Formulate this problem as a max-flow problem, i.e., describe the architecture of the network and the capacities of the edges. Assume a reviewer cannot review more than 11 papers. What is the maximum flow you can expect in your network?

3. (1 point: Hall's theorem) Consider a bipartite matching problem of matching N boys to N girls. Show that there is a perfect match **if and only if** every subset of S of boys is connected to at least |S| girls. Hint: Consider applying the Max-flow Min-cut theorem.

4. (1 point) Suppose you want to find the shortest path from node s to t in a directed graph where edge (u,v) has length $l[u,v] > 0$. Write the shortest path problem as a linear program. Show that the dual of the program can be written as $Max\ X[s]-X[t]$, where $X[u]-X[v] <= l[u,v]$ for all (u,v) *in E.* [An interpretation of this dual is given in page 290 of DPV.]

Instructions on submission:

Label the source file 'hw8.py' and the report report8.pdf. Submit the report on the Canvas site and the code at the teach interface https://teach.engr.oregonstate.edu/Links to an external site. .