This assignment asks you to implement Dinic's algorithm for determining a maximum flow in a directed network $G = (N, E)$ with positive (real) edge capacities. You can assume that the node set is $N \subseteq \{1, 2, \ldots, n\}$, with n being the largest node number appearing. Also $m = |E|$.

Input Data consist of n, source node s, and sink node t, followed by an edge list representation of G: from, to, (real) capacity. No ordering of the input edges is to be assumed. For simplicity, you can assume that if edge (i, j) appears then edge (j, i) does not. (If it is necessary for your implementation, you can also assume that the number of actual network edges m is also input.)

Output should display a maximum flow (only the nonzero edge flows) as well as the maximum flow value. Also produce the corresponding minimum cut: give both the X and $N - X$ sets as well as the (forward) cut edges, and the cut capacity.

(1) Carefully analyze the space complexity of your particular implementation in terms of n, m.

(2) Run your algorithm on the sample problems (to be supplied) and provide detailed output, as above.

Hand in your documented code as well as a clear English description of your data structures, the overall design of your code, and any special issues you encountered. Be sure to discuss the auxiliary data structures needed and how they enable you to carry out important steps efficiently.

Things to Keep in Mind:

- make your code as general and modular as possible
- check that all capacities are positive (on input)
- thoughtfully document your code; let the main routine be "self-documenting"
- work entirely with the *residual* network
- carefully design your stopping criteria
- be efficient and avoid unnecessary/repeated calculations
- after an augment, restart the search from an appropriate place
- explain how you recover the flow from the final residual network
- initialize variables appropriately