Problem Set 2—Due Wed., January 31 3PM

Guidelines for writeups

When describing an algorithm first attempt to give a high level overview of your approach, then fill in details as need to justify your time bound and correctness. Your pseudo-code should always be commented (unless it is self commenting). You are also encouraged to describe your solutions in terms of known algorithms and data structures (e.g. sort the numbers, insert into a balanced binary search tree, delete the minimum of a heap, could all be described with little or no further elaboration).

For proofs, strive for clear, clean arguments. It will always be somewhat a matter of taste which steps can be skipped, but try to avoid proofs of obvious points, and be clear on anything tricky. Define your notation carefully. This will often allow you to give a much crisper argument.

This problem set is intended to let you explore some properties of graph representations as well as timing and testing simple algorithms. There are also paper and pencil questions on network flows.

(60) Problem 1. Implement BFS using three graph implementations: graph is represented by i) an adjacency matrix, ii) adjacency list where the graph is represented by an array of linked lists. List i has the neighbors of vertex i, iii) an adjacency list, but use two arrays instead of linked lists. One array is a neighbor array of length 2m for an undirected graph, or length m for a directed graph. It has the neighbors of vertex 1, followed by the neighbors of vertex 2, ... The second array is an array which has in position i the array index of the first neighbor of i in the neighbor array.

For example, if vertex 1 has neighbors 2,3; vertex 2 has neighbor 1 and vertex 3 has neighbors 1 and 2, the neighbor array is: 2,3,1,1,2 and the index array is: 1,3,4 (assuming the first element of the array is position 1).

Do the BFS starting at vertex number 1. Find the shortest path to each vertex in the graph from vertex 1 (path length= number of arcs). At the end output for each vertex its distance from vertex 1 (or that it couldn't be reached).

Test your implementations on the random graphs generated using the program rand-graph.c (note this produces a graph in adjacency matrix form, you will need to convert this to an adjacency list form for two of your implementations.) First, for correctness, then for efficiency. Try to make your programs as fast as possible (profiling may help).

For timing, run your programs on graphs with 2000 vertices and edge probabilities of 1%, 20% and 70%. For timing purposes you may avoid doing the final output of distances.

(25) Problem 2. Extra credit: extend your BFS to implement a network flow algorithm (that is, use BFS in the ford-fulkerson algorithm to find augmenting paths).

(15) Problem 3. Describe how to modify the Ford-Fulkerson algorithm (using BFS to find augmenting paths) if we add the restriction that for each arc (i,j) there is a lower bound \( l_{ij} \) such
that the flow in arc (i,j) cannot be lower than $l_{ij}$ (you should assume an initial feasible flow, where, the flow $x_{ij}$ on each arc satisfies $l_{ij} \leq x_{ij} \leq c_{ij}$ in addition to the balance constraints).

new To help argue that your program works correctly, prove a version of the max-flow min-cut theorem for this new setting (with lower bounds). First define what the capacity of an S-T cut is in this new setting, then argue that your flow matches the capacity of a (minimum) S-T cut.

Extra credit (10): Describe how to find an initial feasible flow.

(10) Problem 4. Suppose that in addition to each arc having a capacity we also have a capacity on each node (thus if node $i$ has capacity $c_i$ then the maximum total flow which can enter or leave the node is $c_i$). Suppose you are given a flow network with capacities on both arcs and nodes. Describe how to find a maximum flow in such a network (hint: by modifying the network you can use a standard flow algorithm to solve this problem).

(20) Problem 5. Use network flows to solve each of the following problems; new Give the run time of your solution (try and make it as fast as possible)

a) You are given a directed strongly connected graph. Each arc has a cost associated with it. We want to find a minimum cost set of arcs such that removing that set of arcs makes the graph no longer strongly connected.

b) In a computer network there are $n$ processors $P_1, P_2, \ldots, P_n$, and $m$ communication lines $C_1, C_2, \ldots, C_m$. Each processor $i$ has the ability to test $t_i$ lines per day and there is a list $L_i$ which contains the communication lines that processor $i$ is able to test. Subject to these constraints we would like to be able to test all the communication lines every day. A testing schedule determines for each processor the lines it should test. Find a testing schedule or determine that no schedule can test all lines in a single day.

c) Same as b) but find the minimum number of days to test all lines (and a schedule which achieves it). Careful. Be sure your approach actually finds the minimum number of days (some simple greedy solutions fail).