CS 323.001: Data Structures and Algorithms Handout #5
Emory University, Math/CS Dept. February 26, 2019

Written #3: Mostly Graphs
Due: see Canvas

For this assignment, you'll submit a PDF.

Problem 1. Bloom. See the wikipedia article for "Bloom Filter". A Bloom filter is described by three integers: m is the number of bits, k is the number of (independent) hash functions, and n is the number of elements inserted. Typically k is small, but m and n are both large. Let p denote the probability of a false positive. We will use the asymptotic formula $p \sim \left(1 - e^{-kn/m}\right)^k$, which is accurate when m and n are large. In the following exercises, state each answer with at least 3 digits precision.

1(a). Suppose n is large, and $m = 8n$ (8 bits per item). Find the *integer* k minimizing p, and also state that value of p corresponding to this k. Repeat this exercise for $m = 16n$ and $m = 24n$.

1(b). Suppose n is large. Find the smallest ratio m/n (bits per item), and some *integer* k, achieving $p \leq 2^{-4}$. (The ratio m/n does not need to be an integer, but k should be an integer.) Repeat this exercise for $p \leq 2^{-8}$ and $p \leq 2^{-12}$.

Problem 2. TopoSort2. Given a digraph G, consider the following algorithm:

> $G' = G$
> While G' has a vertex v with indegree zero:
>   output v, and delete v (and all its edges) from G'.
> Output G'.

First, argue that if the final G' is empty, then the output vertex ordering is a topological ordering of G. Second, argue that if the final G' is not empty, then G has a cycle. Finally, assuming G is given in adjacency list form, argue that this algorithm can be implemented in $O(V + E)$ time. (Main issue: how to quickly find the next vertex with indegree zero.)

Hint: This is a new topological sorting algorithm, forget DFS here.

Problem 3. Formulas. Suppose we have a graph (or digraph) with $V \geq 4$ vertices, no self-loops, and no parallel edges. For each part, give a formula depending on V, and draw a picture of such a graph for $V = 4$. Each formula should be integer valued. You do not need to prove anything.

3(a). Suppose the graph is undirected, and not connected. What is the maximum possible number of edges?

3(b). Suppose the graph is undirected, and there are no isolated vertices. What is the minimum possible number of edges?

3(c). Suppose the graph is directed and acyclic (a DAG). What is the maximum possible number of edges?

3(d). Suppose the graph is directed and strongly connected. What is the minimum number of edges?

3(e). Suppose the graph is directed and acyclic, with no directed path of length two. What is the maximum possible number of edges? (Hint: bipartite.)

Problem 4. DFS Edge Types. See the wikipedia "Depth-First Search" article for the four types of digraph edges after a DFS traversal: "cross", "back", "tree", and "forward". The program writ3/DFSEdges.java does a DFS traversal of a digraph, saving some information in the arrays `beg`, `end`, and `edgeTo`. Write out (just on paper) a correct version of the `edgeType` method, so that it works in $O(1)$ time, and it correctly identifies the four edge types, as in the example output writ3/test-output.txt.

Problem 5. Red-Black Spanning Trees. As input, you are given a connected graph G, with V vertices and E edges. Each edge of G is colored either red or black. You are also given an integer k, $1 \leq k \leq V$. You want to output a spanning tree T in G with exactly k red edges, or else announce that no such tree exists. Devise an algorithm solving this problem, and running in time $O((k+1)E \lg V)$ (or better!).

Hint: Suppose we are given two distinct spanning trees T_1 and T_2 in G, and edge e in $T_2 - T_1$ (that is, an edge of T_2 that is not an edge of T_1). Then we can find an edge f in $T_1 - T_2$, so that $T_1' = T_1 + e - f$ is another spanning tree, as follows:

Find the cycle C in $T_1 + e$.
Find the vertex cut defined by the two components of $T_2 - e$.
Let f be an edge of C (other than e) crossing the cut.

Note tree T_1' is "one step closer" to tree T_2, in the number of shared edges.