CS 170 - Fall 2003 - Midterm 1

Associate Professor Satish Rao

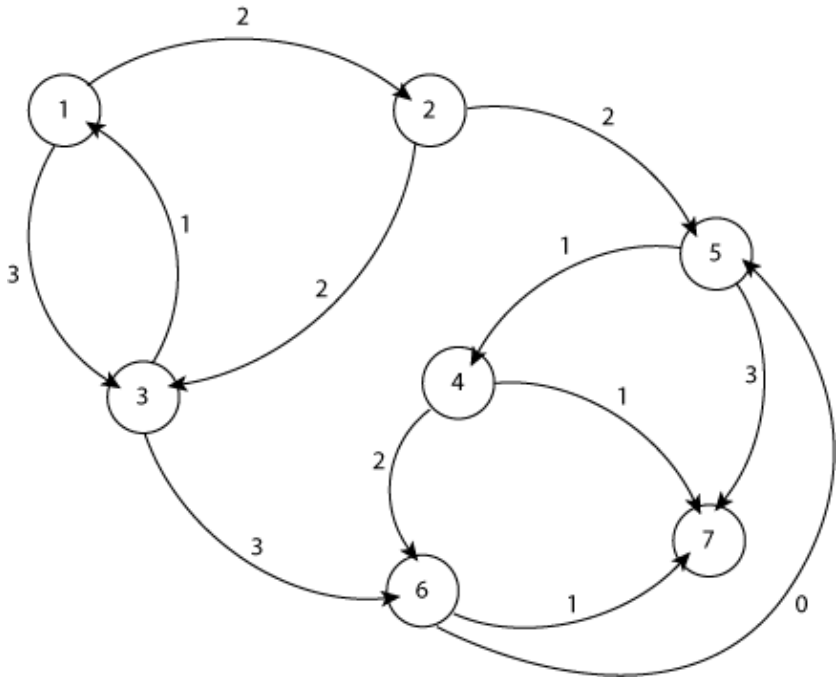
Problem 1: Recurrences (17 points)

Give asymptotically tight solutions for T(n) for the following recurrences. Give only brief justification.

1. $T(n) = T(n/2) + n$
2. $T(n) = T(4n/5) + T(n/5) + n$
3. $T(n) = 8T(n/2) + n^3$
4. $T(n) = 2T(n^{1/3}) + 1$

Problem 2: Graph Algorithms (24 points)

Consider the following graph.



1. Write down an adjacency list representation of the graph. (Write the edges in clock-wise order, where the closest one to twelve o'clock that is "past" twelve is first.)
2. Compute a depth first search tree. (Use the ordering provided above.)
3. What are the strongly connected components?
4. Compute the shortest path distances, give the order in which the nodes are "known" to have the correct distance for Dijkstra's algorithm.

Problem 3: Algorithm Properties (17 points)

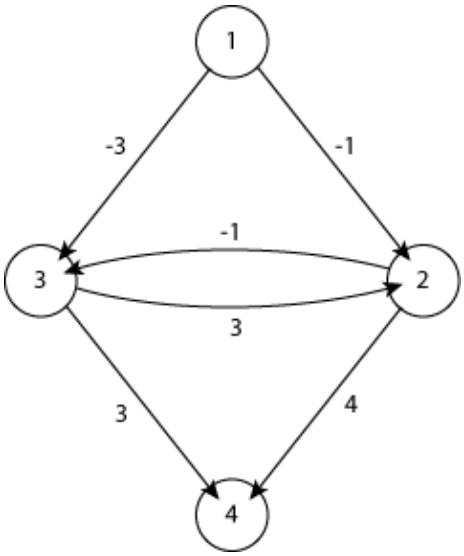
1. Consider a depth first search of a graph. Argue that a node u is in a cycle if and only if it or one of its descendants in a DFS has a back edge to an ancestor of u or to u.
2. Given an s and t in a graph G, a bottleneck path from s to t is the path with the largest minimum weight edge. Give an algorithm for computing the bottleneck s—t path.

Problem 4: Shortest Paths (17 points)

Consider the following version of the Bellman-Ford algorithm.

> Set $d(v) = \infty$ for all $v \in V$, set $d(s) = 0$. Do x times, for all edges, *update*(u,v).
>
> Here *update*(u,v) is defined as follows:
> update(u, v): { if $d(v) > d(u) + d(u, v)$ then $d(v) = d(u) + d(u, v)$. }
>
> Notive that the order of the updates on the edges is arbitrary.

1. Given an n-node graph with no negative cycles, how large should x be to guarantee that the distance labels are correct for a graph with n nodes and m edges. (Be careful, x is the *number of times* that the loop runs.)
2. Given the following graph, what is the smallest value of x where the algorithm has the correct distance labels for any order of the edge relaxations in an iteration.



3. How should you modify the algorithm, so that in the first iteration where the distance labels are correct, you can halt? (Modify update to keep track of something and check for it at the end of the set of relaxations.)
4. For any graph with a well defined shortest path tree, describe an ordering of the edges, where one iteration of Bellman-Ford algorithm suffices to relax all

the edges. (Note: you may assume that you know a shortest path tree.)

Problem 5: Hash Functions (17 points)

A family H of hash functions $h : U \to [m]$ is *2-good* if for any pair $x, y \in U$, and any $s, t \in [m]$, the probability that $h(x) = s$, $h(y) = t$ is at most $1/m^2$.

1. Show that the family of all functions from U to $[m]$ is 2-good.
2. A collision for a pair of elements under a hash function h, is a set of elements $x, y \in U$, $x \neq y$ where $h(x) = h(y)$. Show that the expected number of collisions is at most (n choose 2)/m on n items when using a 2-good hash function.

Problem 6: Shortest Paths and MST's (10 points)

Consider a complete undirected graph G, with edge weights that obey the triangle inequality, i.e. $w(u, v) \leq w(u, x) + w(x, v)$ for all u, v, x.

1. Argue that for any s, the shortest path tree rooted at s forms a star. (That is, a shortest path to any $v \neq s$ consists of a single edge from s.)
2. Give an example of a four node graph of the form above where the minimum spanning tree is a simple path starting at s.
3. Consider the following algorithm for computing a compromise between the MST that is not a star.

   Compute the minimum spanning tree and call it T. Choose the closest node u, where the shortest (and only) path from s to u in T is more than twice as long as $w(s, u)$ (the shortest path in G), remove the edge that leads to s in T, and add the edge (s, u) to T.

   1. Argue that T is a tree after each iteration.
   2. At the end of the algorithm, argue that for any u the path between s and u in T has length no longer than two times $w(s, u)$ (the shortest path in G).
   3. Assuming that the original MST is a simple path with one end at s, upper bound the total weight of the final T in terms of the weight of the MST. Justify your upper bound. (Hint: note that the weight of an added edge (s, u) is at most 1/2 the length of the path in the tree right before it was added. Can this section of path be considered again?)