Algorithms and Complexity
Semester 2, 2012

Problem Set 2

J. Mestre

You should solve all problems above the line before the tutorial. Tutors will call out students to solve these problems in front of the class during the tutorials.

You should attempt to solve some of the problems below the line. Tutors will go over these problems during the tutorial.

Problem 1: An undirected graph $G = (V, E)$ is said to be bipartite if its vertex set V can be partition into two sets A and B such that $E \subseteq A \times B$. Design an $O(n + m)$ algorithm to test whether a given input graph G is bipartite.

1. Suppose we run BFS from some vertex $s \in V$ and obtain layers $L_1, \ldots, L_k$. Let (u, v) be some edge in E. Show that if $u \in L_i$ and $v \in L_j$ then $|i - j| \leq 1$.
2. Suppose run BFS on G. Show that if there is an edge (u, v) such that u and v belong to the same layer then the graph is not bipartite
3. Suppose G is connected and we run BFS. Show that if there are no intra-layer edges then the graph is bipartite
4. Put together all the above to design an $O(n + m)$ time algorithm for testing bipartiteness.

Problem 2: Let T be a rooted tree. For each vertex $u \in T$ we use T_u to denote the subtree of T rooted at u. Assume each vertex $u \in T$ has associated a value $A[u]$. Let $B[u] = \min \{A[v] : v \in T_u\}$. Design an $O(n)$ time algorithm that given A, computes B.

Problem 3: Let G be a connected undirected graph. You task is to design a linear time algorithm for finding all cut edges.

1. Derive a criterion for identifying cut edges that is based on the `down-&-up`$[\cdot]$ values we defined in lecture.
2. Use this criterion to develop an $O(n + m)$ time algorithm for identifying all cut edges.

Problem 4: Let $G = (V, E)$ be an n vertex graph. Suppose G has a pair of vertices s and t such that $\text{dist}(s, t) > n/2$.

1. Argue that there exists a vertex $u \neq s, t$ such that every path from s to t goes through u.
2. Give a linear time algorithm for finding such a vertex u. Analyze your algorithm.

Problem 5: Give an $O(n)$ time algorithm to detect whether a given undirected graph contains a cycle. If the answer is yes, the algorithm should produce a cycle. (Assume adjacency list representation.)

Notice that |E| could be $\Omega(n^2)$, so in such a graph, the algorithm runs in sublinear time!

Problem 6: Let G be an undirected graph with vertices numbered $1 \ldots n$. For a vertex i define $\text{small}(i) = \min\{j \mid j \text{ is reachable from } i\}$, that is, the smallest vertex reachable from i. Design an $O(n+m)$ time algorithm that computes $\text{small}(i)$ for *every* vertex in the graph.

Problem 7 (Advanced): Let G be a directed graph with vertices numbered $1 \ldots n$. For a vertex i define $\text{small}(i) = \min\{j \mid j \text{ is reachable from } i\}$, that is, the smallest vertex reachable from i. Design an $O(n+m)$ time algorithm that computes $\text{small}(i)$ for *every* vertex in the graph.

Problem 8 (Advanced): In a directed graph, a *get-stuck* vertex has in-degree $n-1$ and out-degree 0. Assume the adjacency matrix representation is used. Design an $O(n)$ time algorithm to test if a given graph has a get-stuck vertex. Yes, this problem can be solved without looking at the entire input matrix.

Problem 9 (Advanced): Consider the following random experiment. We start we an empty graph $G = (V, \emptyset)$ and we grow the graph with the following process. Repeatedly pick two vertex $u, v \in V$ uniformly at random, add the edge (u, v) to G, and check whether the graph is connected; the process stops when the graph is connected. Let $T(n)$ be the expected number of edges that this random experiment creates.

Implement a routine for testing connectivity. Use your code to run experiments to make a conjecture about the asymptotic growth of $T(n)$.