**ESO207A/211: Data Structures and Algorithms**

Homework *5* **Due: November 6, 2014**

**Instructions.**

1. Start each problem on a new sheet. Write your name, Roll No., the course number, the problem number, the date and the names of any students with whom you collaborated. Note that you may collaborate with other students but you have to write your own answers.
2. For questions in which algorithms are asked for, your answer should take the form of a short write-up. The first paragraph should summarize the problem you are solving and what your results are (time/space complexity as appropriate). The body of the essay should provide the following:
   (a) A description of the algorithm in English and/or pseudo-code, where, helpful.
   (b) At least one worked example or diagram to show more precisely how your algorithm works.
   (c) A proof/argument of the correctness of the algorithm.
   (d) An analysis of the running time of the algorithm.

   Remember, your goal is to communicate. *Full marks will be given only to correct solutions which are described clearly.* Convoluted and unclear descriptions will receive *low marks*.

---

**Problem 1.** *Breadth-First Search.* Let $G = (V, E)$ be an undirected graph. Let $\pi(v)$ denote the predecessor node of $v$ in a BFS shortest-path tree of $G$.

1. Consider the shortest-path tree for $G$ obtained during the computation of BFS and let $d[]$ be the distance array computed during BFS. Show that if $\{u, v\}$ is an edge of the graph, then, either $d[u] = d[v]$, or, $d[u] = d[v] + 1$, or $d[v] = d[u] + 1$. That is, edges are either tree edges or they go between vertices in consecutive layers.
2. A graph is said to be bi-partite if the set of vertices $V$ can be partitioned into sets $V_1$ and $V \setminus V_1$ such that every edge $\{i, j\}$ exists between $i \in V_1$ and $j \in V \setminus V_1$. That is, there is no edge between pairs of vertices in $V_1$ or between pairs of vertices in $V_2$, edges only go between vertices in $V_1$ and vertices in $V_2$. [1]

   - Show that a graph is bi-partite if and only if it has no cycles of odd length.
   - Give an $O(|V| + |E|)$ algorithm to determine whether $G$ is bi-partite or not.

[1] Another way of defining bi-partiteness is that there is a way to color each of the vertices in $V$ using one of two colors, blue and green, such that the end points of every edge has different color. That is, there are no edges between blue vertices, and no edges between green vertices. The only edges are between some blue vertex and a green vertex.

**Problem 2.** *BFS* The *diameter* of a tree $T = (V, E)$ is defined as $\max_{u,v \in V} \delta(u, v)$, that is, the largest of all pair-wise shortest-path distances. Give an efficient algorithm to compute the diameter of the tree and analyze the running time of your algorithm.

**Problem 3.** *DFS Examples.*

1. Give an example of a directed graph $G = (V, E)$ where there is a path from $u$ to $v$ and in a depth-first search of $G$, $d[u] < d[v]$, but $v$ is not a descendant of $u$ in the depth-first forest produced.

2. Give an example of a directed graph $G = (V, E)$ where there is a path from $u$ to $v$ and in the depth-first search of $G$, $d[v] > f[u]$.

**Problem 4.** *DFS.* A directed graph $G = (V, E)$ is singly connected if between any pair of vertices $u$ and $v$, there is at most one directed path. Give an efficient algorithm to determine whether or not a directed graph is singly connected.

**Problem 5. This problem is not to be submitted. It is for discussion. Problem 22-2 of CLRS.**

Let $G = (V, E)$ be a connected, undirected graph. An *articulation point* of $G$ is a vertex whose removal disconnects $G$. A *bridge* of $G$ is an edge whose removal disconnects $G$. A *biconnected component* of $G$ is a maximal set of edges such that any two edges lie on a common simple cycle. (Or, that a biconnected component is a maximal set of vertices and its induced sub-graph such that it does not contain an articulation point). We can determine articulation points, bridges and biconnected components using depth-first search. Let $G_\pi = (V, E_\pi)$ be a depth-first tree.

**a.** Prove that the root of $G_\pi$ is an articulation point if and only if it has two or more children in $G_\pi$.

**b.** Let $v$ be a non-root vertex of $G_\pi$. Prove that $v$ is an articulation point of $G$ if and only if $v$ has a child $s$ such that there is no backedge from $s$ or any descendant of $s$ to a proper ancestor of $v$.

**c.** Let

$$v.low = \min \begin{cases} v.d, \\ w.d : (u, w) \text{ is a back edge for some descendant } u \text{ of } v \end{cases}$$

Show how to compute $v.low$ for all vertices $v \in V$ in $O(E)$ time.

**d.** Show how to compute all articulation points in $O(E)$ time.

**e.** Prove that an edge of $G$ is a bridge if and only if it does not lies on any simple cycle of $G$.

**f.** Show how to compute all the bridges of $G$ in $O(E)$ times.

**g.** Prove that the biconnected components partition the non-bridge edges of $G$.

**h.** Give an $O(E)$ time algorithm to label each edge $e$ of $G$ with a positive integer $e.bcc$ such that $e.bcc = e'.bcc$ if and only if $e$ and $e'$ are in the same biconnected component.

## Hints

**Problem 1.** If there is an odd cycle $v_1, v_2, \ldots, v_{2k+1}$, then, $v_1$ is colored blue (say), $v_2$ must be green, and then $v_3$ must be blue, and so on. The odd numbered vertices must be blue and even numbered are green. But, then, this is a cycle, so there is an edge from $v_{2k+1}$ to $v_1$, which are both blue. Hence the graph is not bi-partite.

Conversely, without loss of generality assume that $G$ is connected–otherwise, apply the above argument to each of the connected components of $G$. Start with any source vertex $s$, color it blue and start a BFS from $s$. For every vertex $v$ with $d[v]$ even, color $v$ as blue and if $d[v]$ is odd then color it as green. Now scan all the edges $\{u, v\}$ in the adjacency list and check if both $u$ and $v$ are of the same color. If so, then, the graph is not bi-partite and otherwise, the graph is bi-partite.

*Correctness:* if the algorithm reports the graph to be bi-partite then it is correctly bi-partite, since there are no edges between blue vertices or between green vertices. Now suppose that the algorithm reports that the graph is not bi-partite. That is, there is an edge $\{u, v\}$ with $u$ and $v$ both colored the same. Hence, $u$ and $v$ belong to the same layer (i.e., $d[u] = d[v]$), since the other alternative is that $d[u]$ and $d[v]$ differ by some multiple of 2, which is not allowed by part (i). In the BFS tree, let $w$ be least common ancestor of $u$ and $v$, that is, $w$ is the vertex with the highest $d[w]$ value that is a common ancestor of both $u$ and $v$. Then, the cycle consisting of the path in the tree from $w$ to $u$, the edge $\{u, v\}$ and the path in the tree from $v$ to $w$, has cycle length $(d[u]-d[w])+1+(d[v]-d[w])$. Since, $d[u] = d[v]$, the cycle length is odd, and we have an odd cycle, and hence the graph is not bi-partite.

**Problem 2.** Argue the correctness of the following procedure. First do a BFS from the root $r$ of the tree. If the tree is unrooted, do a BFS from any node $r$. Let $u$ be the vertex with the largest value of $d[u]$. Now do a BFS from $u$ and let $v$ be the vertex furthest from $u$ (largest value of $d[v]$). Then, $d[v] = \delta(u, v)$ is the diameter. Note that this algorithm works only when the graph $T$ is a tree. It does not work for general undirected graphs.

**Problem 4.** A standard approach to this problem is to check if a given vertex $u$ has at most one directed path to other vertices. This can be achieved by doing a DFS from $u$ and checking if there are no forward edges and no cross edges within the same component. Now, DFS can be done starting from each vertex in $V$ and the above condition is checked.