Kris Harper
CMSC 27500
May 15, 2009

Homework 5

Problem 1. *Let G be a connected graph in which every DFS-tree is a Hamilton path (rooted at one end). Show that G is a cycle, a complete graph, or a complete bipartite graph in which both parts have the same number of vertices.*

Proof. Suppose that G is not a complete graph or a cycle. Let T be a DFS-tree of G. Since T is a Hamilton path, group every other vertex of T into two sets of vertices. Note that none of these vertices are connected, since G is not a cycle. Moreover, since G is not complete, there are at least two vertices of G which do not share an edge. Then G must be bipartite and complete. On the other hand, if G is not a complete bipartite graph, then we can group the vertices as before, but now we know that two vertices of the same group can be connected. Since T is a path, this must create a cycle. Moreover, since G is not complete, we know there are two vertices which do not share an edge and this ensures us the cycle doesn't overlap. This completes the proof. □

Problem 2. *1) Let G be a connected graph and T a DFS-tree of G, where the edges of T are oriented from parent to child, and the back edges from descendant to ancestor. For $v \in V$, set*

$$g(v) = \min\{f(w) \mid (v,w) \in E(G)\backslash E(T)\}$$

$$h(v) = \min\{f^*(w) \mid (v,w) \in E(T)\}.$$

Show that:
a) The function f^* may be computed recursively by the formula

$$f^*(v) = \min\{f(v), g(v), h(v)\}.$$

Proof. If v has no connection through T and one back edge to an ancestor of v, then $f^*(v) = f(v)$. If v is connected to an ancestor through precisely one back edge, then $g(v)$ will give $f^*(v)$. Now if v is connected through T and one back edge, then we use $h(v)$ to find the ancestors of the lowest valued edge in T along this path. We apply h recursively and eventually use g to obtain $f^*(v)$. Thus $f^*(v) = \min\{f(v), g(v), h(v)\}$. □

b) A nonroot vertex v of T is a cut vertex of G if and only if $f(v) \leq h(v)$.

Proof. Suppose that a nonroot vertex v of T is a cut vertex of G. Then it has a child, u, no descendent of which dominates a proper ancestor of v. Then $(u, v) \in E(T)$ and so $h(v) = f^*(u) = f(u) \geq f(v)$. Conversely, suppose that $f(v) \leq h(v)$. Then

$$f(v) \leq \min\{f^*(w) \mid (v,w) \in E(T)\}.$$

Since $f(v)$ is less than or equal to the f values of all possible ancestors, we must conclude that v has a child no descendent of which dominates v. □

2) Refine the depth-first search algorithm so that it returns the cut vertices and the blocks of a connected graph.

Proof. At each vertex in the stack, check to see if $f(v) \leq h(v)$. If it is, then mark this vertex as a cut vertex, and all of its ancestors as part of a block. This finds all the cut vertices and blocks of a graph. □

Problem 3. *Describe an algorithm based on directed depth-first search which accepts as input a tournament T and returns a directed Hamilton path of T.*

Proof. Create a stack S. Each arc of T is added to S one at a time. After an arc is added, all arcs following it are considered one at time. Any arc pointing in the same direction is candidate to follow. If the path created by following a series of acs leads to a dead end, i.e., an arc in the wrong direction, the path is deconstructed to the last branching point, at which point another arc is chosen. Eventually, no dead ends will be found and the algorithm terminates. □

Problem 4. *Let D be a digraph, and let F be a DFS-branching forest of D. Denote by D' the converse of the digraph obtained from D by deleting all cross edges. Assume D' has just one component.*
1) Show that the set of vertices reachable from the root r in D' induces a strong component of D.

Proof. Let C be a strong component of D. We know that $C \cap F$ is a branching, and this branching is entirely contained in F. Note that $C \cap F$ is a subset of the vertices in D' reachable from r. Moreover, since D' contains no cross arcs of D, no connections are broken in $C \cap F$ from C. Therefore there must be some strong component of D' contained in the set of vertices reachable from r. □

2) Apply this idea iteratively to obtain all strong components of D (taking care to select each new root appropriately).

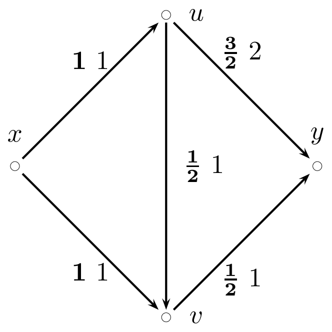
Proof. Select a root r_1 in D'. The set of vertices reachable from r in D' includes a strong component, C_1, of D. Now select r_2 in D' such that $r_2 \notin C_1$. The set of vertices reachable from r_2 includes another strong component of D, and since $r_2 \notin C_1$, these components are distinct. Continue in this way until there exists no such r_n. □

3) Implement this procedure by employing branching-search.

Proof. Take a vertex $r_1 \in D'$ and find the strong component, C_1, as in Part 2). Now add all neighbors of r_1 to a stack, S. For each element in the stack, consider if it's in C_1 and if it has neighbors. If it's not in C_1, call it r_2. If it is in C_1, add it's neighbors to S and throw it away. Continue in this way until a vertex, r_2, is found such that r_2 is not in the first strong component. Find the strong component generated by r_2 and then consider all its neighbors. Continue until all vertices are covered by strong components. □

Problem 5. *Give an example to show that not all maximum flows in an initial-valued integer network need be integer-valued.*

Proof. Consider the following flow where bolded values are the maximum flow and non-bolded values are the capacities.



□