

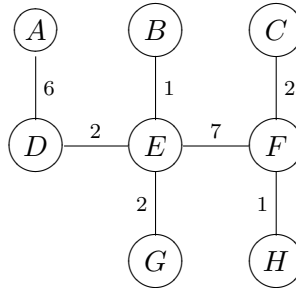
# Problem Set 3

CS157 - Spring 2009

Due: Tuesday March 17, 2009 10:30am

## Problem 1: Minimum Spanning Tree Updates

- (a) Suppose that a graph  $G$  has a minimum spanning tree already computed. Provide an efficient algorithm that updates the MST when a new node of degree 2 is added to  $G$ . No proof of correctness or runtime is necessary.
- (b) Provide the resulting MST after a vertex  $X$  is connected to nodes  $A$ ,  $B$ , and  $C$  of the following MST with weights  $w(XA) = 4$ ,  $w(XB) = 8$ , and  $w(XC) = 3$ .



- (c) Let a *star* be defined as a tree in which all non-root vertices are children of the root. Give an  $O(|V|)$  algorithm to update a star MST with the addition of a new vertex of degree  $k$ , where  $k$  may range from 1 to  $|V|$ . Prove the algorithm's correctness.
- (d) Bonus (write-up not required for full credit): Give an  $O(|V| + k \log k)$  algorithm for the general case of updating an MST with the addition of a node with degree  $k$ . Prove the runtime of the algorithm. Hint: try creating a graph with  $O(k)$  edges upon which you can run Prim or Kruskal's algorithm as an intermediate step to solving the problem.

## Problem 2: Optimized Minimum Spanning Tree

Suppose that all edge weights in a graph  $G$  are integers in the range from 1 to  $W$ , where  $W \ll |V| + |E|$ .

- (a) Provide a more efficient version of Kruskal's algorithm to find the minimum spanning tree of  $G$ . Prove the runtime of your algorithm.
- (b) Provide a more efficient version of Prim's algorithm to find the minimum spanning tree of  $G$ . Prove the runtime of your algorithm.

### Problem 3: Divide-and-Conquer Strongly Connected Components

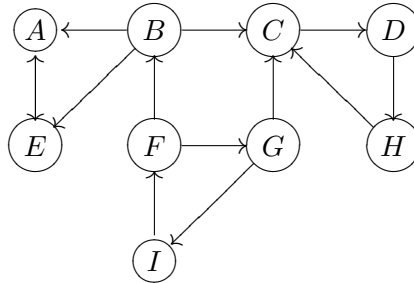
Here is an alternative algorithm for finding the strongly connected components (SCC) of a graph:

```

Components(G):
  If G has no edges then
    forall  $v \in V$  Output  $\{v\}$ 
  Else
    Select a random vertex  $v$  from  $V$ 
     $SCC \leftarrow Pred(G, v) \cap Desc(G, v)$ 
    Output  $SCC$ 
    Components( $\langle Pred(G, v) - SCC \rangle$ )
    Components( $\langle Desc(G, v) - SCC \rangle$ )
    Components( $\langle Rem(G, v) \rangle$ )
  
```

Let  $Desc(G, v)$  be the set of vertices in  $G$  that can be reached from  $v$  and let  $Pred(G, v)$  be the set of vertices from which  $v$  may be reached. Let  $Rem(G, v)$  be all the remaining vertices in  $G$ . Assume that  $\langle V \rangle$  refers to the subgraph containing vertices  $V$  and all of their incident edges in  $G$ .

- (a) Draw the *meta-graph* of the following graph, as defined on page 92 of the text. Label each node of the meta-graph with the node from the corresponding connected component whose label occurs first in the alphabet.



- (b) Suppose that *Components* is run on the original graph. List the nodes of the meta-graph in the order that their corresponding connected components are outputted. For this part of the problem only, assume that the randomly selected vertex is always the node whose label occurs first in the alphabet.
- (c) Prove the correctness of *Components*.
- (d) What are the worst- and best-case runtimes for *Components*? Consider both the organization of the input graph and the choice of the random vertex as potential variables. Give your answers in big- $O$  notation, and explain each case. Assume that the runtimes of *Pred* and *Desc* are linear to the number of edges in their input graphs, the intersection and subtraction operations are linear to the number of nodes, and *Rem* and  $\langle V \rangle$  require constant time.

- (e) Conjecture the average-case runtime for *Components*.
- (f) What are some possible advantages that *Components* might have over the depth-first-search-based algorithm discussed in class and the text?

### Problem 4: Ternary Huffman Encoding

Most hard disks store information in binary digits, or “bits”. Suppose that a ternary hard disk has been developed which uses ternary digits, represented as 0, 1, or 2. We wish to adapt Huffman’s algorithm to efficiently encode data to this disk.

- (a) Suppose we are given the following alphabet  $\Gamma$  with known frequencies:  $\{A : 40, B : 30, C : 20, D : 35, E : 70, F : 5\}$ . Display the most efficient ternary encoding tree for  $\Gamma$  (no proof is necessary). Order the nodes at each level of the tree such that the frequency corresponding to each node decreases from left to right.
- (b) Prove that if the number of symbols in an input alphabet is odd, then the most efficient prefix-free encoding corresponds to a full ternary tree.
- (c) Provide a modified version of Huffman’s algorithm to find the most efficient prefix-free ternary encoding tree for an alphabet of size  $n$ , where the characters occur with known frequencies  $f_1, f_2, \dots, f_n$ .
- (d) Prove the correctness and runtime of your algorithm.