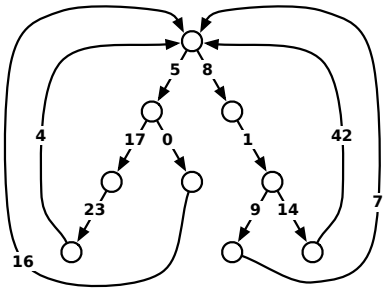
CS 473: Undergraduate Algorithms, Spring 2009
HBS 6.5

1. (a) Describe and analyze and algorithm to find the *second smallest spanning tree* of a given graph G, that is, the spanning tree of G with smallest total weight except for the minimum spanning tree.

   (b)* Describe and analyze an efficient algorithm to compute, given a weighted undirected graph G and an integer k, the k smallest spanning trees of G.

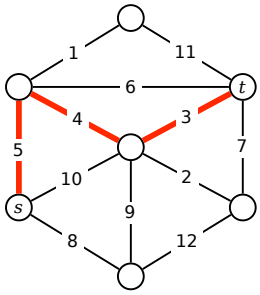
2. A *looped tree* is a weighted, directed graph built from a binary tree by adding an edge from every leaf back to the root. Every edge has a non-negative weight.



   (a) How much time would Dijkstra's algorithm require to compute the shortest path between two vertices u and v in a looped tree with n nodes?

   (b) Describe and analyze a faster algorithm.

3. Consider a path between two vertices s and t in an undirected weighted graph G. The *bottleneck length* of this path is the maximum weight of any edge in the path. The *bottleneck distance* between s and t is the minimum bottleneck length of any path from s to t. (If there are no paths from s to t, the bottleneck distance between s and t is ∞.)



The bottleneck distance between s and t is 5.

   Describe and analyze an algorithm to compute the bottleneck distance between *every* pair of vertices in an arbitrary undirected weighted graph. Assume that no two edges have the same weight.