CSCI 3104 Recitation- Trees

Michael Levet

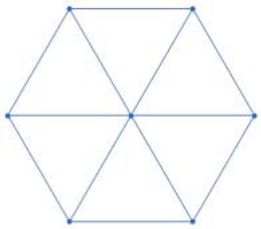
1 Learning Objectives

The primary goal of recitation this week is to (re)familiarize students with basic notions of trees that arise in this course, as well as future CS courses. In particular, we have the following learning outcomes.

- Students will correctly identify and construct examples of trees and non-trees.
- Students will exchange edges to construct one spanning tree from another.
- Students will construct minimum weight spanning trees on weighted graphs.

2 Trees

(Recommended) Problem 1. Recall the Wheel graph on seven vertices, W_7, pictured below.



Do the following.

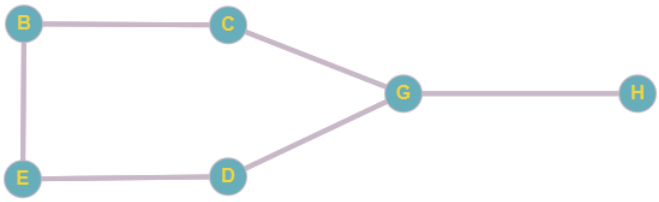
(a) Construct a subgraph of W_7, which we call H, that spans W_7. That is, H uses all the vertices of W_7.

(b) Construct a subgraph of W_7 that spans W_7, contains cycles, and is not connected.

(c) Construct a subgraph of W_7 that spans W_7, does not contain cycles, and is not connected.

(d) Construct a subgraph of W_7 that spans W_7, is connected, but has no cycles. We call such a subgraph a *spanning tree* of W_7.

(Recommended) Problem 2. Consider the following graph, which we call G.



Do the following.

(a) How many spanning trees does G have?

(b) Consider the following spanning trees of G. We note that T' contains an edge e which does not belong to T.What structure do you observe when adding e to T? [**Note:** We denote $T \cup e$ as the graph resulting from adding e to T.]

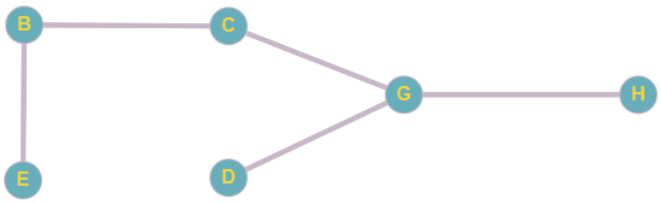


Figure 1. We label this spanning tree as T.

and:

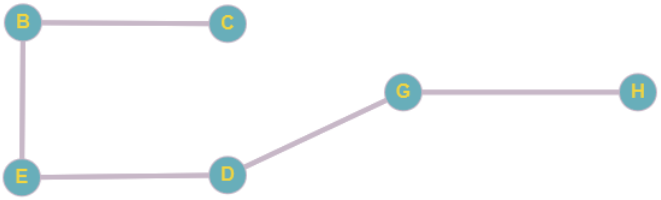


Figure 2. We label this spanning tree as T'.

(c) Let e be the edge in T' that does not belong to T. Identify the edge f in $T \cup e$, such that if we remove f from $T \cup e$, we are left with T'. This process is called *exchanging edges*, and this is a key technique in establishing the correctness of our spanning tree algorithms.

(d) Consider the spanning tree T'' below. Describe how to exchange edges to obtain T'' from T'.

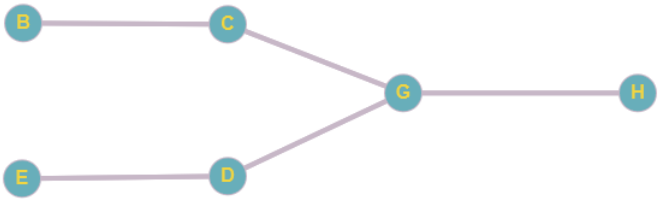
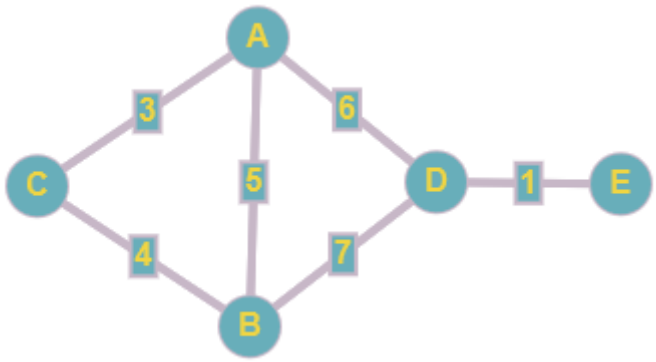


Figure 3. We label this spanning tree as T''.

(Recommended) Problem 3. Consider the following graph G, where the edges are weighted.



Consider the following spanning tree of G.

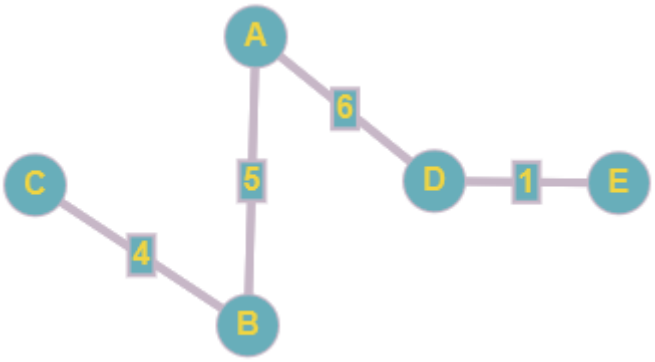


Figure 4. We label this spanning tree as T.

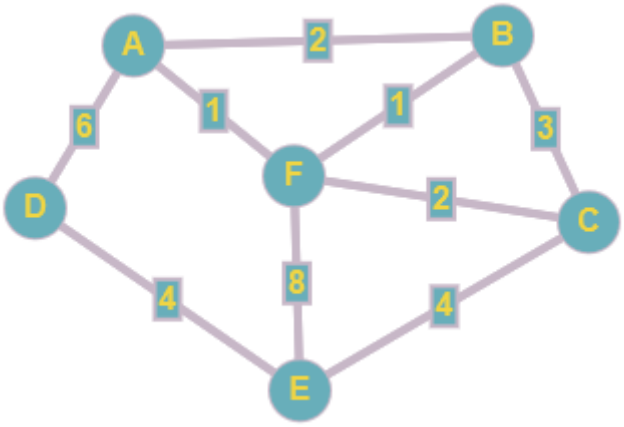
Do the following.

(a) The *weight* of T, denoted $\text{wt}(T)$, is the sum of the edge weights contained in T. Determine $\text{wt}(T)$.

(b) Find a spanning tree T' of T such that $\text{wt}(T') < \text{wt}(T)$. Clearly indicate the edges you exchanged to obtain T' from T.

Remark. Problem 3 served to illustrate the *Cycle Property*, which states the following: Suppose that the graph G has distinct edge weights. Suppose that e is an edge of the graph G and there exists a cycle C in G such that e is the heaviest weight edge on C. Then e does not belong to any minimum-weight spanning trees of G.

(Recommended) Problem 4. In this problem, we seek to build a minimum-weight spanning tree for the graph G below. Use a greedy algorithm that adds edges one at a time to a set S, using the lowest-weight edges first. If we encounter an edge e that would create a cycle (considering **only** the edges in S), we discard e.



Do the following.

(a) Clearly identify the first three edges selected by the algorithm, including the order in which they were selected. It does not matter as to the order in which two edges of the same weight were selected.

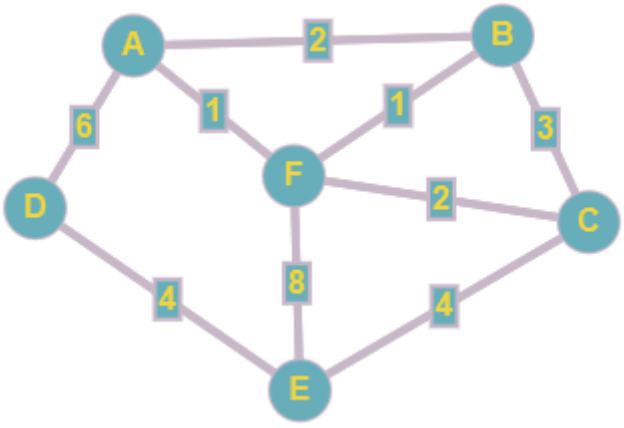
(b) What will the algorithm do when it encounters the edge BC?

(c) What will the algorithm do when it encounters the edge ED?

(d) Draw the final minimum-weight spanning tree and determine its weight.

Remark: The algorithm in Problem 4 is known as *Kruskal's algorithm.*

(Recommended) Problem 5. In this problem, we seek to build a minimum-weight spanning tree for the graph G below. Use a greedy algorithm that examines the edges of G in decreasing order (i.e., the heaviest edge gets examined first). If an edge is the heaviest edge on some remaining cycle of G (after accounting for edge deletions from previous iterations), then that edge gets removed. The remaining graph is a minimum spanning tree.



Do the following.

(a) Clearly identify the first three edges examined by the algorithm, including the order in which they were examined. It does not matter as to the order in which two edges of the same weight were selected. Determine if these edges are to be included in the minimum spanning tree and justify your answer.

(b) What will the algorithm do when it encounters the edge BC?

(c) What will the algorithm do when it encounters the edge ED?

(d) Draw the final minimum-weight spanning tree and determine its weight.

Remark: The algorithm in Problem 5 is known as the *Reverse-Delete algorithm.* In general, we do not expect this algorithm to yield the same minimum spanning tree as Kruskal's algorithm.