Homework Goals

- Develop knowledge of Trees and Tree algorithms;
- Develop knowledge of Binary Trees and properties;
- Develop knowledge of Binary Search Trees (in comparison to heaps);
- Develop knowledge of Graphs and Graph algorithms.

Trees: 10/50 points

1. Text, Problem C-6.28, Page 308. For simplicity, assume T is a binary tree. Also assume that you are given a function depth(Tree, node) that returns depth of node in Tree.

   Algorithm LCA(T, v, w)
   
   Input: tree T, node v, and node w
   
   Output: node common

   vdepth <- depth(T, v)
   wdepth <- depth(T, w)

   while vdepth > wdepth do
       v <- v.parent
       vdepth <- vdepth - 1
   while wdepth > vdepth do
       w <- w.parent
       wdepth <- wdepth - 1
   while v ≠ w do
       v <- v.parent
       vdepth <- vdepth - 1 //unnecessary unless want to know depth
       w <- w.parent
       wdepth <- wdepth - 1 //unnecessary unless want to know depth
       common <- v
   return common

   O(n)

2. The following questions refer to Figure 6.2, page 255 in the text.

   where: ERU is Electronics R·Us; S is Sales; P is Purchasing; M is Manufacturing; D is Domestic; I is International; T is Tuner; C is Canada; SA is S. America; O is Overseas; Af is Africa; E is Europe; As is Asia; and Au is Australia.

   a) List the nodes, given a preorder traversal of the tree.
ERU, R&D, S, D, I, C, SA, O, Af, E, As, Au, P, M, TV, CD, T

b) List the nodes, given a postorder traversal of the tree.

R&D, D, C, SA, Af, E, As, Au, O, I, S, P, TV, CD, T, M, ERU

c) List the nodes, given a level order traversal of the tree.

ERU, R&D, S, P, M, D, I, TV, CD, T, C, SA, O, Af, E, As, Au

d) The tree in Figure 6.2 is a general tree. Convert the general tree into a binary tree as shown in class. Draw the converted binary tree below. List the nodes, given an inorder traversal of the converted binary tree.
Inorder traversal: R&D, D, C, SA, Af, E, A, Au, O, I, S, P, TV, CD, T, M, ERU
**Binary Trees: 10/50 points**

3. Draw a proper binary tree with a *height* of 4 and a maximum number of leaf nodes.

![Binary Tree Diagram](image)

*Note that for the questions below (4a – 4d), the binary tree is not necessarily proper.*

4a) What is the minimum number of *leaf* nodes in a binary tree with height *h*?

**Answer:** 1

4b) What is the maximum number of *leaf* nodes in a binary tree with height *h*?

**Answer:** $2^h$

4c) What is the minimum number of nodes in a binary tree with height *h*?

**Answer:** $h + 1$

4d) What is the maximum number of nodes in a binary tree with height *h*?

**Answer:** $2^{h+1} - 1$

**Min Heaps and Binary Search Trees: 10/50 points**

5. Briefly describe (1-2 sentences) the binary search tree property.

**Binary Search Tree:** As described in the book, a node is $\geq$ its left child and $\leq$ its right child. In shorthand, we will call this BST form $(\geq, \leq)$. Note in class we discussed similar implementations for duplicates in the form $(>, \leq)$ and $(\geq,<)$. All are correct.

6. Briefly describe (1-2 sentences) the *min*-heap property.

**Min Heap:** A binary tree represented as an array in which each node is $\leq$ the keys of all the node’s children

7. Draw a binary search tree containing the following keys: 12, 2, 5, 9, 11, 7, 1. Assume that the keys are inserted into the binary search tree in the order given.
8. Draw a min-heap as a tree containing the following keys: 12, 2, 5, 9, 11, 7, 1. Assume that the keys are inserted into the heap in the order given.

9. Can the binary search tree property be used to print the keys of an n-node binary search tree in sorted order in O(n) time? If so, describe the algorithm. If not, explain why not.

Yes. Inorder traversal gives output of a BST in linear time.

10. Can the min-heap property be used to print the keys of an n-node min-heap in sorted order in O(n) time? If so, give the algorithm. If not, explain why not.

No. There is not particular property of a min-heap that allows sorting in O(n) time. One would have to sort the data, possibly using a heapsort, in O(n lg n) time.
Graphs: 10/50 points
In a directed graph, the out-degree of a vertex is the number of edges leaving it, and the in-degree of a vertex is the number of edges entering it.

11. Given an adjacency matrix representation of a directed graph of $V$ vertices and $E$ edges, how long does it take to compute the out-degree of all vertices in terms of big-oh? Briefly explain.

$O(V)$ for single node, by examining row for individual node
$O(V^2)$ for entire graph, by examining v-length row for each of v nodes

12. Given an adjacency matrix representation of a directed graph of $V$ vertices and $E$ edges, how long does it take to compute the in-degree of all vertices in terms of big-oh? Briefly explain.

$O(V)$ for single node, by examining column for individual node
$O(V^2)$ for entire graph, by examining v-length column for each of v nodes

13. Given an adjacency list representation of a directed graph of $V$ vertices and $E$ edges, how long does it take to compute the out-degree of all vertices in terms of big-oh? Briefly explain.

$O(1 + E/V)$ for single node, by examining adjlist for individual node
$O(V + E)$ for entire graph, by examining adjlist for each of V nodes

14. Given an adjacency list representation of a directed graph of $V$ vertices and $E$ edges, how long does it take to compute the in-degree of all vertices in terms of big-oh? Briefly explain.

$O(V + E)$ for single node, must examine adjlist for entire graph

With additional array of size V, can be done in $O(V+E)$ for entire graph. That is, cycle through all edges and increment counter when vertex is discovered in associated linked list

Without additional array, can be done in $O(V^2 + VE)$. For each node in V, cycle through all adjacency lists (i.e., all V+E edges) looking for membership and increment when found

Graph Algorithms: 10/50 points
You are given the graph shown below:

![Graph Diagram]
15. Show each iteration of Prim’s algorithm on the graph with node B as the starting node. The first iteration is provided for you.

<table>
<thead>
<tr>
<th>Iteration 1</th>
<th>Iteration 2</th>
<th>Iteration 3</th>
<th>Iteration 4</th>
<th>Iteration 5</th>
<th>Iteration 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>d</td>
<td>p</td>
<td>k</td>
<td>d</td>
<td>p</td>
</tr>
<tr>
<td>A</td>
<td>F</td>
<td>3</td>
<td>B</td>
<td>A</td>
<td>T</td>
</tr>
<tr>
<td>B</td>
<td>T</td>
<td>0</td>
<td>-</td>
<td>B</td>
<td>T</td>
</tr>
<tr>
<td>C</td>
<td>F</td>
<td>6</td>
<td>B</td>
<td>C</td>
<td>F</td>
</tr>
<tr>
<td>D</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>D</td>
<td>F</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
<td>8</td>
<td>B</td>
<td>E</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>-</td>
<td>-</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

16. Draw the resulting minimal spanning tree. Include edge weights in your drawing.

```
A
  3
  |
B
  

C

  4

D
  2

E

  1

F
```