CSCI211: Problem Set 2

Due Friday, January 29
Points Possible: 25

1. **(Levitan, 6.4.1)**

   (a) **Construct a heap for the keys 1, 8, 6, 5, 3, 7, 4 by successive key insertions (top-down algorithm). You can draw as an array or a tree. Tree will probably be easier. Show your work.**

   (b) **Construct a heap for the same keys (stored in an array) using the following bottom-up algorithm:**

Algorithm 1 HeapBottomUp(H[1...n])

```
for i = ⌊n/2⌋ downto 1 do
  k = i
  v = H[k]
  heap = false
  while not heap and 2 * k ≤ n do
    j = 2 * k
    if j < n then
      if H[j] > H[j + 1] then
        j = j + 1
      end if
      if v ≤ H[j] then
        heap = true
      else
        H[k] = H[j]
        k = j
      end if
    end if
  end while
  H[k] = v
end for
```

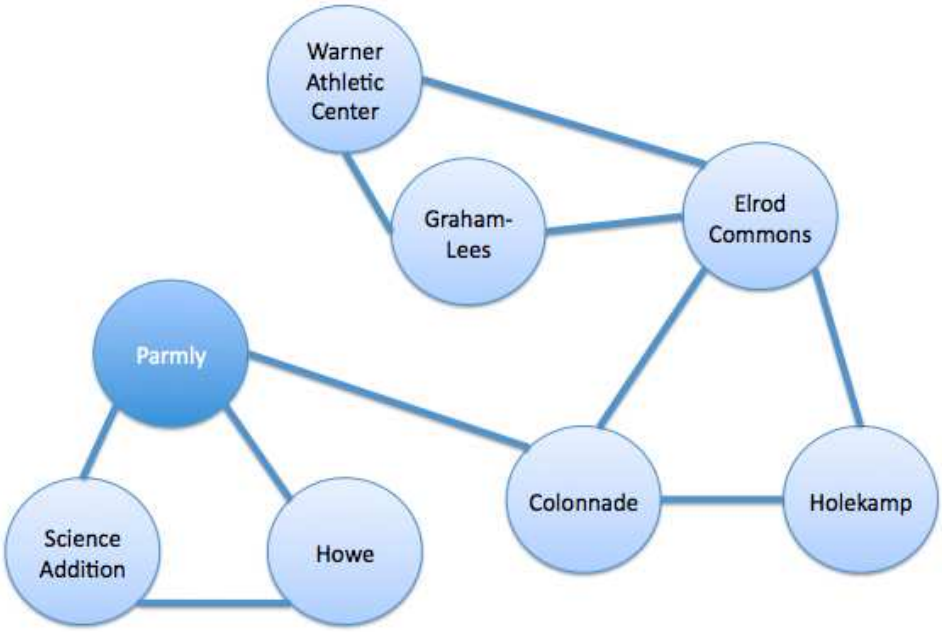
   (c) **Compare the algorithms in terms of runtime and conceptual complexity.**

   (d) **Will the successive-key insertions and bottom-up algorithms always yield the same heap for the same insertions?**

2. **(Levitan, 6.4.2) Design an algorithm for checking whether an array $H[1...n]$ is a heap and determine its runtime efficiency.**

3. **Below is a map of campus. A node in this graph is a buildling (or set of buildings), and an edge represents that there is a route between a pair of buildings.**

   (a) **Do a breadth-first search beginning at Parmly. Indicate which layer each town is in. Draw the BFS tree that you end up with. How long is the shortest path from Parmly to Warner Athletic Center (in number of edges)? What is the shortest path?**



(b) **Do a depth-first search beginning at Parmly. Draw the DFS tree that you end up with.**

4. **(3.9). There's a natural intuition that two nodes that are far apart in a communication network, i.e., separated by many hops, have a more tenuous connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes.**

   Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all $s - t$ paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.) Give an algorithm with running time $O(m + n)$ to find such a node v.