Clare Minnerath
01/22/2020
Assignment 1, #10

2. smallest numbers in a list:

My algorithm first sorts the array using an insertion sort that has a worst time complexity [if all elements are in reverse] of $\Theta(n^2)$:

a. Key operation is setting n[j+1] to n[j]
   This operation can occur anywhere from 0->n times in the while loop & the while loop executes n-1 times where n is the # of elements in the list
   So, in the worst case, we have the key operation occurring: 1+2+…+n-1 = $\frac{n(n-1)}{2} - 1$
   $\lim_{n\to\infty}\left(\frac{\frac{n(n-1)}{2}-1}{n^2}\right) = \frac{1}{2}\lim_{n\to\infty}\frac{n^2-n-2}{n^2}$ $= 1 - 0 - 0 = 1$

b. Key operation is appending small number to list.
   Appending the smallest numbers to an array takes $\Theta(m)$ complexity every time where m is the number of smallest elements looking for (obvious complexity)

The result is a smallest_nums algorithm which runs in $\Theta(n^2) + \Theta(m)$.

3. sets of 3 from a list:

My algorithm has an every time complexity of $\Theta(n^3)$

a. The key operation is printing a combination of 3 from the list
b. The nested for loops are guaranteed to run for n,n-1, and n-2 iterations respectively which results in the key operation occurring n(n-1)(n-1) times where n is the length of the list.

$$\lim_{n\to\infty}\left(\frac{n(n-1)(n-2)}{n^3}\right) = \lim_{n\to\infty}\left(\frac{(n^2-n)(n-2)}{n^3}\right) = \lim_{n\to\infty}\left(\frac{n^3-3n^2+2n}{n^3}\right) = 1 - 0 + 1 = 1$$

The result is a sets of 3 algorithm wish runs in $\Theta(n^3)$.

5. GCD:

My algorithm, a recursive GCD, has a complexity of O(logn) where n is the sum of the two numbers we are finding the GCD of.

a. The key operation is checking if the two numbers are divisible (in which case the GCD has been found), and this occurs at most log(x+y) times since the sum of the numbers decreases by over half each recursive call to the function.
b. For example, if we call GCD(954,276), the next call would be GCD(276,126) where 954+276 = 1230 = (276+126)3.06 = (402)3.06. So, the original input is approximately a third its original size by the next call.

7. Is the binary tree a heap?

My algorithm has a worst time complexity of $\Theta(n)$ where n is the number of nodes in the tree (length of the list). This complexity occurs when the tree is a heap, so it must check every node of the tree.