Megan Ku

DSA: Homework 1 (written)

1/27/2020

1. To implement list concatenation in Python, I would use the built-in `extend()` function where, for any two lists x and y (where y is to be added to the end of x), the code would be the following:

   ```
   x.extend(y)
   ```

   If n_1 = `len(x)` and n_2 = `len(y)`, then the runtime of `extend()` is O(n_2). This is because `extend()` works by iterating over each item in y and appending it to list x. We know that `append()` is a constant operation until the array needs to be resized, but the process of iteration over the list has a runtime of O(n). Since there is no iteration over the initial list, n_1 ,only the size of n_2 affects the runtime.

2.
   a. Pseudocode:

   ```
   def find_longest_asc(x):
         longest_list = []
         current_list = []
         for index, item in enumerate(x):
               current_list.append(item)
               if (index == len(x)-1) or (x[index + 1] <= item):
                     if len(current_list) > len(longest_list):
                           longest_list = current_list
                     current_list = []
         return longest_list
   ```

   b. Runtime Analysis:

   Let `n = len(x)`. The `enumerate()` function has a runtime of O(n), and the `append()` function is O(n) when the array needs to be resized, O(1) otherwise. The other operations within the function are assignment functions and have runtimes of O(1). Therefore, the runtime of this function is, at most, O(n^2) because the O(n) from enumeration and O(n) from the appending compound.