Name: Michael Rogers

ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 7a (14 points) **Fall 2019, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name: $\boxed{\text{Michael Rogers}}$

ID: $\boxed{105667404}$

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**

Problem Set 7a (14 points) **Fall 2019, CU-Boulder**

1. (1 pt) Provide a one-sentence description of each of the components of a divide and conquer algorithm.

   Solution. Divide: Break problems into several smaller instances of the same problem with the same attributes.

   Conquer: If smaller instance is trivial, solve it. Otherwise, divide the array again.

   Combine: Combine results for smaller problems into solution for the larger problem.

2. (3 pts) Use the array $A = [2, 5, 1, 6, 7, 9, 3]$ for the following questions

   (a) (1 pt) What is the value of the pivot in the call $partition(A, 0, 6)$?

   Solution. Pivot = 3, assuming we are using the technique presented in class which takes the pivot to be the last element in the array.

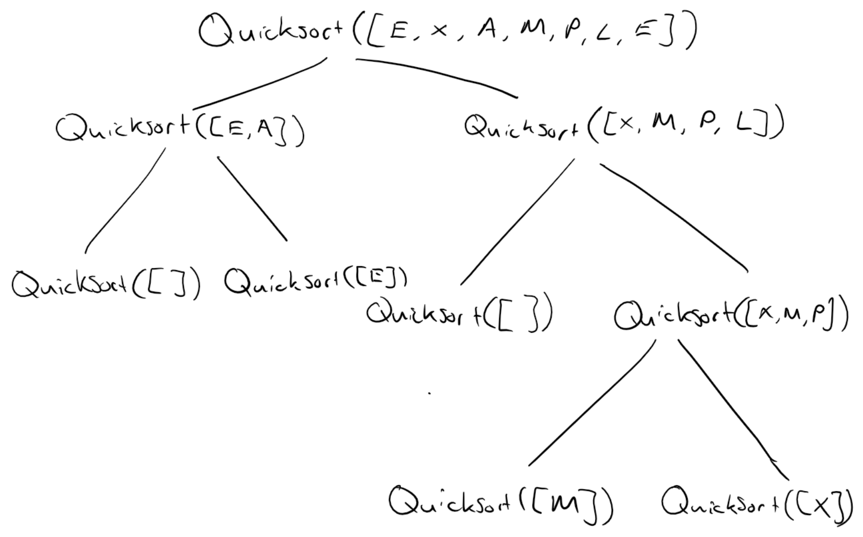
   (b) (1 pt) What is the index of that pivot value at the end of that call to $partition()$?

   Solution. Index of Pivot = 2. All elements to the left of the pivot are less than the pivot and all elements to the right are greater than the pivot.

   (c) (1 pt) On the next recursive call to Quicksort, what sub-array does $partition()$ evaluate?

   Solution. {2,1}, or the left sub-array of the pivot.

3. (4 pts) Draw the tree of recursive calls that Quicksort makes to sort the list E, X, A, M, P, L, E in alphabetical order. Use the last element in the sub-list in each recursive call as the pivot.



   Solution.

Name: Michael Rogers
ID: 105667404

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7a (14 points) **Fall 2019, CU-Boulder**

4. (6 pts) You are given a collection of n bottles of different widths and n lids of different widths and you need to find which lid goes with which bottle. You can compare a lid to a bottle, from which you can determine if the lid is larger than the bottle, smaller than the bottle, or the correct size. However, there is no way to compare the bottles or the lids directly to each other, i.e. you can't compare lids to lids or bottles to bottles. Design an algorithm for this problem with an average-case efficiency of $\Theta(nlgn)$

Solution. Assuming every bottle has a lid and no two lids or bottles are the same size: Spose' B = $\{bottle_1, bottle_2, ..., bottle_n\}$,
L = $\{lid_1, lid_2, ..., lid_n\}$ In order for these two arrays to be sorted correctly, there must be an attribute of each matching bottle and lid that must be the same. Therefore we can use the pivot of one array on the other to acheive the same result because the two arrays must contain the same values in different orders. This will allow us to use quicksort without comparing bottles to bottles or lids to lids.

```
BottleMatch(B, L):
   Quicksortesque(A, B, p, r):
      if p <  r: // If low index is less than the high index
         // Value of the last element of the array B to use for partition of A.
         A-piv = partition(A, p, r, B[r])
         // Partition B by same partition value as A
         partition(B, p, r, A[A-piv])

         Quicksortesque(A, B , p, r - 1) //Sub-array left of pivot
         Quicksortesque(A, B , p + 1, r) //Sub-array right of pivot

   Quicksortesque(B, L, 0, len(B)) // len(B) = len(L)

   return B, L
```

partition() takes in the two arrays and finds the partition of the first array by the last values of the second. Once it finds this partition value it partitions the second array by the same value as the first. After it partitions both of the arrays, it runs quicksort on both arrays making the corresponding lids and bottles have the same index in their respective arrays. The runtime of this algorithm would be $O(2n \log(n))$ because we run quicksort on both arrays, which equals $O(n \log(n))$.