CSCI 3104 PS7a

Jonathan Phouminh

TOTAL POINTS

9.5 / 14

QUESTION 1

1 1 / 1

- ✓ + **0.5 pts** **Divided into 3 parts correctly.**
- ✓ + **0.5 pts** **Reasonable description for each part.**
- + **0.8 pts** Lack the combine part.
- + **0 pts** Incorrect/Not attempted.

QUESTION 2

3 pts

2.1 1 / 1

- ✓ + **1 pts** **Correct value for pivot.**
- + **0 pts** Incorrect/Not attempted.

2.2 1 / 1

- ✓ + **1 pts** **Correct value mentioned.**
- + **0 pts** Incorrect answer or empty solution submitted. Please refer to the solution file.

2.3 1 / 1

- ✓ + **1 pts** **Correct indices range or correct sub array mentioned.**
- + **0 pts** Incorrect indices range or incorrect subarray or empty solution submitted. Please refer to the solution file.
- + **0.5 pts** Minor mistake of adding one extra index in the subarray. Please refer to the solution file.
- + **0.5 pts** The function call would be on the other subarray.

QUESTION 3

3 4 / 4

- ✓ + **4 pts** **Correct!**
- + **3 pts** Forgot to swap the pivot at the end of each iteration, or swapped the wrong element
- + **0.8 pts** Correct position of pivot for the last E in EXAMPLE
- + **0.8 pts** Correct position of pivot for A in EXAMPLE
- + **0.8 pts** Correct position of pivot for X in EXAMPLE
- + **0.8 pts** Correct position of pivot for L in EXAMPLE
- + **0.8 pts** Correct position of pivot for M in EXAMPLE
- + **0 pts** Incorrect or not attempted
- + **3 pts** Did not use the last element as the pivot

QUESTION 4

4 1.5 / 6

- + **3.5 pts** Achieves a correct mapping between lids and bottles
- + **2.5 pts** Uses nlogn solution to do so
- ✓ + **1.5 pts** **Attempts to use quicksort/other sort to achieve solution**
- + **2 pts** Does not use the partition value of first array as pivot for partition of second array
- + **0 pts** Uses the sorting technique, where lids are being compared with lids and bottles are being compared with bottles, to find the solution.
- + **0 pts** You are applying BinarySearch on an unsorted array, which leads to an incorrect solution.
- + **0 pts** Incorrect
- + **0 pts** No answer
- + **0 pts** Providing incorrect arguments for Quicksort
- + **3 pts** Uses the right technique of using the last element (or any other element) of one list to partition another list. But the solution does do not specify precisely how to apply this recursively to find the final matching.

💬 This is O(n*n*logn) because it needs to do an nlogn search for each cap/bottle.

Name: Jonathan Phouminh

ID: 106054641

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: Jonathan Phouminh

ID: 106054641

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 problem into smaller instances of the same problem, each with the same features of the original problem.
   Conquer: If smaller instance is trivial, solve it, otherwise divide again
   Combine: Combine results for smaller problem into solution

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.

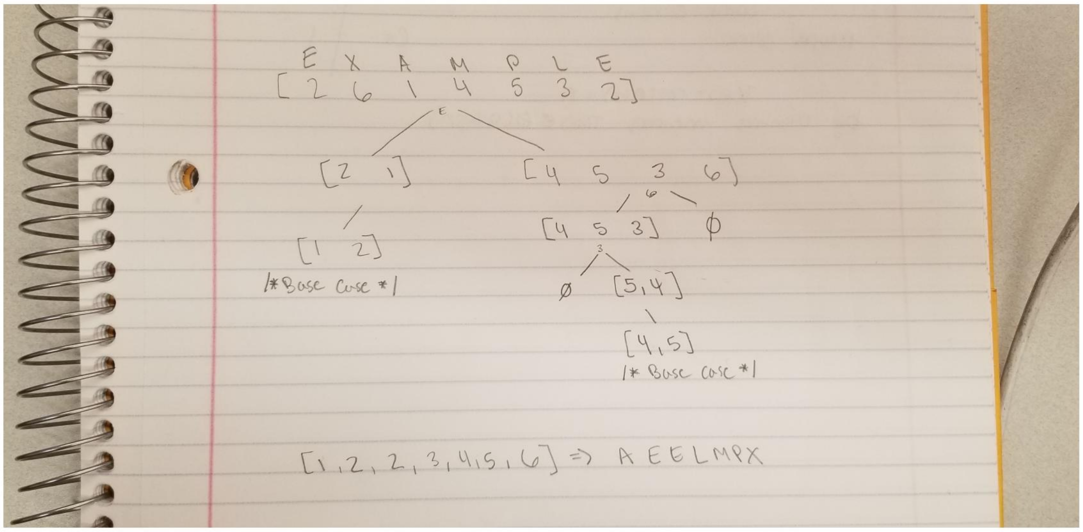
   (b) (1 pt) What is the index of that pivot value at the end of that call to $partition()$?
   Solution.
   Pivot = 3

   Pivot located at index A[2] after end of partition

   (c) (1 pt) On the next recursive call to Quicksort, what sub-array does $partition()$ evaluate?
   Solution.
   The partition will evaluate the left subarray containing $[2, 1]$

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.



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.

Code doesn't actually work but the general idea is to recursively break the array of bottles down to one single bottle and at the base case we would compare it against all the lids in the lid array list. This can be proved to be in $\theta(nlgn)$ via masters theorem.

```
def countMatches(lids,bottles):
            n = len(bottles)
            if (n!=1):
                countMatches(lid,bottles[0:(n//2)]
                countMatches(lid,bottles[(n//2):n]
                return x + y
            for i in range(len(lids)): should only come here for the base cases
                if bottles[0] == lids[i]:
                    count++
                return count
```

Name: Jonathan Phouminh

ID: 106054641

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

Collaborated with: Zachary Chommala, Bao Nguyen