Name: Luke Joyce

ID: 107355873

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

Problem Set 7b (47 points + 10 pts extra credit) **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: Luke Joyce

ID: 107355873

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

Problem Set 7b (47 points + 10 pts extra credit) **Fall 2019, CU-Boulder**

Important: This assignment has two (Q2, Q3) coding questions.

- You need to submit two python files, one for each question.
- The .py file should run for you to get points and name the file as following -
  If Q2 asks for a python code, please submit it with the following naming convention - `Lastname-Firstname-PS7b-Q2.py`.
- You need to submit the code via Canvas but the table/plot/result should be on the main .pdf.

Name: Luke Joyce
ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7b (47 points + 10 pts extra credit) **Fall 2019, CU-Boulder**

1. (7 pts) Suppose that we modify the `Partition` algorithm in QuickSort in such a way that on alternating levels of the recursion tree, `Partition` either chooses the best possible pivot or the worst possible pivot.

   (a) (1 pt) What are the best possible and the worst possible pivots for Quicksort?
   Solution.
   The best case for the index of a pivot is to end up at n/2, as in the middle index of the array / sub-array.
   The worst case for the index of the pivot is to end up as the first or last element of the array. This is because the more even the sub-arrays are, the less recursive levels there are before the array is sorted.

   (b) (4 pts) Write down a recurrence relation for this version of QuickSort and give its asymptotic solution.
   Solution.
   For the best case (only case):
   $T(n) = 2T(n/2) + \theta(n)$
   $T(n) = 2(2T(n/4) + \theta(n)) + \theta(n) = \theta(nlogn)$

   (c) (2 pts) Provide a verbal explanation of how this `Partition` algorithm affects the running time of QuickSort.
   Solution.

   This algorithm makes it so the time complexity is always the best case, since it is always choosing the best pivot value, which in turns always produces two sub-arrays of equal length after each partition call.

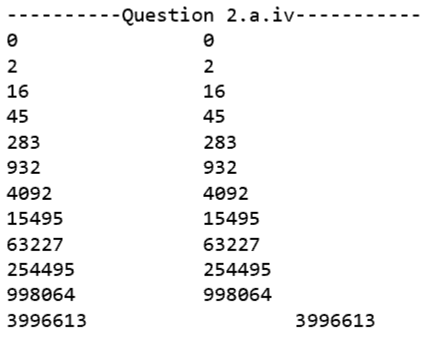
Name: Luke Joyce
ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7b (47 points + 10 pts extra credit) **Fall 2019, CU-Boulder**

2. *(14 pts total) In PS1b, you were asked to count flips in a sorting algorithm with quadratic running time. The problem definition looked something like this:*
Let $A = \langle a_1, a_2, \ldots, a_n \rangle$ be an array of numbers. Let's define a 'flip' as a pair of distinct indices $i, j \in \{1, 2, \ldots, n\}$ such that $i < j$ but $a_i > a_j$. That is, a_i and a_j are out of order.
For example - In the array A = [1, 3, 5, 2, 4, 6], (3, 2), (5, 2) and (5, 4) are the only flips i.e. the total number of flips is 3. (Note that in this example the indices are the same as the actual values)

   (a) (14 pts) Write a Python program with the following features:
      i. (2 pts) Generates a sequence of n numbers in the range $[1, \ldots, n]$ and then randomly shuffles them.
      ii. (2 pts) Implements a $\theta(n^2)$ sorting routine that counts the number of flips in the array.
      iii. (5 pts) Implements a sorting routine with $\theta(nlgn)$ running time that counts the number of flips in the array. **Hint: Mergesort**
      iv. (5 pts) Run your code, both sorting algorithms, on values of n from $[2, 2^2, 2^3, \ldots.2^{12}]$ and present your results in a table or labeled plot. Result with no supporting code will not get points.

   Follow the naming convention for python code mentioned on Page 2.



```
----------Question 2.a.iv-----------
0                0
2                2
16               16
45               45
283              283
932              932
4092             4092
15495            15495
63227            63227
254495           254495
998064           998064
3996613                  3996613
```

Figure 1: Caption

Solution. This image is showing the results for the number of flips for the merge-Sort (right), and selection sort (left).

Name: Luke Joyce

ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7b (47 points + 10 pts extra credit) **Fall 2019, CU-Boulder**

3. (10 pts) Help the Mad Scientist calculate his h-index. According to Wikipedia: "A scientist has index h if h of their N papers have at least h citations each, and the other $N - h$ papers have no more than h citations each."

   For this question, write a Python program that calculates the h-index for a given input array. The array contains the number of citations for N papers, sorted in descending order (each citation is a non-negative integer). Your Python program needs to implement a divide and conquer algorithm that takes the *citations* array as input to outputs the h-index.

   Example:
   Input: citations = [6,5,3,1,0]
   Output: 3
   Explanation: [6,5,3,1,0] means the researcher has 5 papers with 6, 5, 3, 1, 0 citations respectively. Since the researcher has 3 papers with at least 3 citations each and the remaining two with no more than 3 citations each, the h-index is 3.
   Note: If there are several possible values for h, the maximum value is the h-index.
   Hint: Think how will you find it by a linear scan? You can then make your "search" more efficient.

   Do not submit anything on the .pdf for this question.
   Follow the naming convention for the python code mentioned on Page 2.

Name: Luke Joyce
ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7b (47 points + 10 pts extra credit) **Fall 2019, CU-Boulder**

4. (16 pts) Consider the following strategy for choosing a pivot element for the `Partition` subroutine of QuickSort, applied to an array A.

   - Let n be the number of elements of the array A.
   - If $n \leq 15$, perform an Insertion Sort of A and return.
   - Otherwise:
     - Choose $2\lfloor\sqrt{n}\rfloor$ elements at random from A; let S be the new list with the chosen elements.
     - Sort the list S using Insertion Sort and store the median of S as m.
     - Partition the sub-array of A using m as a pivot.
     - Carry out QuickSort recursively on the two parts.

   (a) (4 pts) Using the following array A with $n = 20$, show one iteration of this partitioning strategy on the array

   $$A = [34, 45, 32, 1, 23, 90, 12, 13, 43, 54, 65, 76, 67, 56, 45, 34, 44, 55, 23, 2]$$

   . Clearly identify all variables.

   Solution.
   In this array, since $n > 15$, we will go onto the "Otherwise" part of the description:

   First, since n = 20, we do $2[\sqrt{20}]$ and get $2 * 4 = 8$ (truncated int). Choosing 8 elements sorted from A randomly gets us [1,2,12,43,45,56, 67,90]. The median is at index 7/2 = 3, which is 43 in our new sub-array. Using that as our new pivot, we can partition the original array into two sub-arrays as follows:
   [1,2,12,13,23,23,32,34,34,]
   and [44,45,45,54,55,56,65,67,76,90]
   which leaves us with two arrays of size less than 15, so we can do insertion sort to finish them off.

Name: Luke Joyce
ID: 107355873

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 7b (47 points + 10 pts extra credit) **Fall 2019, CU-Boulder**

(b) (4 pts) If the element m obtained as the median of S is used as the pivot, what can we say about the sizes of the two partitions of the array A? **Hint: Think about the best and worst possible selections for the values in S.**

Solution. The worst case for this algorithm would be the case that we picked either the smallest 8 (or $2[\sqrt{n}]$) elements in the array A or the largest 8 (or $2[\sqrt{n}]$) elements in the array A to make our sorted list S. This would cause us to partition array A in a lopsided fashion. For example, when n=20, if we were to randomly end up with the smallest 8 elements in the array for S, we would find a median at index 3, and partition to a smaller sub-array of size 4, and a larger sub-array of size $20 - 4 - 1 = 15$. On the other hand, the best case is that we randomly choose elements so that our median of S ends up being the same as the median for the whole array A, in which case we would end up with two sub-arrays of the same size (or 1 off in size in this case).

(c) (3 pts) How much time does it take to sort S and find its median? Give a Θ bound.

Solution. For the insertion sort part of this, we have $\theta(n^2)$. Then, finding the median only takes linear time, giving us $T(n) = \theta(n^2) + \theta(n) = \theta(n^2)$. But since the size of array S is only $2[\sqrt{n}]$, we can substitute that in to get $T(n) = \theta((2 * \sqrt{n})^2) = T(n) = \theta(4n) = \theta(n)$

(d) (5 pts) Write a recurrence relation for the worst case running time of QuickSort with this pivoting strategy.

Solution.
$T(n) = \theta(n - \sqrt{n}) + \theta(\sqrt{n}) + \mathcal{O}(n) + \mathcal{O}(n)$
n - $\sqrt{n}$ for the bigger worst case sub-array.

5. (10 pts extra credit) Implement the bottles and lids algorithm that you wrote in assignment 7a and show that it functions correctly on randomly generated arrays representing 100 bottles and lids. Your algorithm needs to use a divide and conquer strategy to receive credit for this question.

Name: $\boxed{\text{Luke Joyce}}$

ID: $\boxed{107355873}$

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

Problem Set 7b (47 points + 10 pts extra credit) **Fall 2019, CU-Boulder**

9th page