COMP285: Analysis of Algorithms
North Carolina A&T State University — Homework #2
Dr. Allison Sullivan — Due: September 7, 2018 (*in class quiz*)

Homework #2

You should try to solve these problems by yourself. I recommend that you start early and get help in office hours if needed. If you find it helpful to discuss problems with other students, go for it. **You do not need to turn in these problems. The goal is to be ready for the in class quiz that will cover the same or similar problems.**

Problem 1: Recurrence Relations

Calculate the time complexity of the below divide-and-conquer algorithm via a recurrence relation. Assume addition is an $\mathcal{O}(1)$.

```
1  getResult(A[0...n - 1]){
2     if (n = 0) { return 0 }
3     else if  (n = 1)  { return A[0] }
4     i =  n/4
5     R1 = getResult(A[0...2i - 1])
6     R2 = getResult(A[i...3i - 1])
7     R3 = getResult(A[3i...n - 1])
8     return R1 + R2 + R3
9  }
```

(a) Write a recurrence relation for `getResult`.

(b) Solve the recurrence relation to find a big $\mathcal{O}$ expression for the number of `getResult` calls as a function of n.

Problem 2: Divide and Conquer, Take 2

Suppose you are given an array A of n sorted numbers that has been *circularly shifted* to the right by k positions. For example $\{35, 42, 5, 15, 27, 29\}$ is a sorted array that has been circularly shifted $k = 2$ positions, while $\{27, 29, 35, 42, 4, 15\}$ has been shifted $k = 4$ positions. Give an $O(\log n)$ algorithm to find the largest number in A. You may assume the elements of A are distinct. Write the recurrence for your algorithm and show that its recurrence solves to $O(\log n)$ (e.g., using the Master Method, a recursion tree, or an inductive proof).

Problem 3: Linear Time Sorting

Suppose you are given the task to sort 10000 numbers between 0 and $10^{12} - 1$ (i.e., the keys are 12 digit numbers). You have decided to use radix sort but need to decide how many digits to group for each radix sort digit. Which is best among having 1 digit per radix sort digit, 3 digits per radix sort digit, 6 digits per radix sort digit, or directly using counting sort (i.e., 12 digits per radix sort digit)? You are provided with a counting sort procedure with exact time complexity of $13n+9k+4$. Show your work.

Problem 4: Politics Invades COMP 285, Along With Trash Pandas

Move over donkeys and elephants, inspired by Rocket the Raccoon[1], raccoons have decided to enter the political fray in the United States.

Suppose that you encounter n raccoons standing in a line.[2] To keep things civil, you decide you want to rearrange the raccoons by their political ideology.

Each raccoon has a political leaning: left, right, or center. You'd like to sort the raccoons so that all the left-leaning ones are on the left, the right-leaning ones are on the right, and the centrist raccoons are in the middle. You can only do two types of operations on the raccoons:

Operation	Result
`poll(j)`	Ask the raccoon in position j about its political leanings
`swap(i, j)`	Swap the raccoon in position j with the raccoon in position i

However, in order to do either operation, you need to pay the raccoons to co-operate (they are politicians now, after all): each operation costs one bundle of trash. Also, you didn't bring a piece of paper or a pencil, so you can't write anything down and have to rely on your memory. Like many humans, you can remember up to seven integers between 0 and $n-1$ at a time[3].

Design an algorithm to sort the raccoons which costs $O(n)$ bundles of trash, and uses no extra memory other than storing at most seven integers between 0 and $n-1$[4].

[1] from Guardians of the Galaxy

[2] They just happen to be for no reason.

[3] `https://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two`

[4] You don't need to use all seven storage spots, but you can if you want to. You actually just need two.

Solution

We can use a variant of the algorithm which we used for QuickSort to partition elements in-place:

```
1  sortRaccoons( raccoons ){
2     i = 0
3     # first, go through all the raccoons and put the left-leaning
4     ones on the left.
5     for j = 0, ..., n-1{
6        if poll(j) == "left"{
7           swap( i, j )
8           i += 1
9        }
10    }
11
12    # next, go through and put the center-leaning ones to the left of
13    the right-leaning ones.
14    for j = 0, ..., n-1{
15       if poll(j) == "center"{
16          swap( i, j )
17          i += 1
18       }
19    }
20    # now we should be all done!
21 }
22
```