

Due: Thursday, October 25

1. Recurrence Relations

You've decided to accept a job as an algorithm analyst at the social media company FacePage. One day, your boss Clark Hackerberg stops by with a problem: he has three different algorithms that all perform the same task, and he needs to choose which one to use.

- Algorithm A solves a problem of size n by dividing it into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time $O(n)$.
- Algorithm B solves a problem of size n by recursively solving two subproblems of size $n - 1$ and then combining the solutions in constant time $O(1)$.
- Algorithm C solves a problem of size n by dividing it into nine subproblems of size $n/3$, recursively solving each subproblem, then combining the solutions in quadratic time $O(n^2)$.

For each of these algorithms you should:

- (a) Write down a recurrence relation for the runtime.
- (b) Derive the runtime of the algorithm.

Which algorithm should your boss implement?

2. The Index Problem

Let A be an array of n distinct integers where A is **already sorted** in ascending order. Our problem is to find an index i , $1 \leq i \leq n$, such that $A[i] = i$ or determine that no such i exists.

Describe an algorithm for this problem with $O(\log n)$ worst case running time. You should give the algorithm (in clear English or in clear high-level pseudo-code) and briefly explain why the running time is $O(\log n)$ in the worst case.

3. Recursive Calls¹

Professor Mae Trix has devised an algorithm for computing the Trixian function on two $n \times n$ matrices. (Nevermind what that function does - we only care about the algorithm!)

- (a) Prof. Trix's first attempt at her algorithm has a worst-case running time described by the recurrence relation:

$$\begin{aligned}T(1) &= c \\T(n) &= 8T(n/2) + cn^4\end{aligned}$$

What is the big-O asymptotic runtime of this algorithm as a function of n ? Show your work.

- (b) By using some clever tricks, Prof. Trix has removed one of the recursive calls and now has an algorithm with a worst-case runtime described by the recurrence relation:

$$\begin{aligned}T(1) &= d \\T(n) &= 7T(n/2) + dn^4\end{aligned}$$

What is the big-O asymptotic runtime of this algorithm as a function of n ? Show your work.

- (c) Is the second algorithm asymptotically better than the first in this case? Briefly, what do you think is the reason for this outcome?

4. Stooge Sort¹

Professors Curly, Mo, and Larry have proposed the following sorting algorithm:

- First sort the first two-thirds of the elements in the array.
- Next sort the last two-thirds of the elements in the array.
- Finally, sort the first two-thirds again.

The code is given below. Notice that the floor function, $\lfloor x \rfloor$, simply rounds down to the nearest integer. This is just used to compute the appropriate two-thirds and round to an integer so that we don't use non-integer indices into our array!

```
def Stooge-Sort(A, i, j):  
  
    if A[i] > A[j]:  
        swap A[i] and A[j]  
  
    if i+1 >= j:  
        return  
  
    k =  $\lfloor (j-i+1)/3 \rfloor$   
    Stooge-Sort(A, i, j-k) # Sort the first two-thirds.  
    Stooge-Sort(A, i+k, j) # Sort the last two-thirds.  
    Stooge-Sort(A, i, j-k) # Sort the first two-thirds again!
```

- (a) Give an informal but convincing explanation (not a rigorous proof by induction) of why the approach of sorting the first two-thirds of the array, then sorting the last two-thirds of the array, and then sorting again the first two-thirds of the array yields a sorted array. A few well-chosen sentences should suffice here.
- (b) Find a recurrence relation for the worst-case runtime of Stooge-Sort. To simplify your recurrence relation, you may assume each of the recursive calls is on a portion of the array that is *exactly* two-thirds the length of the original array.
- (c) Next, solve the recurrence relation using the work tree method. **Show all of your work.** In your analysis, it will be convenient to choose n to be a^k for some fixed constant a . (For example, we used $a = 2$ when analyzing the multiplication problem or Mergesort in class. Here you will want to use a different value of a . The value of a that you choose might not even be an integer! As we've seen in class, this is valid and allows us to significantly simplify the analysis.)
- (d) How does the worst-case runtime of Stooge-Sort compare with the worst-case runtime of the other sorting algorithms that we've seen so far?

¹Adapted from problem sets created by Harvey Mudd College CS Professor Ran Libeskind-Hadas.