Assignment 3

csci2200, Algorithms

Instructions:

- Honor code: Work on this assignment alone, or with one partner. Between different teams, Collaboration is at level 1 [verbal collaboration only]
- Check out the Homework guidelines on class website.

---

1. **Breaking eggs:** Suppose you have an n-stories high building, and a bunch of eggs. An egg has a certain level l at which , if thrown from any level $\geq l$, it breaks. For example, an egg might have $l = 7$ meaning you can safely throw the egg down from levels 1 through 6, and it will not break; but if you through the egg from a level 7 or higher, it breaks.

   You are given a building and a bunch of eggs (all identical) and your goal is to find out the level l of the eggs. While you think about the problem, you can assume $n = 100$ (i.e. 100-level high building). But describe your solutions in terms of n[1]

   (a) Describe an approach that only breaks one egg to find out l. How many throws does it do?

   What we expect: Explain the rationale of the algorithm and give pseudocode. Its analysis as function of n.

   (b) Describe an approach that minimizes the number of throws. How many eggs might it break?

   What we expect: Explain the rationale of the algorithm and give pseudocode. Its analysis as function of n.

   (c) Assume now you have two eggs. Describe an approach that minimizes the number of throws.

   What we expect: Explain the rationale of the algorithm and give pseudocode. Its analysis as function of n.

[1] This is from Kleinberg-Tardos textbook; also reported as an interview question in 2014 by an alum

2. **Stoogesort:** One of your colleagues at work has proposed the following sorting algorithm, and your task is to evaluate it.

```
STOOGE-SORT(A, i, j)
if A[i] > A[j]: swap A[i] ↔ A[j]
if i + 1 ≥ j: return
k ← ⌊(j − i + 1)/3⌋
STOOGE-SORT(A, i, j − k)
STOOGE-SORT(A, i + k, j)
STOOGE-SORT(A, i, j − k)
```

(a) Correctness:

do not turn in Work through an example and argue briefly that STOOGE-SORT correctly sorts any array of one element.

do not turn in Work through an example and argue briefly that STOOGE-SORT correctly sorts any array of two elements.

do not turn in Consider the algorithm but with the first line (that swaps elements $A[i]$ and $A[j]$) missing. Argue that it would not correctly sort by showing a simple counter-example.

do not turn in Work through an example array of 3 elements and see how it is getting sorted by STOOGE-SORT.

i. Consider the state of the array A after the first recursive call finished and before starting the second recursive call (and assume the the recursive call correctly sorts). Consider the largest $n/3$ elements in A. Where might they reside? Make a statement and argue (briefly) why it's correct.
What we expect: Statement: ...... Argument: ...

ii. Consider the state of the array A after the second recursive call finished and before starting the third recursive call (and assume the recursive calls sort correcty). Consider the largest $n/3$ elements in A. Where might they reside? Make a statement and argue (briefly) why it's correct.
What we expect: Statement: ...... Argument: ...

(b) Running time: Give a recurrence for the worst-case running time of STOOGE-SORT and a tight asymptotic (Θ-notation) bound on the worst-case running time.

What we expect: The recurrence, illustrate the process to find its solution, and its solution.

3. **Select the $\sqrt{n}$-closest:** Given an unordered sequence S of n elements (for simplicity, assume items are integers or real numbers), describe an efficient method for finding the $\lceil \sqrt{n} \rceil$ elements whose values are closest to (the value of) the median of S. What is the running time of your method? Aim for linear time.

What we expect: The rationale of the algorithm, pseudocode, analysis

4. **Merging sorted lists:** Assume you have k sorted arrays containing a total of n elements, and you want to merge them together in a single (sorted) array containing all n elements. For simplicity you may assume that the k arrays contain the same number of elements, namely n/k elements each.

   (a) Approach 1: merge array 1 with array 2, then merge the result with array 3, then merge the result with array 4, and so on. What is the worst-case running time ?

   What we expect: Detailed analysis of this approach

   (b) Approach 2: split the set of k arrays into two sets of $k/2$ arrays, merge each one recursively, then use the standard 2-way merge procedure (from mergesort) to combine the two resulting arrays. What is the worst-case running time ?

   What we expect: A recurrence , the recurrence depth, and the solution.

   (c) Approach 3: Give another approach (to merge the k arrays) that uses a heap, and runs in $O(n \lg k)$-time.

   What we expect: The idea of the algorithm, pseudo-code, analysis