CS515: Algorithms and Data Structures, Winter 2021

Homework 1*

Due: Tue, Jan 19, 2021

Homework Policy:

1. Students should work on group assignments in groups of preferably three people. Each group submits to CANVAS a *typeset* report in pdf format.
2. The goal of the homework assignments is for you to learn solving algorithmic problems. So, I recommend spending sufficient time thinking about problems individually before discussing them with your friends.
3. You are allowed to discuss the problems with other groups, and you are allowed to use other resources, but you *must* cite them. Also, you must write everything in your own words, copying verbatim is plagiarism.
4. *I don't know policy:* you may write "I don't know" *and nothing else* to answer a question and receive 25 percent of the total points for that problem whereas a completely wrong answer will receive zero.
5. Algorithms should be explained in plain english. Of course, you can use pseudocodes if it helps your explanation, but the grader will not try to understand a complicated pseudocode.
6. More items might be added to this list. ☺

Problem 1.

(a) Prove that the following algorithm actually sorts its input!

```
StoogeSort(A[0..n−1]):
  if n = 2 and A[0] > A[1]
      swap A[0] ↔ A[1]
  else if n > 2
      m = ⌈2n/3⌉
      StoogeSort(A[0..m−1])
      StoogeSort(A[n−m..n−1])
      StoogeSort(A[0..m−1])
```

(b) Would StoogeSort still sort correctly if we replaced $m = \lceil 2n/3 \rceil$ with $m = \lfloor 2n/3 \rfloor$? Justify your answer.

(c) State a recurrence (including the base case(s)) for the number of comparisons executed by StoogeSort.

*Some of the problems are from the text book. Looking into similar problems from the book, chapters 1 and 2 is recommended.

(d) Solve the recurrence, and prove that your solution is correct. [Hint: Ignore the ceiling.]

(e) Prove that the number of swaps executed by StoogeSort is at most $\binom{n}{2}$.

Problem 2. Consider the following restricted variants of the Tower of Hanoi puzzle. In each problem, the pegs are numbered 0, 1, and 2, and your task is to move a stack of n disks from peg 0 to peg 2, exactly as in problem.

Suppose you are forbidden to move any disk directly between peg 1 and peg 2; every move must involve peg 0. Describe an algorithm to solve this version of the puzzle in as few moves as possible. Exactly how many moves does your algorithm make?

Problem 3. You are a visitor at a political convention with n delegates; each delegate is a member of exactly one political party. It is impossible to tell which political party any delegate belongs to; in particular, you will be summarily ejected from the convention if you ask. However, you can determine whether any pair of delegates belong to the same party by introducing them to each other. Members of the same political party always greet each other with smiles and friendly handshakes; members of dffierent parties always greet each other with angry stares and insults. Suppose more than half of the delegates belong to the same political party. Describe an efficient algorithm that finds out the size of the majority party. The efficiency of your algorithms is measured in terms of the number of pair of delegates that you introduce to each other. We expect that you need about $O(n \log n)$ handshakes.

Problem 4. Design an algorithm that given an array of integers $A[1..n]$ (that contains at least one positive integer), computes

$$\max_{1 \le i \le j \le n} \sum_{k=i}^{j} A[k]$$

For example, if $A = [31, -41, 59, 26, -53, 58, 97, -93, -23, 84]$, the output must be 187. Prove that your algorithm is correct, and analyze its running time. For full credit your algorithm should work in linear time. For any credit (more than "I don't know") you algorithm should work in $O(n^2)$ time.