Data Structures and Algorithms
Homework Assignment 4

Given: February 5, 2018

Due: February 12, 2018
No Late Days Allowed

Note: The homework is due **electronically on Gradescope and Canvas** on Monday, February 12 by 11:59 pm EST. **You may NOT use any late days on this homework**, as solutions will be released on Tuesday so you can use them to study for Midterm 1.

A. **Gradescope**: You must select the appropriate pages on Gradescope. Gradescope makes this easy for you: before you submit, it asks you to associate pages with the homework questions. Failing to do so will get you points off, which cannot be argued against after the fact. Gradescope may prompt you with a warning to select your cover page, please ignore this warning.

B. **LaTeX**: You must use the `hw121.cls` Latex template provided on the course website, or a harsh penalty will be incurred. Handwritten solutions or solutions not typeset in Latex will not be accepted.

C. **Solutions**: Please write concise and clear solutions; you will get only a partial credit for correct solutions that are either unnecessarily long or not clear. Please refer to the Written Homework Guidelines for all the requirements.

D. **Algorithms**: Whenever you present an algorithm, your answer must include 3 separate sections:

1. A precise description of your algorithm in English. No pseudocode, no code.
2. Proof of correctness of your algorithm
3. Analysis of the running time complexity of your algorithm

E. **Collaboration**: You are allowed to discuss **ideas** for solving homework problems in groups of up to 3 people but *you must write your solutions independently.* Also, you must write on your homework the names of the people with whom you discussed. For a clarification on the collaboration policy, please see Piazza @547

F. **Outside Resources**: Finally, you are not allowed to use *any* material outside of the class notes and the textbook. Any violation of this policy may seriously affect your grade in the class. If you're unsure if something violates our policy, please ask.

1. [20pts - Can You Beat Euclid?]

A. Prove the following properties about the GCD of two integers.

1) If x and y are both even, then $\gcd(x, y) = 2\gcd(x/2, y/2)$.
2) If x is odd and y is even, then $\gcd(x, y) = \gcd(x, y/2)$.
3) If x and y are both odd with $x \neq y$, then $\gcd(x, y) = \gcd(|x - y|/2, y)$.

B. Subtracting, testing the parity of an integer, and dividing an integer in half are operations that are generally less expensive than computing remainders. Come up with an efficient divide-and-conquer algorithm to compute the GCD of two integers x and y, where $x \geq y$ and x and y each have n bits. Show that the algorithm runs in $O(n^2)$ time. You may assume that subtracting two n-bit integers takes $O(n)$ time, and that testing parity and halving can be done in constant time.

2. [20pts - Ben's Lonely Hearts] Ben tried to turn this Valentine's Day into a "Palentine's Day" by making plans with his pals to go whale-watching. Sadly, all of his friends already had plans to spend the day with their significant others. Ben, being the only bachelor, felt lonely on Valentine's Day, so on his way home he bought a box of candy hearts. There are n candy hearts in the box (you can suppose n is even but not necessarily a power of two), and each heart has one of k loving messages written on it, like 'All Mine' and 'Soul Mate'. Ben discovered that the candy hearts have a promotion happening, where he will get a Blu-Ray copy of *Thor: Ragnarok* if more than $n/2$ of the hearts have the same message written on them. Hoping to turn his Valentine's Day around, Ben wants to figure out if he has won the prize. Still recovering from his cancelled "Palentine's Day" plans, Ben's mind is a little unfocused. The only thing Ben is able to do is pick two candy hearts, say heart i and heart j, and determine in constant time whether i and j have the same message written on it.

a. Give an $O(n \lg n)$ divide-and-conquer algorithm to determine if Ben has won a copy of *Thor: Ragnarok* or not.

b. Design a linear time algorithm that is based on the following approach:

- Pair up the n candy hearts arbitrarily, to get $n/2$ pairs (each candy heart belongs to at most one pair).
- Consider each pair of candy hearts: if the two candy hearts have different messages written on them, eat both of them; if they have the same message written on it, eat just one of them and keep the other one.

Hint: To prove the correctness of your algorithm, it may help to prove the following: after performing the above procedure, there are at most $n/2$ candy hearts left and if there is a message m that appears on more than $n/2$ candy hearts, then message m appears on more than half of the remaining candy hearts after performing the above procedure.

3. [20pts - Twenty-One Valentines] To finish forming the band Twenty-One Pigeons, Ziad thinks of getting some the singers Valentine's day-themed costumes. However, he does not have the time to pick out good costumes, so he asks Weiwei to scout the c best costumes in town.

Weiwei comes up with a list C of c costumes, with potentially different prices. Since the entertainment studio is on a limited budget, Ziad asks Weiwei to choose s (such that $s \leq c$) costumes from C that are the closest to the median price. For example, if C contains five costumes c_1, c_2, c_3, c_4, c_5, with prices given by $(10, 12, 5, 90, 3)$, and $s = 2$, then your algorithm should return c_1 and c_2.

Additionally, Ziad is in a hurry, so he wants the s costumes as fast as possible. Unfortunately, Weiwei forgot to sort his list of c costumes, and he knows that sorting isn't efficient for this task. But he still comes up with a linear time algorithm to pick the s costumes from C.

Describe a linear time algorithm (with respect to c) that Weiwei could have came up with. Don't forget to prove its correctness and to analyze its runtime.

4. [20pts - Battle of the Bouquets] It's Valentine's Day, and all n female TAs are standing outside Krishna's house in a single file line with a bouquet of flowers waiting to ask him out. They all want to know who Krishna will eventually go out with, and they figure that Krishna will probably go out with someone with a big bouquet. Therefore, they decide to be jealous of the first TA with a bigger bouquet in front of them. If there is no TA with a larger bouquet in front of them, they decide to arbitrarily be jealous of the first TA in line. They only care about TAs in front of them because they all figure that Krishna probably does not care about whoever is behind them.

The bouquets do not necessarily have to be distinct because everyone bought their flowers from the same flower shop on Chestnut anyway.

Naive $\Theta(n^2)$ Algorithm: For each TA t in line, keep checking the bouquet of flowers of the TA forward from their location in line until we either encounter a bigger one, or we reach the first person in line. Denote the TA holding this bigger bouquet as b, and conclude that TA t will be jealous of TA b.

As all the TAs are too busy lining up, they probably need a more efficient algorithm. Please help them think of a more efficient algorithm such that every TA can find out whom they are jealous of in $\Theta(n)$ time (across all TAs). Give a brief argument to prove that your algorithm is correct and derive its running time.

5. [20pts - Vicky's Party] Vicky wants to organize an event for Valentine's day because she's feeling lonely. She decides to host a mingling event to meet new friends. The venue she chooses can hold up $2n$ people. Because the event becomes so popular, all $2n$ tickets get sold out immediately. When Vicky arrives at the event, she notices that everyone has

lined up already. Because there are so many people, they had formed two lines of the same length to save space. She notices that both lines happen to be sorted by height in increasing order. Vicky's feeling generous and decides that the person who's taller or shorter than exactly half the $2n$ people will get a free VIP pass to the event. To be more precise, she's looking for either the n^{th} or $(n+1)^{th}$ tallest person. Give an $O(\lg n)$-time algorithm to help Vicky find this lucky person. Note that for each of the two lines, Vicky knows the exact height of each person in that line, as well as the position of each person in that line.