Midterm 2: November 13, 7-9pm, 2017

A	B	C	D	E	F	G	H	J	K
9am	10am	11am	noon	1pm	1pm	2pm	2pm	3pm	3pm
Rucha	Rucha	Srihita	Shant	Abhishek	Xilin	Shalan	Phillip	Vishal	Phillip
SC 1404	SC 1404	SC 1404	DCL 1320	DCL 1320	DCL 1320	ECE 1002	ECE 1002	ECE 1002	ECE 1002

Name:	
NetID:	
Name on Gradescope:	

- ***Don't panic!***
- Please print your name and NetID **in each page** in the appropriate fields, and circle your discussion section in the boxes above. We will return your exam at the indicated section.
- If you brought anything except your writing implements, your double-sided **handwritten** (in the original) 8½" × 11" cheat sheet, and your university ID, please put it away for the duration of the exam. In particular, please turn off and put away *all* medically unnecessary electronic devices.
  - Submit your cheat sheet together with your exam. An exam without your cheat sheet attached to it will not be graded.
  - If you are NOT using a cheat sheet, please indicate so in large friendly letters on this page.
- Please ask for clarification if any question is unclear.
- **This exam lasts 120 minutes.**
- If you run out of space for an answer, feel free to use the blank pages at the back of this booklet, but please tell us where to look.
- As usual, answering any (sub)problem with "I don't know" (and nothing else) is worth 25% partial credit. Correct, complete, but sub-optimal solutions are *always* worth more than 25%. A blank answer is not the same as "I don't know".
- Total IDK points for the whole exam would not exceed 10.
- **Beware of the Four Deadly Sins.** Give complete solutions, not examples. Declare all your variables. If you don't know the answer admit it and use IDK. Do not write unnecessary text.
- **Style counts.** Please use the backs of the pages or the blank pages at the end for scratch work, so that your actual answers are clear.
- Try to avoid writing in the margins of the pages, as it might not be scanned in.
- Please return ***all*** paper with your answer booklet: your cheat sheet, and all scratch paper. We will **not** return the cheat sheet.
- ***Good luck!***

NETID: NAME:

1 (20 PTS.) Short questions.

1.A. (10 PTS.) Give an asymptotically tight solution to the following recurrence, where $T(n) = O(1)$ for $n < 10$, and otherwise:
$T(n) = T(n/3) + T((2/3)n) + O(n)$.

1.B. (10 PTS.) Given a vertex v in a directed graph G, describe a linear time algorithm to decide if there is a simple cycle in G that contains v. As a reminder, a ***simple cycle*** C is a cycle where all its vertices are different (note, that we consider a cycle of length two to be a valid simple cycle).

NETID: NAME:

2 (20 PTS.) (**Seen in lab**) You are given a directed graph G with n vertices and m edges. Every edge $x \to y \in \mathsf{E}(\mathsf{G})$ has a weight $w(x \to y)$ associated with it. Here, exactly *one* edge $u \to v$ has negative weight (all other weights are positive numbers). Describe an algorithm, as fast as possible, that decides if there is a negative cycle in G, and if not, it computes the shortest path between the two given vertices s and t in G. What is the running time of your algorithm? Argue that your algorithm is correct.

NETID: NAME:

3 (20 PTS.) You are given an array $A[1 \,..\, n]$ of n numbers that used to be sorted, and a parameter k. Unfortunately, Russian interference with your input had corrupted k values somewhere in the array (but you don't know where). An example of the input:

3	14	15	<u>92</u>	65	<u>58</u>	97	<u>23</u>	99	433	795	5028

[The underline denotes entries that were corrupted – note, that you are not given this information.]

Given a value x, describe an algorithm, as fast as possible, that decides if x appears somewhere in A – fortunately, you know that none of the k corrupted entries in A has value equal to x.

What is the running time of your algorithm (as a function of n and k)? Argue that your algorithm is correct.

(Hint: How many consecutive entries in the array do you have to read before you are certain you had read a non-corrupted value, and furthermore, you know that this value is smaller or larger than x.)

NETID: NAME:

4 (20 PTS.) There are n people living along Purple street in Shampoo-Bananananana. The n people live in locations $1, \ldots, n$ along Purple street (which is as straight as a ruler),

It is time for redistricting Purple street. A district can have between Δ and 2Δ people living in it, for some prespecified parameter Δ (where $n/3 > \Delta > 0$). A district is a consecutive interval along Purple street. Every person is assigned to a district containing it. The districts are disjoint.

For every person in Purple street, you know their vote in the last election. Specifically, you are given an array $v[1 \,..\, n]$, where the vote of the ith person is $v[i]$, which is either equal to 0 or 1. A set $S \subseteq \{1, \ldots, n\}$ of people is t-good, if $|\#_0(S) - \#_1(S)| \leq t$, where $\#_0(S)$ (resp. $\#_1(S)$) is the number of people in S that voted for 0 (resp 1) in S.

Describe an algorithm, as fast as possible, that given $\Delta, v[1 \,..\, n]$ and t as input, outputs TRUE if there is a way to redistrict Purple street so that every district is t-good (the algorithm outputs FALSE otherwise). What is the running time of your algorithm? The algorithm you provide should be iterative (i.e., please do not use dictionary/hashing or recursion).

NETID: NAME:

5 (20 PTS.) You are given a graph G with n vertices and m edges. Think about this graph as a network of windows machines. For every vertex $v \in \mathsf{V}(\mathsf{G})$, you also given a value $b(v)$ which is one if it is *infected* (with a virus), and 0 otherwise. A vertex is ***vulnerable***, if it is already infected or there are at least k distinct infected nodes that can reach it (here, think about k as being a small number compared to n).

5.A. (10 PTS.) For the case that G is a DAG, describe an algorithm, as fast as possible, that computes all the vulnerable vertices of G. What is the running time of your algorithm? Argue (shortly) that your algorithm is correct.

5.B. (10 PTS.) Describe an algorithm, as fast as possible, for the case that G is a general directed graph. What is the running time of your algorithm?