CS 341, Spring 2019 A. Jamshidpey, E. Zima

ASSIGNMENT 1

DUE: Friday May 24, 6 PM. DO NOT COPY. ACKNOWLEDGE YOUR SOURCES.

Please read `http://www.student.cs.uwaterloo.ca/~cs341` for general instructions and policies. Also read Assignment section of the course outline for clarification of what "justify" means.
Note: All logarithms are base 2 (i.e., $\log x$ is defined as $\log_2 x$).
Note: For all algorithm design questions, you must give the algorithm, argue the correctness, and analyze time complexity.

1 Order notation and the growth rate of functions.

1.1. [4 marks] Using the basic definition of Θ-notation, prove that $6n(3n-1)(19n+2) \in \Theta(n^3)$. Do not use limit computation!

1.2. [10 marks] For each of the following pairs of functions $f(n)$ and $g(n)$, determine the "most appropriate" symbol in the set $\{o, \omega, \Theta, O, \Omega\}$ to complete the statement that $f(n) \in \quad (g(n))$ (if one of the symbols applies at all). "Most appropriate" means that you should not answer "O" if you could answer "o" or "Θ". Justify your answers even in the cases when none of symbols applies at all.

You may use the following (based on Skeina, p. 56), where $f(n) \ll g(n)$ is shorthand for $f(n) \in o(g(n))$:

$$1 \ll \log\log n \ll \log n \ll \log^2 n \ll \sqrt{n} \ll n \ll n\log n \ll n^2 \ll 2^n \ll n!$$

Furthermore, $n^a \in o(n^b)$ for $0 < a < b$, and $\log^a n \in o(n^b)$ for any $a, b > 0$, and $n^a \in o(2^n)$.

(a) $f(n) = 2019n^3 + 12871n^2 + 19$, $g(n) = \frac{2}{2019}n^4 + 2n$;

(b) $f(n) = \log^2(n^4)$, $g(n) = \sqrt{n}$;

(c) $f(n) = 16^{\log n^3} + n^5$, $\quad g(n) = \frac{1}{2}n^{12} + 100n^6$;

(d) $f(n) = n^2$, $g(n) = (\lceil \frac{n}{2} \rceil - \frac{n}{2})n^2$;

(e) $f(n) = 2^{\sqrt{n}}$, $\quad g(n) = n^{\log n^2}$.

1.3. [6 marks] Prove or disprove each of the following statements (give a counter example to disprove).

(a) If $\log(f(n)) \in \Omega(\log(g(n)))$ then $f(n) \in \Omega(g(n))$.

(b) If $f(n) \in O(h(n))$ and $g(n) \in O(h(n))$ then $\dfrac{f(n)}{g(n)} \in O(1)$.

(c) If $f(n) \in o(g(n))$ then $\log(f(n)) \in o(\log(g(n)))$.

2 Analysis of Run Time.

Analyze the following pseudocode and give a tight (Θ) bound on the running time as a function of n. You can assume that all individual instructions (including logarithm) are elementary, i.e. take constant time. Show your work.

(a) [6 marks]

```
l := 0;
for i = n + 1 to n^2 do
  for j = 1 to ⌈log i⌉ do
     l := l + 1
  od
od.
```

(b) [8 marks]

```
m := 1; l := 0; s := 0;
while m ≤ n do
  for j = n − m to n do
     l := l + 1
  od
  for j = 1 to ⌈log n⌉ do
     s := s + 1
  od
   m := 3m
od.
```

3 Reductions.

Note: For this section you can assume that all basic operations over rational numbers take constant time.

3.1. [6 marks] **Duplicates Problem.** Given a list of n rational numbers $a_1, \ldots, a_n$, the *duplicates decision problem* is to find out if there are distinct indices i and j such that $a_i = a_j$. Design an efficient algorithm for this problem. Your algorithm must have complexity in $o(n^2)$.

3.2. [10 marks] **Collinearity Test Problem.** Given an array of distinct points $P_1, \ldots, P_n \in \mathbb{Z}^2$, the problem is to find out if there are 3 points on the same line (collinear). A naive algorithm will test collinearity of all possible triples of points and have complexity $\Theta(n^3)$. Using reduction to part 3.1., design an efficient algorithm with complexity in $o(n^3)$.

4 Bonus.

[5 marks] Given a binary string (i.e., a finite sequence of 0's and 1's) we choose any two-digit substring 01 and replace it by a substring of the form 100....0 using an arbitrary (but finite) number of zeros. Prove by induction that this transformation can not be performed infinitely many times, i.e. this sequence of transformations must terminate for any input string.