CSCE629 Analysis of Algorithms

Homework 1

Texas A&M U, Fall 2019 *08/30/19*

Lecturer: Fang Song *Due: 09/06/19*

Instructions. Typeset your submission by LaTeX, and submit in PDF format. Your solutions will be graded on *correctness* and *clarity*. You should only submit work that you believe to be correct, and you will get significantly more partial credit if you clearly identify the gap(s) in your solution. You may opt for the "I'll take 15%" option (details in Syllabus). A random subset of the problems will be graded.

You may collaborate with others on this problem set. However, you must **write up your own solutions** and **list your collaborators and any external sources** for each problem.

1. (Piazza) **Attention: this problem is due Friday, August 30, 11:59pm CDT.**
   (a) (2 points) Enroll on Piazza `https://piazza.com/tamu/spring2019/csce440640/`.
   (b) (3 points) Post a note on Piazza describing: 1) a few words about yourself; 2) your strengths in CS (e.g., programming, algorithm, ...); 3) what you hope to get out of this course; and 4) anything else you feel like sharing. See instructions on how to post a note `https://support.piazza.com/customer/en/portal/articles/1564004-post-a-note`.

   The purpose is to help me know you all, and also get you known to your fellow students.

2. (10 points) (Order of growth rate) Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows function $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.
   - $f_1(n) = n^{2.5}$
   - $f_2(n) = \sqrt{2n}$
   - $f_3(n) = n + 10$
   - $f_4(n) = 10n$
   - $f_5(n) = 100^n$
   - $f_6(n) = n^2 \log n$

3. (10 points) (Order of growth rate) Take the following list of functions and arrange them in ascending order of growth rate.
   - $g_1(n) = 2^{\sqrt{n}}$
   - $g_2(n) = 2^n$
   - $g_3(n) = n(\log n)^3$

- $g_4(n) = n^{4/3}$
- $g_5(n) = n^{\log n}$
- $g_6(n) = 2^{2^n}$
- $g_7(n) = 2^{n^2}$
- $g_8(n) = n!$

4. (15 points) (Understanding big-O notation) Assume you have functions f and g such that $f(n)$ is $O(g(n))$. For each of the following statements, decide whether you think it is true or false and give a proof or counterexample.

   (a) $\log_2 f(n)$ is $O(\log_2 g(n))$.

   (b) $2^{f(n)}$ is $O(2^{g(n)})$.

   (c) $f(n)^2$ is $O(g(n)^2)$.

5. (Basic proof techniques) Read the chapter on `Proof by Induction` by Erickson (`http://jeffe.cs.illinois.edu/teaching/algorithms/notes/98-induction.pdf`), and do the following.

   (a) (10 points) Prove that every integer (positive, negative, or zero) can be written in the form $\sum_k \pm 3^k$, where the exponents k are distinct non-negative integers. For example:

$$42 = 3^4 - 3^3 - 3^2 - 3^1$$
$$25 = 3^3 - 3^1 + 3^0$$
$$17 = 3^3 - 3^2 - 3^0$$

   (b) (10 points) A binary tree is *full* if every node has either two children (an internal node) or no children (a leaf). Give two proofs (proof by induction and proof by contradiction) of the following statement: in any full binary tree, the number of leaves is exactly one more than the number of internal nodes.

6. (10 points) (Reduction) Given a set X of n Boolean variables $x_1, \ldots, x_n$; each can take the value 0 or 1 (equivalently, "false" or "true"). By a term over X, we mean one of the variables x_i or its negation $\bar{x}_i$. Finally, a clause is simply a disjunction of distinct terms $t_1 \vee t_2 \vee \ldots \vee t_\ell$. (Again, each $t_i \in \{x_1, x_2, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$.) We say the clause has length l if it contains l terms.

   A truth assignment for X is an assignment of the value 0 or 1 to each x_i; in other words, it is a function $v : X \to \{0, 1\}$. The assignment v implicitly gives $\overline{x_i}$ the opposite truth value from x_i. An assignment satisfies a clause C if it causes C to evaluate to 1 under the rules of Boolean logic; this is equivalent to requiring that at least one of the terms in C should receive the value 1. An assignment satisfies a collection of clauses $C_1, \ldots, C_k$ if it causes all of the C_i to evaluate to 1; in other words, if it causes the conjunction $C_1 \wedge C_2 \wedge \ldots C_k$ to evaluate to 1. In this case, we will say that v is a satisfying assignment with respect to $C_1, \ldots, C_k$; and that the set of clauses $C_1, \ldots, C_k$ is *satisfiable*. For example, consider the three clauses

$$(x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_3}), (x_2 \vee \overline{x_3}) .$$

A truth assignment v that sets all variables to 1 is not a satisfying assignment, because it does not satisfy the second of these clauses; but the truth assignment v' that sets all variables to 0 is a satisfying assignment.

We can now state the `Satisfiability` Problem, also referred to as SAT:

Given a set of clauses $C_1, \ldots, C_k$ over a set of variables $X = \{x_1, \ldots, x_2\}$, does there exist a satisfying truth assignment?

Suppose we are given an oracle $\mathcal{O}$ which can solve the `Satisfiability` problem. Namely if we feed a set of clauses to $\mathcal{O}$, it will tell us YES if there is a satisfying assignment for all clauses or NO if there exists none. Show that you can actually find as atisfying truth assignment v for $C_1, \ldots, C_k$ by asking questions to $\mathcal{O}$. Describe your procedure, and analyze how many questions you need to ask.