CS 170 Spring 2017

Efficient Algorithms and Intractable Problems
Prasad Raghavendra and Sanjam Garg

Homework 1

Instructions: You are welcome to form small groups (up to 4 people total) to work through the homework, but you **must** write up all solutions by yourself. List your study partners for homework on the first page, or "none" if you had no partners.

If using LaTeX (which we recommend), you may use the homework template linked on this Piazza post to get started.

Begin each problem on a new page. Clearly label where each problem and subproblem begin. The problems must be submitted in order (all of P1 must be before P2, etc). For questions asking you to give an algorithm, respond in what we will refer to as the *four-part format* for algorithms: main idea, pseudocode, proof of correctness, and running time analysis.

Read the Homework FAQ Piazza post on Piazza before doing the homework for more explanation on the four-part format and other clarifications for our homework expectations.

No late homeworks will be accepted. No exceptions. This is not out of a desire to be harsh, but rather out of fairness to all students in this large course. Out of a total of approximately 12 homework assignments, the lowest two scores will be dropped.

Special Questions:

- *Shortcut questions*: Short questions are usually easy questions that give you opportunities to practice basic materials. However, we understand that some of you are very familiar with the topics and do not want to spend too much time on easy questions. Therefore, we design shortcut questions for this purpose. A shortcut question usually has multiple parts that build upon each other and are ordered by their difficulty level. You can work on those in order or start from wherever you like. However you only need to submit the last part you are able to solve. For example, if a question has 5 parts (a, b, c, d, e), you are confident about part e, you should submit part e without any of the previous four parts. If you are confident about d but not sure about e, you should submit d for grading purposes. Please clearly indicate in your submission which part you are submitting.
- *Redemption questions*: It is important that you carefully read the posted solutions, even for problems you got right. To encourage this, you have the option of submitting a redemption file, a few paragraphs in which you explain, for each problem you choose to cover, what you did wrong and what the right idea was in your own words (not cutting and pasting from the solution!), and appending it to your homework. For example, suppose that as you review your solutions to HW1, you realize you had misunderstood question 3 and answered it incorrectly. You would write down what you just learned, and then submit it in your HW2 assignment the following week. Because these are mainly for your benefit, feel free to format them however is most useful for you.
- *Extra credit questions*: We might have some extra credit questions in the homework for people who really enjoy the materials. However, please note that you should do the extra credit problems only if you really enjoy working on these problems and want an extra challenge. It is likely not the most efficient manner in which to maximize your score.

Due Tuesday, January 31, at 11:59am

1. (★ level) Course Syllabus

Before you answer any of the following questions, please read over the syllabus carefully. The syllabus is pinned on the Piazza site. For each statement below, write *OK* if it is allowed by the course policies and *Not OK* otherwise.

(a) You ask a friend who took CS 170 previously for her homework solutions, some of which overlap with this semester's problem sets. You look at her solutions, then later write them down in your own words.

(b) You had 5 midterms on the same day and are behind on your homework. You decide to ask your classmate, who's already done the homework, for help. He tells you how to do the first three problems.

(c) You look up a problem online to search an algorithm, write it in your words and cite the source.

(d) You were looking up Dijkstra's on the internet, and run into a website with a problem very similar to one on your homework. You read it, including the solution, and then you close the website, write up your solution, and cite the website URL in your homework writeup.

(e) You are working on the homework in one of the TA's office hours with other people. You hear that student John Doe asked the TA if his solution is correct or not and the TA is explaining it. You join their conversation to understand what John has done.

2. (★★ level) Shortcut Question: Asymptotic Complexity Comparisons

Please read the instruction for shortcut questions carefully. You only need to submit the part that you are able to solve. Please clearly indicate which part you submit.

(a) Order the following functions so that $f_i \in O(f_j) \iff i \leq j$. Do not justify your answers.

(a) $f_1(n) = 3^n$
(b) $f_2(n) = n^{\frac{1}{3}}$
(c) $f_3(n) = 12$
(d) $f_4(n) = 2^{\log_2 n}$
(e) $f_5(n) = \sqrt{n}$
(f) $f_6(n) = 2^n$
(g) $f_7(n) = \log_2 n$
(h) $f_8(n) = 2^{\sqrt{n}}$
(i) $f_9(n) = n^3$

(b) Prove that for any $\varepsilon > 0$ we have $\log x \in O(x^{\varepsilon})$.

(c) Let $f(\cdot)$ be a function. Consider the equality

$$\sum_{i=1}^{n} f(i) \ \in \ \Theta(f(n)),$$

give a function f_1 such that the equality holds, and a function f_2 such that the equality does not hold.

(d) Prove or disprove: If $f : \mathbb{N} \to \mathbb{N}$ is any positive-valued function, then either (1) there exists a constant $c > 0$ so that $f(n) \in O(n^c)$, or (2) there exists a constant $\alpha > 1$ so that $f(n) \in \Omega(\alpha^n)$.

3. (★★★ level) Recurrence Relations

Derive an asymptotic *tight* bound for the following $T(n)$. Cite any theorem you use.

(a) $T(n) = 2 \cdot T(\frac{n}{2}) + \sqrt{n}$.

(b) $T(n) = T(n-1) + c^n$ for constants $c > 0$.

(c) $T(n) = 2T(\sqrt{n}) + 3$, and $T(2) = 3$.

4. (★★★ level) Bound the running time

For each of the following functions, give a tight $\Theta(\cdot)$ bound on the running time. Justify your answer. For this problem you can assume all arithmetic operations run in $O(1)$ time.

Example:

```
function f1(n):
    if n < 3:
         return n
    else:
         return f1(n-1) + f1(n-2)
```

Solution: This algorithm runs in exactly $\Theta(fib(n))$ time, or $\Theta((\frac{1}{2}(1+\sqrt{5}))^n)$. There are exactly $fib(n)$ calls used to compute the nth Fibonacci number.

(a) (Hint: how is this different than the example?)

```
function f2(n):
    if n < 3:
         return n
    else:
         return f2(n/2) + f2(n/4) // division truncates
```

(b)
```
function f3(n):
    if n == 1: return 0
    if n is even:
        return f3(n/2)
    else:
        return f3(n+1)
```

(c) (Extra Credit Question: *For Eternal Fame Or Eternal Misery*)

```
function f4(n):
    if n == 1: return 0
    if n is even:
        return f4(n/2)+1
    else:
        return f4(3n+1)+1
```

5. (★★★★ level) γ-approximate Median

In the lecture, we saw an algorithm to compute Median that runs in *expected linear time*, i.e., an algorithm whose average run-time is linear, but in rare cases could take much longer. In this problem, we will design a different algorithm to compute the median that runs in linear time, always!

Consider an array $A[1 \ldots n]$ of distinct integers. The *median* of A is the $\lceil n/2 \rceil$-th largest integer in the array. A *γ-approximate median* of A is one of the integers that is not one of the $\lfloor \gamma n - 1 \rfloor$ smallest integers, and not one of the $\lfloor \gamma n \rfloor$ largest integers. For example, the median is a $1/2$-approximate median. You may assume that $0 < \gamma \leq \frac{1}{2}$.

(a) You should see that finding GAM is an easier problem than finding the exact median. Now assume that you have a linear time algorithm for GAM. Use this GAM algorithm as a subroutine to design a linear time algorithm for finding the median. Please note that γ is a fixed constant.

(You need to give a four-part solution for this part)

(b) Consider the following high-level algorithm:

1. Partitioning the elements into $\lceil n/5 \rceil$ groups of 5 elements each where the last group may have less than 5.
2. Find the exact median of each group for all groups in $O(n)$ time.
3. Find the exact median, x, of the $\lceil n/5 \rceil$ group medians. If you need to choose between two medians, always prefer medians that derived from a full group of 5, over "pseudo-medians" that derived from a group with less than 5.

Give an algorithm for step 2 (You only need to briefly describe your idea and why it is $O(n)$). Prove that there exists a choice of γ and a constant $n_0 > 0$, such that x is a γ-approximate median when $n \geq n_0$. (Hint: you can consider the choice that $\gamma = \frac{3}{10}$)

6. (★★★★★ level) Merged Median

Given k sorted arrays of length l, design an efficient algorithm to finding the median element of all the $n = kl$ elements. Your algorithm should run asymptotically faster than $O(n)$.

(You need to give a four-part solution for this problem.)