CS 161 (Stanford, Winter 2024)   Homework 1

Style guide and expectations: Please see the "Homework" part of the "Resources" section on the webpage for guidance on what we look for in homework solutions. We will grade according to these standards. You should cite all sources you used outside of the course material.
What we expect: Make sure to look at the "**We are expecting**" blocks below each problem to see what we will be grading for in each problem!

Exercises. The following questions are exercises. We suggest you do these on your own. As with any homework question, though, you may ask the course staff for help.

1 Exercise: Course policies (1 pt.)

Have you thoroughly read the course policies on the webpage?

[We are expecting: The answer "yes."**]**

2 Exercise: Complexity Bounds (6 pt.)

For each blank, indicate whether A_i is in O, Ω, or Θ of B_i. More than one space per row can be valid.

[We are expecting: All valid spaces in the table to be marked (checkmark, "x", etc.). No explanation is required.**]**

(The first two lines are already filled out for you as an example.)

A	**B**	O	Ω	Θ
$10n$	n	✓	✓	✓
10	n	✓		
n^2	n^3			
2^n	n^3			
2^n	2^{2n}			
2^n	2^{n+2}			
$\log(n^n)$	$\log(n!)$			
n^{n+1}	n^n			
$n^{1/n}$	1			
$\log\log n$	$\log n$			
$\log\sqrt{n}$	$\log n$			
1	0.5^n			
$n^{\log 5}$	$5^{\log n}$			
$(1+\sin n)^n$	n			

3 Exercise: Big-Oh Definitions

Decide whether the following are true or false, and then prove it **using the definitions of Big-Oh**.

3.1 (3 pt.)

If $f(n) = 3n + \log n$ and $g(n) = 2n - 1$, then $f(n)$ is $O(g(n))$.

3.2 (3 pt.)

If both $f(n)$ and $g(n)$ are both $O(h(n))$, then $f(n) + g(n)$ is $O(h(n))$.

3.3 (3 pt.)

If both $f(n)$ and $g(n)$ are both $O(h(n))$, then $f(n) \cdot g(n)$ is $O(h(n))$.

[We are expecting: For all parts, a short but formal proof.**]**

Problems. The following questions are problems. You may talk with your fellow CS 161-ers about the problems. However:

- Try the problems on your own *before* collaborating.
- Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
- If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

4 Big-Oh via Induction (8 pt.)

The Fibonacci numbers are a famous sequence defined as

$$F_0 = 0 \qquad F_1 = 1 \qquad F_{n+2} = F_n + F_{n+1}$$

For example, the first few terms of the Fibonacci sequence are

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

Show by induction that

1. $F_n = O(3^n)$.
2. $F_{2n} = \Omega(2^n)$.

[We are expecting: For each part, a formal proof by induction. Make sure to clearly label your inductive hypothesis, base case, inductive step, and conclusion.**]**

5 Probability Refresher (9 pt.)

Plucky the Pedantic Penguin goes fishing every morning before studying for CS161. On any day, with probability 90%, Plucky doesn't catch anything, and with probability 10%, Plucky catches one fish. (Plucky stops fishing on a day as soon as they catch a fish, so Plucky never catches more than one fish on the same day.)

1. What is the expected number of fish that Plucky catches on a given day?

   [We are expecting: A number and a brief but formal explanation.**]**

2. Plucky's success in fishing on different days is independent. How many days, in expectation, will Plucky have to wait until the next time they catch a fish?

   [We are expecting: A number and a brief but formal explanation.**]**

3. What is the expected number of fish that Plucky catches in n days?

   [We are expecting: A number and a brief but formal explanation.**]**

6 Algorithm Design (15 pt.)

In this problem, we will be asking you to write a function in pseudocode. For reference, here is a pseudocode sample.

Algorithm 1: Replace an element in an array of length n with a given element

```
Input: An array $A$ of size $n$, an element $e$ present in the array and the new element $E$
Output: Updated array
$i \leftarrow 0$
while $i < n$ do
    currElement $\leftarrow A[i]$
    if $currElement = e$ then
        $A[i] \leftarrow E$
        return $A$
    else
        $A[i] \leftarrow$ currElement
    $i \leftarrow i + 1$
return $A$        // If element e is not found, return the original array
```

The following definitions will also be important for this problem:

Degree of a polynomial: The degree of a polynomial is the highest degree of the individual terms which have a non-zero coefficient. For example the degree of $P(x) = 3x^4 + 2x^2 + 3x + 1$

is 4 (because the coefficient of x^4 is non-zero)

Product of two polynomials: The product of two polynomials a, b gives a polynomial $c = a \times b$ with degree $\deg(c) = \deg(a) + \deg(b)$. Consider the following example:

Example product: For $a = x^4 + 3x - 5$, $b = x - 1$:

$$\begin{aligned} c = a \times b &= (x^4 + 3x - 5) \times (x - 1) \\ &= x^5 - x^4 + 3x^2 - 3x - 5x + 5 \\ &= x^5 - x^4 + 3x^2 - 8x + 5. \end{aligned}$$

Your objective: You will design an algorithm to efficiently multiply polynomials. Design an algorithm that takes two integers n and m, and also n polynomials, each of degree m, and returns the product of all of these polynomials. You are given a function MultiplyPoly(a, b) which multiplies two polynomials in $O(\deg(a) + \deg(b))$ time and can be used as a subroutine.[1].

Below are two examples of the desired behavior of the algorithm.

Example 1.

Input:

1. $n = 3$
2. $m = 2$
3. $P_1 = x^2 - 1$
4. $P_2 = -2x^2 + x - 3$
5. $P_3 = 6x^2 - 5x$

Output:

$-12x^6 + 16x^5 - 11x^4 - x^3 + 23x^2 - 15x$

Example 2.

Input:

1. $n = 2$
2. $m = 4$
3. $P_1 = 2x^4 - x^2 + x - 1$
4. $P_2 = -x^4 + x^3 + x^2 - 3x - 3$

[1]This is a simplified assumption, as in reality the fastest known algorithm for multiplying two polynomials, the *Fast Fourier Transform (FFT)*, is a little bit slower with running time $O(\deg(a) + \deg(b) \log((\deg(a) + \deg(b)))$. You don't need to know anything about FFT for this problem.

Output:

$-2x^8 + 2x^7 + 3x^6 - 8x^5 - 5x^4 + 3x^3 - x^2 + 3$

[We are expecting: A succinct but clear English description, pseudocode implementation, and running time analysis.**]**