Honors Discrete Mathematics: Homework 6

Due on March 28, 2022 at 11:59pm

Professor Gerandy Brito Spring 2022

You may collaborate with other students in this class, but (1) you must write up your own solutions in your own words, and (2) you must write down everyone you worked with at the top of the page, or "no collaborators" if you did it all on your own. Additionally, if you used any outside websites or textbooks besides the course text, please cite them here. You may not use question-answer sites like Chegg or MathOverflow.

Sarthak Mohanty

Exercise 1

First, read the section in the Appendix below on Upper and Lower Bounds. Then, find upper and lower bounds for the runtime of the algorithm represented by the summation

$$\sum_{i=1}^{n}\sum_{j=1}^{i^2}\sum_{k=j}^{i^2} c,$$

where n is the number of times the outer loop is iterated and c is the execution cost of the statement that contributed most to the running time of the algorithm. Remember that your upper and lower bounds must differ by only a constant.

Exercise 2

Let $a \equiv 4 \pmod{13}$ and $b \equiv 9 \pmod{13}$. Find the smallest positive value of c such that

(a) $c \equiv 9a \pmod{13}$.

(b) $c \equiv 11b \pmod{13}$.

(c) $c \equiv ab \pmod{13}$.

(d) $c \equiv 2a + 3b \pmod{13}$.

(e) $c \equiv a^2 + b^2 \pmod{13}$.

(f) $c \equiv a^3 - b^3 \pmod{13}$.

(g) $c \equiv b^{-1} \pmod{13}$.

(h) $c \equiv a^{12} \pmod{13}$.

(i) $c \equiv a^{2021} \pmod{13}$.

(j) $c \equiv (a - b)^{2021} \pmod{13}$.

(You do not need to justify your work.)

Exercise 3

Important: For this question, you may not use any outside resources besides the textbook.

Consider the Euclidean algorithm, which computes $\gcd(a, b)$ for two integers a, b, such that $0 \le b \le a$.

(a) Find $\gcd(12345, 54321)$ using the Euclidean algorithm. Outline each iteration of the process, as in Chapter 4.3 Example 16 in the textbook.

Next, express $\gcd(12345, 54321)$ as a linear combination of 12345 and 54321. Outline each iteration of the process, as in Chapter 4.3 Example 17 in the textbook.

(b) The *extended Euclidean algorithm* is an algorithm that not only computes $\gcd(a, b)$, but also expresses $\gcd(a, b)$ as a linear combination with Bézout coefficients of the integers a and b. The main advantage of the algorithm is that it only requires one pass through the steps of the Euclidean algorithm, rather than two passes, as in part (a). The procedure is as follows:

Initialization. Set $s_0 = 1$, $s_1 = 0$, $t_0 = 0$, and $t_1 = 1$.

Inductive Step. For $i = 2, 3, \ldots, n$, let $s_i = s_{i-2} - q_{i-1}s_{i-1}$ and $t_i = t_{i-2} - q_{i-1}t_{i-1}$, where each q_i is the i-th quotient in the divisions used when the Euclidean algorithm finds $\gcd(a, b)$.

Termination. Return $\gcd(a, b) = s_n a + t_n b$.

Run the extended Euclidean algorithm to find $\gcd(12345, 54321)$ and express it as a linear combination of 12345 and 54321. Your work should be given in the form of a table, whose first row is given below.

a_i	b_i	q_i	s_i	t_i
54321	12345	N/A	1	0

Exercise 4

(a) Prove that if a and b are both positive integers, then $2^a - 1 \pmod{2^b - 1} = 2^{a \bmod b} - 1$.

Note: The following proposition may be useful: $(\forall a, b \in \mathbb{N}^+)(\exists! q, r \in \mathbb{Z})((0 \le r < b) \land (a = qb + r))$. This is commonly known as Euclid's division lemma.

(b) Use part (a) to show that if a and b are both positive integers, then $\gcd(2^a - 1, 2^b - 1) = 2^{\gcd(a,b)} - 1$.

Exercise 5

Recall that we denote the set of congruency classes modulo n as $\mathbb{Z}_n$.

(a) A *zero divisor* is a nonzero element $\bar{a}$ such that, there exists a nonzero element $\bar{b}$ satisfying $\bar{a}\bar{b} = \bar{0}$. How many elements in $\mathbb{Z}_{16}$ are zero divisors? Show your work.

(b) An element $\bar{a}$ is a *unit* if there exists a nonzero element $\bar{b}$ satisfying $\bar{a}\bar{b} = \bar{1}$. How many elements in $\mathbb{Z}_{14}$ are units? Show your work.

(c) What is the relationship between the number of zero divisors and the number of units in any $\mathbb{Z}_n$? You do not have to justify your answer, but consider possible explanations for this phenomenon.

Exercise 6

(a) Find the last digit of $17^{17^{17}}$.

(b) Find the last two digits of 1987^{1987}.

Hint: It may benefit you to use the CRT.

Exercise 7

(a) Derive an non-recursive formula with finitely many terms for the number of trailing zeroes in $n!$, where n is a positive integer greater than or equal to 5. Thoroughly explain your answer.

Hint: Floor/ceiling functions are feeling lonely in this course. Perhaps it is time to include them.

(b) Using the formula derived in part (a), determine the last $\underline{8}$ digits in 32!. Show your work.

Exercise 8

You have returned to The Farm, equipped with your new knowledge of modular arithmetic. Good timing, as the wise farmer now has a dilemma he'd like your help with.

Yesterday, he instructed his 7 nieces to count the number of cows on the farm. However, it seems the farmer's wisdom was not passed on throughout the family tree, as each of the nieces forgot the exact number overnight. You question each of the nieces to obtain more information:

- Betsy counted the cows, but all she remembers is that her count was off by 1 in the 1's place.
- Clara counted the cows, but all she remembers is that her count was off by 1 in the 10's place.
- Delilah counted the cows, but all she remembers is that her count was off by 1 in the 100's place.
- Evelyn, Fiona, Grace, and Helen all counted the cows. They remember that they got the correct count.

The farmer's one saving grace is that at the end of the day, the nieces tallied up each of their counts and wrote the sum down on a piece of paper, which reads 3162.

Your task is to find the correct total number of cows on the farm. Along the way, you should be able to determine the incorrect counts that Betsy, Clara, and Delilah obtained. Clearly explain each step you take to obtain your answers.

Note: We do not consider changing digit 9 to 0 or 0 to 9 as being off by 1, since in this case we would be off by 9.

Appendix

Upper and Lower Bounds

Some summations are difficult to work with. In this part, we find upper and lower bounds for running time. This allows us to simplify the summations.

In adopting this approach, we require that the upper and lower bounds must be the same function, only differing by a constant. The constant for the upper bound must be larger than or equal the constant for the lower bound.

To obtain an upper bound in this approach, we remove terms of expression being subtracted, if helpful, and substitute terms (usually, but not necessarily, the upper/top bound of the summation) into the expression. To obtain a lower bound, we split summations to reduce size, if helpful, and substitute terms (usually, but not necessarily, the lower/bottom bound of the summation) into the expression.

Example. Find the runtime of the algorithm[1]

$$\sum_{i=1}^{n}\sum_{j=i}^{n} c = c\sum_{i=1}^{n}(n-i).$$

Finding an upper bound is straightforward:

$$c\sum_{i=1}^{n}(n-i) \leq c\sum_{i=1}^{n} n = cn^2.$$

We now find a lower bound.

$$c\sum_{i=1}^{n}(n-i) \geq c\sum_{i=n/2}^{n}(n-i).$$

Our goal is to make the original summation in the previous equation at least as large as a quadratic function. Let us try to substitute $n/2$ or n for i and see what happens.

- Substituting $n/2$ for i, we get

  $$c\sum_{i=n/2}^{n}(n-i) \leq c\sum_{i=n/2}^{n}\left(n-\frac{n}{2}\right),$$

  but this does not work because it makes the lower bound larger instead of smaller.

- Substituting n for i, we get

  $$c\sum_{i=n/2}^{n}(n-i) \geq c\sum_{i=n/2}^{n}(n-n) = 0,$$

  but this does not work either because it makes the lower bound too small.

However, splitting the summation another way can help us get where we want.

$$c\sum_{i=1}^{n}(n-i) \geq c\sum_{i=1}^{n/2}(n-i) \geq c\sum_{i=1}^{n/2}\left(n-\frac{n}{2}\right) = c\left(\frac{n}{2}\right)\left(\frac{n}{2}\right) = \left(\frac{c}{4}\right)n^2.$$

Thus, the runtime is $\bar{c}n^2$ where $\frac{c}{4} \leq \bar{c} \leq c$.

[1] Technically, $\sum_{j=1}^{n} = (n-i+1)$, but we ignore the negligible ‘1’ term in this course.