CS5800 Algorithms *Out: 10 September 2021*

Problem Set 1

Ravi Sundaram *Due: 17 September 2021*

Problem 1 (Asymptotics) 20

1. Return a list of the following functions separated by the symbol $\equiv$ or $\ll$, where $f \equiv g$ means $f = \Theta(g)$ and $f \ll g$ means $f = O(g)$. For example, if the functions are $\log n, n, 5n, 2^n$ a correct answer is $\log n \ll n \equiv 5n \ll 2^n$. All logarithms are in base 2. Prove each relation formally (the adjacent ones in the ordering you come up with).

   (a) $1/n$

   (b) $n \log n$

   (c) $\log n!$

   (d) $n^{1/\log n}$

   (e) $\log^2 n$

   (f) $\log^2(n \log n)$

2. For some given functions, f, g, h, decide which of the following statements is correct and give a formal proof.

   (a) If $f(n) = O(g(n))$, then $f(n) = O(\log g(n))$

   (b) If $f(n) = \Theta(g(n))$, then $(f(n)^2) = \Theta(g(n)^3)$

   (c) If $f(n) = \Omega(n * g(n))$, $g(n) = \Omega(n * h(n))$, then $f(n) = \Omega(n^2 * h(n))$

Solution:

1. The ordering is:

$$1/n \ll n^{1/\log n} \ll \log^2 n \equiv \log^2(n \log n) \ll n \log n \equiv \log n!$$

   Proof:
   We have $\lim_{n\to\infty} 1/n = 0$ and $\log n^{1/\log n} = 1$, thus $1/n \ll n^{1/\log n}$.

   Since $\lim_{n\to\infty} \log^2 n = \infty$, it leads to $n^{1/\log n} \ll \log^2 n$.

   Using L'Hospital's rule,

$$\lim_{n\to\infty} \frac{\log^2(n \log n)}{\log^2 n} = \lim_{n\to\infty} \frac{\log(n \log n)}{\log n} = \lim_{n\to\infty} \frac{\log n + 1}{\log n} = 1$$

   thus, $\log^2 n \equiv \log^2(n \log n)$.

   Again,

$$\lim_{n\to\infty} \frac{n \log n}{\log^2(n \log n)} = \lim_{t\to\infty} \frac{t}{\log^2 t} = \lim_{t\to\infty} \frac{t}{2 \log t} = \infty$$

   thus, $\log^2(n \log n) \ll n \log n$.

Using Stirling's approximation that $\log n! = n\log n - n + O(\log n)$, then

$$\lim_{n\to\infty} \frac{\log n!}{n\log n} = \lim_{n\to\infty} \frac{n\log n - n + O(\log n)}{n\log n} = 1$$

thus, $n\log n \equiv \log n!$.
Q.E.D.

2. (a) False. We can raise a contradiction by letting $f(n) = n, g(n) = n^2$, but $f(n) = O(\log g(n))$ does not hold.

   (b) False. We can raise a contradiction by letting $f(n) = n, g(n) = 2n$, then $(f(n)^2) = O(g(n)^3)$.

   (c) True. Given that $f(n) \gg n * g(n)$ and $g(n) \gg n * h(n)$, we could find c_1, c_2 such that $f(n) \geqslant c_1 n g(n)$ for $n \geqslant n_1$ and $g(n) \geqslant c_2 n h(n)$ for $n \geqslant n_2$. Then, let $n_0 = \max\{n_1, n_2\}$, we have $f(n) \geqslant c_1 c_2 n^2 h(n)$ for $n \geqslant n_0$. Thus, $f(n) \gg n * g(n)$.

■

Problem 2 (Induction) 10+15

1. Let q be a real number other than 1. Use induction on n to prove that $\sum_{i=0}^{n-1} q^i = (q^n - 1)/(q-1)$
2. Show that any value of 12 cents or more can be attained using some arbitrary amount of 4 and 5 cent denomination coins (this problem is an example of strong induction).

Solution:

1. Proof:
   Basic case: when $n = 1$, $\sum_{i=0}^{n-1} q^i = 1 = (q^n - 1)/(q - 1)$
   Hypothesis: assuming $\sum_{i=0}^{n-1} q^i = (q^n - 1)/(q - 1)$ holds when $n = k - 1$
   Then we can show that when $n = k$,

$$\begin{aligned}\sum_{i=0}^{k-1} q^i &= \left(\sum_{i=0}^{k-2} q^i\right) + q^{k-1} \\ &= (q^{k-1} - 1)/(q - 1) + q^{k-1} \\ &= (q^k - 1)/(q - 1)\end{aligned}$$

   Q.E.D.

2. Proof:
   Basic case:

$$\begin{aligned}12 &= 4(3) + 5(0) \\ 13 &= 4(2) + 5(1) \\ 14 &= 4(1) + 5(2) \\ 15 &= 4(0) + 5(3)\end{aligned}$$

   Hypothesis: there exsist a value of k cents with $k \geqslant 15$ that can be attained using 4 and 5 cent denomination coins.

Then, $k-1$ can be expressed as $k-1 = k-5(1)+4(1)$, showing that it can be attained using coints of 4 and 5 cents. Similarly, we have $k-3 \geqslant 12$ which can also be attained using 4 and 5 cent denomination coins. By adding a 4 cent coin to the amount of $k-3$ cents, we now attain the value of $k-1+3 = k+1$.
Q.E.D.

■

Problem 3 (Modular Arithmetic) 25

We are going to prove Fermat's Little Theorem.

1. Let p be a prime. Show that for any integer x not divisible by p, $x^{p-1} \equiv 1 \bmod p$. Hint: Consider the sequence $x, 2x, ..., (p-1)x$.

Solution:

Proof: Any two numbers of the sequence $x, 2x, ..., (p-1)x$, say mx and nx, have different results on mod p since x is not divisible by p. Then the sequence must have $1, 2, ..., p-1$ as their results of mod p, i.e.,

$$\begin{aligned} & x \cdot 2x \cdots (p-1)x \equiv 1 \cdot 2 \cdots (p-1) \bmod p \\ \Leftrightarrow & (p-1)! x^{p-1} \equiv (p-1)! \bmod p \\ \Leftrightarrow & x^{p-1} \equiv 1 \bmod p \end{aligned}$$

■

Problem 4 (Programming: modular-exponentiation) 30

The Hackerrank link will be posted on Canvas. In addition to the below description, it also contains more formal requirements for how your program should behave.

Description: Implement modular exponentiation in a way that outputs the intermediary steps of the algorithm.

Problem Statement: **IMPORTANT** You are NOT allowed to use built in language functions which trivialize the task of computing exponents. Any submissions which use this and avoid the task at hand will be given a 0.

In this problem, you will have to efficiently implement modular exponentiation. Recall that the problem of modular-exponentiation is, given positive integers a and n, and a non-negative integer x, calculate $a^x \mod n$.

One way of doing this is exponentiation by squaring. It involves repeatedly squaring the base a and reducing it using modulus: $a^2 \mod n$. Doing so yields the values a, $a^2 \mod n$, $a^4 \mod n$, $a^8 \mod n, \ldots$, etc. By combining these in the correct way, and using the fact that every number has a binary representation, we can compute $a^x \mod n$ in time $O(\log x)$.

Implement modular-exponentiation by squaring, and output the intermediary values of $a^{2^i} \mod n$, as well as the final value $a^x \mod n$.

Note that in addition to the above, the challenge page also describes the input/output format, and gives a few examples. While some of it is reproduced here, the hackerrank version is authoritative.

Input Format: The input will be exactly one line, with three space delimited integers a, x, and n.

Constraints: The integers will satisfy $2 \leq a, x, n \leq 2^{64}$.

Output Format: On the first line, output the d-bit binary representation of x.

On the next d lines, output the values:

$a^1 \mod n$,

$a^2 \mod n$,

$a^4 \mod n$,

$\vdots$

$a^{2^d} \mod n$,

On the last line, output $a^x \mod n$

Sample Input: 3 7 10

Sample Output: 111

3

9

1

7

Explanation: 7 in binary is written as 111.

$$3^1 = 3 \mod 10 \quad 3^2 = 9 \mod 10 \quad 3^4 = 81 = 1 \mod 10 \quad 3^7 = 2187 = 7 \mod 10$$