NYU TANDON SCHOOL OF ENGINEERING

CS-UY 2413: Design&Analysis of Algorithms—Spring 2017 Prof. Boris Aronov
Department of Computer Science and Engineering Prof. Lisa Hellerstein

Homework 2

Due Feb. 7 at the start of lecture (Hellerstein's sections).
Due Feb. 8 at the start of lecture (Aronov's section).
This homework should be handed in on paper, not electronically. No late homeworks accepted. Contact your professor for special circumstances.

Policy on collaboration on this homework: Feel free to discuss this homework and write the answers in groups. Each group must have no more than THREE (3) people in it; all from the same section. Hand in ONLY ONE homework per group. On that homework, put the names, netIDs, and sections of ALL the people in the group. Your name should appear on only one homework that is handed in. Homeworks handed in separately with identical or similar answers will be considered academic dishonesty.

Any homework submitted is expected to be the work of the students whose names are on it. If there is evidence that the work is not your own (such as copying from others, from the Internet, paying a third party to carry out the work, etc.), it will be treated as academic dishonesty and will be reported to the department and the Dean of Student Affairs. You will receive a zero on the homework, and if it happens again, you will receive an F in the course. It does not matter who copied from whom.

Format: Type or write very clearly, as described in the previous homework.

---

1. Consider the following function. It takes as input a number N, written in binary with no leading 0's. Let n be the number of bits in the binary representation of N. The function computes the value $1 + 2 + \ldots + N$.

```
function computesum(N)
-------------------------------------------------------------
Input: a number N > 0, written in binary (with no leading 0's)
Output: 1+2+...+N
-------------------------------------------------------------

result = 0
for i=1...N
     result = result + i
return result
```

   Answer the following questions about the executation of computesum(N). Note: Whenever you give an answer in big-Oh notation, the expression inside the big-Oh should

NOT contain a floor, ceiling, or the base of a logarithm. Of course, it shouldn't contain a leading constant either.

In giving running times, use the *bit model*, where we charge for bit operations.

(a) How many times is the statement `result=result+i` executed? Give an exact answer, and express it as a function of N.

(b) Repeat the previous question, but this time express the answer as a function of n. For this question, assume the worst case, namely that all the bits in the binary representation of N are 1's.

(c) What is the maximum time taken to execute the statement `result=result+i` once, out of all the times it is executed? Express your answer in big-Oh notation as a function of N.

(d) Repeat the previous question, but this time give the running time as a function of n.

(e) What is the running time of computesum(N), expressed as a function of N? Give your answer in big-Oh notation.

(f) Repeat the previous question, but this time give the running time as a function of n.

2. The function computesum(N) is not the fastest way to compute $1 + 2 + \ldots + N$.

(a) Write a new function, with no loops or recursion, that computes $1 + 2 + \ldots + N$. As before, assume that N is given in binary, without leading 0's, and that n is the number of bits.

(b) Express the running time of your function, in big-Oh notation, as a function of N. Use the bit model.

(c) Repeat the previous question, but this time give the running time as a function of n.

3. Let A be an array of length n. Each element of array A contains an n-bit binary number. Consider the problem of computing the product of all the numbers in A.

Here is one procedure for solving this problem:

```
function firstalg(A[1...n])

result = 1
for i=1 to n do
    result = result * A[i]
```

What is the running time of this procedure? Give your answer in big-Oh notation. Use the bit model.

Hint: Make sure that you consider how long `result` can get!

4. Find the value of each of the following expressions. That is, for each expression of the form $a \bmod b$, find the value c such that $c \equiv a \bmod b$ and $0 \leq c \leq b-1$.

   (a) $13 \bmod 7$

   (b) $100 \bmod 9$

   (c) $100 \bmod 20$

   (d) $-21 \bmod 10$

5. The textbook gives some substitution rules without proof. One says that $x \equiv x' \bmod N$ and $y \equiv y' \bmod N$ implies $x + y \equiv x' + y' \bmod N$.

   This can be shown using the definition of $a \equiv b \bmod N$ (meaning that N divides $a-b$), as follows:

   If N divides $x - x'$, and N divides $y - y'$, then $x - x' = q_1 N$ and $y - y' = q_2 N$ for some integers q_1 and q_2. Therefore,

$$\begin{aligned}(x+y) - (x'+y') &= (x-x') + (y-y') && (1)\\ &= q_1 N + q_2 N && (2)\\ &= (q_1+q_2)N && (3)\end{aligned}$$

   so N divides $(x + y) - (x' + y')$. It immediately follows, from the definition of $a \equiv b \bmod N$ that $(x + y) \equiv (x' + y') \bmod N$.

   (a) Prove the following, different substitution rule. In your proof, you can use the substitution rule above, and the definition of $a \equiv b \bmod N$. Do not use any other substitution rules.

   $x \equiv z \bmod N$ and $y \equiv z \bmod N$ implies $xy \equiv z^2 \bmod N$.

   (Hint: Multiply $x - z$ by y, and $y - z$ by x.)

   (b) Either prove the following statement or disprove it. If you prove it, you may use the above two substitution rules in your proof, but no others.

   For any integers a, b and c, if $a(b-c) \equiv 0 \pmod N$, then $a \equiv 0 \pmod N$ or $b \equiv c \pmod N$ (or both).

6. For each modular division problem below, specify whether it has a solution, or whether the answer is undefined. We consider an answer to the problem $a/b \pmod c$ to be undefined if b does not have an inverse $\pmod c$. *You do not have to compute the answer to the modular division problem, if it exists.* Justify your answer.

(a) $20/15 \pmod{50}$

(b) $17/14 \pmod{33}$

(c) $109/19 \pmod{115}$

(d) $y/16 \pmod{x}$, for integers $x, y > 1$. (In this subquestion, is the answer always the same or does it depend on the value of x? Of y? Justify your answer.)

7. In this problem, use **Fermat's Little Theorem**. If p is prime, then for every integer a with $1 \leq a < p$,

$$a^{p-1} \equiv 1 \pmod{p}.$$

You may also use substitution rules from class and/or the textbook.

(a) What is the value of $225^{52} \bmod 53$? Use the fact that 53 is prime.

(b) What is the value of $(256^{104} + 7 \cdot 512^{104}) \bmod 53$?

(c) If p prime, what is $(p+1)^{(p-1)} \bmod p$? What about $p^{p-1} \bmod p$?

8. Use `extended-euclid(90, 66)` to find d, x, and y where $d = \text{GCD}(90, 66)$ and $d = 90x + 66y$.

Note: Using our textbook, you can calculate these values using two different but closely related approaches. You can either execute the recursive pseudocode that is given, or you can iteratively calculate the values, bottom-up, with back-substitution. If you find the textbook explanation confusing, try another one, such as `http://www-math.ucdenver.edu/~wcherowi/courses/m5410/exeucalg.html`

There is more than one pair of values x, y such that $ax + by = d$. Different versions of `extended-euclid` may find different values for x and y, depending on how the base of the recursion is handled! You will get full credit for any triple (d, x, y) provided d is the correct GCD, and $d = 90x + 66y$. Check your answer and show your work.