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

Solutions to Homework 2

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.
Solution: N times.

(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.
Solution: $2^n - 1$ times. The maximum value of N which has n 1's in its binary representation is $2^n - 1$.

(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.
Solution: $O(log(N))$

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

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

(f) Repeat the previous question, but this time give the running time as a function of n.
Solution: $O(n \cdot 2^n)$. There are $2^n - 1$ numbers less than the maximum n bit number. Adding each of them to the maximum number takes $O(n)$ time each.

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.
Solution:

```
function computesum(N)
     return N(N+1)/2
```

(b) Express the running time of your function, in big-Oh notation, as a function of N. Use the bit model.
Solution: $O(log^2(N))$

(c) Repeat the previous question, but this time give the running time as a function of n.
Solution: $O(n^2)$. The addition takes $O(n)$ time while the multiplication and division each takes $O(n^2)$ time. The total running time is then $O(n^2) + O(n^2) + O(n)$ which is $O(n^2)$.

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!

Solution: $O(n^4)$. Long multiplication of a n bit number by a m bit number takes

$O(mn)$ time and will result in a product with approximately $(m+n)$ bits. Therefore, multiplication takes $O(n^2)$ for two n-bit numbers and will result in a product with approximately $2n$ bits. There are n multiplications and every number is n bits. The first multiplication will multiply two n bit numbers. This will take at most kn^2 time for some constant k and produce a $2n$ bit number. The second multiplication will multiply an n bit number by a $2n$ bit number. This will take at most $2kn^2$ time and produce a $3n$ bit number. As we keep multiplying, the nth multiplication multiplies an approximately n^2 bit number by an n bit number. This will take at most kn^3 time and produce a $n(n+1)$ bit number. So the total time would be $1(kn^2)+2(kn^2)+...+n(kn^2)$. This becomes $(kn^2)(1+2+...+n) = (kn^2)(n(n+1)/2) = O(n^4)$.

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 mod 7
   Solution: 6

   (b) 100 mod 9
   Solution: 1

   (c) 100 mod 20
   Solution: 0

   (d) -21 mod 10
   Solution: 9

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 z.)

Note: When we say $a \in \mathbb{Z}$, this means a is in $\mathbb{Z}$ (the integers) (i.e. a is an integer)

Solution:
$x \equiv z \bmod N$ implies $N \mid (x - z)$
$y \equiv z \bmod N$ implies $N \mid (y - z)$
Important fact: If $a \mid b$, then $a \mid bc$ for any integer c
If $N \mid (x - z)$ then $(x - z) = q_1 N$ for some integer q_1

$$x = q_1 N + z \tag{4}$$

If $N \mid (y - z)$ then $(y - z) = q_2 N$ for some integer q_2

$$y = q_2 N + z \tag{5}$$

Multiplying equation 4 with equation 5:
$xy = (q_1 N + z)(q_2 N + z)$
Subtracting z^2 from both sides:
$xy - z^2 = (q_1 N + z)(q_2 N + z) - z^2$
$xy - z^2 = q_1 q_2 N^2 + q_1 z N + q_2 z N + z^2 - z^2$
$xy - z^2 = q_1 q_2 N^2 + q_1 z N + q_2 z N$
$xy - z^2 = N(q_1 q_2 N + q_1 z + q_2 z)$ where $(q_1 q_2 N + q_1 z + q_2 z) \in \mathbb{Z}$
Thus, $N \mid (xy - z^2)$ and $xy \equiv z^2 \bmod N$

Alternative solution using hint:
$x \equiv z \bmod N$ implies $(x - z) = q_1 N$ for some integer q_1
$y \equiv z \bmod N$ implies $(y - z) = q_2 N$ for some integer q_2

$$(x - z) = q_1 N \tag{6}$$
$$(y - z) = q_2 N \tag{7}$$

Multiply equation 6 by y and equation 7 by z:

$$y(x - z) = q_1 y N \tag{8}$$
$$z(y - z) = q_2 z N \tag{9}$$

Adding equation 8 and 9:

$xy - yz + yz - z^2 = q_1yN + q_2zN$
$xy - z^2 = N(q_1y + q_2z)$ where $(q_1y + q_2z) \in \mathbb{Z}$
Thus, $N \mid (xy - z^2)$ and $xy \equiv z^2 \bmod N$

(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).

Solution: False. Let $N = 6$, $a = 2$, $b = 4$, and $c = 1$.
$a(b - c) \equiv 0 \pmod N$ becomes $2(4 - 1) \equiv 0 \pmod 6$ or $6 \equiv 0 \pmod 6$.
However, $2 \equiv 0 \pmod 6$ and $4 \equiv 1 \pmod 6$ are both false.
Thus, the statement is false.

Note: In general, it is possible for $ab \equiv 0 \pmod N$ even if $a \not\equiv 0 \pmod N$ and $b \not\equiv 0 \pmod N$. For example, if $a \cdot b = N$, $ab \equiv 0 \pmod N$ becomes $N \equiv 0 \pmod N$ which is certainly true. However, $a \equiv a \pmod N$ and $b \equiv b \pmod N$ since $a, b < N$.

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) $20/15 \pmod{50}$
Important Fact: $(a/b) (\mathrm{mod} N)$ exists if and only if $GCD(b, N) = 1$
Solution: $GCD(15, 50) = 5$. Thus, the solution is undefined.

(b) $17/14 \pmod{33}$
Solution: $GCD(14, 33) = 1$. Thus, the solution is defined.

(c) $109/19 \pmod{115}$
Solution: $GCD(19, 115) = 1$. Thus, the solution is defined.

(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.)
Solution: If we want the solution to be defined, the $GCD(16, x) = 1$.
The answer depends on x. More specifically, x cannot be a multiple of 2 since x and 16 need to be relatively prime.

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.
Useful Fact: $AB \bmod C = ((A \bmod C)(B \bmod C)) \bmod C$
Solution: $225^{52} \bmod 53 = [(25^{52})(9^{52})] \bmod 53 = (25^{52} \bmod 53)(9^{52} \bmod 53)$
By Fermats Little Theorem, we know that both $25^{52} \bmod 53$ and $9^{52} \bmod 53$ are equal to 1.
Thus, $(1 \cdot 1) \bmod 53 = 1$

(b) What is the value of $(256^{104} + 7 \cdot 512^{104}) \bmod 53$?
Solution:
$(256^{104} + 7 \cdot 512^{104}) \bmod 53$
$= 256^{104} \bmod 53 + (7 \cdot 512^{104}) \bmod 53$
$= (256^{52} \bmod 53) \cdot (256^{52} \bmod 53) + (7 \bmod 53) \cdot (512^{52} \bmod 53) \cdot (512^{52} \bmod 53)$
By Fermats Little Theorem:
$= 1 \cdot 1 + 7 \cdot 1 \cdot 1$
$= 1 + 7$
$= 8$

(c) If p prime, what is $(p+1)^{(p-1)} \bmod p$? What about $p^{p-1} \bmod p$?
Solution:
$(p+1)^{(p-1)} \bmod p$
Useful fact: $A^B \bmod C = ((A \bmod C)^B) \bmod C$
Therefore, $(p+1)^{(p-1)} \bmod p$ becomes:
$(((p+1) \bmod p)^{(p-1)}) \bmod p$
Since $((p+1) \bmod p) = 1$, we get:
$(1^{(p-1)}) \bmod p = 1 \bmod p = 1$.

$p^{(p-1)} \bmod p$
This is trivial. $ab \bmod a = 0$ regardless of the value of a.
Thus, $p^{(p-1)} \bmod p = (p \cdot p \cdot \ldots \cdot p \cdot p) \bmod p = 0$

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.

Solution:

```
extended-euclid(90, 66):
d = 90x + 66y

Forward:
90 = 66(1) + 24
66 = 24(2) + 18
24 = 18(1) + 6
18 = 6(3) + 0

Backwards:
6 = 24(1) + 18(-1)
6 = 24(1) + (66 + 24(-2))(-1)
6 = 66(-1) + 24(3)
6 = 66(-1) + (90(1) + 66(-1))(3)
6 = 90(3) + 66(-4)

Thus, we get d = 6, x = 3, y = -4
```