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

Homework 3 Solutions

Due Feb. 21 at the start of lecture (Hellerstein's sections).
Due Feb. 22 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: The policy for collaboration on this homework is the same as in HW1 and HW2. By handing in this homework, you accept that policy. Remember: A maximum of 3 people per group.

Notes: (i) Every answer has to be justified unless otherwise stated. Show your work!
(ii) You may use any theorem/property/fact proven in class or in the textbook without re-proving it.
(iii) **Compute all running times in this homework in the bit model.**

1. The following is an exercise in modular exponentiation.

   (a) Let $a_i = 5^{2^i} \bmod 18$. Using the following iterative procedure, compute a_i for $i = 1$ to 4:

   $a_0 = 5$
   for $i = 1$ to 4 do
   $\quad a_i = a_{i-1} * a_{i-1} \bmod 18$

   List the values of $a_0, a_1, \ldots, a_4$.
   Solution: $a_0 = 5,\ a_1 = 7,\ a_2 = 13,\ a_3 = 7,\ a_4 = 13$

   (b) The number 23, written in binary is 10111, which is 16 + 4 + 2 + 1.
   Using that fact, express $5^{23} \bmod 18$ as a function of $a_0, a_1, \ldots, a_4$. (Your expression should be a mod 18 expression.) Then calculate $5^{23} \bmod 18$ using the values for a_i that you computed in the previous problem.
   Solution:

$$\begin{aligned} 5^{23} \bmod 18 &= 5^{16+4+2+1} \bmod 18 \\ &= 5^{16} \cdot 5^4 \cdot 5^2 \cdot 5^1 \bmod 18 \\ &= 5^{2^4} \cdot 5^{2^2} \cdot 5^{2^1} \cdot 5^{2^0} \bmod 18 \\ &= a_4 \cdot a_2 \cdot a_1 \cdot a_0 \bmod 18 \\ &= 13 \cdot 13 \cdot 7 \cdot 5 \bmod 18 \\ 13 \cdot 13 &\equiv 7 \bmod 18 \\ 7 \cdot 7 &\equiv 13 \bmod 18 \\ 13 \cdot 5 &\equiv 11 \bmod 18 \end{aligned}$$

   Thus $5^{23} \equiv 11 \bmod 18$

2. When doing ordinary division, if we calculate a/b and get the value x, then x is the solution to the equation $bx = a$.

   Suppose that $GCD(b, c) = 1$. Then the division problem $a/b \bmod c$ has a solution x. The solution x satisfies $bx \equiv a \pmod{c}$ (bx does not have to be equal to a, but they must be equivalent, modulo c).

   Consider the division problem $6/5 \bmod 49$. Verify that it has a solution x and calculate x. Show your work.

   Hint: Check your answer. If your answer x is correct, then it should satisfy $5x \equiv 6 \bmod 49$.

   Solution: $6/5 \bmod 49$ has a solution because $GCD(5, 49) = 1$. The modular inverse of 5 mod 49 can be computed using `extended-euclid(49, 5)` or through back-substitution.

```
extended-euclid(49, 5):
d = 49x + 5y

Forward:
49 = 5(9) + 4
5 = 4(1) + 1
4 = 1(4) + 0

Backwards:
1 = 5 + 4(-1)
1 = 5 + (49 + 5(-9))(-1)
1 = 49(-1) + 5(10)

We get d = 1, x = -1, y = 10
Thus, 10 is the modular inverse of 5 mod 49
```

   Using the modular inverse: $6/5 \bmod 49 = 6 \cdot 10 \bmod 49 = 11$
   We can verify this because $5 \cdot 11 \bmod 49 \equiv 6 \bmod 49$

3. As discussed in class and in the textbook, if N is a composite number (that is not a Carmichael number), then *at least* half the integers a between 1 and $N - 1$ satisfy $a^{N-1} \bmod N \neq 1$.

   For $N = 6$, what fraction of the numbers a between 1 and $N-1$ satisfy $a^{N-1} \bmod N \neq 1$? List all of them, and specify the fraction.

   Solution:

   $1^5 \bmod 6 = 1$
   $2^5 \bmod 6 = 2$
   $3^5 \bmod 6 = 3$

$4^5 \bmod 6 = 4$
$5^5 \bmod 6 = 5$
4/5 satisfy the property

4. If it takes n bits to represent the number N in binary, how many bits does it take to represent the number $\sqrt{N}$? Choose the correct expression from the list below, AND justify your answer:

   (a) $\Theta(n)$
   (b) $\Theta(\sqrt{n})$
   (c) $\Theta(\log(n))$

   Solution: $\Theta(n)$. The bit sizes will approximately be $n/2$. Recall that the number of bits in N is $\lceil \log_2(N) \rceil$. The number of bits in $\sqrt{N}$ is then $\lceil \log_2(\sqrt{N}) \rceil = \lceil \log_2(N^{\frac{1}{2}}) \rceil \approx \frac{1}{2} \log_2(N)$. This means the bit size of $\sqrt{N}$ will be approximately $\frac{n}{2} = \Theta(n)$

5. Describe an algorithm that takes as input an n-bit positive integer a, and determines whether or not it is the product of two consecutive integers, x and $(x+1)$. Analyze the running time of your algorithm in the bit model. It should run in time polynomial in n.

   Solution: We are trying to determine if an integer x such that $x(x+1) = a$ exists. We will solve the problem by looking for such a value x in the set $1, \ldots, a$. We will use binary search, since for a given value x, it is easy to determine whether x is too low, too high, or just right. The following is psuedocode of a modification on binary search. We use the value of `i = mid * (mid + 1)` to determine whether we are too low, too high, or just right:

```
function is_product_of_consecutive(a)
-----------------------------------------------------------------------
Input: an n-bit number a ≥ 0
Output: a truth value determining whether a is the product of two
consecutive integers
-----------------------------------------------------------------------
low = 1
high = a
while low ≤ high:
    mid = ⌊(low + high)/2⌋
    i = mid * (mid + 1)
    if i = a:
        return true
    if i > a:
        high = mid - 1
    else:
```

```
        low = mid + 1
return false
```

This is a modification of binary seach. The while loop will take $O(n)$ in the worst case because the values are getting cut in half each iteration. Inside the loop the most expensive operation are the multiplications which will take $O(n^2)$, so the total running time is $O(n^3)$.

6. As described in the textbook, the probability that a uniformly random n-bit number is prime is approximately $1.44/n$. For this problem, assume that it is is exactly $1.44/n$.

   Note: Give exact numerical answers to the questions below (with a few significant digits). No big-Oh's etc.

   (a) Suppose we generate, uniformly, a random 100-bit number. What is its probability of being prime?

   Note: Just to be clear, a random 100-bit number is generated by executing the following loop:

   for i=1 to 100

   generate a random bit (either 0 or 1, each with equal probability) and assign it to b_i

   Output the binary number $b_1 \dots b_{100}$

   Solution: There are 100 bits so the probability of producing a prime is 1.44/100 = .0144

   (b) If we repeatedly generate uniformly random 100-bit numbers, until we get a prime, how many numbers do we expect to generate?

   Solution: The probability of producing a prime is .0144 and we want 1 of them so the expected number of times we need to generate it is $1/.0144 = 69.44 \approx 70$ times

7. In an RSA cryptosystem, choose $p = 13$ and $q = 17$.

   (a) Consider the following potential values for e in this cryptosystem: 6, 7, 3, 4, 9. Which one of them is a legal value for e?
   Solution: 7 is the only value relatively prime to $(p-1)(q-1) = 192$

   (b) Using that legal value of e, what is the corresponding d?
   Solution: $d = 55$ The inverse of 7 mod 192 is 55. It can be found through back-substitution or `extended-euclid(192, 7)`

   (c) Using your values for d and e, encrypt the message 21.
   Solution: $y = 21^7 \bmod 221 = 200$

   (d) Using your values for d and e, decrypt the message 3.
   Solution: $x = 3^{55} \bmod 221 = 198$

8. Consider the following, silly randomized method for searching an *unsorted* array $A[1 \ldots n]$ of integers.

    function SillyRandomSearch(A,x)

    Repeat forever:

    Uniformly and randomly generate an integer i such that $1 \leq i \leq n$
    if $A[i] = x$: Return i

    (a) If x is contained in exactly one position of array A, how many times do we expect SillyRandomSearch(A, x) to execute the repeat loop. Give your answer as a function of n. Do *not* use asymptotic notation in your answer (that is, give an exact bound, not an asymptotic bound).

    Solution: The probability of picking x is $1/n$ so the expected number of runs is n

    (b) Repeat the previous question, but this time assume that x is contained in exactly two positions in the array.

    Solution: The probability of picking x is $2/n$ so the expected number of runs is $n/2$

    (c) Now suppose we replace the Repeat loop with the following:

    Repeat forever:

    Uniformly and randomly generate an integer i such that $1 \leq i \leq n$
    Uniformly and randomly generate an integer j such that $1 \leq j \leq n$
    if $A[i] = x$ and $A[j] = x$: Return (i, j)

    If x is contained in exactly two positions in array A, how many times do we expect this new repeat loop to be executed? Again, give your answer as a function of n and do *not* use asymptotic notation.

    Solution: The probability of picking x is $2/n$ and we must pick it twice independently. The probability of picking it twice independently is $4/n^2$ so the expected number of runs is $n^2/4$

9. In the RSA cryptosystem, Eve, the eavesdropper, can easily find out Bob's public key (N, e) because Bob has made it public. Suppose Alice sends a message x to Bob, after encrypting it using Bob's public key. Recall that x is a number in $\{0, \ldots, N-1\}$. Let y be the encrypted message that she sends. Eve sees y and would like to decrypt it. But Eve doesn't know how to do it, because RSA is designed to make it difficult for her.

    Suppose Eve is lucky enough to find out some additional information I. This information may or may not make it easier for Eve to decrypt y. For each of the following pieces I of additional information, indicate whether it seems to make it easier for Eve, or if it doesn't help. If your answer is "easier," then describe how Eve can decrypt y

in time polynomial in n, using N, e and I. If your answer is "it doesn't help," then describe how Eve could compute I herself in polynomial time, just using Bob's public key (N, e).

For example, if I is the decryption key d, then it makes things much easier for Eve, because Eve can compute $y^d \bmod N$ to find the value of x. On the other hand, if I is the single number $N-1$, then it doesn't help, because she could compute $N-1$ herself by subtracting 1 from N.

(a) a number that is equal to $p-1$
Solution: Given $p-1$ we can compute p by adding 1. Eve can then get $q = N/p$. Using the factors p and q, Eve can compute $(p-1)(q-1)$ and then compute the decryption key since e is public.

(b) a number r such that $0 < r < (p-1)(q-1)$ and $re-1$ is divisible by $(p-1)(q-1)$
Solution: Given r such that $re-1$ is divisible by $(p-1)(q-1)$, this means that $re \equiv 1 \bmod (p-1)(q-1)$. Thiis means that r is the inverse of e and is the decryption key.

(c) a number that is equal to $(p-1)(q-1)$
Solution: Given $(p-1)(q-1)$, Eve can compute the modular inverse of e to get the decryption key

(d) a number that is equal to $(p-1)^{p-1} \bmod p$
Solution: This value is equal to 1 and does not help Eve.

10. Suppose you have a randomized algorithm $\mathcal{A}$ for testing whether a number x is prime, with the following properties:

- If x is prime, $\mathcal{A}$ will always return "prime"
- If x is not prime, $\mathcal{A}$ returns "not prime" with probability 4/5 and "prime" with probability 1/5

(a) If x is not prime, what is the probability that $\mathcal{A}$ will return the wrong answer?
Solution: 1/5

(b) Using $\mathcal{A}$ as a subroutine, describe an algorithm $\mathcal{B}$ with the following properties:
- If x is prime, $\mathcal{B}$ will always return the correct answer.
- If x is not prime, $\mathcal{B}$ will return the correct answer with probability 124/125.

Solution: A wrong answer occurs only when x is not prime and $\mathcal{A}$(x) returns "prime". The probability of this happening is 1/5 so we can chain 3 calls to $\mathcal{A}$(x) to make the probability of producing a wrong answer to be 1/125.

```
for i = 1..3:
    if A(x) = ''not prime'':
        return ''not prime''
return ''prime''
```

This works because the function will only return "prime" if all 3 calls to $\mathcal{A}(x)$ returns "prime". When x is prime, the probability of this happening is 1, but when x is not prime, the probability of this happening is $(1/5)^3 = 1/125$