Homework 3

Name:	Rohan Karamel
NetID:	rak218
Course:	Algorithms 2
Instructor:	Professor Szegedy
Date:	February 26, 2024

Problem 1.27.

Solution. The value of d should be the multiplicative inverse of 3 (mod 352). Using Euclid's algorithm first:

$$352 = 3 \cdot 117 + 1$$

$$3 = 1 \cdot 3 + 0$$

Extended Euclid's is thus:

$$1 = 352 + 3 \cdot (-117)$$

$$-117 \equiv 235 \bmod 352$$

Therefore, $d = 235$. Thus, we only need to calculate $41^3 \pmod{391}$ to find the encryption:

$$\begin{aligned} 41^3 &\equiv 68921 \pmod{391} \\ &\equiv 68921 - 176 \times 391 \\ &\equiv 68921 - 68616 \\ &\equiv 105 \pmod{391} \end{aligned}$$

Therefore, $41^3 \pmod{391} = 105$.

Problem 1.28.

Solution. Given $p = 7$ and $q = 11$, calculate:

$$\begin{aligned} n &= pq \\ &= 7 \times 11 \\ &= 77 \end{aligned}$$

Calculate $\phi(n)$:

$$\begin{aligned} \phi(n) &= (p-1)(q-1) \\ &= (7-1)(11-1) \\ &= 6 \times 10 \\ &= 60 \end{aligned}$$

Now, choose a public exponent e. Let's say $e = 17$ (common choice). Calculate the private exponent d such that $ed \equiv 1 \pmod{\phi(n)}$:

$$\begin{aligned} de &\equiv 1 \pmod{60} \\ 17d &\equiv 1 \pmod{60} \end{aligned}$$

Solving for d:

$$d \equiv 6 \pmod{60}$$

So, $d = 6$. We pick $e = 7$ because it is the smallest number that is relatively prime to 60.

Problem 1.29.

a

Solution. This function is universal. By calculation, the appropriate number of bits is $2log_2m$

b

Solution. H is not universal as m is no longer prime so the number of bits is not applicable to solve for.

c

Solution. This function is universal, thus, the number of bits is

$$\log m - 1^m = m \log m - 1$$

Problem 1.31.

Solution. If N is an n-bit number, $N!$ is approximately $\Theta(n^3)$ bits long. We reach this approximation by recognizing that $N! < N^N$ for all positive integers. So, by calculating the number of bits of N^N, we can find a decent upper bound. The number of bits is as follows:

$$\log_2\left(N^N\right) = N\log_2(N) = N \cdot n$$

From the previous homework, we know we can also find a decent lower bound by finding the number of bits in $\left(\frac{N}{2}\right)^{\frac{N}{2}}$. The number of bits is as follows:

$$\log_2\left(\frac{N}{2}^{\frac{N}{2}}\right) = \frac{N}{2}\log_2(\frac{N}{2}) = \frac{N}{2} \cdot (n-1)$$

Thus, the upper bound is $O(Nn)$ and the lower bound is $\Omega(Nn)$. Therefore, because these bounds are approximately equal, then the number of bits of $N!$ is $\Theta(Nn)$.

To compute $N!$, we proceed by the following algorithm:

1 factorial (int n):

2 if n is 0: return 0

3 else return $n \cdot$ factorial$(n-1)$

This algorithm runs N times, each of which performing a multiplication of n bits and $n-1$ bits long. Therefore, the running time is $\mathrm{O}(N \cdot n^2)$.

If we are using Karatsuba's algorithm, then our algorithm improves to $\mathrm{O}(N \cdot n^{1.585})$

Problem 2.4.

Solution. a Algorithm A

> *Solution.* The runtime of this algorithm is $T(n) = 5T(n/2) + O(n)$. By Master's Theorem, we have that this algorithm runs in $\Theta(n^{\log_2 5})$.

b Algorithm B

> *Solution.* The runtime of this algorithm is $T(n) = 2T(n-1)+O(1)$. By Master's Theorem, we have that this algorithm runs in $\Theta(2^n)$.

c Algorithm C

> *Solution.* The runtime of this algorithm is $T(n) = 9T(n/3) + O(n^2)$. By Master's Theorem, we have that this algorithm runs in $\Theta(n^2 \log_3 n)$

d Conclusions: We choose Algorithm C because the exponent is smaller and we can completely ignore the log.

Problem 2.5.

Solution. a $T(n) = 2T(n/3) + 1$

> *Solution.* Using Master's Theorem $e = \log_3 2, d = 0$. Because $e > d$, we have $\Theta(n^{\log_3 2})$.

b $T(n) = 5T(n/4) + n$

> *Solution.* $e = \log_4 5, d = 1$. $e > d$, so we have $\Theta(n^{\log_4 5})$

c $T(n) = 7T(n/7) + n$

> *Solution.* $e = \log_7 7 = 1, d = 1$. $e = d$, so we have $\Theta(n \log_7 n)$.

d $T(n) = 9T(n/3) + n^2$

> *Solution.* $e = \log_3 9 = 2, d = 2$. Because $e = d$, we have $\Theta(n^2 \log n)$.

e $T(n) = 8T(n/2) + n^3$

> *Solution.* $e = \log_2 8 = 3, d = 3$. Because $e = d$, we have $\Theta(n^3 \log n)$.

f $T(n) = 49T(n/25) + n^{3/2}\log(n)$

> *Solution.* $e = \log_{25} 49, d = 3/2$. Because $d > e$, we have $\Theta(n^{1.5})$.

g $T(n) = T(n-1) + 2$

> *Solution.* $\Theta(n)$

h $T(n) = T(n-1) + n^c$, where $c \geq 1$ is a constant

> *Solution.* $\Theta(n^{c+1})$

i $T(n) = T(n-1) + c^n$, where $c > 1$ is some constant

> *Solution.* $\Theta(n \cdot c^n)$

j $T(n) = 2T(n-1) + 1$

> *Solution.* $\Theta(2^n)$

k $T(n) = T(\sqrt{n}) + 1$

> *Solution.* $\Theta(n) = \log_2(\log_2 n)$

Problem 2.8.

Solution. The appropriate value of ω_4 is i as $\omega_4 = e^{\frac{\pi i}{2}}$. We apply the FFT Algorithm on $(1, 0, 0, 0)$:

$$\begin{aligned}
X[0] &= 1 + 0 + 0 + 0 = 1 \\
X[1] &= 1 + \omega_4 \cdot 0 + \omega_4^2 \cdot 0 + \omega_4^3 \cdot 0 = 1 \\
X[2] &= 1 + \omega_4 \cdot 0 + \omega_4^2 \cdot 0 + \omega_4^3 \cdot 0 = 1 \\
X[3] &= 1 + \omega_4 \cdot 0 + \omega_4^2 \cdot 0 + \omega_4^3 \cdot 0 = 1
\end{aligned}$$

The result is: $(1, 1, 1, 1)$

We apply the FFT Algorithm on $(1, 0, 1, -1)$:

$$\begin{aligned}
X[0] &= 1 + 0 + 1 + (-1) = 1 \\
X[1] &= 1 + \omega_4 \cdot 0 - \omega_4^2 \cdot 1 + \omega_4^3 \cdot -1 = 0 - \omega_4 = -\omega_4 \\
X[2] &= 1 + \omega_4 \cdot 0 + \omega_4^2 \cdot 1 + \omega_4^3 \cdot -1 = 1 + \omega_4^2 = 2 + 1 = 3 \\
X[3] &= 1 + \omega_4 \cdot 0 - \omega_4^2 \cdot 1 - \omega_4^3 \cdot -1 = 0 + \omega_4 = \omega_4
\end{aligned}$$

The results is: $(1, -i, 3, i)$