Matheus Peixoto Ribeiro Vieira - 221 4104

a) Faça a divisão de 44 por 5 usando o algoritmo Divide(x,y) do slide 5 da aula sobre Teste de primalidade.

44 = 101100 5 = 101

divide(44,5) = (8,4)

divide(22,5) = (4,2)

divide(11,5) = (2,1)

divide(5,5) (1,0)

divide(2,5) = (0,2)

divide(1,5) = (0,1)

divide(0,5) = (0,0)

divide(1,5)	divide(2,5)	divide(5,5)	divide(11,5)	divide(22,5):
q=0 r=0	q=0 r=1	q=0 r=2	q=1 r=0	q=2 r=1
r=1	q=0 r=2	q=0 r=4	q=2 r=0	q=4 r=2
return(0,1)	return(0,2)	q=0 r=5	q=2 r=1	return(4,2)
		q=1 r=0	return(2,1)	
		return(1,0)		

divide(44,5)

q=4 r=2

q=8 r=4

return(8,4)

b) Faça a análise de complexidade da função modexp(x, y, N) do slide 7 no pior caso.

function mode exp(x, y, N)

if y = 0: return 1 — O(n): verifica se todos os bits são zero

z = modeexp(x, [y/2], N) → realização do shift que possui custo O(n), e, como estamos analisando o pior caso, isso implica que o 0 bit mais significativo está como 1. Logo, até o número se tornar zero serão feitas n chamadas recursivas.

if y is even: O(1)

return z^2 mod N — $O(n^2 + n^2) = O(n^2)$

↳ divisão = $O(n^2)$

↳ multiplicação = $O(n^2)$

else:

return x · z · z mod N — $O(n^2 + n^2 + n^2) = O(n^2)$

$n \cdot O(n^2) = O(n^3)$

c) Faça a análise de complexidade da função primality2(N) no pior caso (slide 15).

1 - function primality2(N)
2 - pick positive integers $a_1, a_2, \ldots, a_k < N$ at random
3 - if $a_i^{N-1} \equiv 1 \pmod{N}$ for all $i = 1, 2, \ldots, k$:
4 - return yes
5 - else:
6 - return no

2 - Considerando que a escolha de um número aleatório seja $O(1)$ e são k valores, logo $k \cdot O(1) = O(k) = O(1)$
3 - É realizado a chamada da função modexp que possui complexidade $O(n^3)$ no pior caso, todavia ela será chamada k vezes, gerando um custo de $k \cdot O(n^3)$, porém considerando k como constante teremos o custo de $O(n^3)$

Assim, a complexidade da função primality 2 é $O(n^3)$