Teste de primalidade

Carlos Eduardo Gonzaga Romaniello de Souza - 19.1.4003

18 de abril de 2022

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

- 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)

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

Para a complexidade local temos as seguintes análises:

- $O(n)$ para conferir se y vale 0
- $O(n)$ para o pior caso nas chamadas recursivas da função modexp
- $O(1)$ para verificar se o número é par
- $O(n^2)$ para as multiplicações
- $O(n^2)$ para a divisão inteira

Somando tudo temos que a complexidade local é $O(n^2)$, porém a função pode ser chamada n vezes no pior caso, com isso temos que a complexidade será $n \times O(n^2) = O(n^3)$. A imagem 1 mostra as complexidades citadas a cima.

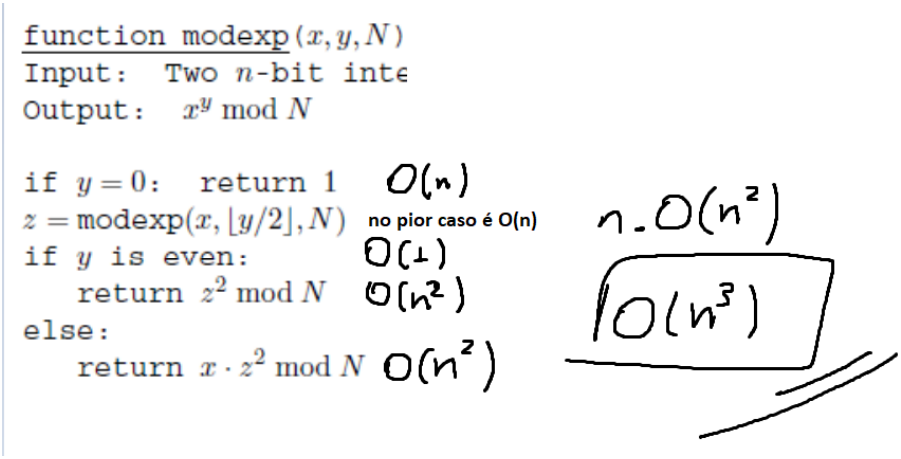

Figure 1: Resolução letra B

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

Para a complexidade local temos as seguintes análises:

- $O(n^3)$ para a exponenciação
- $O(n^2)$ para a divisão inteira
- $O(n)$ para a comparação
- Há um for que será executado k vezes

Com isso temos que a exponenciação, que é a operação com a maior complexidade, será executada k vezes, portanto temos que $k \times O(n^3) = O(n^3)$. A imagem 2 mostra todas as complexidades citadas a cima.

```
function primality2(N)
Input:  Positive integer N
Output:  yes/no

Pick positive integers a_1, a_2, ..., a_k < N at random
if a_i^{N-1} ≡ 1 (mod N) for all i = 1,2,...,k:
   return yes
else:
   return no
```

$k * O(n^3)$

a_i^{N-1} $O(n^3)$

(mod N) $O(n^2)$ $O(n^3)$

if $a_i^{N-1} \equiv 1 \pmod{N}$ $O(n)$

for all $i = 1,2,\ldots,k$: $O(k)$

Figure 2: Resolução letra C