Complexidade Computacional

Parte 01 - Algoritmos Iterativos



1. E-mail \*

[kawhan.laurindo@dcx.ufpb.br](mailto:kawhan.laurindo@dcx.ufpb.br) | [vinicius.teixeira@dcx.ufpb.br](mailto:vinicius.teixeira@dcx.ufpb.br) |[sidney.jose@dcx.ufpb.br](mailto:sidney.jose@dcx.ufpb.br)

1. As afirmações abaixo tratam do assunto de complexidade. Marque a(s) alternativa(s) verdadeiras(s).

*Marque todas que se aplicam.*

(V) Instruções de atribuição, soma, subtração, multiplicação e divisão custam O(1)

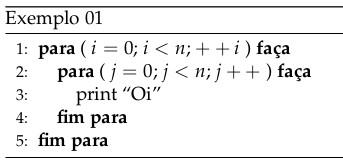
(V) Laços são operações compostas e custam o número de vezes que é executado

(F) Chamadas a funções custam O(1)

(V) A complexidade de um algoritmo ou método é medida pelo custo de sua instrução mais cara

(F) Quanto maior a complexidade de um algoritmo, mais rápido ele é.

1. Qual a complexidade do Algoritmos abaixo?



*Marcar apenas uma oval.*

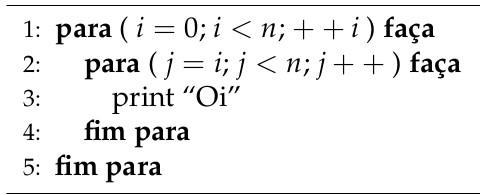
( ) O(n)

(✅) O(n²)

( ) O(nlg(n))

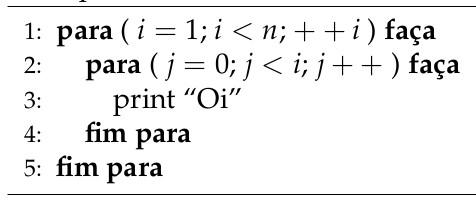
( ) O(n³)

4. Qual a complexidade do Algoritmo abaixo? Justifique sua resposta.

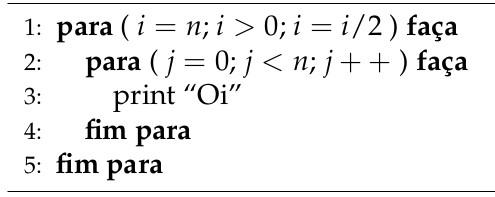


**A complexidade desse algoritmo é O(n²) pois pela expressão 1(Linha 1) vai de “ i ” até n ou seja “n” vezes e o de baixo(Linha 2) vai colocar o j sendo igual a “i” indo até n também, fazendo pelo cálculo O(n) \* O(n) = O(n²), representado também a soma dos números naturais.**

5. Qual a complexidade do Algoritmos abaixo? Justifique sua resposta.

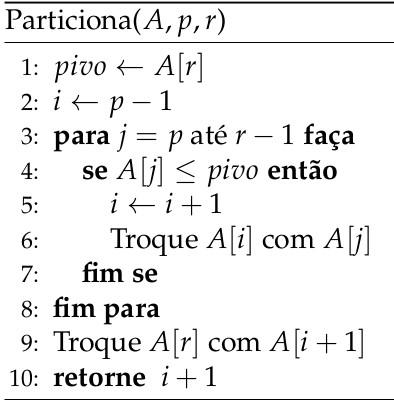


**A complexidade deste algoritmo é O(n²), na expressão da linha 1 ele vai de i até n vezes, porque começa em 1 e vai até n, a expressão da linha 2 vai até i começando em 0, i representa n vezes de execução e o “j” representa “i” vezes execuções logo os 2 valem n execuções, ou seja, O(n) \* O(n) = O(n²), representando também a soma dos números naturais.**

6. Qual a complexidade do algoritmo abaixo? Justifique sua resposta.

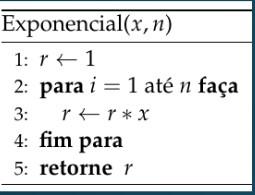
**A complexidade desse algoritmo é O(nlog2n) a instrução da linha 1 vai ser verdadeira até que o número de i não possa ser dividido por 2, ou seja log2n, sendo n na atribuição “i=n” o valor que ele começa, a linha 2 vai de j com começo 0 até n, ou seja n vezes, fazendo o cálculo das 2 instruções fica O(log2n) \* O(n) = O(nlog2n).**

7. Qual a complexidade do algoritmo abaixo? Justifique sua resposta.



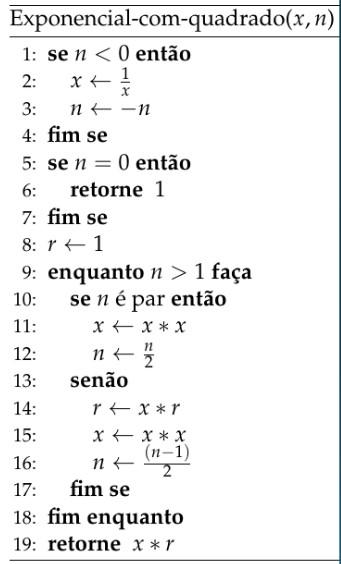
**A complexidade desse algoritmo é O(n), pois o tempo de execução aumenta linearmente de acordo com o tamanho da entrada, na linha podemos ver que o laço vai de p até r-1, sendo p o início e r o final, tendo isso podemos perceber que o tempo de execução do algoritmo vai depender do valor de r**.

8. Qual a complexidade do algoritmo abaixo? Justifique sua resposta.



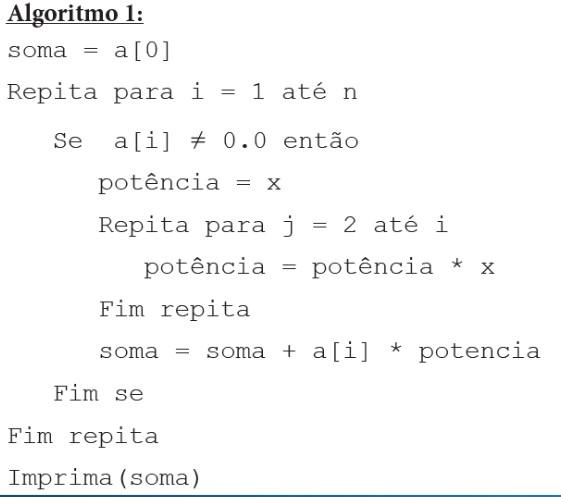
**A complexidade desse algoritmo é O(n) sendo n o valor do expoente e x a base do exponencial, a execução da linha 2 que é a maior começa em 1 que é o valor de i e vai até n que é o valor do expoente ou seja essa execução vai ter n vezes, sendo assim a complexidade desse algoritmo é O(n)**

9. Qual a complexidade do algoritmo abaixo? Justifique sua resposta.

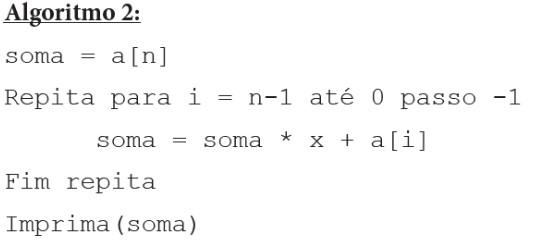


**A complexidade desse algoritmo é O(log2n) a instrução da linha 9 vai ser verdadeira até que o número de n não possa ser dividido por 2, ou seja, log2n,, a linha 9 vai vai rodar enquanto n for maior que 1, ou seja n-1 vezes, fazendo o cálculo das instruções fica O(log2n)**

10. Qual a complexidade do algoritmo abaixo? Justifique sua resposta. (Observe que "repita" é apenas outra forma de laço)

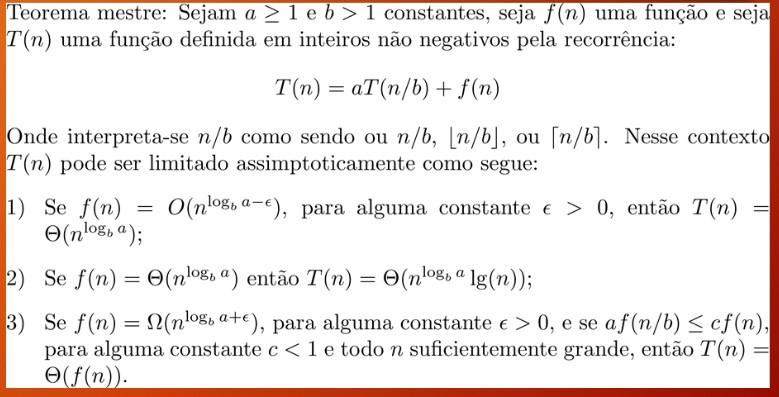


**A complexidade deste algoritmo é O(n²) , seguindo o seguinte raciocínio na linha 2 o valor vai até n vezes, começa em 1 com valor de i = 1 e vai até n, na linha 5 sendo o valor de i determinado por n, a execução dele também vai ser n, sendo assim calculando a complexidade fica O(n) \* O(n) = O(n2)**

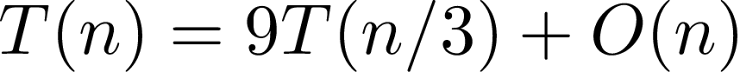
1. Qual a complexidade do Algoritmos abaixo? Justifique sua resposta. 

**A complexidade desse algoritmo é O(n) pois tem um laço na linha 2 que vai de n-1 ate 0, devido a isso o tempo do algoritmo vai depender do tamanho de n.**

Teorema Mestre

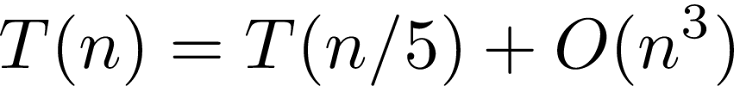


12. Forneça as Complexidades para a equação Recorrente. Inclua em sua resposta as contas, o caso do teorema mestre a aplicar



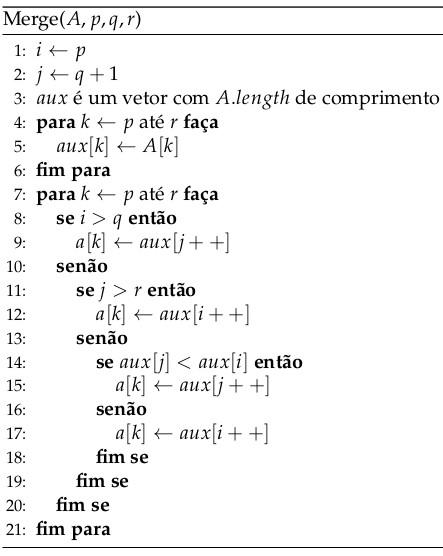
**Consultando o teorema temos T(n) = aT(n/b) + f(n) sendo assim a = 9, b=3 e f(n) = n, agora precisamos resolver logba = log39 = 2, agora precisamos resolver nlog39 = n², coo-relacionando agora f(n) com nlog39 dessa forma vamos ver que n < n² ou seja f(n) < nlog39, dessa forma ele se encaixa na primeira das 3 equações sendo então representada como ( nlogba) = n², ou seja a complexidade para a equação Recorrente é (n²)**

13. Forneça as Complexidades para a equação Recorrente. Inclua em sua resposta as contas, o caso do teorema mestre a aplicar

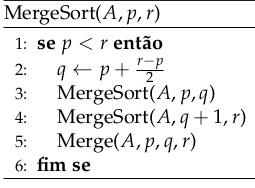


**Consultando o teorema temos T(n) = aT(n/b) + f(n) sendo assim a = 1, b = 5 e f(n) = n3, agora precisamos resolver logba = log51 = 0, depois resolvendo nlogba = n0 = 1, agora comparando f(n) = n³ com nlogba = 1, f(n³) > nlogba, sendo assim se encaixa no terceiro caso ( f(n)) = (n³), ou seja complexidade para a equação Recorrente é (n³).**

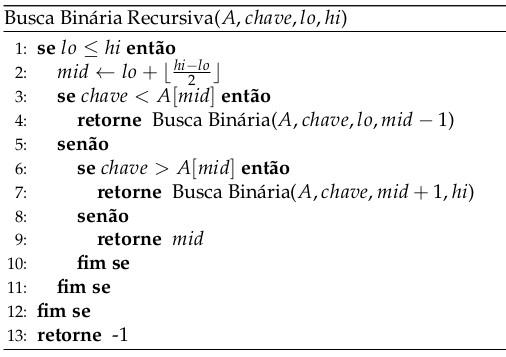
Merge



Mergesort



Busca Binária



14. Usando o teorema Mestre, indique a complexidade do Algoritmo Mergesort

**Seguindo o teorema mestre aplicando na equação do merge sort, temos de calcular as chamadas não recursivas e recursivas para aplicar na fórmula, primeiro calculando a não recursiva Merge (A, p,q,r) temos complexidade O(n), pois temos 2 laços que vão até n separados, ou seja, a complexidade é O(n) linear, partindo para as chamadas recursivas temos 2 obrigatórias, sendo uma que vai do início até a metade e outra que vai da metade até o fim, ou seja, as 2 possuem a complexidade de O(n/2), aplicando o teorema mestre temos T(n) = T(n/2) + T(n/2) + O(n) = T(n) = 2T(n/2) + O(n), comparando agora com a equação T(n) = aT(n/b) + f(n), sendo assim, a = 2, b = 2 e f(n) = n, precisamos agora calcular logba = log22 = 1 e passando para calcular nlogba = n1 = n, comparando agora f(n) e nlogba temos, n = n , ou seja, nlogba = f(n) se encaixando no caso 2 ( nlogba \* lg(n)) = (n \* lgn), ou seja, complexidade para a equação Recorrente é (nlgn)**

15. Usando o teorema Mestre, indique a complexidade do Algoritmo de Busca Binária Recursiva

**Seguindo o teorema mestre aplicando na equação do algoritmo de busca binária, temos de calcular as chamadas não recursivas e recursivas para aplicar na fórmula, primeiro calculando as não recursivas temos complexidade O(1), pois só temos operações simples ou seja o número de elementos não vai interferir no tempo de execução do algoritmo, partindo para as chamadas recursivas temos 1 obrigatória, se a chave for maior que o número vamos analisar a parte de cima do vetor, se a chave for menor que número vamos analisar a parte de baixo do vetor, aplicando o teorema mestre temos T(n) = T(n/2) + f(n) = T(n) = 1T(n/2) + O(1), comparando agora com a equação T(n) = aT(n/b) + f(n), sendo assim, a = 1, b = 2 e f(n) = 1, precisamos agora calcular logba = log21 = 0 e passando para calcular nlogba = n0 = 1, comparando agora f(n) e nlogba temos 1=1, ou seja, nlogba = f(n) se encaixando no caso 2 ( nlogba \* lg(n)), substituindo temos a complexidade (lg 1).**