**8ECM306 – TÓPICOS AVANÇADOS EM ESTRUTURA DE DADOS ENGENHARIA DA COMPUTAÇÃO – 3ª SÉRIE – 2025 – Prof. Calvetti**

**Exercícios propostos da Aula 5**

**Pedro Wilian Palumbo Bevilacqua – RA: 23.01307-9**

**Efetuar a análise assintótica das funções**

**Resultados dos Exercícios**

**Exercício 1 –**

Exercício implementado na pasta zip, com nome “Ex1.java”.

**Exercício 2 –**

Para o pior caso, o insertion sort tem ordem de complexidade do tipo ***O(n²)***.

**Exercício 3 –**

A linha 1 será executada f(n) = n² vezes considerando n inteiro e maior ou igual a zero.

**Exercício 4 –**

A linha 1 será executada f(n) = (n-2)² vezes considerando n inteiro e maior ou igual a 2.

**Exercício 5 –**

A linha 1 será executada f(n) = ((n-4)/2) + 1 simplicando fica f(n) = (n/2) – 1 considerando n inteiro e maior ou igual a 2.

**Exercício 6 –**

A linha 1 será executada f(n) = vezes considerando n inteiro e maior ou igual a 0.

**Exercício 7 –**

A linha 1 será executada em forma de uma progressão aritmética (PA) portanto generalizando usando a formula de PA para este caso f(n) = (n(n+1))/2 também podendo ser escrita como f(n) = para n inteiro e maior que 0.

**Exercício 8 –**

Algoritmo de ordenação por inserção: f1(n) = 8n²

Algoritmo de ordenação por intercalação: f2(n) = 64 n ln (n)

Analiticamente podemos encontrar o ponto de encontro dessas 2 funções e após isso avaliar como cada uma delas se comporta um instante antes ou depois para avaliar qual algoritmo age melhor para quais interações:

f1(n) = f2(n)

8n² = 64 n ln(n)

n = 8 ln(n)

Portanto podemos determinar aproximadamente que n1 = 1,155 e n2 = 26,1.

Analisando os pontos antecedentes a esse como n=1 e n=26 e os pontos posteriores como n=2 e n=27 podemos determinar que o algoritmo de ordenação por inserção é mais eficiente que o outro algoritmo entre.

**Exercício 9 –**

fA(n) = 100n²

fB(n) =

Analiticamente podemos encontrar o ponto de encontro dessas 2 funções e após isso avaliar como cada uma delas se comporta um instante antes ou depois para avaliar qual algoritmo age melhor para quais interações:

fA(n)=fB(n)

100n² =

Portanto podemos determinar aproximadamente n1 = 0,10 e n2=14,32.

Analisando os pontos antecedentes a esse como n=0 e n=14 e os pontos posteriores como n=2 e n=15 podemos determinar que o algoritmo 100n² é mais eficiente para o menor valor em n = 15.

**Exercício 10 –**

fA(n) = 128n²

fB(n) = 4n³

Analiticamente podemos encontrar o ponto de encontro dessas 2 funções e após isso avaliar como cada uma delas se comporta um instante antes ou depois para avaliar qual algoritmo age melhor para quais interações:

fA(n)=fB(n)

128n² = 4n³

Portanto podemos determinar aproximadamente n = 32.

Analisando o ponto antecedente a esse como n=31 e o ponto posterior como n=33 podemos determinar que o algoritmo 4n³ é mais eficiente tendo o maior valor de n como n = 31.

**Exercício 11 –**

Computador C1: computa op/s

Computador C2: computa op/s

Algoritmo A: fA(n) = 5n²

Algoritmo B: fB(n) = 40 n log n

Pelo exercício n = , portanto:

Operações do algoritmo A: fA() = 5\*()² = 5 \* operações.

Operações do algoritmo B: fB() = 40\* \* log = 3,2 \* operações.

Computador C1:

Algoritmo A: tA = total de operações / op/s -> ta= (5 \* )/() = 15,8 anos

Algoritmo B: tB = total de operações / op/s -> tb = (3,2 \*)/() = 5,3 minutos

Computador C2:

Algoritmo A: tA = total de operações / op/s -> ta= (5 \* )/() = 57,9 dias

Algoritmo B: tB = total de operações / op/s -> tb = (3,2 \*)/() = 3,2 segundos

**Exercício 12 –**

Computador 1: f1(n) = , para tamanho 25

Computador 2: f2(n) = 100\*f1(n) = -> f2(n) = , para tamanho x

Computador 1 para tamanho 25: f1(25) = tempo

Computador 2 para tamanho x: f2(x) = tempo

t1 = t2

=

Portanto, x = 31,64

O tamanho do problema x para que o computador 1 tenha o mesmo tempo que o computador 2 é x=31.