• Resolva as equações abaixo:

a) 210 = 1024

b) lg(1024) = 10

c) lg(17) = 4.0874628412503

d) lg(17) (teto) = 5

e) lg(17) (piso) = 4

• Plote um gráfico com todas as funções abaixo:

a) f(n) = n3

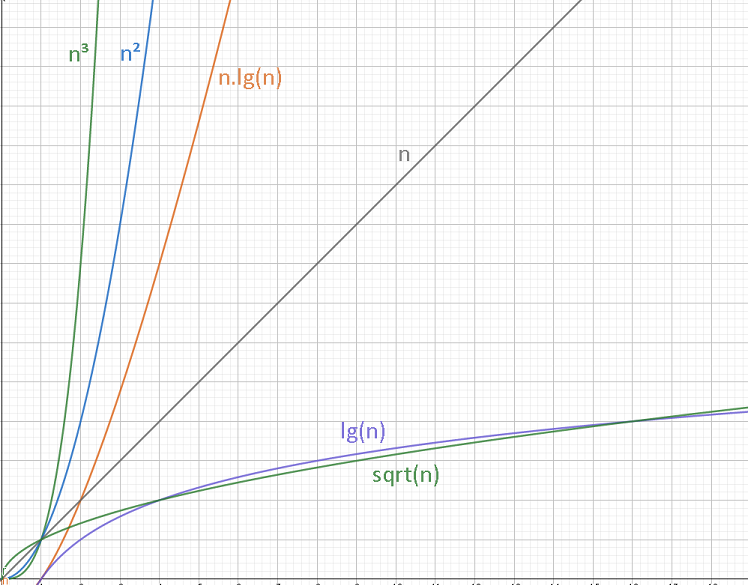
b) f(n) = n2

c) f(n) = n.lg(n)

d) f(n) = n

e) f(n) = sqrt(n)

f) f(n) = lg(n)



• Calcule o número de subtrações que o código abaixo realiza:

for (int i = 0; i < n; i++){

if (rand() % 2 == 0){

a--;

b--;

} else {

c--;

}

}

Melhor caso: n => O(n), Ω(n) e 𝚯(n)

Pior caso: 2n => O(n), Ω(n) e 𝚯(n)

• Calcule o número de subtrações que o código abaixo realiza:

for (int i = 3; i < n; i++){

a--;

}

Resultado = n - 3 => O(n), Ω(n) e 𝚯(n)

• Calcule o número de multiplicações que o código abaixo realiza:

for (int i = n; i > 0; i /= 2)

a \*= 2;

será realizado lg(n) (piso) + 1

O(lg n), Ω(lg n) e 𝚯(lg n)

• Encontre o menor valor em um array de inteiros

1º) Qual é a operação relevante

2º) Quantas vezes ela será executada?

int min = array[0];

for (int i = 1; i < n; i++){

if (min > array[i]){

min = array[i];

}

}

R1 = a mais relevante é a comparação entre elementos no array

R2 = sera executado n-1 vezes

• Exercício Resolvido (8): Encontrar Mínimo

int min = array[0];

for (int i = 1; i < n; i++){

if (min > array[i]){

min = array[i];

}

}

1º) Qual é a operação relevante?

A comparação entre elementos do array

2º) Quantas vezes ela será executada?

Ela será executada n-1 vezes

3º) O nosso T(n) = n – 1 é para qual dos três casos?

Nesse algoritmo será para os três casos, porque não existe condição de parada

4º) O nosso algoritmo é ótimo? Por que?

O algoritmo é ótimo porque ele realiza o mínimo necessário para realizar o problema, não sendo possível ser menor do que isso nesse caso.

• Exercício Resolvido (9): Pesquisa Sequencial

boolean resp = false;

for (int i = 0; i < n; i++){

if (array[i] == x){

resp = true;

i = n;

}

}

1º) Qual é a operação relevante?

A comparação entre elementos do array

2º) Quantas vezes ela será executada?

No melhor caso 1 vez, no pior caso N vezes e no caso médio (n+1)/2

3º) O nosso algoritmo é ótimo? Por que?

O algoritmo é ótimo porque ele realiza o mínimo necessário para realizar o problema, não sendo possível ser menor do que isso nesse caso.

• Encontre o maior e menor valores em um array de inteiros e, em seguida, encontre a função de complexidade de tempo para sua solução

int menor = array[0];

int maior = array[0]

for (int i = 1; i < n; i++){

if (array[i] < menor){

menor = array[i];

} else if (array[i] > maior){

maior = array[i]

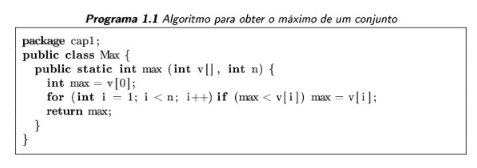
}

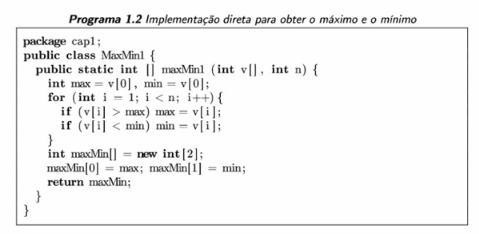
}

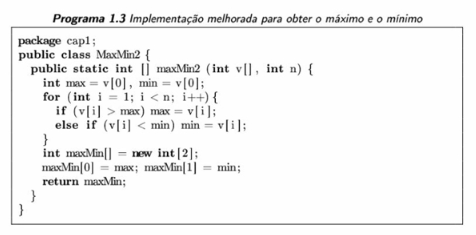
Melhor caso = 2 + (n-1)

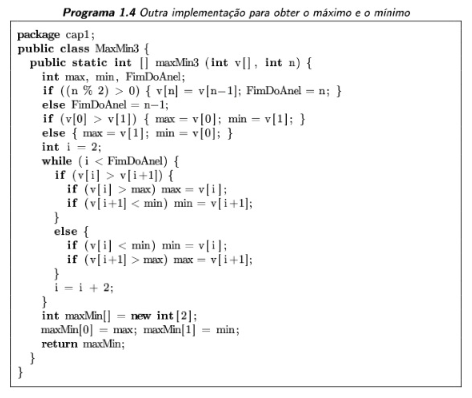
Pior caso = 2+ 2(n-1)

• Considerando o problema de encontrar o maior e menor valores em um array de inteiros, veja os quatro códigos propostos e analisados no livro do Ziviani









• Um aluno deve procurar um valor em um array de números reais. Ele tem duas alternativas. Primeiro, executar uma pesquisa sequencial. Segundo, ordenar o array e, em seguida, aplicar uma pesquisa binária. O que fazer?

Depende de quantas vezes o array será consultado, a pesquisa tem custo N, e a ordenação possui custo N.lg(n), se a quantidade de pesquisas a serem feitas for maior do que o custo da ordenação ira valer mais apena ordenar primeiro, caso contrario apenas pesquisar sequencialmente.

• Responda se as afirmações são verdadeiras ou falsas:

a) 3n2 + 5n + 1 é O(n): Falso

b) 3n2 + 5n + 1 é O(n2 ): Verdadeira

c) 3n2 + 5n + 1 é O(n3 ): Verdadeira

d) 3n2 + 5n + 1 é Ω(n): Verdadeira

e) 3n2 + 5n + 1 é Ω(n2 ): Verdadeira

f) 3n2 + 5n + 1 é Ω(n3 ): Falso

g) 3n2 + 5n + 1 é 𝚯(n): Falso

h) 3n2 + 5n + 1 é 𝚯(n2 ): Verdadeira

i) 3n2 + 5n + 1 é 𝚯(n3 ): Falso

• Preencha verdadeiro ou falso na tabela abaixo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | O(1) | O(lg(n)) | O(n) | O(n.lg(n)) | O(n2) | O(n3) | O(n5) | O(n20) |
| F(n) = lg(n) | x | V | V | V | V | V | V | V |
| F(n) = n.lg(n) | x | x | x | V | V | V | V | V |
| F(n) = 5n+1 | x | x | V | V | V | V | V | V |
| F(n) = 7n5-3n2 | x | x | x | x | x | x | V | V |
| F(n) = 99n3-1000n2 | x | x | x | x | x | V | V | V |
| F(n) = n5-99999n4 | x | x | X | x | x | x | V | V |

• Preencha verdadeiro ou falso na tabela abaixo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Ω(1) | Ω(lg(n)) | Ω(n) | Ω(n.lg(n)) | Ω(n2) | Ω(n3) | Ω(n5) | Ω(n20) |
| F(n) = lg(n) | V | V | x | x | x | x | x | x |
| F(n) = n.lg(n) | V | V | V | V | x | x | x | x |
| F(n) = 5n+1 | V | V | V | X | x | x | x | x |
| F(n) = 7n5-3n2 | V | V | V | V | V | V | V | x |
| F(n) = 99n3-1000n2 | V | V | V | V | V | V | x | x |
| F(n) = n5-99999n4 | V | V | V | V | V | V | V | x |

• Preencha verdadeiro ou falso na tabela abaixo:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Θ(1) | Θ(lg(n)) | Θ(n) | Θ(n.lg(n)) | Θ(n2) | Θ(n3) | Θ(n5) | Θ(n20) |
| F(n) = lg(n) | x | V | x | x | x | x | x | x |
| F(n) = n.lg(n) | x | x | x | V | x | x | x | x |
| F(n) = 5n+1 | x | x | V | x | x | x | x | x |
| F(n) = 7n5-3n2 | x | x | x | x | x | x | V | x |
| F(n) = 99n3-1000n2 | x | x | x | x | x | V | x | x |
| F(n) = n5-99999n4 | x | x | x | x | x | x | V | x |

• Sabendo que o Algoritmo de Seleção faz 𝚯(n2 ) comparações entre registros, quantas dessas comparações temos no código abaixo? Justifique

for (int i = 0; i < n; i++){

seleção();

}

Como o algoritimo acontece N vezes é feito uma multiplicação de n.n2 logo serão n3 vezes:

Θ(n3)

• Sabendo que o limite inferior da ordenação é 𝚯(n.lg n) e que o custo da pesquisa binária é 𝚯(lg n), qual é a ordem de complexidade de uma solução em que ordenamos um array e efetuamos uma pesquisa binária. Justifique sua resposta

Neste caso é feito uma soma pois ambas operações acontecem separadamente sem influenciar na outra, somando o custo de operação dos dois algoritimos chegamos a equação: n.lg(n) + lg(n), a qual possui um custo de:

Θ(n.lg(n))

• Dado f(n)=3n2 -5n-9, g(n) = n.lg(n), l(n) = n.lg2 (n) e h(n) = 99n8 , qual é a ordem de complexidade das operações abaixo. Mostre sua resposta usando as notações O, Ω e 𝚯:

a) h(n) + g(n) - f(n) = O(n8), Ω(n8), Θ(n8)

b) 𝚯(h(n)) + 𝚯(g(n)) - 𝚯(f(n)) = O(n8), Ω(n8), Θ(n8)

c) f(n) x g(n) = O(n3.lg(n)), Ω(n3.lg(n)), Θ(n3.lg(n))

d) g(n) x l(n) + h(n) = O(n8), Ω(n8), Θ(n8)

e) f(n) x g(n) x l(n) = O(n4.lg3(n)), Ω(n4.lg3(n)), Θ(n4.lg3(n))

f) 𝚯(𝚯(𝚯(𝚯(f(n))))) = O(n2), Ω(n2), Θ(n2)

• Dada a definição da notação O:

a) Mostre os valores de c e m tal que, para n ≥ m, |3n2 + 5n +1| ≤ c x |n2 |, provando que 3n2 + 5n + 1 é O(n2 )

Para isso ser verdade C precisa ser maior do que 3 (igual ou maior a 4)

b) Mostre os valores de c e m tal que, para n ≥ m, |3n2 + 5n +1| ≤ c x |n3 |, provando que 3n2 + 5n + 1 é O(n3 )

Qualquer valor acima de 4 atende este requisito

c) Prove que 3n2 + 5n + 1 não é O(n)

Não existe valor que seja capaz de fazer com que a curva da inequação se demonstre verdadeiro. Aumentar o valor de C apenas ira retardar o momento em que a curva quadrática ira superar a Linear

• Dada a definição da notação Ω:

a) Mostre os valores de c e m tal que, para n ≥ m, |g(n)| ≥ c x |f(n)|, provando que 3n2 + 5n + 1 é Ω(n2 )

Qualquer valor abaixo de 3 atende esse requisito (igual ou menor a 2)

, e m = 0 porque as 2 linhas nunca se tocam neste cass

b) Mostre os valores de c e m tal que, para n ≥ m, |g(n)| ≥ c x |f(n)|, provando que 3n2 + 5n + 1 é Ω(n)

qualquer valor de N ira gerar uma linha que por mesmo que em determinado momento ultrapasse a curva quadrática em outro seguinte será superada e passara novamente a ser um limite inferior. Mas de toda forma o valor mais apropriado de C para que M = 0 seria C = 8

c) Prove que 3n2 + 5n + 1 não é Ω(n3 )

Em um gráfico cubico por mesmo que inicialmente seja inferior ao quadrático sempre haverá um momento em que este ira ultrapassa-lo, por exemplo nesse caso quando de n³ e c = 1, o gráfico ira ultrapassa-lo quando m = 4.2...

• Dada a definição da notação 𝚯:

a) Mostre um valor para c1 , c2 e m tal que, para n ≥ m, c1 x |f(n)| ≤ |g(n)| ≤ c2 x |f(n)|, provando que 3n2 + 5n +1 é 𝚯(n2 )

Podemos recuperar os dados descobertos anteriormente ao calcular O e Ω da equação.

O valor de C1 = 4 e C2 = 2

b) Prove que 3n2 + 5n +1 não é 𝚯(n)

Assim como demonstrado anteriormente não existe valor que satisfaça c1 para que crie um limite superior

c) Prove que 3n2 + 5n +1 não é 𝚯(n3 )

Assim como demonstrado anteriormente não existe valor que satisfaça c2 para que crie um limite inferior

• Faça um resumo sobre Teoria da Complexidade, Classes de Problemas P, NP e NP-Completo. Use LaTeX e siga o modelo de artigos da SBC (sem abstract, resumo nem seções) com no máximo duas página.

A questão acima será respondida em um link para o site Overleaf onde o documento LaTeX foi escrito.

<https://pt.overleaf.com/read/mqmsbzvzchbd>

• Apresente a função e a complexidade para os números de comparações e movimentações de registros para o pior e melhor caso

void imprimirMaxMin(int [] array, int n){

int maximo, minimo;

if (array[0] > array[1]){

maximo = array[0];

minimo = array[1];

} else {

maximo = array[1];

minimo = array[0];

}

for (int i = 2; i < n; i++){

if (array[i] > maximo){

maximo = array[i];

} else if (array[i] < minimo){

minimo = array[i];

}

}

}

|  |  |  |
| --- | --- | --- |
|  | Função |  |
|  | Movimentações | Comparações |
| Pior | f(n) = 2 + (n-2) | f(n) = 1 + 2(n-2) |
| Melhor | f(n) = 2 | f(n) = 1 + (n-2) |

|  |  |  |
| --- | --- | --- |
|  | Complexidade |  |
|  | Movimentações | Comparações |
| Pior | O(n), Ω(n) e Ɵ(n) | O(n), Ω(n) e Ɵ(n) |
| Melhor | O(1), Ω(1) e Ɵ(1) | O(n), Ω(n) e Ɵ(n) |

• Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

i = 0;

while (i < n) {

i++;

a--;

}

if (b > c) {

i--;

} else {

i--;

a--;

}

|  |  |  |
| --- | --- | --- |
|  | Função | Complexidade |
| Pior | F(n) = n + 2 | O(n), Ω(n) e Ɵ(n) |
| Melhor | F(n) = n + 1 | O(n), Ω(n) e Ɵ(n) |

• Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

a--;

b--;

}

c--;

}

|  |  |  |
| --- | --- | --- |
|  | Função | Complexidade |
| Todos | F(n) = (2n + 1)n | O(n2), Ω(n2) e Ɵ(n2) |

• Apresente a função e a complexidade para o número de subtrações para o pior e melhor caso

for (i = 0; i < n; i++) {

for (j = 1; j <= n; j\*=2) {

b--;

}

}

|  |  |  |
| --- | --- | --- |
|  | Função | Complexidade |
| Todos | F(n) = n\*lg(n) + n | O(n\*lg(n)), Ω(n\*lg(n)) e Ɵ(n\*lg(n)) |

• Suponha um sistema de monitoramento contendo os métodos telefone, luz, alarme, sensor e câmera, apresente a função e ordem de complexidade para o pior e melhor caso: (a) método alarme; (b) outros métodos.

void sistemaMonitoramento() {

if (telefone() == true && luz() == true){

alarme(0);

} else {

alarme(1);

}

for (int i = 2; i < n; i++){

if (sensor(i- 2) == true){

alarme (i - 2);

} else if (camera(i- 2) == true){

alarme (i - 2 + n);

}

}

}

|  |  |  |
| --- | --- | --- |
|  | Alarme |  |
|  | Função | Complexidade |
| Todos | F(n) = 1 + (n-2) | O(n), Ω(n) e Ɵ(n) |

|  |  |  |
| --- | --- | --- |
|  | Sensor |  |
|  | Função | Complexidade |
| Todos | F(n) = n-2 | O(n), Ω(n) e Ɵ(n) |

• Apresente um código, defina duas operações relevantes e apresente a função e a complexidade para as operações escolhidas no pior e melhor caso

Resolução de uma questão retirada de um repositório de perguntas comuns feitas em entrevistas para a google, que eu resolvi usando a linguagem Go:

Enunciado da questão: Recebido um array, retorno outro contendo a multiplicação dos elementos não pertencentes a posição ‘i’. exemplo: [1, 2, 3, 4, 5] => [120, 60, 40, 30, 24]

func function(slice []int) []int {

    if len(slice) == 1 {

        slice\_1 := make([]int, 0)

        slice\_1 = append(slice\_1, 0)

        return slice\_1

    }

    slice\_1 := make([]int, len(slice))

    i, temp := 0, 1

    for ; i < len(slice); i++ {

        slice\_1[i] = temp

        temp \*= slice[i]

    }

    i, temp = len(slice)-1, 1

    for ; i >= 0; i-- {

        slice\_1[i] \*= temp

        temp \*= slice[i]

    }

    return slice\_1

}

Operação relevante escolhida: Atribuição de elementos do array, comparações.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Atribuição no array | | |  |
|  | | Função | Complexidade | |
| Todos | | F(n) = 2n + 2n = 4n | O(n), Ω(n) e Ɵ(n) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Comparações | | |  |
|  | | Função | Complexidade | |
| Todos | | F(n) = 1 + n + n = 1 + 2n | O(n), Ω(n) e Ɵ(n) | |

• Apresente o tipo de crescimento que melhor caracteriza as funções abaixo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Constante | Linear | Polinomial | Exponencial |
| 3n |  | V |  |  |
| 1 | V |  |  |  |
| (3/2)n |  | V |  |  |
| 2n3 |  |  | V |  |
| 2n |  |  |  | V |
| 3n2 |  |  | V |  |
| 1000 | V |  |  |  |
| (3/2)n |  |  |  | V |

• Classifique as funções f1 (n) = n2 , f2 (n) = n, f3 (n) = 2n , f4 (n) = (3/2)n , f5 (n) = n3e f6 (n) = 1 de acordo com o crescimento, do mais lento para o mais rápido

f6 (n) = 1

f2 (n) = n

f1 (n) = n2

f5 (n) = n3

f4 (n) = (3/2)n

f3 (n) = 2n

• Classifique as funções f1 (n) = n.log6 (n), f2 (n) = lg(n), f3 (n) = log8 (n), f4 (n) = 8n2 , f5 (n) = n.lg(n), f6 (n) = 64, f7 (n) = 6n3 , f8 (n) = 82n e f9 (n) = 4n de acordo com o crescimento, do mais lento para o mais rápido

f6 (n) = 64

f3 (n) = log8 (n)

f2 (n) = lg(n)

f9 (n) = 4n

f1 (n) = n.log6 (n)

f5 (n) = n.lg(n)

f4 (n) = 8n2

f7 (n) = 6n3

f8 (n) = 82n

• Faça a correspondência entre cada função f(n) com sua g(n) equivalente, em termos de 𝚯. Essa correspondência acontece quando f(n) = 𝚯(g(n))

|  |  |
| --- | --- |
| f(n) | g(n) |
| n+30 | 3n-1 |
| n2+2n-10 | n2+3n |
| n3.3n | n4 |
| lg(n) | lg(2n) |

• No Exercício Resolvido (10), verificamos que quando desejamos pesquisar a existência de um elemento em um array de números reais é adequado executar uma pesquisa sequencial cujo custo é 𝚯(n). Nesse caso, o custo de ordenar o array e, em seguida, aplicar uma pesquisa binária é mais elevado, 𝚯(n \* lg(n)) + 𝚯(lg(n)) = 𝚯(n \* lg(n)). Agora, supondo que desejamos efetuar n pesquisas, responda qual das duas soluções é mais eficiente

Para realizar N pesquisas teremos gasto n em todas as execuções, o que ira gerar um custo total de n\*n, levando a um 𝚯(n²). Mas se aplicarmos uma ordenação iremos executar lg(n) em todas as pesquisas, levando a um custo total de n.lg(n) \* lg(n), o que se torna 𝚯(n.lg²(n)). Sendo o crescimento de n.lg²(n) menor do que o crescimento de n² teremos um custo menor ao realizarmos a ordenação com a pesquisa binaria.