Lista de Ejercicios: Análisis de Complejidad Algorítmica

I. Definición de Orden Asintótico: O, Ω, Θ

Universidad UNICAMP

1. Sejam $f(n)$ e $g(n)$ funções assintoticamente não-negativas. Usando a definição básica da notação Θ, mostre que a função $h(n) = \max\{f(n), g(n)\}$ pertence a $\Theta(f(n) + g(n))$.

2. É verdade que $2^{n+1} \in O(2^n)$? E $2^{2n} \in O(2^n)$?

3. Mostre que $n! \in o(n^n), n! \in \omega(2^n)$ e $\log n! \in \Theta(n \log n)$.

4. Prove ou apresente um contra-exemplo para cada uma das afirmações abaixo.

   (a) se $f(n) \in O(g(n))$ então $g(n) \in O(f(n))$
   (b) $f(n) + g(n) \in \Theta(\min(f(n), g(n)))$
   (c) se $f(n) \in O(g(n))$ então $2^{f(n)} \in O(2^{g(n)})$
   (d) se $f(n) \in O(g(n))$ então $g(n) \in \Omega(f(n))$
   (e) se $h(n) \in o(f(n))$ então $f(n) + h(n) \in \Theta(f(n))$

5. João era um aluno de MC448 que gostava de implementar e testar os algoritmos vistos em aula. Ele percebeu que o algoritmo InsertionSort (Ordena-Por-Inserção) era bem eficiente para vetores com poucos elementos. Ele implementou então o seguinte algoritmo:

   ```
   ALGORITMO-DO-JOAO(A, n)
   se n ⩽ 100
       então INSERTIONSORT(A, n)
       senão MERGESORT(A, n)
   ```

   Pedro, um colega de João, sabia que o InsertionSort tinha complexidade de pior caso $\Theta(n^2)$ e concluiu que o algoritmo do João tinha complexidade $\Theta(n^2)$. João, por outro lado, afirmou que seu algoritmo tinha complexidade $\Theta(n \lg n)$. Quem está certo? Por quê?

6. Sejam $T(n)$ e $f(n)$ funções dos inteiros no reais.

   a) O que significa "$T(n)$ é $O(f(n))$"?
   b) É verdade que $20n^3 + 10n \lg n + 5$ é $O(n^3)$? Justifique.
   c) É verdade que $\frac{1}{2}n^2$ é $O(n)$? Justifique.
   d) O que significa "$T(n)$ é $\Omega(f(n))$"?
   e) O que significa "$T(n) = n + \Omega(n \lg n)$"?

Universidad MIT

7. Problem 1-1. Asymptotic Notation

   For each of the following statements, decide whether it is *always true*, *never true*, or *sometimes true* for asymptotically nonnegative functions and. If it is *always true* or *never true*, explain why. If it is *sometimes true*, give one example for which it is true, and one for which it is false.

   (a) $f(n) = O(f(n)^2)$
   (b) $f(n) + g(n) = \Theta(\max(f(n), g(n)))$
   (c) $f(n) + O(f(n)) = \Theta(f(n))$
   (d) $f(n) = \Omega(g(n))$ and $f(n) = o(g(n))$ (note the little-o)
   (e) $f(n) \neq O(g(n))$ and $g(n) \neq O(f(n))$

Lista de Exercícios sobre Ordem de Complexidade

8. O que significa dizer que uma função g(n) é O(f(n))?

9. Indique se as afirmativas a seguir são verdadeiras ou falsas e justifique a sua resposta:

   a. $2^{n+1} = O(2^n)$.
   b. $2^{2n} = O(2^n)$.
   c. É melhor um algoritmo que requer 2^n passos do que um que requer $10n^5$ passos.
   d. f(n) = O(u(n)) e g(n) = O(v(n)) => f(n) + g(n) = O(u(n) + v(n))
   e. f(n) = O(u(n)) e g(n) = O(v(n)) => f(n) - g(n) = O(u(n) - v(n))

10. Suponha um algoritmo A e um algoritmo B com funções de complexidade de tempo a(n) = n^2 – n + 549 e b(n) = 49n + 49, respectivamente. Determine quais são os valores de n pertencentes ao conjunto dos números naturais para os quais A leva menos tempo para executar do que B.

11. Defina um Tipo Abstrato de Dados *TMatriz*, para representar matrizes quadradas de tamanho n. Implemente as operações para somar e multiplicar 2 matrizes. Explique qual é a ordem de complexidade dessas duas operações. Se você tivesse a opção de utilizar um algoritmo exponencial O(2^n) para multiplicar duas matrizes, qual algoritmo você iria preferir? Justifique. Qual seria a modificação necessária em seu tipo abstrato de dados para representar matrizes genéricas com dimensões (m,n)? Nesse caso, qual seria a ordem de complexidade para multiplicar 2 matrizes: (m,n) * (n, k)?

12. O *Casamento de Padrões* é um problema clássico em ciência da computação e é aplicado em áreas diversas como pesquisa genética, editoração de textos, buscas na internet, etc. Basicamente, ele consiste em encontrar as ocorrências de um *padrão P* de tamanho *m* em um *texto T* de tamanho *n*. Por exemplo, no texto T = “PROVA DE AEDSII” o padrão P = “OVA” é encontrado na posição 3 enquanto o padrão P=“OVO” não é encontrado. O algoritmo mais simples para o casamento de padrões é o algoritmo da “Força Bruta”, mostrado abaixo. Analise esse algoritmo e responda: Qual é a função de complexidade do número de comparações de caracteres efetuadas no melhor caso e no pior caso. Dê exemplos de entradas que levam a esses dois casos. *Explique sua resposta!*

```c
#define MaxTexto 100
#define MaxPadrao 10

/* Pesquisa o padrao P[1..m] no texto  T[1..n] */
void ForcaBruta(   char T[MaxTexto], int n,
                   char T[MaxPadrao], int m)
{
    int i,j,k;
    for( i = 0 ; i < n - m + 1 ; i++ )
    {
         k = i;
         j = 0;
         while ( ( j <= m ) && ( T[k] == P[j] ) )
         {
              j = j + 1;
              k = k + 1;
         }
         if (j > m)
         {
              printf("Casamento na posicao %d",i);
              break;
         }
    }
}
```

13. Considere que você tenha um problema para resolver e duas opções de algoritmos. O primeiro algoritmo é quadrático tanto no pior caso quanto no melhor caso. Já o segundo algoritmo, é linear no melhor caso e cúbico no pior caso. Considerando que o melhor caso ocorre 90% das vezes que você executa o programa enquanto o pior caso ocorre apenas 10% das vezes, qual algoritmo você escolheria? Justifique a sua resposta em função do tamanho da entrada.

Demostración de Complejidad, PUC-Rio

14. Considerar las siguientes Funciones

(a) $10.n^{\pi}$

(b) $\log n$

(c) $\log(n^2)$

(d) $0.005.n^{0.00001}$

(e) $1000.(\log n)^2$

(f) $30.n^3.\log n$

(g) $50.n.\log^2 n$

(h) $(\log n)^{\log n}$

(i) $\frac{n}{\log n}$

(j) $70.n$

(k) $\log \log n$

(l) $(1.5)^{\log^2 n}$

(m) $90.n^2.\log n + n^3.\log n$

- Coloque las funciones de arriba en orden de crecimiento Asintótico, i.e. Valor cuando $n \to \infty$.
- Agrupar aquellas funciones por su respectivo Orden O, Ω, Θ.

(n)

15. ¿Qué significa que una función g(n) es O(f (n)).

16. ¿Qué significa que una función g(n) es Θ(f (n)).

17. ¿Qué significa que una función g(n) es Ω(f (n)).

18. Supongamos un algoritmo A y un algoritmo B con funciones de complejidad de tiempo $a(n) = n^2\ n+549$ y $b(n) = 49n+49$, respectivamente. Determine cuáles son los valores de n pertenecientes al conjunto de los números naturales para los cuales A tiene menor tiempo para ejecutar que B.

19. Exprese una función $10n^3 - 5n^2 - 10n + 3$ en términos de notación Θ.

20. ¿Es verdad que $2n^3 + 5 = \Theta(n^3)$? Explique.

21. Dos algortimos A y B poseen complejidad n^5 y 2^n respectivamente. ¿Usted utilizará el algoritmo B en lugar de A, en que caso? Explicar

22. Cuál es el orden de complejidad en el peor caso de:

(a) $2n + 10$

(b) $(1/2)n(n + 1)$

(c) $n + \sqrt{n}$

(d) $n/1000$

(e) $(1/2)n^2$

(f) $(1/2)n^2 - 3n$

23. ¿Cuáles son las magnitudes físicas que influencian la eficiencia de tiempo de un algoritmo en la práctica?

24. Explique qué tipos de problemas o algoritmos suelen tener complejidad del orden de $n \log n$ y cómo los identificamos.

25. ¿Qué problemas suelen tener complejidad del orden de $\log n$?

26. ¿Cuáles son los problemas que suelen ser exponenciales?

27. Escribir las siguientes funciones en notación *O*.

(d) $3n^3 + 20n^2 \log n$

(b) $3n^n + (5)2^n$

(c) $(n - 1)^n + n^{(n-1)}$

(d) $4n + 2n \log n$

(e) 34

28. Verdadero o Falso, Justificar

(a) $(\log n)^{100} = O(n^{\varepsilon}),\ \forall\, \varepsilon > 0$.
(b) $2^{n+1} = O(2^n)$.
(c) Se $g(n, m) = m \log_d n$ onde $d = [m/n + 2]$
($[x]$ é o menor inteiro maior que x),
$m = O(n^2)$, então $g(n, m) = O(m^{(1+\varepsilon)})\ \forall \varepsilon > 0$.

29. Usando la definición de O, pruebe formalmente los siguientes enunciados

(a) $2\sqrt{n} + 6 = O(\sqrt{n})$
(b) $n^2 = \Omega(n)$
(c) $\log_2(n) = \Theta(ln(n))$
(d) 4^n no es $O(2^n)$.

[Se espera para cada parte, una prueba rigurosa (pero breve), usando la definició de O, Ω y Θ]

30. Busca dos funciones $f(n)$ y $g(n)$ que satisfagan las siguientes relaciones. Si no existen tales $f(n)$ y $g(n)$, explica brevemente por qué. [P2, HW2, Leitert, Summer 2017, Kent]

(a) $f(n) \in O(g(n))$ y $f(n) \notin \Theta(g(n))$
(b) $f(n) \in \Theta(g(n))$ y $f(n) \in O(g(n))$ (c) $f(n)$
$\in \Theta(g(n))$ y $f(n) \notin O(g(n))$ (d) $f(n) \in \Omega(g(n))$
y $f(n) \notin O(g(n))$

31. Sejam $f(n)$ e $g(n)$ funções assintoticamente não-negativas. Usando a definição básica da notação Θ, mostre que a função $h(n) = \max f(n), g(n)$ pertence a $\Theta(f(n) + g(n))$.

32. Mostre que para quaisquer constantes a, b onde $b > 0$ temos que $(n + a)^b$ Pertenece $\Theta(n^b)$.

33. É verdade que $2^{n+1} \in O(2^n)$? E $2^{2n} \in O(2^n)$?

34. Explique por que a afirmação "o tempo de execução do algoritmo A é pelo menos $O(n^2)$" não faz sentido.

35. Mostre que $n!$ $o(n^n)$, $n!$ $w(2^n)$ e $\log n!$ $\Theta(n \log n)$. Não utilize a aproximação de Stirling.

36. Prove ou apresente um contra-exemplo para cada uma das afirmacões abaixo.

(e) se $f(n) \in O(g(n))$ então $g(n) \in O(f(n))$
(b) $f(n) + g(n) \in \Theta(\min(f(n), g(n)))$
(c) se $f(n) \in O(g(n))$ então $2^{f(n)} \in O(2^{g(n)})$
(d) se $f(n) \in O(g(n))$ então $g(n) \in (f(n))$
(e) se $h(n) \in o(f(n))$ então $f(n) + h(n) \in \Theta(f(n))$

II. Calcular la Complejidad de los Algoritmos (Scripts de Código)

1. Obtenha como função de n a melhor análise de complexidade possıvel para os dois pseudo-codigos apresentados abaixo [Laber p1, 26/06/2007, PUC-Rio]

(a)

```
Pseudocodigo-1
soma ← 0
for i ← 1  to  n do
      for j ← 1 to n² do
            soma + +
            aux ← soma
            while aux > 1 do
                  aux ← aux/2
            end while
      end for
end for
```

(b)

```
Pseudocodigo-2
soma ← 0
for i ← 1 to n do
      for j ← 1 to n³ do
            soma ← soma + 2
            aux ← soma
            while aux ≥ 1 do
                  aux ← aux/2
            end while
      end for
end for
```

2. Obtenha como função a melhor análise de complexidade possível para os dois pseudo-códigos apresentados abaixo. [Laber p1, 24/04/2007, PUC- Rio]

```
t ← 0
Cont ← 1
for (i = 1; i ≤ n; i + +) do
      Cont ← 2 ∗ Cont
end for
while Cont ≥ 1 do
      Cont ← Cont/2
      for (j = 1; j ≤ n; j + +) do
            t + +
      end for
end while
```

3. Calcule a complexidade, no pior caso, do fragmento de codigo abaixo:

Pseudocodigo-1
```
int i, j, k;
for (i = 0; i < N; i + +) do
    for (j = 0; j < N; j + +) do
        R[i][j] = 0;
        for (k = 0; k < N ; k + +) do
            R[i][j]+ = A[i][k]*B[k][j];
        end for
    end for
end for
```

4. Calcule a complexidade, no pior caso, do fragmento de codigo abaixo:

Pseudocodigo-2
```
int i, j, k, s;
for (i = 0; i < N   1; i + +) do
    for (j = i + 1; j < N; j + +) do
        for (k = 1; k < j   1; k + +) do
            s = 1;
        end for
    end for
end for
```

5. Calcule a complexidade, no pior caso, do fragmento de codigo abaixo:

Pseudocodigo-3
```
int i, j, s;
for (i = 0; i < N   1; i + +) do
    for (j = 1; j < 2*N; j + +) do
        s = s + 1;
    end for
end for
```

6. Obter uma equação matemática relativa a uma análise do melhor e melhor caso do fragmento de código abaixo:

Pseudocodigo-1
```
for (i = 0; i < N; i = i + 2) do

    for (j = N   i; j >= 0; j      ) do

        if (V[i] < V[j]) then
            printf(i)
        end if
    end for
end for
```

7. Escreva um algoritmo eficiente que procure por um dado número em vetor ordenado. Quais são sua funçã de custo e ordens de complexidade O e Ω?

8. Supongamos que disponemos de la siguiente definició de tipo:

```
CONST n = ...;
TYPE vector = ARRAY [1..n] OF INTEGER;
```

Consideramos entonces los procedimientos y funciones siguientes:

```
PROCEDURE Algoritmo (VAR a:vector)
for (i = 0; i < N; i = i + 2) do
    for (j = N    i; j >= 0; j--    ) do
        if (V[i] < V[j]) then
            printf( "%d " , i ) ;
        end if
    end for
end for
```

9. Considere os pseudo-códigos abaixo. .   [Laber p1, 27/06/2011, PUC- Rio]

a) Faça a análise assintótica do procedimento abaixo, ou seja, determine uma função f(n) tal que T(n) = θ(f(n))

```
Pseudo1
    t ← 0
    Cont ← 1
    Para i=1 até n
        Cont ← cont+1
    Fim Para
    Enquanto cont ≥ 1
        Cont ← cont/2
        Para j = 1 a n
            t++
        Fim Para
    Fim Enquanto
```

b) Faça a análise assintótica do procedimento abaixo, ou seja, determine uma função f(n) tal que T(n) = θ(f(n))

```
Pseudo2
    t ← 1
    Enquanto t < n
        t ← 2t
        Para i = 1 a t
            cont ← 0
        Fim Para
    Fim Enquanto
```

10. Considere os pseudo-códigos abaixo. [Laber p1, 03/10/2011, PUC- Rio]

Faça uma análise assintótica de pior caso do trecho de código abaixo em função de m e n. Quanto mais justa a análise maior a pontuação. Assuma que a função rand(m) devolve, em tempo O(log m), um número aleatório do conjunto {1, ..., m}.

```
cont ← 0
For i := 1 to n
    a ← rand(m)
    If cont ≤ m
            For j := 1 to a
                    cont ++
            End For
    End if
End For
```

11. Obtenha como função de *n* a melhor análise de complexidade possível para o pseudo-código apresentado abaixo. [Laber p1, 24/04/2007, PUC- Rio]

```
Pseudo-Código-2
        Cont ← 0
        Para  i=1 até n
                Aux ← i
                Enquanto Aux ≥ 0 e Cont < n^(3/2)
                        Cont ++
                        Aux - -
                Fim Enquanto
        Fim Para
```

12. Analise a complexidade do pseudo código abaixo. [Laber p1, 18/12/2007, PUC- Rio]

```
Para  i = 1 até n faça
        Para  j = 1 até n^2 faça
                k ← 1
                Enquanto k < n
                        k ← 2 * k
                        Para  ℓ = 1 até k faça
                                Print 'Hello"
```

13. Análise em função de N a complexidade de tempo o procedimento. [Laber pf, 06/12/2010, PUC- Rio]

```
Proc1(N)
        t ← N
        Enquanto t > 0 faça.
                q ← 1
                Enquanto 2q ≤ t
                        q ← 2q
                Fim Enquanto
                t ← t − 1
        Fim Enquanto
Fim Proc1
```

III. Diseñar el seudocodigo y analizar su complejidad

Universidad de Kent

1. Use Binary Search to search for the numbers 2, 43, and 70 in the sequence:

   2 5 11 17 19 21 26 33 39 43 51 65 79 88 99

   Show for each iteration which item is selected and which part of the sequence remains.

2. Suppose you are given an array A of n distinct and sorted numbers that has been circularly shifted k positions to the right. For example, $\{35, 42, 5, 15, 27, 29\}$ is a sorted array that has been circularly shifted $k = 2$ positions, while $\{27, 29, 35, 42, 5, 15\}$ has been shifted $k = 4$ positions.
   – Suppose you know what k is. Give an $O(1)$ time algorithm to find the largest number in A.
   – Suppose you do not know what k is. Give an $O(\log n)$ time algorithm to find the largest number in A.

3. You have given two sorted arrays A and B of size n, respectively. Find the median of the two sorted arrays, i. e., of $A \cup B$. The overall run time complexity should be $O(\log n)$

4. Rank the following functions by order of growth. That is, find an arrangement $f1, f2, \ldots$ of the functions satisfying $f1 \in O(f2)$, $f2 \in O(f3)$, ... Partition your list into equivalence classes such that functions f_i and f_j are in the same class if and only if $f_i \in \Theta(f_j)$.

$$\left(\sqrt{2}\right)^{\log n} \quad n^2 \quad n! \quad \left(\tfrac{3}{2}\right)^n \quad \log^2 n$$

$$4^{\log n} \quad n \quad 2^n \quad n \log n \quad 2^{2^{n+1}}$$

Universidad UFOP Lista de Exercícios sobre Limite Inferior

5. Considere um arranjo A com n elementos não ordenados. O problema é achar o maior valor dentre estes n elementos.
   (a) Apresente um algoritmo ótimo para resolver esse problema. Implemente o seu algoritmo, ou.
   (b) Apresente um algoritmo para resovler esse problema. Qual é a função de complexidade desse algoritmo no melhor, pior caso, e no caso médio?

6. Considere um arranjo A com n elementos não ordenados. O problema é achar o maior e o menor valor dentre estes n elementos.
   (c) Apresente um algoritmo ótimo para resolver esse problema. Implemente o seu algoritmo, ou.
   (d) Apresente um algoritmo para resovler esse problema. Qual é a função de complexidade desse algoritmo no melhor, pior caso, e no caso médio?

7. Considere um arranjo A com n elementos não ordenados. O problema é achar o maior e o segundo maior valor dentre estes n elementos.
   (e) Apresente um algoritmo ótimo para resolver esse problema. Implemente o seu algoritmo, ou.
   (f) Apresente um algoritmo para resovler esse problema. Qual é a função de complexidade desse algoritmo no melhor, pior caso, e no caso médio?

8. São dados 2n números distintos distribuídos em dois arranjos A e B, cada um com n elementos ordenados, tal que: A[0] < A[1] < … < A[n-1] e B[0] < B[1] < … < B[n-1]. O problema é achar o n-ésimo maior número dentre estes 2n elementos.
   (a) Apresente um algoritmo ótimo para resolver esse problema. Implemente o seu algoritmo, ou.
   (b) Apresente um algoritmo para resovler esse problema. Qual é a função de complexidade desse algoritmo no melhor, pior caso, e no caso médio?