Projeto e Análise de Algoritmos
2024.2

Complexidade e Notação Assintótica

Prof. Marcio Costa Santos DCC/UFMG

COMPLEXIDADE DE ALGORITMOS

Algoritmos

- Conjunto finito de passos;
- Resolução de um problema;
- Dados de Entrada $\equiv$ *instância*;
- Dados de Saída $\equiv$ *resposta ou retorno*;
- Algoritmo é uma função que transforma a entrada na saída.

Pseudo-Código

- SE;
- PARA;
- ENQUANTO;
- Atribuição : $\leftarrow$;
- Matemática e regras.
- Estruturas de Dados Simples (Listas, Matrizes, Vetores e Árvores).

Exemplo Algoritmo 1

Entrada: vetor de inteiros x, tamanho n de x.
Saída: ?
$s \leftarrow 0$;
para $i = 1$ *até* n **faça**
 $s \leftarrow s + x[i]$;
fim
retorna s;

Exemplo Algoritmo 2

Entrada: vetor de inteiros x, tamanho n de x.
Saída: ?

$i \leftarrow 2$;
enquanto $i \leq n$ **faça**
    se $x[i]$ *é par* **então**
        $s \leftarrow s + 1$;
    fim
    $i \leftarrow i + 1$;
fim
retorna s;

Exemplo Algoritmo 3

Entrada: matrizes de inteiros x e y, número de linhas n de x e y; e número de colunas m de x e y.

Saída: ?

Matriz z;

para $i = 1$ *até* $i \leq n$ **faça**

  para $j = 1$ *até* $j \leq m$ **faça**

    $z[i][j] \leftarrow x[i][j] + y[i][j]$;

  fim

fim

retorna z;

Exemplo Algoritmo 4

Entrada: matrizes de inteiros x e y, número de linhas e colunas n de x e y

Saída: ?

Matriz z;

para $i = 1$ *até* $i \leq n$ **faça**

  para $j = 1$ *até* $j \leq n$ **faça**

    $z[i][j] \leftarrow 0$;

    para $k = 1$ *até* $k \leq n$ **faça**

      $z[i][j] \leftarrow z[i][j] + x[i][k].y[k][j]$;

    fim

  fim

fim

retorna z;

Exemplo Algoritmo 5

Entrada: vetor de inteiros x, tamanho n de x.
Saída: ?
$i \leftarrow 2$;
$s \leftarrow x[1]$;
enquanto $i \leq n$ e $x[i] \leq s$ **faça**
  $s \leftarrow x[i]$;
  $i \leftarrow i + 1$;
fim
retorna i;

Complexidade

Função de Complexidade

Complexidade de um algoritmo é uma FUNÇÃO que calcula, para cada entrada, o número de operações que este algoritmo realiza.

Complexidade

- SE = Escolha, subconjunto;
- PARA = Somatório;
- ENQUANTO = Somatório;
- Atribuição : $\leftarrow$; = Tempo Unitário
- Matemática e regras. = Tempo Unitário, a depender da complexidade das regras!
- Estruturas de Dados Simples (Listas, Matrizes, Vetores e Árvores). = Tempo de cada operação!

Exemplo Algoritmo 1

Entrada: vetor de inteiros x, tamanho n de x.
$s \leftarrow 0$;
para $i = 1$ *até* n **faça**
    $s \leftarrow s + x[i]$;
fim
retorna s;

Exemplo Algoritmo 1

Entrada: vetor de inteiros x, tamanho n de x.

$s \leftarrow 0$ (1);

for $i = 1$ *até* n $(\sum_{i=1}^{n})$ (2) **do**

$\quad s \leftarrow s + x[i]$(3) ;

end

retorna s;

Exemplo Algoritmo 1

$$
\begin{aligned}
F(x,n) = &\quad 1 + \sum_{i=1}^{n}(2+3) \\
= &\quad 1 + 4n
\end{aligned}
$$

Função de Complexidade Algoritmo 1

$F(x,n) = 1 + 5n$

Exemplo Algoritmo 2

Entrada: vetor de inteiros x, tamanho n de x.
$i \leftarrow 2$;
enquanto $i \leq n$ **faça**
    se $x[i]$ *é par* **então**
        $s \leftarrow s + 1$;
    fim
    $i \leftarrow i + 1$;
fim
retorna s;

Exemplo Algoritmo 2

Entrada: vetor de inteiros x, tamanho n de x.

$i \leftarrow 2$ (1);

enquanto $i \leq n$ $(\sum_{i=2}^{n})$ (1) **faça**

  se $x[i]$ *é par* (2) **então**

    $s \leftarrow s + 1$ (2);

  fim

  $i \leftarrow i + 1$ (2);

fim

retorna s;

Exemplo Algoritmo 2

y_i é 1 se $x[i]$ é par e 0 em caso contrário.

$$
\begin{aligned}
F(x,n) = &\ 1 + \sum_{i=2}^{n}(1+2+2+2y_i) = 1 + \sum_{i=2}^{n} 5 + \sum_{i=2}^{n} 2y_i \\
= &\ 1 + 5n - 5 + \sum_{i=2}^{n} 2y_i = 5n - 4 + \sum_{i \in \{x[i] \text{é par}\}, i \geq 2} 2
\end{aligned}
$$

Função de Complexidade Algoritmo 2

$F(x,n) = 5n - 4 + \sum_{i \in \{x[i] \text{é par}\}, i \geq 2} 2$

Exemplo Algoritmo 3

Entrada: matrizes de inteiros x e y, número de linhas n de x e y; e número de colunas m de x e y.

Matriz z;
para $i = 1$ *até* $i \leq n$ **faça**
    para $j = 1$ *até* $j \leq m$ **faça**
        $z[i][j] \leftarrow x[i][j] + y[i][j]$;
    fim
fim
retorna z;

Exemplo Algoritmo 3

Entrada: matrizes de inteiros x e y, número de linhas n de x e y; e número de colunas m de x e y.

Matriz z;

para $i = 1$ *até* $i \leq n$ $(\sum_{i=1}^{n})$ (2) **faça**

  para $j = 1$ *até* $j \leq m$ $(\sum_{j=1}^{m})$ (2) **faça**

    $z[i][j] \leftarrow x[i][j] + y[i][j]$ (8);

  fim

fim

retorna z;

Exemplo Algoritmo 3

$$
\begin{aligned}
F(x, n, m) &= \sum_{i=1}^{n}\left(2 + \sum_{j=1}^{m}(2+8)\right) \\
&= \sum_{i=1}^{n} 2 + \sum_{i=1}^{n}\sum_{j=1}^{m} 10 \\
&= 2n + \sum_{i=1}^{n} 10m = 2n + 10mn
\end{aligned}
$$

Função de Complexidade Algoritmo 3

$F(x, n, m) = 2n + 10mn$

Exemplo Algoritmo 4

Entrada: matrizes de inteiros x e y, número de linhas e colunas n de x e y

Matriz z;
para $i = 1$ *até* $i \leq n$ **faça**
    para $j = 1$ *até* $j \leq n$ **faça**
        $z[i][j] \leftarrow 0$;
        para $k = 1$ *até* $k \leq n$ **faça**
            $z[i][j] \leftarrow z[i][j] + x[i][k].y[k][j]$;
        fim
    fim
fim
retorna z;

Exemplo Algoritmo 4

Entrada: matrizes de inteiros x e y, número de linhas e colunas n de x e y

Matriz z;

para $i = 1$ *até* $i \leq n$ $(\sum_{i=1}^{n})$ (2) **faça**

  para $j = 1$ *até* $j \leq n$ $(\sum_{j=1}^{n})$ (2) **faça**

    $z[i][j] \leftarrow 0$ (1);

    para $k = 1$ *até* $k \leq n$ $(\sum_{k=1}^{n})$ (2) **faça**

      $z[i][j] \leftarrow z[i][j] + x[i][k] + y[k][j]$ (11);

    fim

  fim

fim

retorna z;

Exemplo Algoritmo 4

$$\begin{aligned} F(x,n) &= \sum_{i=1}^{n}(2+\sum_{j=1}^{n}(2+1+\sum_{k=1}^{n}11)) \\ &= \sum_{i=1}^{n}2+\sum_{i=1}^{n}\sum_{j=1}^{n}3+\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}11 \\ &= 2n+3n.n+11.n.n.n \end{aligned}$$

Função de Complexidade Algoritmo 4

$F(x,n,m) = 2n + 3n^2 + 11n^3$

Exemplo Algoritmo 5

Entrada: vetor de inteiros x, tamanho n de x.
$i \leftarrow 2$;
$s \leftarrow x[1]$;
enquanto $i \leq n$ *e* $x[i] \leq s$ **faça**
    $s \leftarrow x[i]$;
    $i \leftarrow i + 1$;
fim
retorna i;

Exemplo Algoritmo 5

Entrada: vetor de inteiros x, tamanho n de x.

$i \leftarrow 2$ (1);

$s \leftarrow x[1]$ (1);

enquanto $i \leq n$ *e* $x[i] \leq s$ $\sum_{i=2}^{\max\{i|\forall j \leq i, x[j-1] \leq x[j]\}}$ *(2)*

faça

$s \leftarrow x[i]$ (2);

$i \leftarrow i + 1$ (2);

fim

retorna i;

Exemplo Algoritmo 5

$$F(x,n) = 1 + 1 + \sum_{i=2}^{\max\{i \mid \forall j \leq i, x[j-1] \leq x[j]\}} (2 + 2 + 2)$$
$$= 2 + \sum_{i=2}^{\max\{i \mid \forall j \leq i, x[j-1] \leq x[j]\}} 6$$

Função de Complexidade Algoritmo 5

$F(x,n) = 2 + \sum_{i=2}^{\max\{i \mid \forall j \leq i, x[j-1] \leq x[j]\}} 6$

Complexidade de Algoritmos

Instância : conjunto de dados de entrada de um algoritmo, $\mathcal{I}$

Tamanho de uma instância : tamanho em bits da entrada.

$\mathcal{I}_n$: conjunto de instâncias de tamanho n.

Complexidade de um algoritmo: É uma função que leva o tamanho da instância em ....

Complexidade de Algoritmos - Pior caso

Complexidade de Pior Caso

O maior (pior) número de passos para uma instância de tamanho n.

$$T(n) = \max_{x \in I_n} F(n, x)$$

Complexidade de Algoritmos - Melhor caso

Complexidade de Melhor Caso

O menor (melhor) número de passos para uma instância de tamanho n.

$$T(n) = \min_{x \in I_n} F(n, x)$$

Complexidade de Algoritmos - Caso Médio

Complexidade de Médio Caso

O número esperado de passos para uma instância de tamanho n.

$$T(n) = \sum_{x \in I_n} F(n, x) p_x$$

Exemplo Algoritmo 1

Entrada: vetor de inteiros x, tamanho n de x.
Saída: ?
$s \leftarrow 0$;
para $i = 1$ *até* n **faça**
  $s \leftarrow s + x[i]$;
fim
retorna s;

Função de Complexidade Algoritmo 1

$F(x, n) = 1 + 5n$

Complexidade do Algoritmo 1

Melhor: $T(n) = \min_{x \in I_n} F(x, n) = \min_{x \in I_n}(1 + 5n) = 1 + 5n$

Pior: $T(n) = \max_{x \in I_n} F(x, n) = \max_{x \in I_n}(1 + 5n) = 1 + 5n$

Médio: $T(n) = \sum_{x \in I_n} F(x, n)p_x = (1 + 5n) \sum_{x \in I_n} p_x =$
$1 + 5n$

Exemplo Algoritmo 2

Entrada: vetor de inteiros x, tamanho n de x.
Saída: ?

$i \leftarrow 2$;
enquanto $i \leq n$ **faça**
    se $x[i]$ *é par* **então**
        $s \leftarrow s + 1$;
    fim
    $i \leftarrow i + 1$;
fim
retorna s;

Função de Complexidade Algoritmo 2

$$F(x, n) = 5n - 4 + \sum_{i \in \{x[i] \text{é par}\}, i \geq 2} 2$$

Complexidade do Algoritmo 2

Melhor: $T(n) = \min_{x \in I_n} F(x,n) =$
$\min_{x \in I_n}(5n - 4 + \sum_{i \in \{x[i] \text{é par}\}, i \geq 2} 2) = 5n - 4$

Pior: $T(n) = \max_{x \in I_n} F(x,n) =$
$\max_{x \in I_n}(5n - 4 + \sum_{i \in \{x[i] \text{é par}\}, i \geq 2} 2) =$
$5n - 4 + 2(n-1) = 5n - 4 + 2n - 2 = 7n - 6$

Médio: ...

Complexidade do Algoritmo 2 - Caso Médio

$$
\begin{aligned}
T(n) = &\sum_{x \in I_n} F(x,n) p_x = \sum_{x \in I_n} (5n - 4 + \sum_{i \in \{x[i] \text{é par}\}, i \geq 2} 2) p_x \\
= &\sum_{i=1}^{n-1} (5n - 4 + 2i) p_i = \sum_{i=1}^{n-1} (5n-4) p_i - \sum_{i=1}^{n-1} 2 i p_i \\
= & \qquad 5n - 4 + \sum_{i=1}^{n-1} i \frac{\binom{n}{i}}{2^n}
\end{aligned}
$$

Exemplo Algoritmo 3

Entrada: matrizes x e y, número de linhas n de x e y; e número de colunas m de x e y.

Saída: ?

Matriz z;

para $i = 1$ *até* $i \leq n$ **faça**

  para $j = 1$ *até* $j \leq m$ **faça**

    $z[i][j] \leftarrow x[i][j] + y[i][j]$;

  fim

fim

retorna z;

Função de Complexidade Algoritmo 3

$F(x, n, m) = 2n + 4mn$

Complexidade do Algoritmo 3

$$
\begin{aligned}
\text{Melhor: } & T(n,m) = \min_{x \in I_n} F(x,n,m) = \\
& \min_{x \in I_n} (2n + 4mn) = 2n + 4mn \\
\text{Pior: } & T(n,m) = \max_{x \in I_n} F(x,n,m) = \\
& \max_{x \in I_n} (2n + 4mn) = 2n + 4mn \\
\text{Médio: } & T(n,m) = \sum_{x \in I_n} F(x,n,m) p_x = \\
& (2n + 4mn) \sum_{x \in I_n} p_x = 2n + 4mn
\end{aligned}
$$

NOTAÇÃO ASSINTÓTICA

Análise Assintótica

- Precisamos mesmo calcular essas funções de forma exata?
- Faz diferença uma complexidade de pior caso de $2n$ para uma de $3n$?
- E $2n + 4$ para $3n - 1$?
- Análise quando os tamanhos crescem.

Análise Assintótica

O: Limite Superior

o: Limite Superior estrito

Ω: Limite Inferior

ω: Limite Inferior estrito

Θ: Equivalência

Limite Superior

$f = O(g)$

Existem n_0 e c tal que:

$$f(n) \leq c.g(n) \text{ para todo } n \geq n_0$$

Limite Superior Estrito

$f = o(g)$

Para todo $c > 0$ existe n_0 tal que:

$$f(n) < c.g(n) \text{ para todo } n \geq n_0$$

Limite Inferior

$f = \Omega(g)$

Existem n_0 e c tal que:

$$f(n) \geq c.g(n) \text{ para todo } n \geq n_0$$

Limite Inferior Estrito

$f = \omega(g)$

Para todo $c > 0$ existe n_0 tal que:

$$f(n) > c.g(n) \text{ para todo } n \geq n_0$$

Equivalência

$f = \Theta(g)$

Existem n_0, c_1 e c_2 tal que:

$$c_1.g(n) \leq f(n) \leq c_2.g(n) \text{ para todo } n \geq n_0$$

Exemplos

- $n^2 + n = O(n^3)$
- $n^2 + n = O(n^2)$
- $n^2 + n = o(n^3)$
- $n^2 + n = \Omega(n^2)$
- $n^2 + n = \Omega(n)$
- $n^2 + n = \Theta(n^2)$

Propriedades

- $k.f(n) = O(f(n))$ para k real.
- $f(n).g(n) = O(f(n).g(n))$.
- $O(f(n)) + O(g(n)) = O(f(n) + g(n))$.
- $O(f(n)).O(g(n)) = O(f(n).g(n))$.

Exemplo Algoritmo 1

Entrada: vetor de inteiros x, tamanho n de x.
$s \leftarrow 0$;
para $i = 1$ *até* n **faça**
  $s \leftarrow s + x[i]$;
fim
retorna s;

Exemplo Algoritmo 1

Entrada: vetor de inteiros x, tamanho n de x.
$s \leftarrow 0$ (1);
for $i = 1$ *até* n $(\sum_{i=1}^{n})$ (2) **do**
   $s \leftarrow s + x[i]$(3) ;
end
retorna s;

Exemplo Algoritmo 1

Entrada: vetor de inteiros x, tamanho n de x.
$s \leftarrow 0$ $(O(1))$;
for $i = 1$ *até* n $(O(n))$ **do**
  $s \leftarrow s + x[i](O(1))$;
end
retorna s;

Exemplo Algoritmo 1

$$O(1) + O(n).O(1) = O(1) + O(n.1) = O(1 + n)$$
$$O(n)$$

Complexidade Algoritmo 1

$O(n)$

Exemplo Algoritmo 2

Entrada: vetor de inteiros x, tamanho n de x.

$i \leftarrow 2$;

enquanto $i \leq n$ **faça**

    se $x[i]$ *é par* **então**

        $s \leftarrow s + 1$;

    fim

    $i \leftarrow i + 1$;

fim

retorna s;

Exemplo Algoritmo 2

Entrada: vetor de inteiros x, tamanho n de x.

$i \leftarrow 2$ (1);

enquanto $i \leq n$ $(\sum_{i=2}^{n})$ (1) **faça**

  se $x[i]$ *é par* (2) **então**

    $s \leftarrow s + 1$ (2);

  fim

  $i \leftarrow i + 1$ (2);

fim

retorna s;

Exemplo Algoritmo 2

Entrada: vetor de inteiros x, tamanho n de x.

$i \leftarrow 2$ $(O(1))$;

enquanto $i \leq n$ $(O(n))$ **faça**

  se $x[i]$ *é par* $(O(1))$ **então**

    $s \leftarrow s + 1$ $(O(1))$;

  fim

  $i \leftarrow i + 1$ $(O(1))$;

fim

retorna s;

Exemplo Algoritmo 2

$$O(1) + O(n).O(1) = O(n)$$

Complexidade Algoritmo 2

$O(n)$