GA-026: Algoritmos I

Prof. Luiz Gadelha

Programa de Pós-Graduação em Modelagem Computacional, P4/2019
Laboratório Nacional de Computação Científica

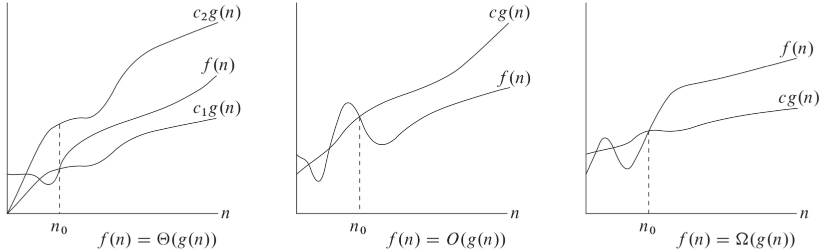
26 de setembro de 2019

Laboratório
Nacional de
Computação
Científica

Notação para Crescimento Assintótico de Funções

- Seja $g(n) : \mathbb{R} \to \mathbb{R}$.
- Podemos definir as seguintes classes de crescimento assintótico de funções:
  - $$\Theta(g(n)) = \{f(n) : \text{existem constantes positivas } c_1, c_2, \text{ e } n_0$$
    $$\text{tais que } 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ para todo} n \geq n_0\}$$
  - $$O(g(n)) = \{f(n) : \text{existe constante positiva } c \text{ e } n_0$$
    $$\text{tal que } 0 \leq f(n) \leq cg(n) \text{ para todo} n \geq n_0\}$$
  - $$\Omega(g(n)) = \{f(n) : \text{existe constante positiva } c \text{ e } n_0$$
    $$\text{tal que } 0 \leq cg(n) \leq f(n) \text{ para todo} n \geq n_0\}$$

Notação para Crescimento Assintótico de Funções



- Fonte: Cormen, T., Leiserson, C., Rivest, R., Stein, C. (2009). Introduction to Algorithms. MIT Press.

Laboratório Nacional de Computação Científica

Projeto de Algoritmos: Dividir para Conquistar

- Muitos algoritmos usam recursão, chamando a si mesmos para resolver algum problema computacional.
- São típicos da técnica de **dividir para conquistar** de projeto de algoritmos:
  - **Dividir**: um problema computacional é dividido em subproblemas computacionais menores similares ao original;
  - **Conquistar**: o algoritmo é aplicado recursivamente aos subproblemas para solucioná-los;
  - **Combinar**: as soluções dos subproblemas são combinadas para se obter uma solução para o problema original.

Mergesort

- O **Mergesort** segue a estratégia de dividir para conquistar:
  - **Dividir**: dividir um vetor de tamanho n em dois subvetores de tamanho $\frac{n}{2}$;
  - **Conquistar**: ordenar os dois subvetores recursivamente com o Mergesort;
  - **Combinar**: fundir dois subvetores ordenados para obter o resultado ordenado.

MERGE-SORT(A, p, r)

if $p < r$
    $q = \lfloor (p + r)/2 \rfloor$
    MERGE-SORT(A, p, q)
    MERGE-SORT$(A, q + 1, r)$
    MERGE(A, p, q, r)

- Fonte: Cormen, T., Leiserson, C., Rivest, R., Stein, C. (2009). Introduction to Algorithms. MIT Press.

Mergesort

```
MERGE(A, p, q, r)
n1 = q - p + 1
n2 = r - q
let L[1 .. n1 + 1] and R[1 .. n2 + 1] be new arrays
for i = 1 to n1
    L[i] = A[p + i - 1]
for j = 1 to n2
    R[j] = A[q + j]
L[n1 + 1] = ∞
R[n2 + 1] = ∞
i = 1
j = 1
for k = p to r
    if L[i] ≤ R[j]
        A[k] = L[i]
        i = i + 1
    else A[k] = R[j]
        j = j + 1
```

- Fonte: Cormen, T., Leiserson, C., Rivest, R., Stein, C. (2009). Introduction to Algorithms. MIT Press.

Mergesort: Correção

- **Invariante de laço**.

  No início do **for** (linhas 12-17), o subvetor $A[p, ..., k-1]$ contem os $k-p$ menores elementos de $L[1, ..., n_1+1]$ e $R[1, ..., n_2+1]$ ordenados. Além disso, $L[i]$ e $R[j]$ são os menores elementos dos seus vetores.
- Inicialização:
  - Antes do 1^o laço temos $k = p$. Logo $A[p, ..., k-1]$ é vazio e satisfaz o IdL.
  - Como $i = j = 1$, $L[i]$ e $R[j]$ são os menores elementos dos seus vetores que não foram copiados para A ainda.

Mergesort: Correção

- Manutenção:
  - Suponha que $L[i] \leq R[j]$:
    - Logo, $L[i]$ é o menor elemento ainda não copiado para A.
    - Como $A[p, ..., q-1]$ tem os $k-p$ menores elementos, a linha 14 copia $L[i]$ para $A[k]$.
    - Logo, $A[p, ..., q]$ conterá os $k-p+1$ menores elementos.
    - Incrementando k (linha 12) e i (linha 15), reconfigura o IdL para a próxima iteração.
  - O caso $L[i] > R[j]$ é análogo.

Laboratório Nacional de Computação Científica

Mergesort: Correção

- Terminação:
  - Quando o algoritmo para, $k = r + 1$.
  - Pelo IdL, $A[p, ..., k-1] = A[p, ..., r]$ contem os $k - p = r - p + 1$ menores elementos de $L[1, ..., n_1 + 1]$ e $R[1, ..., n_2 + 1]$ ordenados.
  - Os vetores L e R contém $n_1 + n_2 + 2 = r - p + 3$ elementos. Todos foram copiados de volta para A, exceto pelos dois sentinelas ∞.

Mergesort: Análise de Complexidade

- O **Mergesort** segue a estratégia de dividir para conquistar.
- Podemos pensar no custo de cada etapa:
  - **Dividir**: calcula índice do meio do vetor: $D(n) = \Theta(1)$;
  - **Conquistar**: divide em dois problemas de tamanho $\frac{n}{2}$, assim essa etapa tem custo $2T(\frac{n}{2})$;
  - **Combinar**: Corresponde ao MERGE, logo tem custo $C(n) = \Theta(n)$.
- Assim, podemos considerar que o custo total do MERGE-SORT, $T(n)$ é dado por:

$$T(n) = \begin{cases} \Theta(1), & \text{se } n = 1. \\ 2T(\frac{n}{2}) + D(n) + C(n) & \text{se } n > 1. \end{cases}$$

- Que corresponde a:

$$T(n) = \begin{cases} \Theta(1), & \text{se } n = 1. \\ 2T(\frac{n}{2}) + \Theta(n) & \text{se } n > 1. \end{cases}$$

Mergesort: Análise de Complexidade

- Podemos resolver a equação de recorrência (quadro), para mostrar que $T(n) = \Theta(n \lg n)$.

Exercício

- **Problema da busca**.
  - **Entrada**: Sequência $A = \langle a_1, a_2, ..., a_n \rangle$ e um valor v.
  - **Saída**: i, índice de uma ocorrência de v em A, ou NULO, caso v não ocorra em A.
- Escrever pseudo-código para algoritmo que solucione o problema da busca.
- Demonstre que o algoritmo é correto.
- Analise o tempo de execução no melhor e no pior caso.

Obrigado!

E-mail: lgadelha@lncc.br