Arrays, Linked Lists e Double Linked Lists

Rafael Alves da Costa

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
FATEC Carapicuíba

Aula 4 - Estrutura de dados

09/2025

Sumário

Relembrando: Notação O!!!

Relembrando: Notação O!!!

Para caracterizar a **eficiência** de um **algoritmo** em termos de **tempo de execução**, quantificamos o número de operações ou passos necessários. O tempo de execução é expresso pelo número de passos para resolver o problema[1].

[1] Miller, B. N.; Ranum, D. L. Problem solving with algorithms and data structures using python, 2Ed. Franklin, Beedle and Associates Inc., 2011.

Unidade Básica de Computação

Para comparar algoritmos, usamos o número de instruções de atribuição[1].

- Na função `sumOfN()`, a soma é representada por `T(n) = 1 + n`, onde `n` é o tamanho do problema[1].
- **T(n)** é o tempo gasto ou consumido para resolver um problema de tamanho n, ou seja, 1+n passos[1].

Ordem de Magnitude

O termo dominante da função T(n) é o mais relevante. Isso é chamado de notação O (Big-O) e é escrito como O(f(n))[1].

- Fornece uma aproximação da parte dominante da função[1].

Exemplo

Para `T(n) = 1 + n`, a constante 1 se torna insignificante à medida que n cresce, então `T(n)` é $O(n)^1$.

- Outro exemplo: `T(n) = 5n² + 27n + 1005`. Quando n é grande, o termo dominante é n^2, então `T(n)` é $O(n^2)^1$.

Desempenho em Casos Diferentes

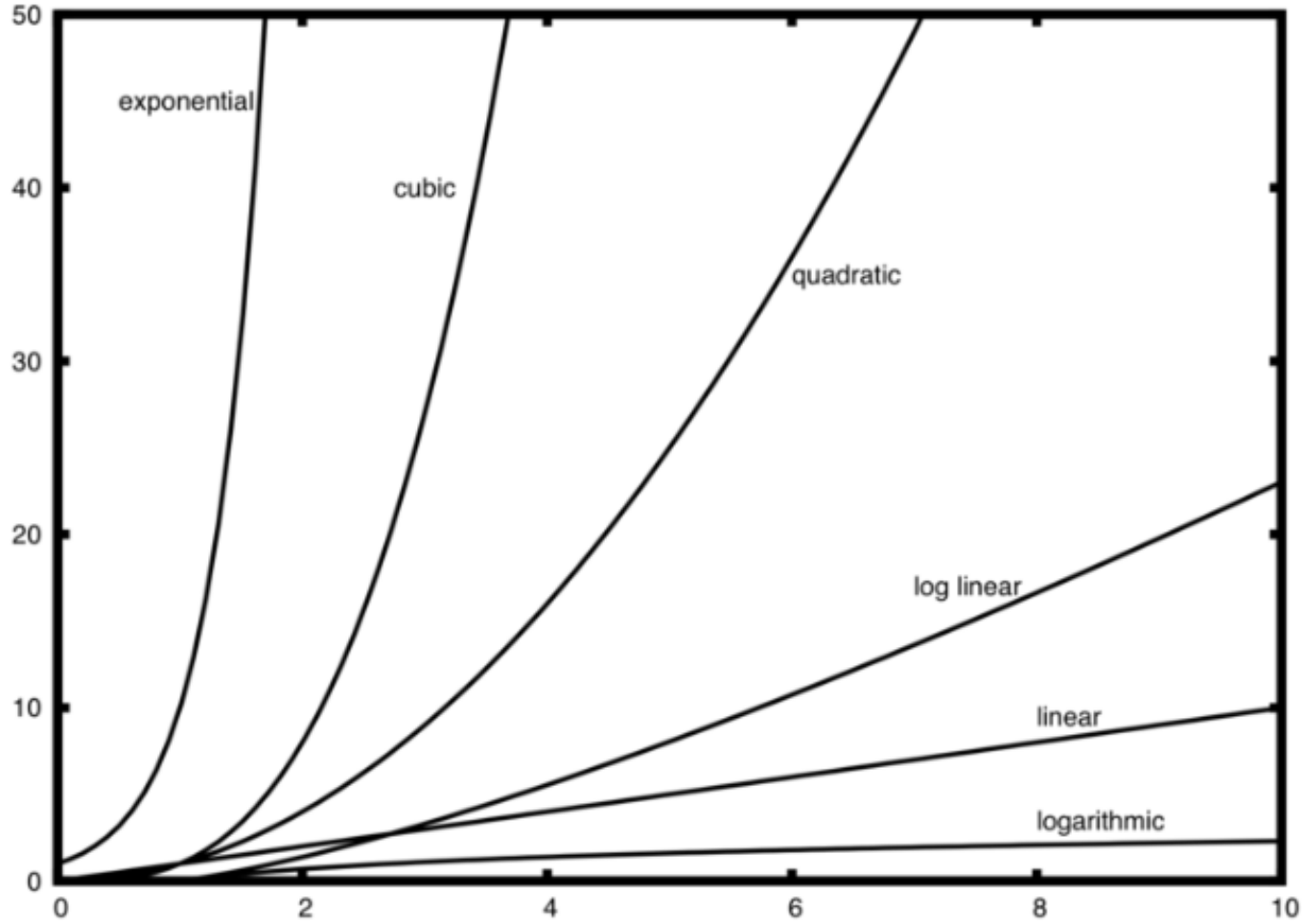
Algoritmos podem ter diferentes desempenhos para diferentes conjuntos de dados[1]:

- Melhor caso
- Pior caso

Funções Comuns de Ordem de Magnitude

f(n)	**Nome**
1	Constante
$\log n$	Logarítmica
n	Linear
$n \log n$	Log Linear
n^2	Quadrática
n^3	Cúbica
2^n	Exponencial

Gráficos das funções comuns



Gráficos das funções comuns $O(\cdot)$

Fonte: Problem solving with algorithms and data structures using python[1]

Arrays

Arrays

Definição e Uso:

- Um array é uma coleção de elementos do mesmo tipo, armazenados em posições contíguas de memória.
- Em Java:

```java
int[] array = new int[10];
array[0] = 1;
```

- Em Python:

```python
array = [0] * 10
array[0] = 1
```

Arrays em Java e Python

- **Similaridades**:
  - Ambos armazenam coleções de elementos.
  - Acesso aos elementos é feito por índices (ex: `array[0]`).
  - Permitem iteração sobre os elementos (usando loops).
- **Diferenças**:
  - **Tamanho**:
    - Java: Tamanho fixo (definido na criação).
    - Python: Tamanho dinâmico (pode crescer ou diminuir).
  - **Tipagem**:
    - Java: Arrays são fortemente tipados (ex: `int[]` só armazena inteiros).
    - Python: Listas são heterogêneas (podem armazenar qualquer tipo de dado).

Arrays em Java e Python

- **Sintaxe**:
  - Java: Declaração explícita do tipo e tamanho.
  - Python: Uso de colchetes e inicialização simplificada.

```java
// Java: Array de inteiros
int[] array = new int[10];
array[0] = 1;
```

```python
# Python: Lista(aprox. arrays)
array = [0] * 10
array[0] = 1
```

Sobre a eficiência na criação de Arrays (Listas) em Python

Analisando criação de Arrays (listas) em Python

```python
# Definindo as funções a serem testadas
def teste1():
    l = []
    for i in range(100):
        l = l + [i]

def teste2():
    l = []
    for i in range(100):
        l.append(i)

def teste3():
    l = [i for i in range(100)]

def teste4():
    l = list(range(100))
```

Métodos da Classe `java.util.Arrays`

Funções Úteis para Manipulação de Arrays

- A classe `java.util.Arrays` fornece métodos estáticos para manipulação de arrays.
- Métodos comuns:
  - `equals(A, B)`: Verifica se dois arrays são iguais.
  - `fill(A, x)`: Preenche o array com um valor específico.
  - `sort(A)`: Ordena o array.
  - `binarySearch(A, x)`: Busca um valor em um array ordenado.
- Exemplo de uso:

```
int[] array = {5, 3, 1, 4, 2};
Arrays.sort(array); // Ordena o array
System.out.println(Arrays.toString(array)); // [1, 2, 3, 4, 5]
```

Geração de Números Pseudoaleatórios

- Números pseudoaleatórios são sequências de números que parecem aleatórios, mas são gerados por algoritmos determinísticos.
- A classe `java.util.Random` é usada para gerar números pseudoaleatórios.
- Métodos comuns:
  - `nextInt()`: Gera um número inteiro aleatório.
  - `nextDouble()`: Gera um número decimal aleatório entre 0.0 e 1.0.
  - `setSeed(s)`: Define a semente para o gerador.
- Exemplo de uso:

```
Random rand = new Random();
rand.setSeed(System.currentTimeMillis()); // Define a semente
int num = rand.nextInt(100); // Gera um número entre 0 e 99
```

- Sobre "O que o número retornado por System.currentTimeMillis representa?"

Exemplo Completo

- Exemplo de programa que gera um array de números pseudoaleatórios, ordena e compara:

```java
import java.util.Arrays;
import java.util.Random;

public class ArrayTest {
    public static void main(String[] args) {
        int[] data = new int[10];
        Random rand = new Random();
        rand.setSeed(System.currentTimeMillis());

        for (int i = 0; i < data.length; i++) {
            data[i] = rand.nextInt(100);
        }

        int[] orig = Arrays.copyOf(data, data.length);
        Arrays.sort(data);

        System.out.println("Original: " + Arrays.toString(orig));
        System.out.println("Ordenado: " + Arrays.toString(data));
    }
}
```

- Saída esperada:

```
Original: [41, 38, 48, 13, 28, 46, 33, 19, 10, 58]
Ordenado: [10, 13, 19, 28, 33, 38, 41, 46, 48, 58]
```

Encadeamento Simples

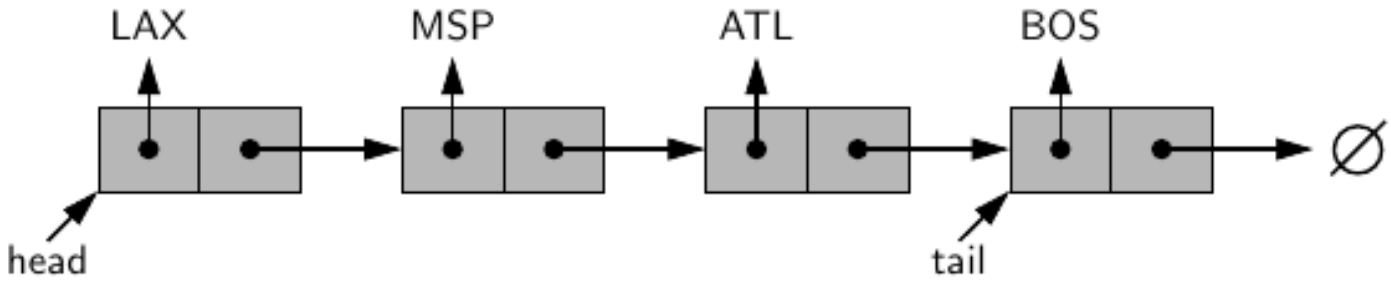
Listas Simplesmente Encadeadas

- **Lista Simplesmente Ligada:** Cada nó possui uma referência apenas para o próximo nó[2].
- **Benefícios:**
  - Inserções e exclusões rápidas no início da lista (tempo constante $O(1)$).
  - Uso eficiente de memória (apenas uma referência por nó).
- **Classe Base:** A classe 'SinglyLinkedBase' serve como uma fundação para implementações mais complexas[2].
- A classe 'Node' é uma classe leve e não pública para armazenar um nó simplesmente ligado[2].
  - **Estrutura:** A classe 'SinglyLinkedBase' gerencia uma lista simplesmente ligada com um ponteiro para o primeiro nó (cabeça).
  - **Obs:.** A lista simplesmente encadeada não possui nós sentinelas, ao contrário da lista duplamente encadeada.
- **Exemplo no notebook!!!**

[2]Goodrich, M. T., Tamassia, R., Goldwasser, M. H. (2013). Data structures and algorithms in Python.

Lista Simplemente Encadeada ou Lista Ligada

- Exemplo de uma lista simplesmente encadeada cujos elementos são strings indicando códigos de aeroportos. A instância da lista mantém um membro chamado head que identifica o primeiro nó da lista e, em algumas aplicações, outro membro chamado tail que identifica o último nó da lista. O objeto **None** é denotado como Ø.



- O primeiro e o último nó de uma lista encadeada são conhecidos, respectivamente, como a cabeça e a cauda da lista. Começando na cabeça e movendo-se de um nó para outro, seguindo a referência próxima de cada nó, podemos alcançar a cauda (tail) da lista. Podemos identificar a cauda como o nó que possui **None** como sua próxima referência. Esse processo é comumente conhecido como percorrer a lista encadeada. Como a referência próxima de um nó pode ser vista como um link ou ponteiro para outro nó, o processo de percorrer uma lista também é conhecido como **salto de link** ou **salto de ponteiro**.

Implementando uma Classe de Lista Encadeada Simples

Apresentaremos uma implementação completa de uma classe `SinglyLinkedList`, a partir dos seguintes métodos:

Métodos (1)

- `size()`: Retorna o número de elementos na lista.
- `isEmpty()`: Retorna `true` se a lista estiver vazia, e `false` caso contrário.
- `first()`: Retorna (mas não remove) o primeiro elemento da lista.
- `last()`: Retorna (mas não remove) o último elemento da lista.

Métodos (2)

- `addFirst(e)`: Adiciona um novo elemento ao início da lista.
- `addLast(e)`: Adiciona um novo elemento ao final da lista.
- `removeFirst()`: Remove e retorna o primeiro elemento da lista.

Encadeamento Duplo

Listas duplamente encadeadas

- **Lista Duplamente Ligada:** Cada nó possui referências para o próximo e o anterior.
- **Benefícios:** Permite inserções e exclusões em tempo constante em ambas as extremidades[2].
- **Classe Base:** A classe 'DoublyLinkedBase' serve como uma fundação para implementações mais complexas[2].
- A classe 'Node' é uma classe leve e não pública para armazenar um nó duplamente ligado[2].
  - **Estrutura:**A classe 'DoublyLinkedBase' pode gerenciar uma lista duplamente ligada com nós sentinelas[2]. **Obs:.** A sentinela garante que o elemento procurado faça parte do vetor.
- **Exemplo no notebook!!!**

Listas Duplamente Encadeadas

- Para proporcionar maior simetria, defini-se uma lista encadeada na qual cada nó mantém uma referência explícita para o nó anterior e uma referência para o nó seguinte. Tal estrutura é conhecida como **lista duplamente encadeada**. Essas listas permitem uma maior variedade de operações de atualização em tempo $\mathbf{O(1)}$, incluindo inserções e exclusões em posições arbitrárias dentro da lista. Utiliza-se o termo **"next"** para a referência ao nó que segue outro, e introduz-se o termo **"prev"** para a referência ao nó que o precede.

Implementando uma Classe de Lista Duplamente Encadeada

Apresentaremos uma implementação completa de uma classe `DoublyLinkedList`, a partir dos seguintes métodos públicos:

Métodos (1)

- `size()`: Retorna o número de elementos na lista.
- `isEmpty()`: Retorna `true` se a lista estiver vazia, e `false` caso contrário.
- `first()`: Retorna (mas não remove) o primeiro elemento da lista.
- `last()`: Retorna (mas não remove) o último elemento da lista.

Métodos (2)

- `addFirst(e)`: Adiciona um novo elemento ao início da lista.
- `addLast(e)`: Adiciona um novo elemento ao final da lista.
- `removeFirst()`: Remove e retorna o primeiro elemento da lista.
- `removeLast()`: Remove e retorna o último elemento da lista.

Considerações

Considerações

- **Recordando!!!**
  - Relembrando Notação Big-O!!!.
  - Arrays.
  - Listas eficiêntes em Python
  - Classes uteis em Java.
  - Encadeamento simples.
  - Encadeamento duplo.
  - Exercícios.
  - Figuras obtidas dos livros de referência!