

# **LISTA COMPLETA DE CONCEITOS - Algoritmos e Complexidade**

**Teóricos, Práticos e Exemplos do Dia a Dia**

# PARTE I: CONCEITOS TEÓRICOS FUNDAMENTAIS

## 1. ANÁLISE DE ALGORITMOS

### 1.1 Complexidade de Tempo

**Definição:** Medida de quanto tempo um algoritmo leva para executar em função do tamanho da entrada.

**Notações Assintóticas:**

- Big O (O): Limite superior - "no máximo"
- Big  $\Omega$  ( $\Omega$ ): Limite inferior - "no mínimo"
- Big  $\Theta$  ( $\Theta$ ): Limite exato - "exatamente"

**Exemplo Matemático:**

$$f(n) = 3n^2 + 2n + 1$$

$O(n^2)$  porque  $3n^2$  domina para  $n$  grande

## PARTE II: ESTRUTURAS DE DADOS

### 4. ESTRUTURAS LINEARES

#### 4.1 Arrays (Vetores)

Características:

- Acesso aleatório  $O(1)$
- Inserção/remoção  $O(n)$  no meio
- Memória contígua (cache-friendly)

Código Conceitual:

```
typedef struct {  
    int dados[1000];  
    int tamanho;  
} Array;
```

## PARTE III: ALGORITMOS FUNDAMENTAIS

### 7. ALGORITMOS DE ORDENAÇÃO

#### 7.1 Algoritmos Simples $O(n^2)$

Bubble Sort:

```
void bubble_sort(int arr[], int n) {  
    for (int i = 0; i < n-1; i++) {  
        int trocou = 0;  
        for (int j = 0; j < n-i-1; j++) {  
            if (arr[j] > arr[j+1]) {  
                trocar(&arr[j], &arr[j+1]);  
                trocou = 1;  
            }  
        }  
        if (!trocou) break; // Otimização: já ordenado  
    }  
}
```

# PARTE IV: CONCEITOS PRÁTICOS E APLICAÇÕES

## 10. TÉCNICAS DE IMPLEMENTAÇÃO

### 10.1 Gerenciamento de Memória em C

#### Alocação Estática:

```
int array[1000];           // Stack - liberada automaticamente
static int global[1000]; // Segmento de dados - vida do programa
```

#### Alocação Dinâmica:

```
// Malloc - alocar memória
int* ptr = malloc(n * sizeof(int));
if (ptr == NULL) {
    fprintf(stderr, "Erro de alocação!\n");
    exit(1);
}
```

# PARTE V: APLICAÇÕES DO MUNDO REAL

## 13. APLICAÇÕES EM TECNOLOGIA

### 13.1 Sistemas de Busca (Google, Bing)

#### Problemas:

- Indexar bilhões de páginas web
- Responder consultas em milissegundos
- Ranquear resultados por relevância

#### Estruturas de Dados Utilizadas:

- **Trie:** Para autocompletar buscas
- **Hash Tables:** Indexação de palavras
- **Grafos:** Algoritmo PageRank para relevância

# RESUMO EXECUTIVO - GUIA RÁPIDO

## Escolha Rápida de Estruturas de Dados

Operação Principal	Melhor Estrutura	Complexidade	Quando Usar
Acesso por índice	Array	$O(1)$	Dados de tamanho fixo
Inserção frequente no início	Lista Ligada	$O(1)$	Tamanho variável
Busca frequente	Hash Table	$O(1)$ médio	Chave-valor rápido
Dados sempre ordenados	BST	$O(\log n)$	Busca e inserção ordenada
LIFO (desfazer ações)	Pilha	$O(1)$	Histórico, navegador
FIFO (fila de espera)	Fila	$O(1)$	Processos, impressão

# CONCEITOS PARA ENTREVISTAS

## Perguntas Frequentes

### 1. "Explique Big O Notation"

- Foca no crescimento assintótico
- Ignora constantes e termos menores
- Exemplo:  $3n^2 + 2n + 1 = O(n^2)$

### 2. "Array vs Lista Ligada"

- Array: acesso  $O(1)$ , inserção  $O(n)$
- Lista: acesso  $O(n)$ , inserção  $O(1)$
- Cache locality favorece arrays

### 3. "Como implementar Hash Table?"




# REFERENCIAS E PRÓXIMOS PASSOS

## Bibliografia Recomendada

1. **Cormen, Leiserson, Rivest, Stein** - "Introduction to Algorithms"
2. **Robert Sedgewick** - "Algorithms in C"
3. **Steven Skiena** - "The Algorithm Design Manual"
4. **Jon Bentley** - "Programming Pearls"

## Recursos Online

1. **LeetCode** - Problemas práticos
2. **HackerRank** - Challenges algorítmicos
3. **Coursera/edX** - Cursos de universidades
4. **GitHub** - Implementações open source

 Esta lista completa de conceitos serve como referência abrangente para algoritmos e estruturas de dados, cobrindo teoria, prática e aplicações do mundo real.

*Última atualização: 27 de agosto de 2025*