

2024.1

ALGORITMOS E ESTRUTURAS DE DADOS I

Professora: Dr. Rosana Cibely

BUSCA LINEAR

ALGORITMOS DE BUSCA E ORDENAÇÃO

COMPONENTES

RENATO BENTO

KAYC HENDERSON

O QUE SÃO ALGORITMOS DE BUSCA ?

Os algoritmos de busca são usados para encontrar um item específico em uma coleção de dados.

O funcionamento desses algoritmos é baseado em uma estratégia de busca que pode variar dependendo do tipo de problema a ser resolvido.



ALGORITMOS DE BUSCA LINEAR

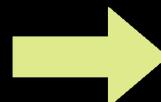
O algoritmo de busca linear é bastante versátil e pode resolver diversos tipos de problemas, especialmente quando os dados não estão ordenados.

Pesquisa em Listas Desordenadas



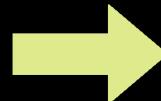
Encontrar um número específico em uma lista de números desordenados.

Verificação de Existência

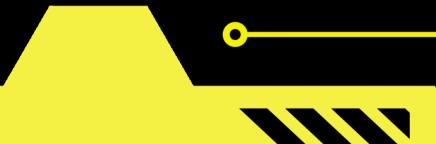


Verificar se um determinado nome está presente em uma lista de convidados.

Busca em Estruturas de Dados Simples



Procurar um valor específico em um array ou lista encadeada.



IMPLEMENTAÇÃO DA BUSCA LINEAR

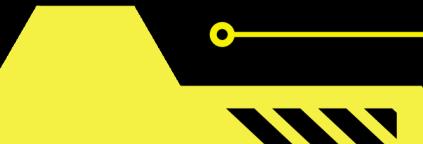
- PROPOSTA DE IMPLEMENTAÇÃO DE ALGORITMO USANDO BUSCA LINEAR

O algoritmo a seguir, feito em C, serve para simular a busca de produtos em um mercado, utilizando uma lista pré-definida de produtos. Ele ilustra como organizar dados de diferentes tipos usando uma estrutura (struct) e como realizar uma busca simples por nome entre esses dados.



produto.h

```
C produto.h > buscarProduto(char *, Produto *)
1   // Define o tamanho máximo do nome do produto
2   #define MAX_NOME 100
3
4   // Define o tamanho máximo do nome do setor
5   #define MAX_SETOR 50
6
7   /**
8    Estrutura que representa um produto no sistema.
9
10   codigo: Código único do produto.
11   nome: Nome do produto (string).
12   preco: Preço do produto (valor em ponto flutuante).
13   setor: Nome do setor onde o produto está localizado no supermercado.
14 */
15  typedef struct {
16      int codigo;
17      char nome[MAX_NOME];
18      float preco;
19      char setor[MAX_SETOR];
20  } Produto;
21
22 /**
23  Função que busca um produto pelo nome.
24
25  Parâmetros:
26  nomeProduto: Nome do produto a ser buscado (string).
27  produtoEncontrado: Ponteiro para a estrutura Produto onde o resultado da busca será armazenado.
28  return: Retorna 1 se o produto for encontrado, 0 caso contrário.
29 */
30 int buscarProduto(char* nomeProduto, Produto* produtoEncontrado);
```



produto.c

```
C produto.c > ⌂ buscarProduto(char *, Produto *)
1  #include <stdio.h>
2  #include <string.h>
3  #include "produto.h"
4
5  #define ARQUIVO_PRODUTOS "produtos.txt"
6
7  int buscarProduto(char* nomeProduto, Produto* produtoEncontrado) {
8      FILE *arquivo = fopen("produtos.txt", "r");
9      if (arquivo == NULL) {
10          printf("Erro ao abrir o arquivo de produtos.\n");
11          return 0;
12      }
13
14      Produto produto;
15      while (fscanf(arquivo, "%d %s %f %s", &produto.codigo, produto.nome, &produto.preco, produto.setor) != EOF) {
16          if (strcmp(produto.nome, nomeProduto) == 0) {
17              *produtoEncontrado = produto;
18              fclose(arquivo);
19              return 1; // Produto encontrado
20          }
21      }
22
23      fclose(arquivo);
24      return 0; // Produto não encontrado
25  }
```



main.c

```
C main.c > ...
1 #include <stdio.h>
2 #include <time.h>
3 #include "produto.h"
4 #include <locale.h>
5
6 int main() {
7     setlocale(LC_ALL, "Portuguese");
8     char nomeProduto[MAX_NOME];
9     Produto produtoEncontrado;
10    clock_t inicio, fim;
11    double tempo_execucao;
12
13    printf("Digite o nome do produto: ");
14    scanf("%s", nomeProduto);
15
16    // Inicia a medição de tempo
17    inicio = clock();
18
19    if (buscarProduto(nomeProduto, &produtoEncontrado)) {
20        printf("+-----+\n");
21        printf("| Produto encontrado: | \n");
22        printf("+-----+\n");
23        printf("| Código: %d\n", produtoEncontrado.codigo);
24        printf("| Nome: %s\n", produtoEncontrado.nome);
25        printf("| Preço: %.2f\n", produtoEncontrado.preco);
26        printf("| Setor: %s\n", produtoEncontrado.setor);
27        printf("+-----+\n");
28    } else {
29        printf("+-----+\n");
30        printf("| Produto não cadastrado. | \n");
31        printf("+-----+\n");
32    }
33
34    // Finaliza a medição de tempo
35    fim = clock();
36
37    // Calcula o tempo de execução em segundos
38    tempo_execucao = ((double) (fim - inicio)) / CLOCKS_PER_SEC;
39    printf("\n Tempo de execução: %.6f segundos\n", tempo_execucao);
40
41    return 0;
42 }
```

produtos.txt

```
C: > Users > danta > Downloads > C produtos.txt
1 1001 Arroz 20.50 Alimentos
2 1002 Feijao 7.90 Alimentos
3 1003 Farinha 4.50 Alimentos
4 1004 Macarrao 5.50 Alimentos
5 1005 oleo 9.50 Alimentos
6 1006 Bolacha 9.50 Alimentos
7 1007 Carne_Bovina 39.99 Carnes
8 1008 Carne_Suina 29.99 Carnes
9 1009 Frango 15.50 Carnes
10 1010 Peixe 25.00 Carnes
11 1011 Yogurte 3.99 Refrigerados
12 1012 Queijo 24.99 Refrigerados
13 1013 Ovos 12.99 Refrigerados
14 1014 Alface 2.99 Hortifruti
15 1015 Tomate 6.50 Hortifruti
16 1016 Cenoura 4.99 Hortifruti
17 1017 Maca 7.99 Frutas
18 1018 Banana 3.99 Frutas
19 1019 Pao 8.99 Padaria
20 1020 Bolo 15.00 Padaria
21 1021 Refrigerante 6.50 Bebidas
22 1022 Detergente 3.99 Limpeza_Doméstica
23 1023 Shampoo 12.99 Cuidados_Pessoais
24 1024 Pasta_de_Dente 7.50 Cuidados_Pessoais
```

EXECUTAR O PROGRAMA



Complexidade

PIOR CASO

O pior caso ocorre quando o elemento procurado está no final da lista ou não está presente, obrigando o algoritmo a verificar todos os elementos da lista. Portanto, a complexidade para o pior caso é $O(n)$.

$$c_1 * 1 + c_2 * 1 + c_3 * 1 + \dots + c_n * 1 = O(n)$$



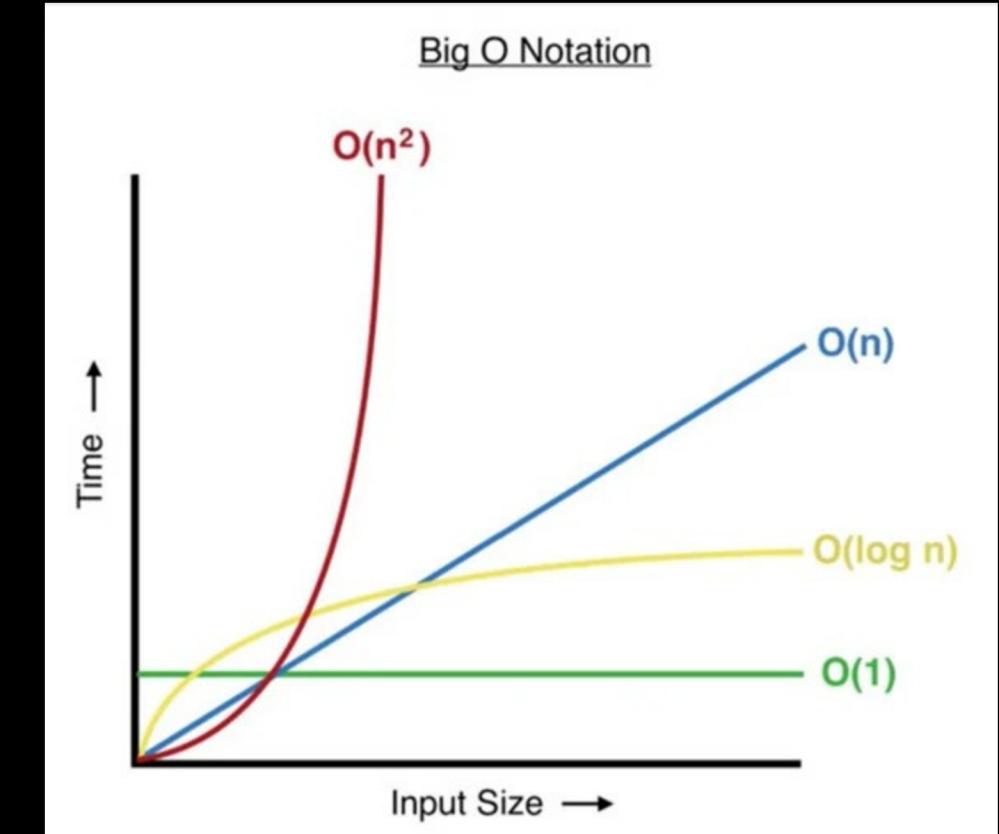
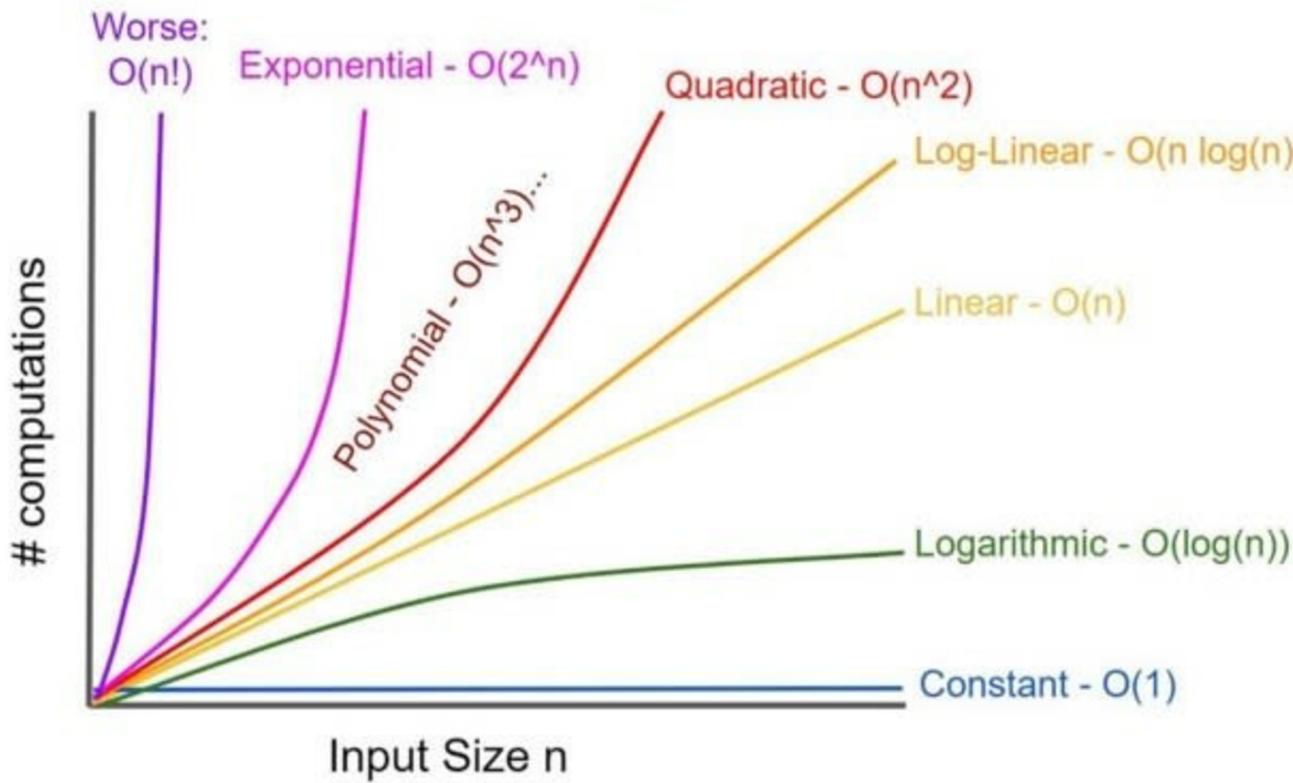
Complexidade

MELHOR CASO

O melhor caso ocorre quando o elemento procurado está no início da lista, e o algoritmo encontra o elemento imediatamente. Portanto, a complexidade para o melhor caso é $O(1)$.

$$c_1 * 1 + c_2 * 1 = O(1)$$





Referências

RDSWEB, O que é: Algoritmo de busca

dio.me, Análise de Algoritmos: Tipos de Busca - Busca Linear -
Wesley Ferraz (13/01/2024)

