



Licenciatura em Informática Aplicada

Nome do Estudante: João de Deus Chichava Júnior e Elcidio Joaquim João

Disciplina: Inteligência Artificial

Tema: Estratégias de Busca em Resolução de Problemas em Inteligência Artificial

Bibliografia:

Silva, A. C. (2014). Algoritmos e Estruturas de Dados: Busca em Largura (BFS). Disponível em: <https://www.inf.ufsc.br/~alexandre.goncalves.silva/courses/14s2/ine5408-1/BFS.pdf>

GeeksforGeeks. Breadth First Search or BFS for a Graph. Disponível em: <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Algoritmos: Teoria e Prática. Editora Campus.

Sedgewick, R., & Wayne, K. (2011). Algorithms. Addison-Wesley Professional.

Skiena, S. S. (2008). The Algorithm Design Manual. Springer Science & Business Media.

Título: Busca em Largura (Breadth-First Search - BFS).

Introdução:

A busca em largura (Breadth-First Search - BFS) é um algoritmo de busca utilizado para explorar ou percorrer grafos. Essa estratégia de busca começa pelo vértice inicial e explora todos os seus vizinhos antes de avançar para os vértices adjacentes. A BFS é amplamente utilizada em problemas que envolvem a exploração de grafos, como encontrar o caminho mais curto entre dois vértices, determinar a conectividade de um grafo e realizar a ordenação topológica.

Descrição da Estratégia:

A busca em largura é implementada utilizando uma estrutura de dados chamada fila. O algoritmo começa inserindo o vértice inicial na fila e marca-o como visitado. Em seguida, repetidamente, até que a fila esteja vazia, o algoritmo realiza as seguintes etapas:

1. Retira o primeiro vértice da fila.
2. Verifica todos os vértices adjacentes não visitados e os insere na fila.
3. Marca os vértices adjacentes como visitados.

Essa estratégia garante que todos os vértices sejam visitados em níveis, ou seja, primeiro os vizinhos directos do vértice inicial, depois os vizinhos dos vizinhos e assim por diante. Isso resulta em uma exploração em largura do grafo, onde o algoritmo visita todos os vértices que estão à mesma distância do vértice inicial antes de prosseguir para os vértices mais distantes.

Exemplo Concreto de Aplicação:

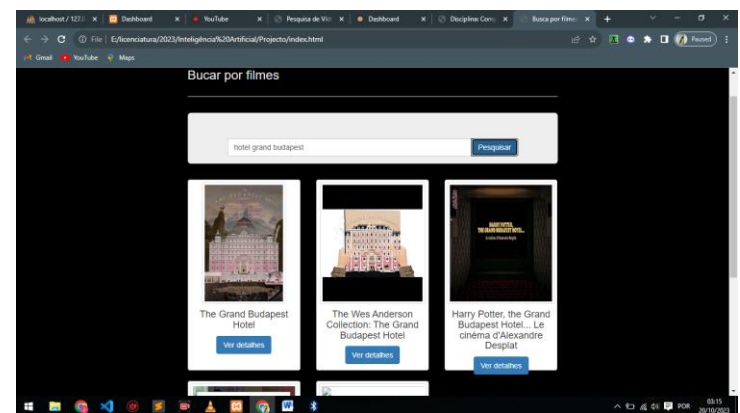
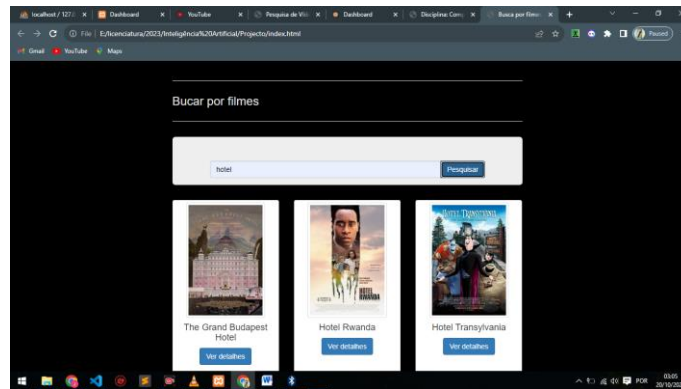
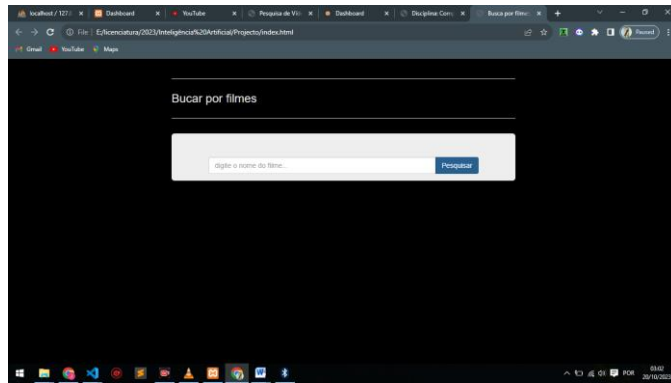
Um exemplo prático de aplicação do algoritmo BFS é a busca pelo caminho mais curto entre dois pontos em um mapa. Suponha que temos um mapa de uma cidade com ruas e intersecções. Podemos representar esse mapa como um grafo, onde as intersecções são os vértices e as ruas são as arestas.

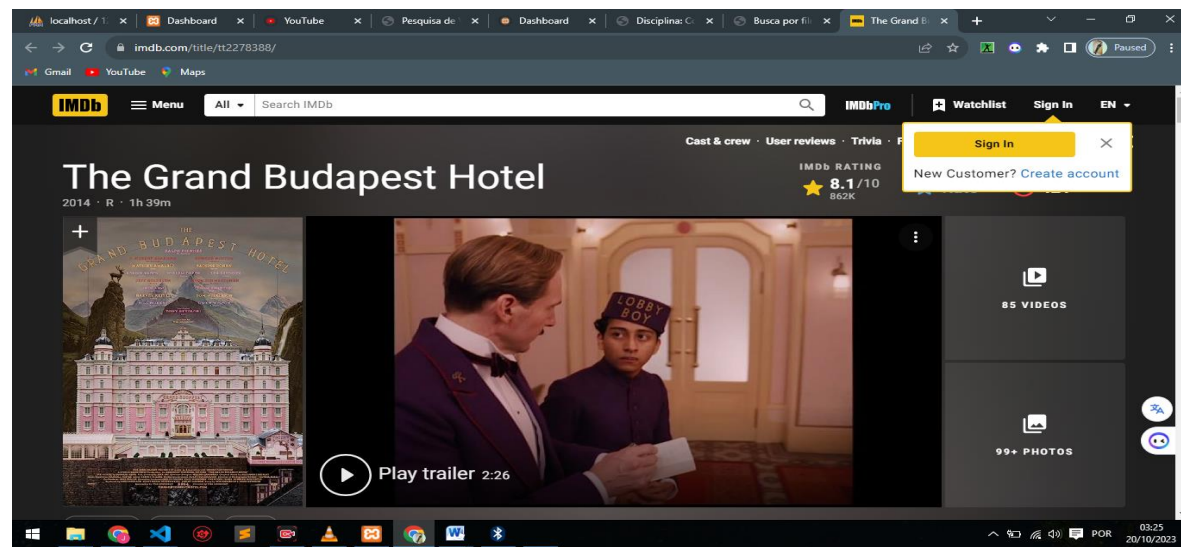
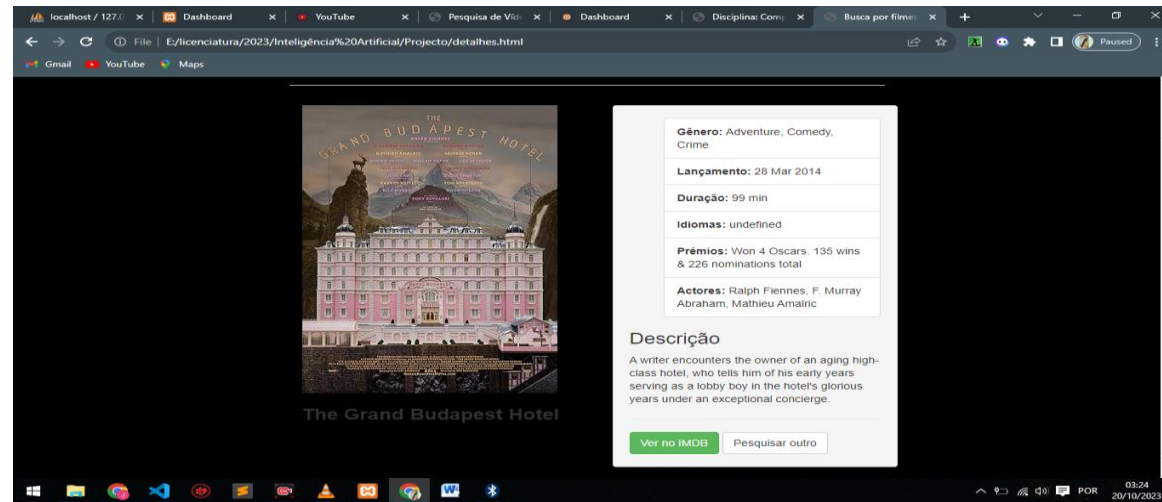
Para encontrar o caminho mais curto entre duas intersecções, podemos utilizar a busca em largura. Começamos pela intersecção inicial e exploramos todas as intersecções vizinhas antes de avançar para as intersecções mais distantes. Ao encontrar a intersecção de destino, paramos a busca e temos o caminho mais curto.

A BFS garante que se existir um caminho entre as duas intersecções, ele será encontrado e será o caminho mais curto possível. Além disso, a BFS também pode ser utilizada para determinar se duas intersecções são alcançáveis uma a partir da outra.

Projecto de busca por Filmes

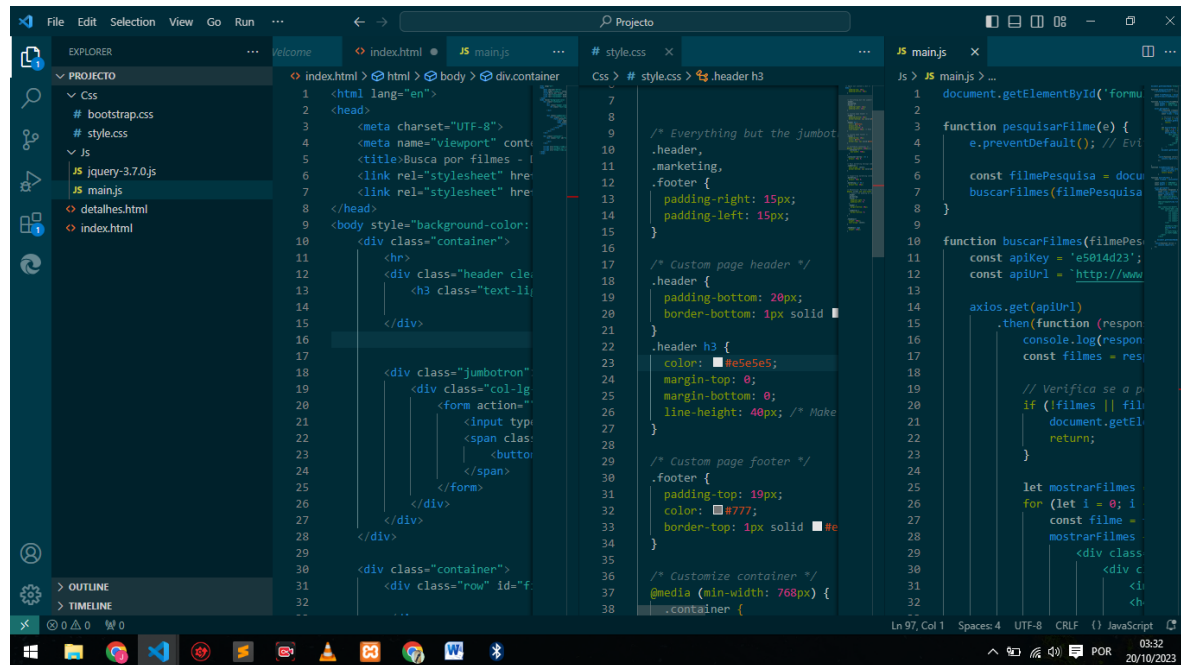
É um sistema de busca por filmes que consiste em o usuário introduzir um nome e clicar no botão pesquisar que sistema faz uma busca de todos filmes de forma ampla relacionados com o que o usuário digitou. Este sistema mostra os filmes buscados com as opções de ver os detalhes do filme e ser redireccionado a um site em que pode ver o trailer. Neste projecto foram usados os princípios da busca em largura (Breadth-First Search - BFS).





No exemplo acima fez uma busca pelo nome hotel e forneceu todos os filmes com a palavra hotel. E vai reduzindo o campo da busca a medida que a especificação do texto e colocada.

Este sistema foi desenvolvido usando HTML, CSS (BOOTSTRAP-como framework) e JAVASCRIPT. O Javascript é consumida uma API através do função `axios.get(URL)`, que faz uma requisição a API e retorna um array de filmes e sua descrição.



```
1 <html lang="en">
2 <head>
3   <meta charset="UTF-8">
4   <meta name="viewport" content="width=device-width, initial-scale=1">
5   <title>Busca por filmes - 1</title>
6   <link rel="stylesheet" href="style.css">
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body style="background-color: #f0f0f0;">
10   <div class="container">
11     <hr>
12     <div class="header clearfix">
13       <h3 class="text-left">Busca por filmes</h3>
14     </div>
15
16     <div class="jumbotron">
17       <div class="col-lg-6">
18         <form action="#">
19           <input type="text" value="Pesquisar por filme">
20           <span class="btn btn-primary">Pesquisar</span>
21         </form>
22       </div>
23     </div>
24
25     <div class="container">
26       <div class="row" id="filmes">
27       </div>
28     </div>
29
30     <div class="container">
31       <div class="row" id="filmes">
32       </div>
33     </div>
34   </div>
35 </body>
36 </html>
```

```
1 /* Everything but the jumbotron is part of the main body
2   * (because it has to scroll within the jumbotron and the
3   * header) - hence this extra rule.
4   */
5 .main-body {
6   padding-top: 20px;
7 }
8
9 /* Custom page header */
10 .header {
11   padding-bottom: 20px;
12   border-bottom: 1px solid #ccc;
13 }
14
15 /* Custom page footer */
16 .footer {
17   padding-top: 20px;
18   padding-bottom: 20px;
19   border-top: 1px solid #ccc;
20 }
21
22 /* Customize container */
23 @media (min-width: 768px) {
24   .container {
25     padding: 0 15px;
26   }
27 }
```

```
1 document.getElementById("form").addEventListener("submit", function(e) {
2   e.preventDefault(); // Evita o envio do formulário
3   const filmePesquisa = document.querySelector("input").value;
4   buscarFilmes(filmePesquisa);
5 });
6
7 function buscarFilmes(filmePesquisa) {
8   const apiKey = 'e5014d73';
9   const apiUrl = `http://www.omdbapi.com/?apikey=${apiKey}&t=${filmePesquisa}&type=movie`;
10
11   axios.get(apiUrl)
12     .then(function(response) {
13       console.log(response);
14       const filmes = response.data;
15
16       // Verifica se a pesquisa foi bem-sucedida
17       if (filmes) {
18         document.getElementById("filmes").innerHTML += `
19           <div class="col-md-6">
20             <div class="card">
21               <img alt="Filme ${filmes.Title}" data-bbox="100 100 200 200"/>
22               <div class="card-body">
23                 <h4>${filmes.Title}</h4>
24                 <p>${filmes.Year}</p>
25                 <p>${filmes.Plot}</p>
26               </div>
27             </div>
28           </div>
29         `;
30       }
31     })
32     .catch(function(error) {
33       console.log(error);
34     });
35 }
```

A busca em largura (Breadth-First Search - BFS), foi aplicada os seus princípios básicos no momento em que o usuário faz a busca e o sistema navega pela API e compila em largura todos os resultados possíveis através do texto e coloca a disposição do usuário. O que podemos verificar a partir da linha 14 do arquivo main.js.

Conclusão:

A busca em largura (Breadth-First Search - BFS) é uma estratégia eficiente para explorar grafos. Sua natureza em largura garante que todos os vértices sejam visitados em níveis, o que pode ser útil em várias aplicações, como encontrar o caminho mais curto entre dois pontos em um mapa ou determinar a conectividade de um grafo. A BFS é amplamente utilizada na área de ciência da computação e é uma ferramenta essencial no arsenal de algoritmos de busca e exploração de grafos.

As estratégias de busca são fundamentais para a resolução de problemas em Inteligência Artificial. Elas permitem que o agente inteligente encontre soluções para problemas complexos, através da exploração de um espaço de estados.