Relatório da Atividade 5: Algoritmos de Busca sem Informação

Este relatório detalha a resolução da Atividade 5 da disciplina de Inteligência Artificial, focada na implementação e comparação de algoritmos de busca em largura (BFS) e busca em profundidade (DFS).

Grupo:

- Rafael Henrique
- Hugo Ryan
- Caio Vitor

Questão 1: Busca em Largura (BFS) na Romênia

Objetivo: Encontrar uma rota entre duas cidades da Romênia usando BFS.

Cidades Escolhidas:

- **Origem:** Arad
- **Destino:** Bucharest

A busca em largura explora o grafo em camadas, garantindo que o caminho mais curto em número de arestas seja encontrado primeiro.

Passo a Passo da Busca (Resumido):

1. **Início:** Fronteira = [Arad], Visitados = {Arad}
2. **Expande Arad:** Fronteira = [Zerind, Sibiu, Timisoara]
3. **Expande Zerind:** Fronteira = [Sibiu, Timisoara, Oradea]
4. **Expande Sibiu:** Fronteira = [Timisoara, Oradea, Fagaras, Rimnicu Vilcea]
5. ... a busca continua expandindo nó por nó na ordem em que foram adicionados à fronteira.
6. **Fim:** A busca encontra Bucharest através do caminho vindo de Fagaras ou Pitesti.

Resultado Final (BFS):

- **Caminho Encontrado:** ['Arad', 'Sibiu', 'Rimnicu Vilcea', 'Pitesti', 'Bucharest']
- **Custo Total:** 418

Questão 2: Implementação da Busca em Profundidade (DFS)

A implementação da Busca em Profundidade foi criada a partir da classe BuscaLargura. A única alteração estrutural necessária foi na forma como os nós são removidos da fronteira.

- **Busca em Largura (Fila - FIFO):** Usa fronteira.pop(0), removendo o elemento mais antigo.
- **Busca em Profundidade (Pilha - LIFO):** Usa fronteira.pop(), removendo o elemento mais recente.

Essa simples mudança faz com que o algoritmo explore um ramo do grafo o mais fundo possível antes de retroceder (backtracking).

Questão 3 e 4: DFS na Romênia e Comparação

Objetivo: Usar a implementação de DFS para encontrar uma rota entre **Arad** e **Bucharest** (mesmo par da Questão 1, conforme a correção do professor) e comparar os resultados.

Passo a Passo da Busca (Resumido):

1. **Início:** Fronteira = [Arad]
2. **Expande Arad:** Fronteira = [Zerind, Sibiu, Timisoara] (Timisoara é o último a entrar, primeiro a sair)
3. **Expande Timisoara:** Fronteira = [Zerind, Sibiu, Lugoj]
4. **Expande Lugoj:** Fronteira = [Zerind, Sibiu, Mehadia]
5. ... a busca continua descendo pelo ramo de Timisoara -> Lugoj -> Mehadia -> Drobeta -> Craiova, etc.

Resultado Final (DFS):

- **Caminho Encontrado:** ['Arad', 'Sibiu', 'Fagaras', 'Bucharest']
- **Custo Total:** 450

Comparativo: BFS vs. DFS (Arad ⟶ Bucharest)

Critério	**Busca em Largura (BFS)**	**Busca em Profundidade (DFS)**
Caminho	Arad ⟶ Sibiu ⟶ R. Vilcea ⟶ Pitesti ⟶ Bucharest	Arad ⟶ Sibiu ⟶ Fagaras ⟶ Bucharest
Custo Total	**418** (Ótimo em custo)	450 (Não ótimo em custo)
Nós no Caminho	5	**4** (Ótimo em nº de paradas)
Estratégia	Explora em camadas, mais lento mas garante otimalidade (se custos forem uniformes).	Explora um caminho até o fim, mais rápido para encontrar *uma* solução, mas não garante ser a melhor.

Memória	Geralmente consome mais memória, pois armazena todos os nós de uma camada.	Mais eficiente em memória, pois só armazena o caminho atual.

Questão 5: Novo Problema - Os Jarros de Água

Objetivo: Medir exatamente 4 litros de água usando apenas jarros de 3 e 5 litros.

Modelagem do Problema:

- **Estado:** Uma tupla (j3, j5) representando a quantidade de água em cada jarro.
- **Estado Inicial:** (0, 0)
- **Estado Objetivo:** Qualquer estado onde j3 = 4 ou j5 = 4. Como o jarro de 3L não pode ter 4L, o objetivo é (x, 4).
- **Ações:** Encher, esvaziar, ou despejar a água de um jarro para o outro.

Resultados para o Problema dos Jarros

Busca em Largura (BFS):

- **Resultado:** Encontrou a solução ótima (mais curta).
- **Solução:** (0,0) → (0,5) → (3,2) → (0,2) → (2,0) → (2,5) → (3,4)
- **Número de Passos:** 6

Busca em Profundidade (DFS):

- **Resultado:** Encontrou uma solução, mas muito mais longa.
- **Solução:** A DFS encontrou um caminho com 16 passos, explorando um ramo muito profundo antes de encontrar a solução.
- **Número de Passos:** 16

Conclusão: Para problemas onde o caminho mais curto é desejado, como o dos jarros, a BFS é a abordagem mais adequada, pois garante a otimalidade. A DFS pode ser útil quando qualquer solução é aceitável e a memória é uma restrição.