UNIVERSIDADE FEDERAL DO TOCANTINS CAMPUS DE PALMAS
COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO 109
Norte, Avenida NS 15, ALCNO 14, Bloco III | 77402-970 | Palmas/TO
(63) 3229-4527 | http://palmas.uft.edu.br/grad/ccomp |
comppalmas@mail.uft.edu.br

Aluno: Eduardo Henrique, Samara, Shayla
Disciplina: Inteligência Artificial
Profa: Dra. Glenda Botelho
Algoritmo: Busca em Profundidade (Depth-First Search - DFS)
Entrega: 08/05/2025

Introdução

O algoritmo de Busca em Profundidade (DFS) é um dos métodos clássicos utilizados em Inteligência Artificial para a exploração de grafos. Este projeto visa a implementação de uma solução para encontrar caminhos entre dois nós em um grafo, aplicando o DFS de forma recursiva. Além disso, o projeto inclui a visualização dos grafos e do caminho encontrado para fornecer uma maneira intuitiva de entender a execução do algoritmo.
O objetivo principal é demonstrar como o DFS pode ser utilizado para explorar diferentes tipos de grafos, incluindo grafos simples, com ciclos, com múltiplos caminhos e disconexos. A implementação do algoritmo tem como foco a busca eficiente e a visualização clara do processo de exploração.

Conceitos Importantes

Busca em Profundidade (DFS):
A Busca em Profundidade (Depth-First Search - DFS) é um algoritmo de busca utilizado para explorar grafos e árvores. Ele funciona de forma recursiva, explorando todos os nós filhos de um nó antes de retornar e explorar outros caminhos. O algoritmo segue uma estratégia LIFO (último a entrar, primeiro a sair), utilizando uma pilha para manter o controle dos nós a serem explorados.

Principais características do DFS:

- **Estratégia de exploração:** O algoritmo vai o mais fundo possível em um caminho antes de voltar e explorar outros caminhos.
- **Memória:** O DFS utiliza memória proporcional à profundidade do grafo, o que pode ser eficiente em termos de uso de memória, mas pode causar problemas em grafos muito profundos.
- **Eficiência:** O DFS não garante o caminho mais curto, mas é eficiente para explorar grafos onde encontrar qualquer caminho já é suficiente.
- **Ciclo:** O algoritmo pode entrar em ciclos infinitos se não for implementada uma verificação para nós visitados.

Grafos:
Um grafo é uma estrutura de dados composta por nós (ou vértices) e arestas que os conectam. É utilizado para modelar uma ampla gama de problemas, como redes sociais, roteamento de tráfego, e busca em jogos.
No projeto, os grafos são representados como dicionários de listas de adjacência, onde cada chave é um nó e os valores são os nós adjacentes a ele. Este tipo de representação é simples e eficiente para algoritmos de busca como o DFS.

Metodologia

Estruturação dos Grafos:
Os grafos utilizados neste projeto foram definidos diretamente no arquivo grafos.py, onde cada grafo é representado como um dicionário. Exemplos de grafos incluem:

- **Grafo Simples:** Um grafo com poucas conexões entre os nós.
- **Grafo com Ciclo:** Um grafo onde um ciclo é formado entre os nós, o que testa a capacidade do DFS de lidar com loops.
- **Grafo Disconexo:** Um grafo onde dois subgrafos não estão conectados, testando se o DFS reconhece a impossibilidade de encontrar um caminho.
- **Árvore Binária:** Uma árvore binária simples para testar a exploração hierárquica.

Implementação do Algoritmo:
O algoritmo DFS foi implementado de forma recursiva na função busca_profundidade(), que percorre o grafo começando do nó inicial até encontrar o nó objetivo ou esgotar todas as possibilidades. A cada nó visitado, ele é marcado como "visitado" para evitar ciclos e garantir que o algoritmo não fique preso em loops infinitos.
A função desenhar_grafo() foi implementada para visualização do grafo. Usando o networkx, os grafos são desenhados e exibidos ao usuário para que ele compreenda facilmente a estrutura e a execução do DFS.

Visualização:
A visualização foi implementada para mostrar a estrutura dos grafos antes e depois da execução do DFS. Utilizamos o layout graphviz_layout para representar os grafos de forma hierárquica e intuitiva, com o nó inicial no topo e os caminhos de exploração claramente visíveis.

Experimentos Realizados

Foram realizados diversos experimentos para validar a eficácia do algoritmo, considerando grafos com diferentes estruturas e características:

1. **Grafo Simples:**
   - O algoritmo foi capaz de encontrar um caminho simples entre o nó inicial e o objetivo.
   - Resultado esperado: Caminho direto, sem complicações.
2. **Grafo com Ciclo:**
   - Teste para garantir que o DFS não entrasse em loops infinitos. Implementamos um conjunto de "nós visitados" para impedir a visita a um nó mais de uma vez.
   - Resultado esperado: O DFS corretamente evitou ciclos e completou a busca.
3. **Grafo Disconexo:**
   - Teste para verificar se o DFS reconhece que não há caminho entre nós desconexos.
   - Resultado esperado: O algoritmo corretamente retornou a mensagem de que não havia caminho disponível entre os nós escolhidos.
4. **Árvore Binária:**
   - Teste para verificar a exploração hierárquica e a capacidade do DFS de navegar pela árvore de maneira eficiente.
   - Resultado esperado: O DFS explorou todos os nós, respeitando a hierarquia da árvore binária.

Resultados

Os resultados mostraram que o algoritmo de Busca em Profundidade (DFS) foi eficaz para explorar grafos com diferentes estruturas. Em todos os testes realizados, o algoritmo:

- Encontrou os caminhos entre os nós quando possível;
- Evitou ciclos infinitos em grafos com ciclos;
- Reconheceu quando não havia caminho entre nós desconexos.

A visualização do grafo e do caminho encontrado ajudou a entender o funcionamento do DFS e a validar sua eficácia em diferentes cenários.

Conclusão

A Busca em Profundidade (DFS) é uma técnica eficiente para explorar grafos e encontrar caminhos, especialmente em problemas onde a memória é uma preocupação importante. Embora o DFS não garanta o caminho mais curto, ele é uma abordagem útil em muitos problemas de IA, como quebra-cabeças, jogos de tabuleiro e planejamento. Este projeto demonstrou a implementação do DFS, a construção e exploração de diferentes grafos, e a visualização do processo de busca. O algoritmo foi eficaz em todos os cenários testados, e a visualização foi crucial para entender o funcionamento do algoritmo de forma intuitiva.