Simulador do Quebra-Cabeça 8-Puzzle

Gabriela Christani Pires (RA: 24001636)
Giulia Monteiro Garrido (RA: 24010281)

Curso: Ciência de Dados e Inteligência Artificial
Disciplinas: Programação e Estrutura de Dados (Prática), Inteligência Artificial
Data de Entrega: 04/12/2024

Introdução

O problema do 8-puzzle é um quebra-cabeça clássico no qual peças numeradas de 1 a 8 são dispostas em uma grade 3×3, com uma única posição vazia representada pelo número 0. O objetivo é reorganizar as peças até atingir uma configuração final onde os números estejam em ordem crescente, com o espaço vazio no canto inferior direito.

O simulador desenvolvido apresenta duas funcionalidades principais: permitir que o usuário resolva o quebra-cabeça manualmente ou visualizar a solução automática gerada pelo sistema. Para isso, foram implementados dois algoritmos de busca distintos: Busca em Largura (BFS) e Busca em Profundidade Limitada Iterativa (IDDFS). Esses métodos foram escolhidos por suas diferentes abordagens para explorar o espaço de estados do problema, demonstrando vantagens e desvantagens em diferentes contextos.

O objetivo deste projeto foi proporcionar uma experiência prática para a aplicação de conceitos teóricos estudados na disciplina, como estruturas de dados dinâmicas e algoritmos de busca. Além disso, o desenvolvimento do simulador envolveu desafios como gerenciamento de memória, controle de estados visitados e criação de uma interface amigável para o usuário.

Desenvolvimento do Projeto

Para representar o estado do quebra-cabeça, foi utilizada uma matriz 3×3, onde cada célula armazena um valor correspondente a uma peça ou o espaço vazio. Essa matriz permitiu a manipulação direta dos estados do tabuleiro, além de possibilitar a verificação de condições, como a detecção de soluções e a geração de estados vizinhos através de movimentos válidos (cima, baixo, esquerda e direita).

Estruturas dinâmicas desempenharam um papel essencial na manipulação e exploração dos estados durante a execução dos algoritmos de busca. A lista encadeada

foi utilizada como base para a implementação de filas e pilhas, proporcionando flexibilidade e eficiência no armazenamento de estados.

A fila foi empregada no algoritmo BFS para garantir a exploração dos estados em ordem de inserção (FIFO). Este método garantiu que todos os estados em uma profundidade fossem explorados antes de passar para a próxima, resultando em uma solução ótima em termos de movimentos. Já a pilha foi utilizada no IDDFS, explorando estados em profundidade crescente até um limite iterativo, o que permitiu equilibrar a eficiência em memória com a segurança de evitar ciclos.

Os algoritmos de busca foram implementados de maneira iterativa, como exigido no projeto, evitando o uso de recursão. A lógica central, comum a todos os métodos, consiste em adicionar um estado inicial à estrutura apropriada (fila ou pilha) e processá-lo até encontrar a solução ou esgotar os estados possíveis. Em cada passo, o estado é removido da estrutura, avaliado para verificar se é a solução, e, caso não seja, gera novos estados adjacentes que são adicionados à estrutura.

Desafios e Soluções

O desenvolvimento do simulador apresentou diversos desafios técnicos, que foram superados com estratégias específicas:

Gerenciamento de Memória Dinâmica

A manipulação de estados dinâmicos exigiu um gerenciamento cuidadoso de memória para evitar vazamentos. Foram implementadas funções específicas para alocar e desalocar memória de forma eficiente, garantindo que cada estado criado fosse devidamente descartado após seu uso.

Controle de Estados Visitados

Evitar a reexploração de estados já processados foi essencial para a eficiência dos algoritmos. Para isso, foi implementada uma estrutura auxiliar que armazenava os estados visitados, permitindo comparações rápidas para verificar duplicatas.

Interface Amigável

Tornar o simulador intuitivo e agradável para o usuário foi um objetivo importante. Para isso, utilizamos cores para realçar os movimentos e destacar soluções, além de mensagens claras que orientavam o jogador durante cada etapa do jogo.

Funcionalidades Extras

O simulador incluiu funcionalidades que vão além dos requisitos básicos, como:

- Exibição de estatísticas, como o número de estados visitados.
- Interface visual melhorada, com destaque para interações intuitivas.

Conclusão

O projeto cumpriu todos os objetivos estabelecidos, implementando de maneira eficiente um simulador para o 8-puzzle com algoritmos de busca otimizados. O desenvolvimento proporcionou uma experiência prática no uso de estruturas de dados e algoritmos, além de superar desafios como gerenciamento de memória e interface de usuário.

Referências Bibliográficas

- Documentação oficial do C - `https://en.cppreference.com/`, acessada em 23/11/2024.
- "Introduction to Algorithms" - Thomas H. Cormen et al., 3ª edição, MIT Press.
- Material fornecido na disciplina de Programação e Estrutura de Dados.
- Material fornecido na disciplina de IA.