UNIVERSIDADE FEDERAL DO CEARÁ

INTELIGÊNCIA ARTIFICIAL

Jose Wellington Franco Da Silva

Aluno: Luiz Zairo Bastos Viana - 499995

Aluno: Marlon Gonçalves Duarte - 493408

Crateús - CE

2022

INTRODUÇÃO

Um problema de busca pode ser definido como um conjunto de estados, cujos quais, constam de um estado inicial (trata-se do primeiro estado da árvore), um estado objetivo (o estado que se busca alcançar), uma função sucessora (função responsável por transitar entre os estados da árvore) e uma função de verificação (responsável por verificar se chegou no objetivo). Existem vários métodos que podem ser utilizados para realizar buscas em estruturas de dados.

Um puzzle deslizante, é um quebra-cabeça de blocos deslizantes que desafia um jogador a deslizar peças (frequentemente planas) ao longo de certas rotas (geralmente em um tabuleiro) para estabelecer uma determinada configuração final que pode ser uma imagem ou uma sequência numérica. As peças do jogo podem ser facilmente abstraídas para o universo computacional e assim buscarmos soluções mais rápidas e efetivas para o mesmo.

No trabalho corrente, iremos abordar a busca em profundidade, largura, A* e busca gulosa para solucionar determinados estados do jogo. Esses métodos de busca supracitados podem ser utilizados para resolver variados problemas de busca.

DESCRIÇÃO DO PROBLEMA

O Puzzle que o trabalho propõe é o jogo do 15, trata-se de um quebra-cabeças de quinze peças, gravados com números. Tendo como objetivo, ordenar os quadrados, da esquerda para a direita e de cima para baixo, isto é, obter a disposição original dos contadores depois de terem sido aleatoriamente deslocados.

Nos foi dado, duas configurações do jogo e por meio dessas duas configurações, teremos que chegar a um objetivo específico, utilizando os algoritmos de busca solicitados:

Configuração 01

1	2	3	4
5	6	8	12
13	9		7
14	11	10	15

Configuração 02

1	2	3	4
13	6	8	12
5	9		7
14	11	10	15

Objetivo

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Estratégia de Busca

As estratégias de busca podem ser divididas em: busca cega e busca heurística. Uma busca é tida como cega, se ela não leva em conta informações específicas sobre o problema a ser resolvido. Já a busca heurística, por sua vez, possui uma informação que é repassada junto aos dados do problema e realiza a quantificação de proximidade a um determinado objetivo.

a) Busca em profundidade

É um algoritmo utilizado para percorrer ou buscar itens dentro das estruturas de dados, grafos ou árvores. Sua característica básica é percorrer todos os nós filhos ao nó raiz o mais profundo possível para somente depois retroceder.

b) Busca em largura (ou em amplitude, ou em extensão)

A busca em largura é um algoritmo utilizado para percorrer ou buscar itens dentro das estruturas de dados, grafos ou árvores. Como característica temos que a busca sempre ocorre nos filhos ou nos mais próximos ao nó pelo qual a busca foi iniciada.

c) Busca heurística pelo melhor primeiro (gulosa)

Busca gulosa pela melhor escolha expande o nó que parece mais próximo ao objetivo de acordo com a função heurística.Função de avaliação f(n) = h (n) (heurística) = estimativa do custo de n até o objetivo ex., hDLR (n) = distância em linha reta de n até Bucareste.

d) Busca A*

No caso da busca A*, a função de avaliação é F(n) = h(n) + g(n), onde h é uma função heurística admissível. A "estrela", muitas vezes denotado por um asterisco, refere-se ao fato de que A* usa uma função heurística admissível, o que significa que A* é ótima, ou seja, sempre encontra um caminho ótimo entre o nó inicial e o nó de objetivo.

IMPLEMENTAÇÃO

A linguagem utilizada com o objetivo de solucionar o problema do Puzzle foi o Python, uma linguagem simples e que tem uma curva de aprendizado elevadíssima. A escolha dessa linguagem se deu a um agregado de fatores, tais como:

a) Conhecimento e familiaridade de ambos os membros da equipe, com a linguagem.
b) Ser uma linguagem que possui uma maior variedade de tutoriais na internet.
c) Ser uma linguagem com vasto número de bibliotecas de fácil acesso.

Quanto às estruturas de dados, todas solicitados no trabalho foram atendidas: Busca em largura, profundidade, gulosa e A*. Quanto às suas performances, foi utilizado um script

do linux que calcula o tempo de execução dos scripts rodados no terminal. Também foi criado um módulo que é responsável por mensurar informações de número de nós até chegar ao objetivo, questões de profundidade e custo.

Para melhor manutenibilidade do código, realizamos uma fatoração das estruturas necessárias ao funcionamento de forma que cada estrutura do código ficou dividida da seguinte forma:

- **Class.py** - Fica responsável por administrar a estrutura do tabuleiro bem como regulamentar os movimentos das peças;
- **Tabuleiro.txt -** Esse arquivo foi criado simplesmente para organizarmos e visualizar da melhor forma como fica o tabuleiro, quando inserirmos os valores no input de inicialização do programa.
- **Buscas.py -** Por sua vez, esse arquivo possui todas as funções necessárias das buscas que foram utilizadas.
- **Puzzle.py -** Por fim, o puzzle é o arquivo que deve ser executado já que ele importa ambos os arquivos anteriores e também é por meio dele que o usuário deve interagir.

Resultados - Configuração 01

Estratégia	Tempo	Espaço	Encontrou a Solução?	Profundidade / Custo
DFS	33.41s	675027 nós	sim	18 movimentos
BFS	3.9s	78259 nós	sim	12 movimentos
GULOSA	+60s	-	não	-
A*	0.08s	135 nós	sim	12 movimentos

Configuração 02

Estratégia	Tempo	Espaço	Encontrou a Solução?	Profundidade / Custo
DFS	+60s	-	não	-
BFS	+60s	-	não	-
GULOSA	+60s	-	não	-
A*	+60s	-	não	-

CONCLUSÃO

A utilização das estruturas de dados que contam com pelo menos uma heurística se apresentaram de forma a possuírem muito mais eficiência em termos de tempo de execução do que as ditas buscas cegas. Por vezes nos deparamos com a situação em que o código criou tantas possibilidades de solução que chegou até a nem exibir o resultado, finalizar ou até mesmo a sinalizar que o tabuleiro era inválido, mesmo que o mesmo possuísse todas as características válidas do jogo.

O algorítmo que mais se destacou, segundo nosso entendimento, foi o A* que finaliza o processo em menor espaço de tempo desde que sejam usadas as heurísticas corretas para o mesmo. O algoritmo Guloso, apesar de contar com uma heurística, ainda é bem inferior ao A*. A surpresa ficou por conta do bfs, que em alguns casos encontrou o resultado em tempo considerado aceitável para fins práticos.