DEPARTAMENTO
DE CIÊNCIA DE
COMPUTADORES

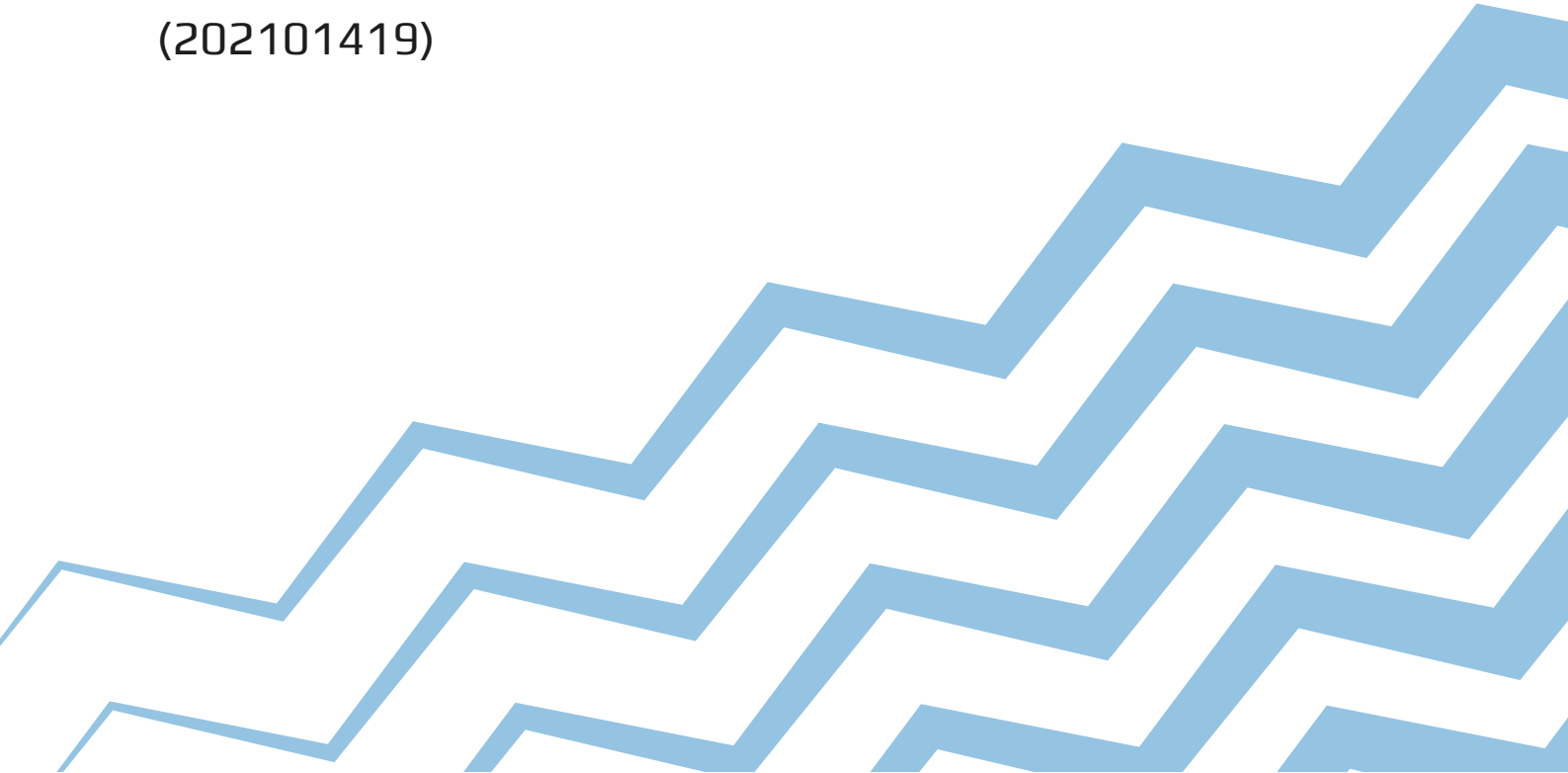
MARÇO DE 2023

Relatório de Inteligência Artificial

Trabalho Prático n.º 1

REALIZADO POR

Matheus Bezerra
(202101391)
Gustavo Schiavo
(202101419)



Sumário

Objetivos

O objetivo deste relatório é apresentar e discorrer sobre algoritmos capazes de encontrar soluções para o problema do jogo dos 15.

Introdução

Problemas de busca/procura é quando temos um estado inicial de alguma situação e temos que, a partir de um estado inicial, encontrar um estado final/objetivo por meio de um caminho.

Os métodos utilizados para resolver os problemas de procura serão pesquisa em profundidade(BFS), pesquisa em largura (DFS), iterativa em profundidade (IDFS), Gulosa e o caminho A*.

Descrição do Problema de Procura

O objetivo será transformar uma matriz com o formato 4x4 com 15 números (de 1 a 15) e um espaço vazio e o objetivo, como o seguinte exemplo:

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

A única ação possível é mover os números adjacentes(em cima, em baixo, esquerda ou direita) para o espaço vazio sucessivamente, com o objetivo de chegar na seguinte posição final (padrão):

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

Porém, neste jogo, tem de ser levado em consideração que há a possibilidade do resultado final não ser possível. O jogo é possível de ser solucionado atendendo alguns requisitos, para isso, é preciso calcular o número de inversões (quantidade de números menores após cada número da matriz) dos estados finais e iniciais, e fazer a seguintes operações booleanas:

CondI = ((InvI % 2 == 0) == (blankRowI % 2 == 1))

CondF = ((InvF % 2 == 0) == (blankRowF % 2 == 1))

CondI == CondF

InvI = soma de inversões do estado inicial
blankRowI = número da coluna do estado inicial
CondI = condição inicial (True or False)
InvF = soma de inversões do estado final
blankRowF = número da coluna do estado final
CondF = condição final (True or False)

Então, para o jogo ser solucionável, é preciso que CondI e CondF sejam ambos True ou ambos False.

Após a verificação da solubilidade da tabela, a busca pelo objetivo do jogo inicia-se, então ocorre as mudanças de posições para que seja encontrado o resultado final.

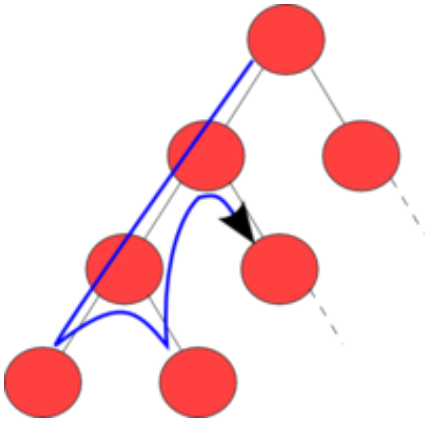
Estratégias de Procura

Nesta seção, será abordado quais são e como funcionam as estratégias de procura para encontrar soluções, evidenciando como funciona, onde se aplica e a complexidade temporal e espacial de cada estratégia apresentada.

As estratégias são divididas em duas classes, a primeira é a procura não guiada, que corresponde às estratégias de procura em profundidade, largura e iterativa limitada em profundidade, estas técnicas não possuem critérios para buscar otimizar a busca; todavia, a outra classe de algoritmos que são a gulosa e a A*, possuem um critério chamado heurística, que visa orientar o algoritmo para otimizá-lo e assim, encontrar a solução mais rapidamente.

- **Procura em Profundidade**

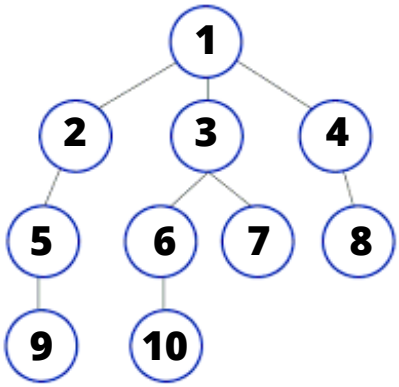
Procura em profundidade (DFS - Depth-First Search) é um algoritmo de busca que progride para a primeira descendência adjacente e assim consecutivamente, até que encontre o resultado final e retorne-o ou não encontre descendentes a partir deste, neste caso, a busca retorna para o descendente anterior (backtrack) e recomeça a busca a partir do descendente seguinte.



Este tipo de busca só deve ser usado para situações em que a árvore de procura não seja muito profunda. A complexidade temporal é O(b^d), em que 'b' é a média de ramificações e 'd' é a profundidade da busca. A complexidade espacial é O(b), onde 'b' representa a profundidade máxima da busca.

- **Procura em Largura**

A procura em largura (BFS - Breadth-First Search) é uma estratégia de busca não informada em que há a verificação de todos os descendentes em cada nível de profundidade antes de ir ao próximo nível de profundidade.



Da mesma forma que na busca em profundidade, o maior problema da busca e largura o maior problema é que se o resultado esperado estiver e algum grau muito elevado, será exigido uma altíssima quantidade de tempo e memória, por exemplo, se o jogo tiver resultado no nível 12 e houver 4 instâncias, em média, será necessário ter mais de 16 milhões de estados criados e armazenados. Tanto a complexidade temporal e complexidade espacial é O(b^d), onde 'b' é o fator de ramificação e 'd' é a profundidade da solução.

- **Procura Iterativa Limitada em Profundidade**

Busca iterativa limitada em profundidade utiliza os melhores fatores da busca em profundidade e busca em largura. Em resumo, é um algoritmo que faz a busca em profundidade, porém possui um determinado limite para o nível de busca, inicialmente o limite sendo 1, e o limite é incrementado cada vez que ocorre a busca e não é achada a solução, e então ocorre novamente a busca com o limite sendo o próximo nível. É uma busca ótima e completa.

Esta estratégia é ideal quando a árvore de procura é muito grande e o nível onde a solução está é desconhecido. A complexidade temporal é O(b^l), onde 'b' é o fator de ramificação e o 'l' é o limite de profundidade, já a complexidade espacial é O(b x l).

- **O que é uma heurística?**

A palavra heurística provém do grego antigo, que possui a pronúncia "heurísko" que em tradução simples significa "eu acho", este termo também originou o termo "heureca".

Heurística significa algum fator/cálculo que, a partir dele, oriente na tomada de decisões e encontre caminhos de sucessão que consigam aproximar-se do resultado esperado com maior facilidade.

A heurística deve ser, preferencialmente, a melhor possível, pois se for utilizado uma heurística de má qualidade, o resultado pode nunca ser encontrado.

- **Procura Gulosa**

A busca gulosa é uma estratégia que visa selecionar apenas o descendente com a melhor heurística (descendente ideal), e assim sucessivamente.

Este algoritmo é interessante quando não há interesse de ter o resultado ótimo, pois não necessariamente utilizará ao melhor caminho possível. A complexidade temporal e espacial são ambas O(b^m), onde 'b' é o fator de ramificação e 'm' é o nível de profundidade.

- **Procura A***

O algoritmo A* tem por objetivo minimizar o custo do caminho, possui uma lista de pontos visitados e outra lista ordenada dos melhores candidatos, por meio da heurística, para encontrar o caminho ideal. A partir do nó inicial, ocorre a verificação se é o objetivo e, se não for, expandir os descendentes, adicioná-los na lista de candidatos e ordená-los pelo melhor heurística, e após isso ocorre uma nova verificação do melhor candidato e repete-se este ciclo até encontrar o resultado.

A heurística utilizada é o somatório de peças fora do lugar.

Descrição da Implementação

A linguagem de programação que será utilizada o python3 por conta de sua facilidade de implementação.

Foi utilizado pilhas e fila como estruturas de dados.

Todo o código está implementado em apenas um ficheiro.

Resultados

Estratégia	Tempo	Espaço	Solução?	Profundidade
DFS	-	-	Não	-
BFS	6.4s	13309	Sim	-
IDFS	1.5s	531441	Sim	12
Gulosa	-	-	Não	-
A*	0.003s	16	Sim	15

Conclusões

Mesmo não conseguindo alcançar os resultados na busca em profundidade(recursion limit) e na busca gulosa(looping), fica evidenciado que a busca orientada, neste caso, o A*, é multiplas vezes mais eficiente em termos de memória e espaço.

Referências

- MATERIAL DE ESTUDO DO MOODLE
- https://pt.slideshare.net/RafaelCoelhoSilva1/busca-em-largura-breadth-first-search
- w3schools.com
- stackoverflow.com
- wikipedia.com