Trabalho Prático 1 de Introdução à Inteligência Artificial

Julia Tiemi Alkmim Morishita e Marcelo Luiz Harry Diniz Lemos

1. Introdução

Esse trabalho consiste em ajudar Márcio, um administrador de sistemas a programar um AGV - *Automated Guided Vehicle* - que percorrerá sua fábrica e coletará a produção. Cabe ao programa definir as rotas do veículo a partir de um mapa estático para reduzir custos. O trabalho consiste na implementação e comparação de quatro algoritmos de busca para encontrar o menor caminho entre o ponto inicial de cada instalação até o ponto de coleta: BFS (Busca em Largura), DFS (Busca em Profundidade), IDS (Busca com Aprofundamento Iterativo) e A*.

2. Modelagem

A solução da tarefa envolve um agente que resolve o problema através de busca, chamado de *Problem-Solving Agent*. O objetivo do agente é um *goal*, ou seja um conjunto de estados, e ele busca uma sequência de ações para atingí-lo a partir do seu estado inicial.
Como visto na descrição do problema, o ambiente que o agente irá trabalhar é estático, observável, discreto e determinístico.
A implementação foi feita da seguinte maneira: primeiramente, buscamos e salvamos as informações da entrada (as dimensões e o valor de W e o mapa). A partir disso, foi gerado um grafo. A peculiaridade desse grafo é que ele inclui apenas os pontos relevantes para a solução do problema, ou seja, os pontos de entrada, de abastecimento e o *goal*. Isso foi feito rodando o algoritmo de Dijkstra para encontrar a distância entre eles.
A partir do comando identificado na instrução de execução, ele segue a aplicar um dos quatro algoritmos no grafo.
Os estados do problema são todas as posições alcançáveis no mapa, ou seja, o conjunto de coordenadas que não estejam identificadas pelo símbolo de obstáculo *. Além disso, excluímos os estados não alcançáveis devido à restrição pela necesidade do estado de abastecimento ('#') após w passos, pois o veículo não pode trafegar até o local apesar dos comandos enviados.
Os estados iniciais definidos para cada mapa de dimensão x por y é dado pelo conjunto de coordenadas das laterais do mapa que não estejam identificadas pelo símbolo de obstáculo *, ou:
[0, i], para todo 0 $\leq$ i $<$ x && [0, i] $\neq$ '*'
$\cup$
[j, 0], para todo 0 $\leq$ j $<$ y && [j, 0] $\neq$ '*'
$\cup$
[k, i], para todo 0 $\leq$ i $<$ x && k = y - 1 && [k, i] $\neq$ '*'
$\cup$
[j, l], para todo 0 $\leq$ j $<$ y && l = x - 1 && [j, l] $\neq$ '*'

As ações são as direções que o AGV pode andar, ou seja, baixo, cima, esquerda, direita.
O teste do objetivo consiste em saber se o veículo, depois de uma sequência de instruções, chegou até o *goal*, ou seja, a coordenada do mapa identificada por '$'.
O custo de cada ação é 1.

3. Algoritmos

3.1. BFS

O BFS é um algoritmo de busca sem informação completo, ou seja, através do estado inicial, das funções de transição e seus custos e o estado final, ele consegue encontrar a solução se ela existir. Ele consiste em expandir o nó mais raso que ainda não foi expandido. Em outras palavras, ele organiza os nós da fronteira em uma fila. Por esse motivo, não é necessário fazer a expansão antes de verificar se estamos no *goal*. Como nosso grafo não tem custo uniforme, ele não sempre encontra a solução logo de cara. É necessário continuar a busca enquanto houver fronteira. Para analisar a complexidade do algoritmo para a solução, vamos considerar o branch factor de 4 (pois temos quatro direções possíveis na maioria dos casos) e o nível da solução d: $O(4^{d})$. Em questão do espaço, ele também é exponencial, pois os nós precisam ser mantidos na memória.

3.2. DFS

Assim como o BFS, o DFS é um algoritmo de busca sem informação. Porém, ao contrário do BFS, ele não consegue sempre encontrar a solução, pois ele falha em loops ou caminhos infinitos. Ele consiste em expandir o nó mais profundo, ou seja, ele organiza os nós da fronteira em uma pilha. O DFS não garante otimalidade, pois pela maneira que ele caminha, existe a possibilidade de expandir um *goal* de custo maior e retorná-lo como solução. A grande vantagem d DFS está na complexidade de espaço, que é linear, pois ele armazena apenas a subárvore que ele está, ou seja, $O(4m)$, sendo m a profundidade máxima da árvore. Em questão de tempo, ele é exponencial, assim como o BFS, porém ao invés de considerar a profundidade da solução, ele considera a profundidade da árvore: $O(4^{m})$. Isso pode ser muito ruim se a profundidade da árvore for muito maior que a profundidade da solução.

3.3. IDS

Esse algoritmo define um limite e várias buscas em profundidade dentro desse limite, aumentando-o a cada iteração. A vantagem dele é a combinação dos benefícios do BFS e do DFS: ele é completo e ótimo e com custo de memória linear. Pode ser considerado o melhor entre os algoritmos citados até agora.

3.4. A*

Ao contrário dos algoritmos anteriores, o A* é um algoritmo de busca com informação. Isso quer dizer que ele tenta utilizar alguma informação além da que está presente na formulação do problema para realizar a busca de forma mais eficiente. Para isso, ele usa uma função de avaliação $f(n)$ para cada nó, decidindo expandir o nó com menor $f(n)$. Em geral, $f(n)$ utiliza alguma heurística para guiar a ordem de expansão dos nós. A heurística para um nó n $h(n)$ é o custo estimado do nó até o *goal*. O algoritmo A* é possivelmente o algoritmo de busca com informação mais rápido, que expande a menor quatidade de nós e garante a solução ótima (se utilizar uma heurística admissível, que será explicado a seguir). Ele consiste em escolher o nó a ser investigado considerando $f(n) = g(n) + f(n)$, no qual $g(n)$ é o custo real para chegar no nó em questão e $h(n)$ é o custo

estimado para chegar ao *goal*. O A* é completo e sua complexidade de tempo é exponencial, dependendo da solução (e não da profundidade), assim como sua complexidade de espaço, pois os nós são guardados na memória.

4. Heurística – Distância de Manhattan

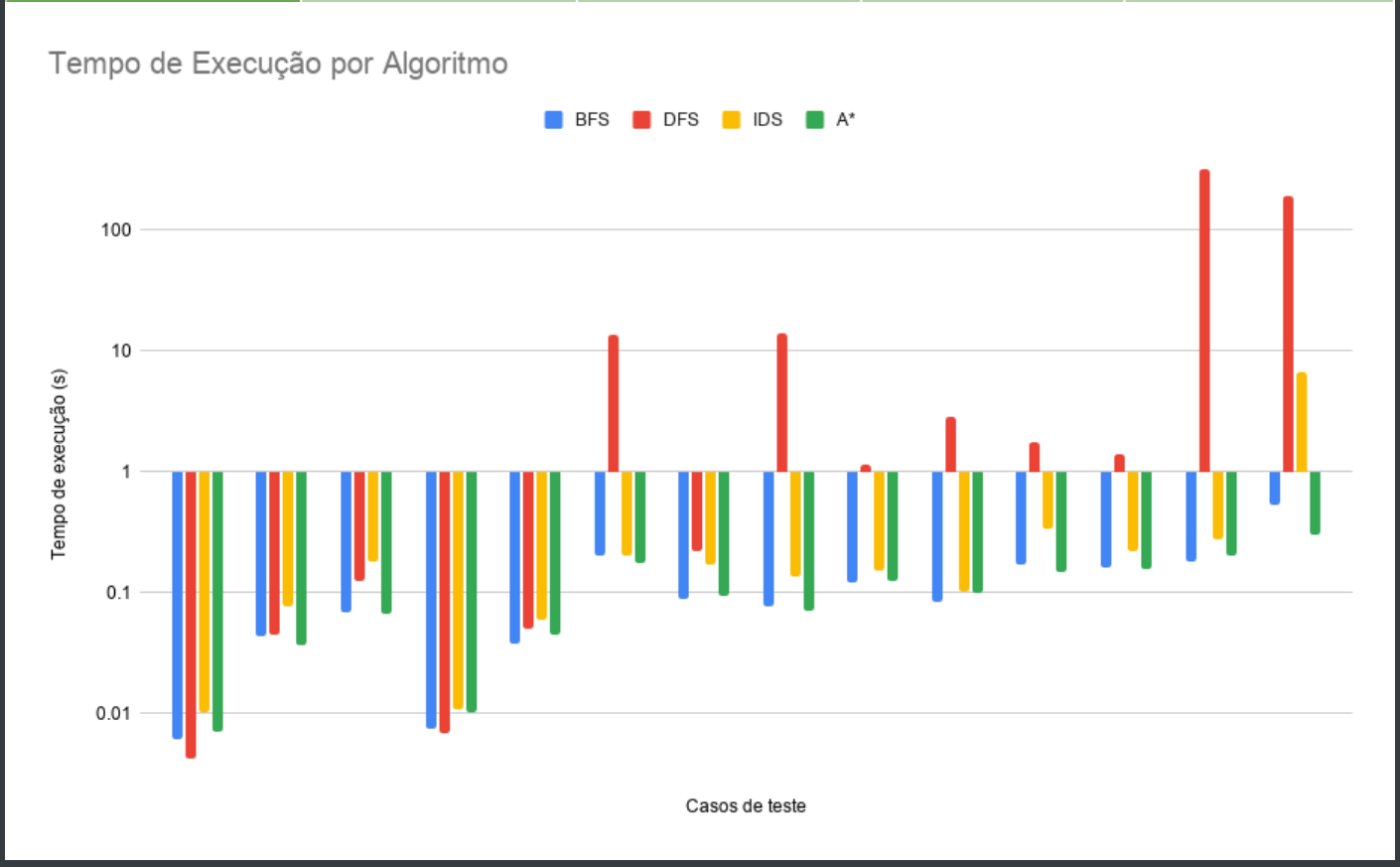
Para o algoritmo A*, devido à natureza bimensional do mapa do problema, foi escolhida uma heurística chamada Distância de Manhattan. Ela consiste em calcular a soma dos comprimentos da projeção da linha que une dois pontos num espaço euclidiano com um sistema de coordenadas fixo. Por exemplo, para dois pontos $P1$ e $P2$ contidos em um plano com coordenadas $(x1, y1)$ e $(x2, y2)$, respectivamente, a Distância de Manhattan é definida por: $|x1 - x2| + |y1 - y2|$. Ela é uma heurística admissível, pois o custo indicado por ela é sempre menor ou igual ao custo real para o *goal*, uma vez que ela define a menor distância possível percorrendo em direções cima, baixo, direita, esquerda.

5. Análise dos Algoritmos

Com a ajuda das bibliotecas *time* e *tracemalloc*, foi possível medir o tempo e espaço alocado durante a execução do programa para cada um dos casos de teste. A seguir vamos apresentar o resultado das execuções em formato de tabela e gráfico. Vale ressaltar que, para o DFS, não foi possível mensurar os resultados devido o elevado tempo de execução.

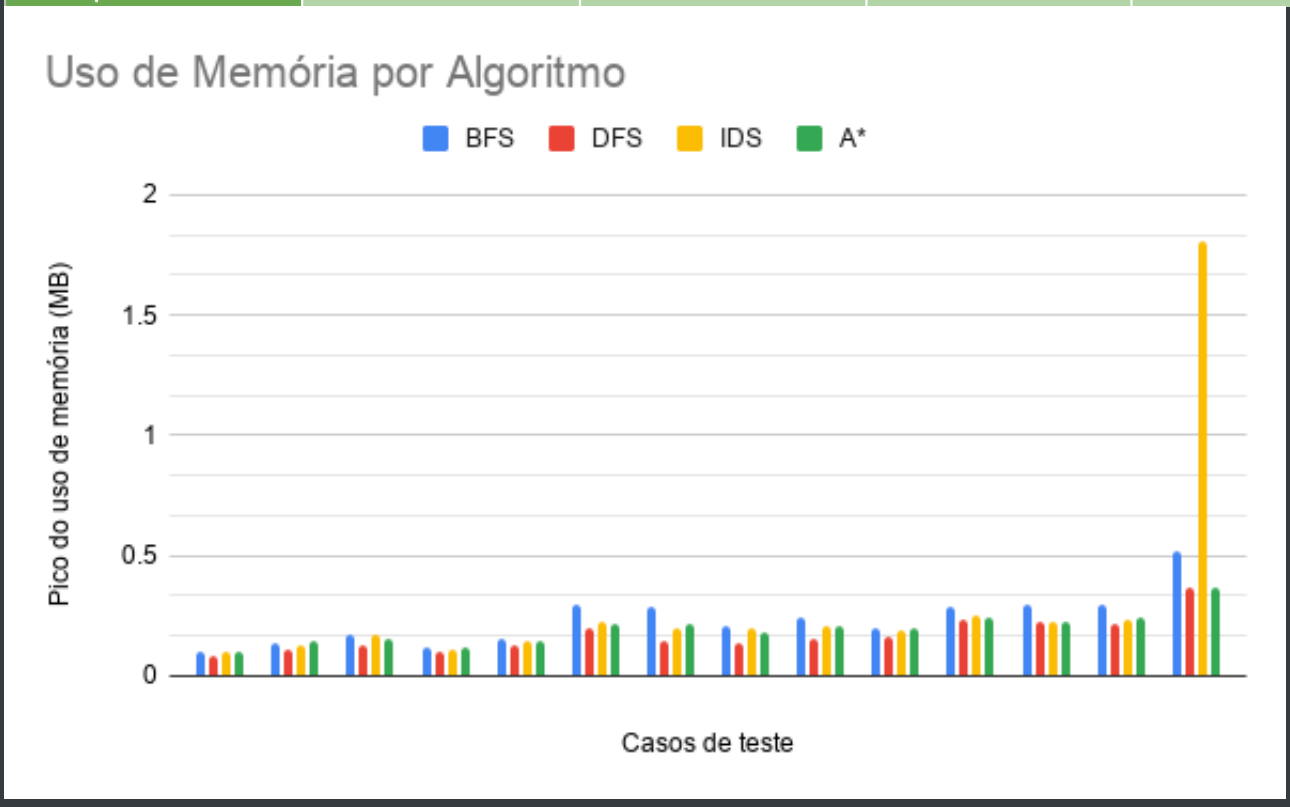
Primeiramente, a tabela e o gráfico relativos ao tempo. O gráfico está em escala logarítmica para ser possível visualizar melhor a comparação entre os valores. Os valores registrados indicam que o DFS demorou consideravelmente mais para executar, enquanto os outros, em geral, não apresentam uma variação tão grande.

	Tempo de execução gasto (s)			
Caso de Teste	BFS	DFS	IDS	A*
input1	0.006178855896	0.004220247269	0.01033234596	0.007180213928
input2	0.04410886765	0.04445505142	0.07790231705	0.03644895554
input3	0.06906604767	0.1254599094	0.1831071377	0.06746482849
input5x5-5	0.007553100586	0.006805896759	0.01094603539	0.01015782356
input5x13-4	0.03812503815	0.05016279221	0.05923509598	0.04557800293
input9x13-5	0.2015328407	13.4512012	0.2019090652	0.1782050133
input9x14-5	0.08832001686	0.2177631855	0.1699340343	0.09466290474
input16x8-3	0.07630681992	13.82647705	0.1375479698	0.07098007202
input16x8-5	0.1222651005	1.15308094	0.1515741348	0.1233947277
input16x9-2	0.08288383484	2.863431931	0.1017310619	0.0993270874
input16x9-3	0.1726570129	1.745886326	0.3336770535	0.1497900486
input16x9-4	0.1621267796	1.392436028	0.2194311619	0.1590750217
input16x9-5	0.1830279827	318.4152598	0.2787339687	0.2036397457
input19x19-2	0.5376501083	191.803278	6.713634014	0.3037118912



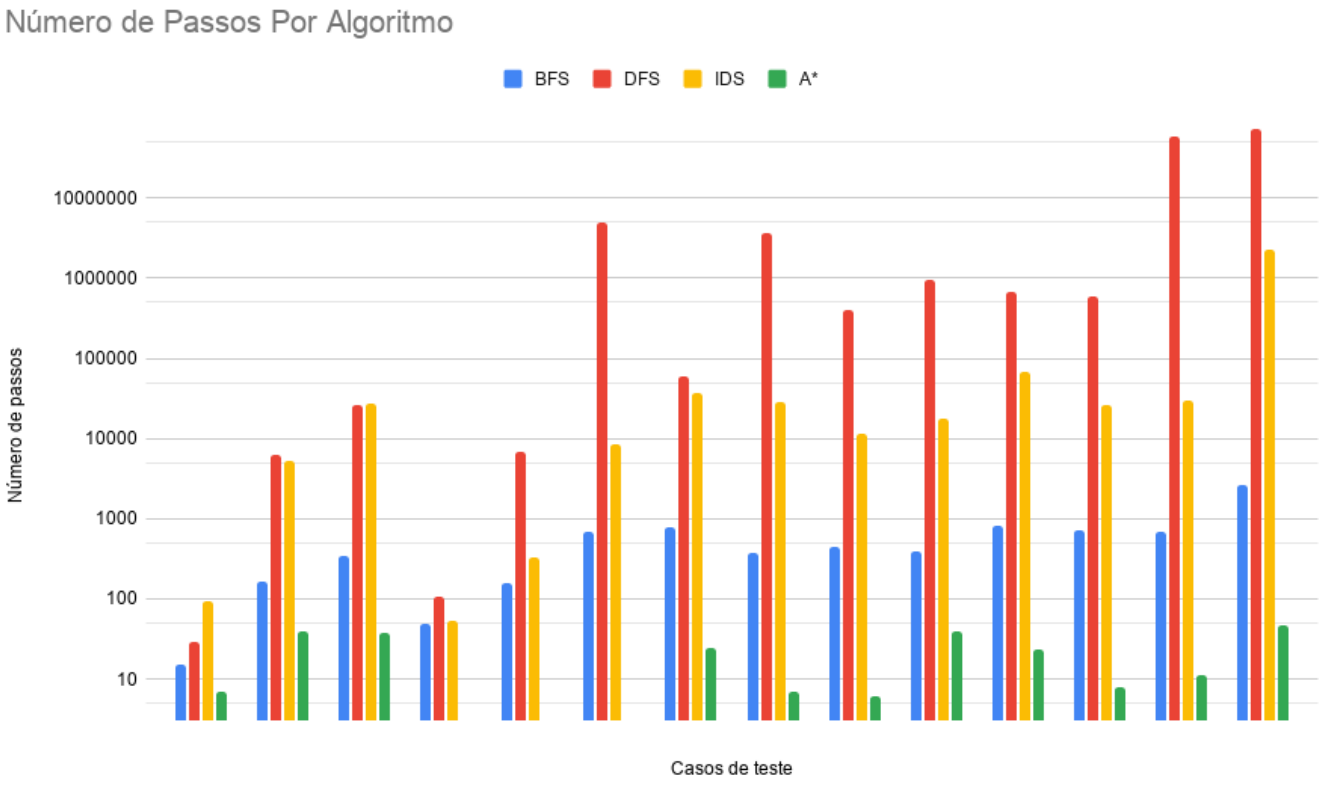
Para o espaço, o gráfico não está em escala logarítmica. Em geral, os algoritmos foram bem uniformes nesse aaspecto. A única execução que apresentou um valor elevado foi o IDS para o caso de teste input19x19-2.

	Espaço alocado (MB)			
Caso de Teste	BFS	DFS	IDS	A*
input1	0.10278	0.085612	0.103093	0.104294
input2	0.135668	0.110267	0.128407	0.144181
input3	0.169582	0.13011	0.171858	0.153073
input5x5-5	0.116369	0.098377	0.114088	0.115626
input5x13-4	0.155141	0.132577	0.143311	0.147053
input9x13-5	0.300893	0.201664	0.224041	0.217781
input9x14-5	0.285806	0.142896	0.197122	0.21759
input16x8-3	0.206695	0.135052	0.197268	0.185051
input16x8-5	0.245839	0.153277	0.205595	0.205643
input16x9-2	0.202263	0.167471	0.191443	0.197794
input16x9-3	0.288431	0.231535	0.251907	0.240785
input16x9-4	0.297967	0.223399	0.226731	0.222355
input16x9-5	0.300447	0.214156	0.238859	0.241505
input19x19-2	0.520743	0.365458	1.811258	0.370096



Por fim, temos a tabela de número de nós visitados por cada algoritmo em cada caso de teste. Como previsto pela definição dos algoritmos, o BFS e, especialmente, o A* têm um tendência de visitar menos nós que o DFS e o IDS.

	Número de Passos			
Caso de Teste	BFS	DFS	IDS	A*
input1	15	29	93	7
input2	161	6228	5275	39
input3	336	26091	27754	38
input5x5-5	49	108	53	3
input5x13-4	160	6778	322	3
input9x13-5	689	4905569	8311	3
input9x14-5	772	60569	36682	24
input16x8-3	367	3615933	28571	7
input16x8-5	448	400368	11437	6
input16x9-2	389	943999	17693	40
input16x9-3	807	682280	67403	23
input16x9-4	727	584654	25940	8
input16x9-5	677	59227486	29900	11
input19x19-2	2682	73387731	2261628	47



6. Discussão dos Resultados

Como podemos ver pelos resultados - ou pela falta deles - o DFS apresentou a pior performance entre os quatro algoritmos. Especialmente em questão de tempo, que foi o que limitou dois casos de testes. Ainda sobre tempo, o A* e o BFS tiveram tempos de execução bem similares.
Não foi possível enxergar a vantagem do DFS em relação ao espaço, devido à adapatação dele para sempre encontrar a solução ótima.
Sobre a quatidade de passos para cada algorítmo, foi nesse ponto que o algoritmo A*, uma busca com

informação, mostrou-se valioso. É possível notar como ele expandiu uma menor quatidade de nós em relação aos outros três.