INSTITUTO FEDERAL ESPÍRITO SANTO Campus Serra	**IFES** **Inteligência Artificial** Exercício de Programação 1: Solução de Problemas por Busca

Professor: Sérgio Nery Simões
Nome:Erikson Eler Ferreira

Data: 04/06/2023
Turma: PPCOMP-2023-1

Relatório Trabalho 1

Fundamentação Teórica:

A fundamentação teórica deste relatório aborda os algoritmos de busca utilizados, que são o Breadth-First Search (BFS), Depth-First Search (DFS), Uniform Cost Search (UCS) e A* (A-Star). Cada algoritmo tem suas características e diferenças, influenciando sua eficiência e o tipo de solução encontrada.

O BFS é um algoritmo de busca em largura, que explora todos os nós vizinhos antes de avançar para os nós mais distantes. Ele garante que a solução encontrada seja a mais curta em termos de número de passos, mas pode ser ineficiente em termos de tempo e espaço, especialmente em grafos grandes.

O DFS, por outro lado, é um algoritmo de busca em profundidade, que explora o máximo possível em uma ramificação antes de retroceder. Ele é eficiente em termos de espaço e pode encontrar soluções rapidamente em árvores ou grafos profundos. No entanto, não garante a solução mais curta e pode ficar preso em ciclos infinitos.

O UCS é um algoritmo de busca de custo uniforme, onde todos os caminhos são considerados, mas são priorizados aqueles com menor custo. Ele é adequado para problemas em que o custo é uma consideração importante, pois garante a solução com o menor custo total. No entanto, pode expandir muitos nós e ter um desempenho inferior em comparação com o A*.

O A* é um algoritmo de busca informada que combina a busca em largura com uma heurística. Ele utiliza uma função heurística para estimar o custo restante até o objetivo, permitindo priorizar nós promissores. Isso torna o A* eficiente em

termos de tempo e nós expandidos, encontrando soluções ótimas se a heurística for admissível e consistente.

Experimentos Questão 1 (Q1):

Para comparar os algoritmos, foi utilizado um labirinto de tamanho 300x300 com percentual de bloqueio de 50% e *seed* igual a 42.

A heurística utilizada para o A-star foi ao cálculo da distância euclidiana do nó atual até o nó destino.

Configurações do computador utilizado no problema:

O experimento foi executado em um servidor Dell PowerEdge com as seguintes especificações:

Processador: AMD EPYC 7453 2.75GHz, 28C/56T - Cache 64M

Memória RAM: 128GB, 3200mhz Dual Rank

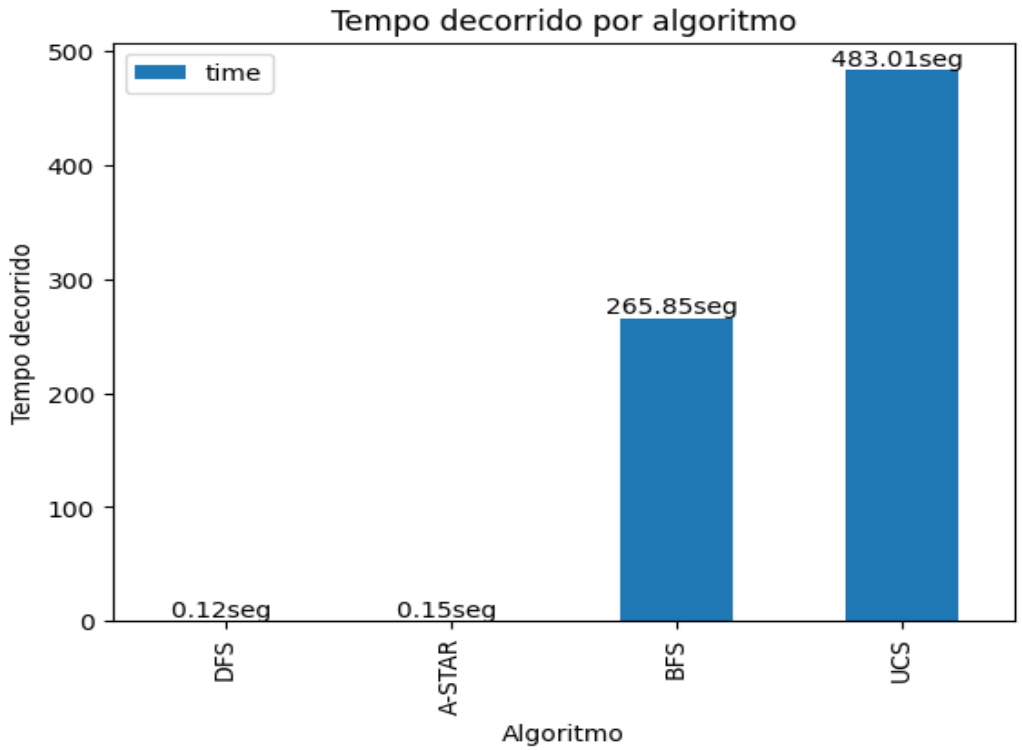
Resultados (Q1):

Após realizar os experimentos de busca no labirinto utilizando os algoritmos BFS, DFS, UCS e A*, obtivemos os seguintes resultados.

Algoritmo	Tempo (s)	Nós Expandidos	Custo Total do Caminho	Número de Passos
BFS	265.85	54,623	474.47	356
DFS	0.12	995	959.95	769
A*	0.15	725	501.15	386
UCS	483.01	54,623	462.14	359

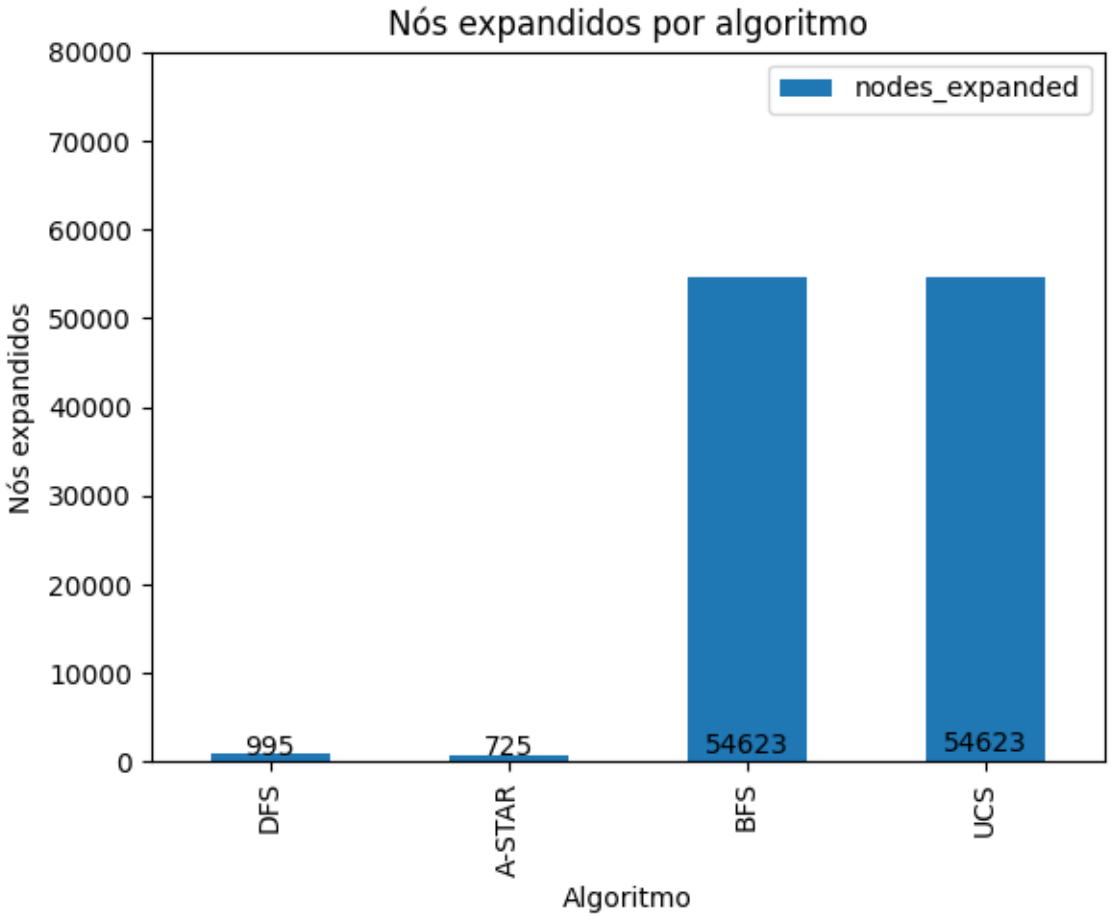
No que diz respeito ao tempo gasto para encontrar o destino, observamos que o DFS foi o algoritmo mais rápido, levando apenas uma fração de segundo. Por outro lado, o UCS foi o algoritmo mais demorado, levando cerca de 8 minutos. O BFS e o A* ficaram em um meio termo, com tempo de execução em torno de 4 segundos.

No gráfico de barras abaixo, podemos visualizar essa diferença de tempo entre os algoritmos:



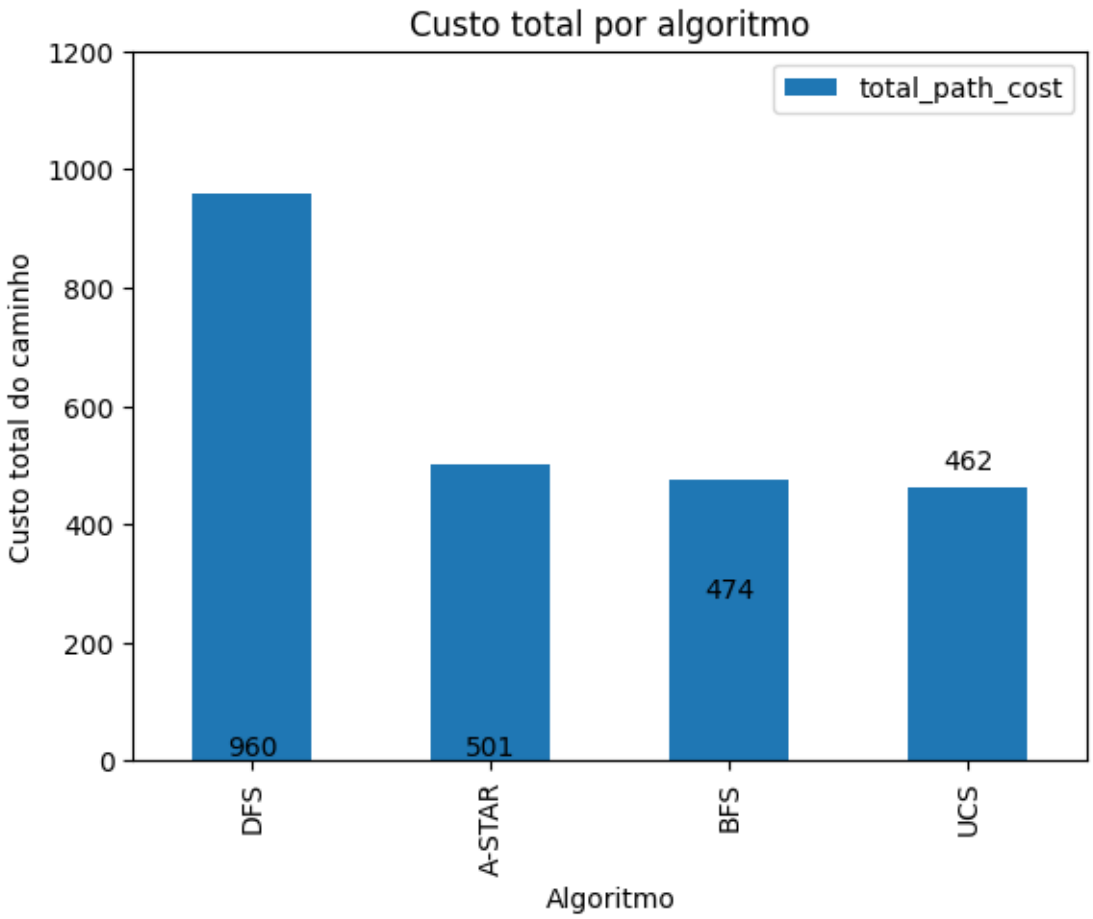
Quanto ao número de nós expandidos durante a busca, tanto o BFS quanto o UCS expandiram uma quantidade considerável de nós em comparação com o DFS e o A*. Isso ocorre devido à natureza exploratória desses algoritmos, que avaliam várias opções antes de encontrar o destino.

O gráfico de barras abaixo ilustra a quantidade de nós expandidos por cada algoritmo:



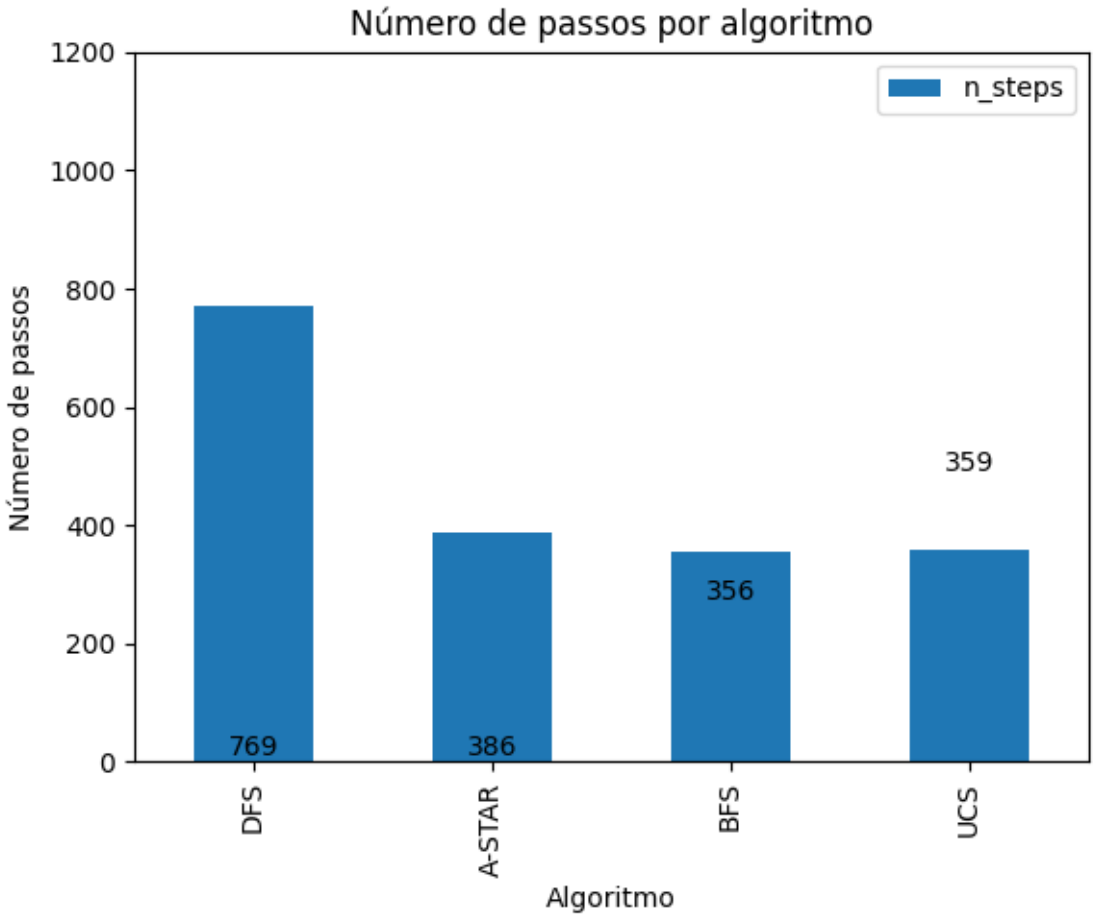
Em relação ao custo total do caminho encontrado, observamos que tanto o BFS quanto o A* alcançaram os menores custos. Isso se deve às características desses algoritmos, que garantem a busca pela solução de menor custo. Por outro lado, o DFS apresentou um custo total significativamente maior devido à sua abordagem não otimizada.

O gráfico de barras abaixo representa o custo total do caminho para cada algoritmo:



Por fim, analisando o número de passos necessários para encontrar o destino, podemos observar que o DFS exigiu o maior número de passos, seguido pelo A*, UCS e BFS. Essa diferença ocorre devido às estratégias de busca adotadas por cada algoritmo.

O gráfico de barras abaixo mostra o número de passos para cada algoritmo:



Em resumo, os resultados obtidos foram consistentes com a teroira. O DFS apresentou a menor velocidade de execução, mas com um custo total do caminho elevado. O BFS e o A* encontraram soluções de menor custo, porém com maior tempo de execução. O UCS teve um desempenho mais demorado devido à expansão de um grande número de nós.

Experimentos Questão 2:

No experimento realizado, o objetivo foi encontrar o caminho mais curto entre as cidades 'Arad' e 'Bucharest' usando os algoritmos BFS, UCS e A* Search. O grafo utilizado continha as conexões e os custos entre as cidades. Foi fornecida uma estimativa heurística das distâncias de todas as cidades até o destino 'Bucharest'.

As configurações do computador são as mesmas da Q1.

Resultados (Q2):

Os resultados obtidos foram os seguintes:

- BFS:
    - Custo: 450;
    - Caminho: Arad – Sibiu – Fagaras – Bucharest;

- UCS:
    - Custo: 418;
    - Caminho: Arad – Subiu – Riminicu Vilcea – Pitesti – Bucharest;

- A-STAR:
    - Custo: 418;
    - Caminho: Arad – Subiu – Riminicu Vilcea – Pitesti – Bucharest;

Algoritmo	Custo
BFS	265.85
DFS	0.12
A*	0.15
UCS	483.01

Comparando os resultados dos três algoritmos, podemos observar que:

- UCS e A* encontraram o mesmo caminho de menor custo totalizando 418.
- BFS encontrou um caminho de maior custo, porém com menor quantidade de nós. Isso de se dá pelo fato de não considerar o peso entre as cidades ou uma heurística como no UCS e A* respectivamente.

Esses resultados estão consistentes com a teoria esperada. O UCS e o A* Search são algoritmos informados, que consideram o custo acumulado do caminho. Portanto, eles são capazes de encontrar o caminho de menor custo com eficiência. Por outro lado, o BFS é um algoritmo não informado que explora todos os nós em uma camada antes de avançar para a próxima, o que pode levar a soluções subótimas em termos de custo.

Portanto, concluímos que, para problemas de roteamento entre cidades, o UCS e o A* Search são mais adequados, pois garantem a otimalidade do caminho encontrado. No entanto, é importante considerar o custo computacional e a disponibilidade de informações adicionais ao escolher o algoritmo a ser utilizado.