Trabalho Prático 1: Algoritmo de Busca para Menor Caminho

Felipe Louzada Mingote[1]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais

1. Introdução

O presente trabalho demonstra os algoritmos de busca sem informação (em largura, com aprofundamento iterativo e de custo uniforme) e com informação (gulosa e A*), junto com a heurística consistente para A* que minimize o número de nós expandidos e diferente das já fornecidas. O trabalho se propõe a resolver o *Path-Finding*, que consiste em encontrar o menor caminho entre dois pontos em um mapa com obstáculos.

2. Solução do Problema

Para construção do mapa, foi implementado um grafo contendo as coordenadas dos mapas e as ligações entre os nós, além de uma matriz de referência para os pesos dos caminhos no mapa. O grafo foi feito utilizando dicionários, evitando a utilização de qualquer biblioteca não nativa do Python 3.10.4.

Um detalhe extra na implementação é de que o algoritmo lê a entrada como se fosse a posição de uma matriz, ou seja, para leitura, foi considerado que as coordenadas passadas na entrada estivessem da forma (coluna, linha), ou (y, x) no plano cartesiano, mantendo a construção do mapa intacto, além do gabarito passado na especificação desse trabalho.

Foram implementados cinco métodos distintos para solucionar o problema:

- Busca em Largura (BFS)
- Aprofundamento Iterativo (IDS)
- Busca de Custo Uniforme (UCS)
- Busca Gulosa
- Busca A*

2.1. Busca em Largura (BFS)

A BFS é um estratégia de busca *sem informação* simples, onde o nó raiz é expandido primeiro, em seguida todos os sucessores do nó raiz são expandidos, seguidos pelos sucessores destes nós, e assim por diante.[Russell and Norvig 2004]

A BFS expande todos os nós de uma determinada profundidade antes dos nós do próximo nível de profundidade da árvore de busca, portanto sendo **FIFO**, onde cada novo nó, que chega vai para o fim da fila, enquanto nós mais antigos são expandidos primeiro.

A BFS é um método completo, onde se existir o nó objetivo, o método irá encontrá-lo.

Se o custo de cada caminho for o mesmo, a BFS é um método ótimo de se alcançar o nó objetivo.

A complexidade de espaço é o problema maior da BFS, já que expande todos os nós em níveis superiores ao nó objetivo.

2.2. Aprofundamento Iterativo (IDS)

O método IDS combina as buscas em largura e em profundidade, aumentando gradualmente o limite de profundidade da busca. Encontra o nó objetivo mais raso, combinando a baixa necessidade de memória da busca em profundidade com a completude da busca em largura, que sempre encontra o nó objetivo se ele existir.[Russell and Norvig 2004]

É um método ótimo quando o quando o custo do caminho é uma função não decrescente da profundidade do nó.

A implementação do mesmo foi dada seguindo o exemplo apresentado em sala de aula, onde o método itera sobre a altura da árvore e possui um limite iterável. Nesse caso, o algoritmo irá iterar em até todos os nós de uma árvore de altura 20.

2.3. Busca de Custo Uniforme (UCS)

Diferente da BFS, que expande o nó mais raso, a UCS expande o nó com o custo mais baixo, utilizando uma fila de prioridade para armazenar os nós da fronteira.[Russell and Norvig 2004]

Além da ordem da fila por custo do caminho, há duas outras diferenças significativas na busca em largura. A primeira é que o teste de objetivo é aplicado a um nó quando ele é selecionado para a expansão, e não quando é gerado pela primeira vez. A razão é que o primeiro nó objetivo que é gerado pode estar em um caminho abaixo do ótimo. A segunda diferença é que é adicionado um teste, caso seja encontrado um caminho melhor para um nó atualmente na borda.

É um algoritmo completo, desde que se adicione um custo ϵ mínimo em cada nó, de modo a se evitar que o algoritmo fique preso em um loop de caminhos com custo igual a zero. Também é ótimo, já que sempre expande o nó com o menor custo de caminho.

2.4. Busca Gulosa

A busca gulosa é também chamada de busca de melhor escolha, onde um nó é selecionado para a expansão com base em uma função de avaliação, f(n).[Russell and Norvig 2004]

Como função de avaliação se usa, em geral, uma heurística, que foi a distância em largura e altura de um ponto ao outro, ou seja, distância de Manhattan, no caso desse trabalho.

No caso da busca Gulosa, a heurística visa expandir o nó mais próximo do objetivo, para se reduzir o tempo de se encontrar a solução.

2.5. Busca A*

A forma de solução mais amplamente conhecida da busca com informação é denominada de busca A*. Ela avalia os nós através da combinação de g(n), o custo para alcançar o nó, e h(n), o custo para ir do nó ao objetivo.[Russell and Norvig 2004]

Desde que a função heurística h(n) satisfaça as condições de admissibilidade e consistência, a busca A* será completa e ótima. O algoritmo é idêntico ao da Busca de Custo Uniforme, exceto que A* usa g + h em vez de g. No caso, a heurística utilizada para o A* também foi a distância de Manhattan.

3. Análise Experimental

Foram realizados testes para se averiguar o funcionamento dos métodos implementados para a solução dos problemas propostos.

Os testes foram realizados em um computador equipado com um I7 2600 com as seguintes entradas:

- mapa teste.map: 1 1 3 1
- cidade.map: 117 86 119 91
- floresta.map: 67 25 71 24

mapa	algoritmo	custo	expandidos	tempo
mapa_teste.map	BFS	7	6	0.0001278
mapa_teste.map	IDS	4.5	3	0.0009996
mapa_teste.map	UCS	4.5	6	0.0009999
mapa_teste.map	Greedy	7	5	0.0005689
mapa_teste.map	Astar	4.5	5	0.0009982
cidade.map	BFS	17.5	58	0.0090558
cidade.map	IDS	17.5	53	0.4314928
cidade.map	UCS	17.5	53	0.06115341
cidade.map	Greedy	17.5	8	0.0539486
cidade.map	Astar	17.5	8	0.056982040
floresta.map	BFS	7.5	159	0.0019991
floresta.map	IDS	7.5	42	0.49987554
floresta.map	UCS	7.5	42	0.0499875
floresta.map	Greedy	7.5	6	0.0548974
floresta.map	Astar	7.5	6	0.06816768

3.1. Tempo de Execução

A seguir o tempo de execução dos métodos para cada um dos mapas:

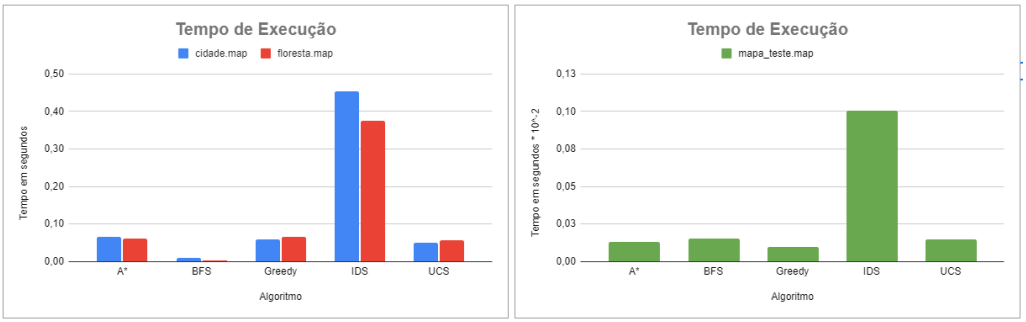

Figura 1. Gráficos com tempo de execução. O gráfico do mapa de teste foi separado para facilitar a escala dos gráficos.

3.2. Número de Nós Expandidos

A seguir a quantidade de nós expandidos dos métodos para cada um dos mapas:

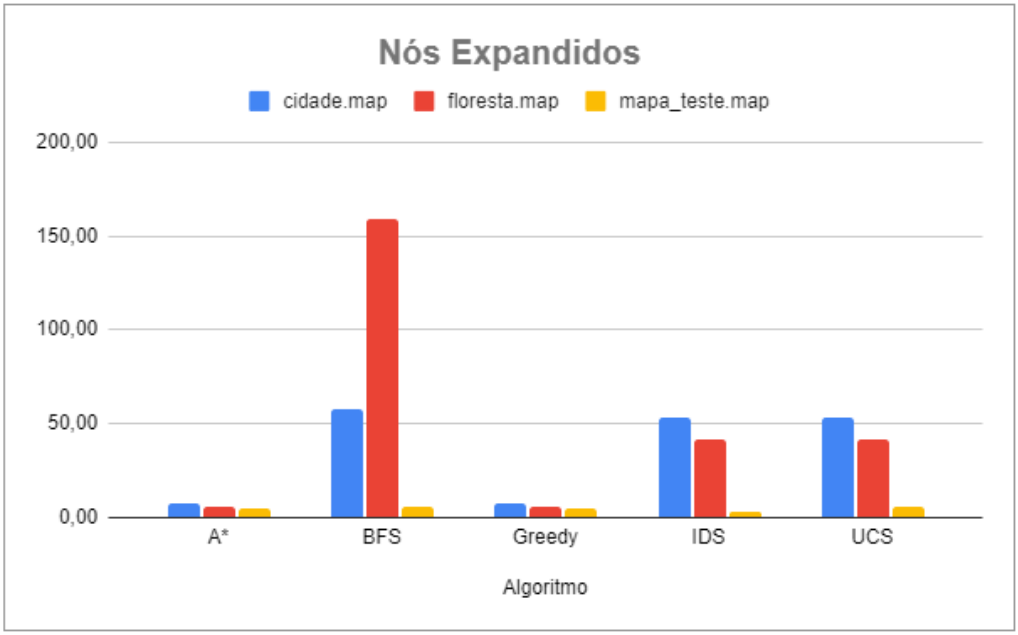

Figura 2. Gráfico relacionando algoritmos com a quantidade de nós expandidos.

4. Conclusão

Analisando os resultados obtidos, podemos observar uma grande vantagem do método BFS em relação ao tempo, uma vez que ele expande sem procurar ou fazer uma busca sobre qual nó é o melhor né a se expandir, o que os outros algorimos fazem e gastam tempo com isso. Por outro lado, a quantidade de nós expandidos é incrivelmente maior, o que pode oferecer um grande problema de espaço na memória em mapas com distâncias muito maiores.

Sobre o tempo, podemos ter uma boa ideia quanto a isso, mas como gerar os testes em mapas e distâncias poderiam levar muito tempo, os testes feitos foram em curtas distâncias, como descrito anteriormente.

No geral, o trabalho teve seu propósito cumprido, a análise prática consegue demonstrar muito bem o porquê o A* é muito usado, tanto em tempo quanto em consumo de memória, além de sempre retornar o menor caminho possível no contexto desse trabalho.

Referências

Russell, S. and Norvig, P. (2004). Inteligência artificial-uma abordagem moderna (2a edição). *Rio de Janeiro, Brasil. Editora Campus*.