UNIVERSIDADE ESTADUAL DO OESTE DO PARNÁ – UNIOESTE

CAMPUS DE FOZ DO IGUAÇU

CENTRO DE ENGEHARIAS E CIÊNCIAS EXATAS – CECE

GUILHERME BACHEGA GOMES

RELATÓRIO DO PRIMEIRO PROJETO DA DISCIPLINA DE INTELIGÊNCIA ARTIFICAL

FOZ DO IGUAÇU

2022

Introdução

Um grafo é uma estrutura de dados que representa relações entre objetos. Os objetos são denominados nós e as relações entre eles são denominadas arestas. Logo, pode-se compreender um grafo como um conjunto de vértices e arestas. (Stanford University, 2016).

Diversos problemas do mundo real podem ser modelados através de grafos, por exemplo:

- O mapeamento de amigos em redes sociais onde os usuários são descritos como vértices e, caso estejam adicionados, seu relacionamento representado como uma aresta que os conecta (Stanford University, 2016).
- A descrição de cidades em uma região, como vértices, e suas estradas que as conectam como arestas. Pode-se incluir pesos em cada aresta representando a distância que seria percorrida por aquela estrada (Cormen *et al*., 2009).

Além de ser possível descrever problemas do mundo real, a solução destes também pode acabar sendo retratada como uma solução de problemas em grafos (Cormen *et al*., 2009). Tomando o caso das cidades em uma região, interligadas por estradas, pode-se ter os seguintes problemas:

- Para otimizar o tempo de uma viagem de uma cidade até outra, deve-se encontrar o menor caminho entre as duas (Cormen *et al.*, 2009);
- Para otimizar a entrega de produtos para um grupo de cidades, deve-se encontrar o melhor caminho que passa por todas essas cidades.

Ambas as questões acabam sendo também problemas de grafos. No caso do primeiro, se trata de um problema de menor caminho de uma única fonte, onde dado um grafo direcionado e com custos, um vértice inicial e um vértice objetivo (ou vértices objetivos), deve-se encontrar o caminho menos custoso. Há diversos algoritmos para solucionar este problema como, por exemplo, Busca em Profundidade, Busca em Largura, Busca de Custo Uniforme, Bellman-Ford, entre outros. (Cormen *et al*., 2009).

No caso do segundo, pode se interpretar como um problema de Caixeiro Viajante (caso o grafo seja completo), um problema que até a escrita deste relatório se trata de ser NP-completo, tendo algumas soluções que dão resultados aproximados. (Cormen *et al*., 2009).

O problema tratado neste trabalho é o de encontrar uma solução para um mapa do jogo *The Backrooms*, que essencialmente é um labirinto de salas onde se deve encontrar uma estratégia de navegação a fim de partir de uma sala inicial até uma sala objetivo. Assim como em outros exemplos já mencionados neste trabalho, a navegação de um labirinto também pode ser mapeada para um problema de grafos. Descreve-se cada sala como sendo um vértice e salas que possuem conexão umas com as outras são descritas como arestas que possuem custos de se transitar entre salas. Desta forma, tem-se um grafo dirigido onde cada vértice representa a distância de ir de uma sala à outra.

Mapeado o labirinto como um grafo, pode-se desenvolver uma solução computacional para ele que manipulará a estrutura. Como há a necessidade de encontrar um caminho entre um vértice inicial e final, define-se o problema como sendo encontrar o menor caminho entre dois vértices partindo de um vértice de fonte única.

O objetivo deste trabalho é resolver o problema de menor caminho entre uma sala inicial e uma sala final de um labirinto, utilizando no mínimo duas abordagens, sendo uma solução mais adequada e a outra menos adequada, e compará-las.

Materiais e Métodos

Os materiais utilizados no desenvolvimento do trabalho foram:

- Notebook Dell, com processador i5 de oitava geração, 16GB de memória RAM, placa de vídeo Nvidia GeForce GTX 1050 e Sistema Operacional Windows 10;
- Linguagem de programação Java (Oracle, 2022);
- Ambiente de desenvolvimento integrado Eclipse (Eclipse Foundation, 2022).

A fim de compreender melhor os temas relacionados a este trabalho, foram realizados estudos sobre grafos, representação de grafos em memória, o problema de menor caminho e algoritmos que encontram um menor caminho entre um vértice inicial e final. (Cormen *et al*., 2009; Russel e Norvig, 2010).

Os algoritmos foram analisados a fim de se definir medidas de desempenho que possam ser utilizadas para compará-los.

Tendo definido estas medidas, os algoritmos foram implementados e as suas métricas de desempenho comparadas.

Resultados

A escolha da estrutura que representa o grafo em memória foi a lista de adjacência. Essa abordagem é uma forma mais compacta de representar grafos que são esparsos, ou seja, que possuem muito menos arestas do que vértices. Além disso, alguns algoritmos na literatura consultada foram feitos considerando listas de adjacência, facilitando o processo de implementação. A sua complexidade de espaço na média é $O(V + E)$, chegando, no pior caso, à $O(V^2)$. Em comparação, a matriz de adjacência possui tanto o caso médio quanto o melhor caso com complexidade de espaço de $O(V^2)$, sendo menos eficiente. (Cormen *et al*., 2009).

Os algoritmos escolhidos como melhor e pior foram o Busca de Custo Uniforme (*Dijkstra*) e a Busca em Profundidade, respectivamente. Além disso, o algoritmo de Busca em Largura também foi implementado. (Cormen *et al*., 2009).

O algoritmo Busca de Custo Uniforme é um algoritmo guloso, ótimo e completo para o problema de menor caminho em grafos dirigidos com pesos não negativos, justamente por isso foi escolhido como a melhor solução (Cormen *et al*., 2009).

É importante ressaltar que essa abordagem não é ótima em grafos que contenham arestas com custo negativo (Cormen *et al*., 2009), como o problema de distância entre salas não corre esse risco (não existe distância negativa), pode-se afirmar que a abordagem é ótima para ele.

As etapas desse algoritmo são descritas a seguir (Cormen *et al.*, 2009):

1. O vértice inicial é explorado e marcado como visitado;

2. Os vértices adjacentes a ele são enfileirados em uma fila de prioridade onde o elemento de maior prioridade é o que possui o menor custo. Todos os vértices enfileirados recebem como antecessor o nó atual e recebem como peso o custo do nó atual mais o custo da aresta que os conecta;

3. O próximo nó da fila de prioridade é explorado e marcado como visitado. Caso seja o vértice objetivo, o algoritmo para e o caminho da sala inicial até a final pode ser obtida ao se percorrer os vértices antecessores partindo do vértice objetivo (note que o caminho será de trás para frente). Caso este nó não seja o objetivo, o passo 2 é executado.

O algoritmo de Busca em Profundidade não é completo e nem ótimo, justo por esse motivo foi escolhido como a pior solução. Funciona percorrendo o grafo de forma cada vez mais profunda sempre que possível até ter que retroceder para explorar outros caminhos (Cormen *et al*., 2009).

Funciona marcando o vértice inicial com uma coloração cinza, que indica que aquele nó foi visitado mas que ainda há arestas não exploradas, e definindo seu antecessor como nulo. Em seguida, explora o primeiro nó não explorado de sua lista de adjacência, o colore como cinza e marca como seu antecessor o nó detentor da lista de adjacência. Caso atinja um vértice onde todas as suas arestas já foram visitadas (possuem coloração cinza ou preta), colore aquele vértice como preto e retrocede a fim de explorar arestas não visitadas de nós que ficaram para trás. O algoritmo é interrompido caso o nó visitado seja um vértice objetivo, sendo possível recuperar o caminho percorrendo os antecessores de cada nó partindo do vértice final (note que o caminho será de trás para frente) (Cormen *et al.*, 2009).

O algoritmo de Busca em Largura foi implementado como extra e trata-se, essencialmente, de um algoritmo de Busca de Custo Uniforme sem levar em consideração os custos das arestas (Cormen *et al*., 2009).

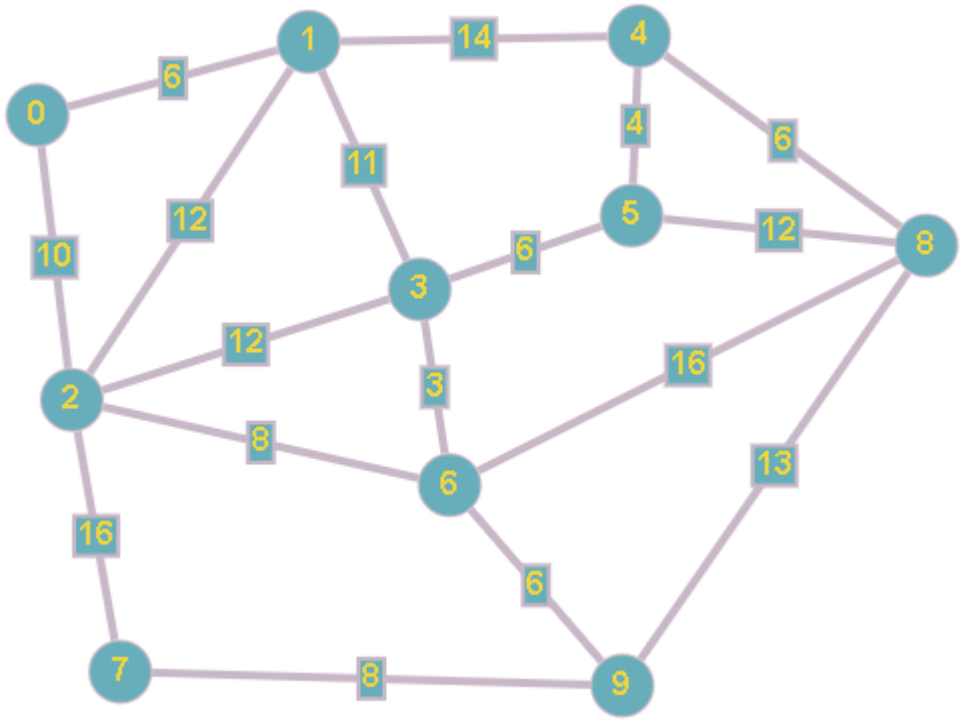
Funciona visitando o nó inicial, o inserindo em uma fila, o colorindo como cinza e definindo seu antecessor como nulo. Em seguida, remove o primeiro nó da fila e o colore como preto. A partir do primeiro nó da fila removido, percorre todos os seus vértices adjacentes que não sejam de coloração cinza ou preta, marcando este nó como antecessor de todos os seus nós adjacentes que são marcados como cinza e enfileirados. Este processo se repete até que o nó atingido seja o objetivo, onde o algoritmo é encerrado e o caminho pode ser recuperado percorrendo os antecessores de cada nó partindo do vértice final (note que o caminho será de trás para frente) (Cormen *et al.*, 2009).

Das medidas utilizadas para comparar os algoritmos, foram utilizados a quantidade de salas que são visitadas, a quantidade de nós que são enfileirados (no caso de Busca de Custo Uniforme e Busca em Largura), o custo total do caminho e o tempo de execução.

As tecnologias utilizadas no desenvolvimento deste trabalho foram a linguagem de programação Java (Oracle, 2022) para a implementação de toda a aplicação e o ambiente de desenvolvimento integrado Eclipse (Eclipse Foundation, 2022) para auxiliar nesta implementação.

Os testes da análise de desempenho foram realizados em um grafo de teste de 10 vértices apresentado a seguir:

Figura 1 – Grafo utilizado para os testes da análise de desempenho



Mesmo que o grafo seja visualmente não dirigido, nos testes dos algoritmos se trata de um grafo dirigido onde havia duas arestas de ida e de volta com mesmo custo.

Nas tabelas a seguir, se encontra os resultados de tempo de execução, número de salas visitadas, número de salas enfileiradas (caso aplique) e o custo total do caminho para a execução da busca tendo o vértice de número 0 como inicial com todos os outros vértices como finais.

Tabela 1 – Tempo de execução (milissegundos) tomando o vértice 0 como inicial.

Vértice final	Busca de Custo Uniforme	Busca em Profundidade	Busca em Largura
1	1.5972	0.0171	0.8392
2	0.0859	0.0408	0.0874
3	1.0061	0.0275	1.0560
4	0.1697	0.0197	0.0423
5	0.1195	0.0181	0.0783
6	0.0531	0.0294	0.0763
7	0.1400	0.0225	0.1301
8	0.0716	0.0438	0.0607
9	0.0530	0.0192	0.0707
Média	0.4190	0.0294	0.2712

Tabela 2 – Número de salas visitadas

Vértice final	Busca de Custo Uniforme	Busca em Profundidade	Busca em Largura
1	2	2	2
2	3	3	3
3	4	4	4
4	6	6	5
5	7	5	8
6	5	8	6
7	9	10	7
8	10	7	9
9	8	9	10
Média	6	6	6

Tabela 3 – Número de salas enfileiradas.

Vértice final	Busca de Custo Uniforme	Busca em Profundidade
1	3	3
2	5	5
3	7	7
4	10	8
5	11	10
6	8	9
7	11	10
8	11	10
9	11	10
Média	9	8

Tabela 4 – Custo do caminho

Vértice final	Busca de Custo Uniforme	Busca em Profundidade	Busca em Largura
1	6	6	6
2	10	18	10
3	17	30	17
4	20	40	20
5	23	36	23
6	18	62	18
7	26	76	26
8	26	46	26
9	24	68	24
Média	19	43	19

A aplicação, quando iniciada, pede ao usuário que entre com o caminho do arquivo texto que contém o mapa do jogo. Após iniciada, ela apresenta as seguintes opções:

- Carregar novo mapa de jogo: requisita um novo caminho de arquivo texto para carregar um novo labirinto;

- Rodar Busca de Custo Uniforme: requisita um vértice inicial e final, executa o algoritmo de Busca de Custo Uniforme e imprime informações relacionadas a seu desempenho;

- Rodar Busca em Profundidade: requisita um vértice inicial e final, executa o algoritmo de Busca em Profundidade e imprime informações relacionadas a seu desempenho;

- Rodar Busca em Largura: requisita um vértice inicial e final, executa o algoritmo de Busca em Largura e imprime informações relacionadas a seu desempenho;

- Rodar Mais Eficiente por Custo: roda os três algoritmos e indica qual ou quais deles teve ou tiveram o caminho com menor custo;

- Rodar Menos Eficiente por Custo: roda os três algoritmos e indica qual, ou quais, deles teve, ou tiveram, o caminho com maior custo;

- Rodar Menos Salas Visitadas: roda os três algoritmos e indica qual, ou quais, deles visitou, ou visitaram, o menor número de salas;

- Rodar Mais Salas Visitadas: roda os três algoritmos e indica qual, ou quais, deles visitou, ou visitaram, o maior número de salas;

- Rodar Todos os Três: roda os três algoritmos e imprime as informações relacionadas a seu desempenho.

Discussão

Ao se analisar as métricas definidas para se comparar os algoritmos, pode-se perceber que o algoritmo de Busca em Profundidade possui tempo de execução, na média, menor do que o algoritmo de Busca em Custo Uniforme para o caso testado.

O tempo de execução foi mensurado a fim de encontrar o algoritmo que encontra uma resposta de maneira mais rápida.

Em relação ao número de salas visitadas, na média, o algoritmo de Busca em Profundidade e Busca de Custo Uniforme visitaram o mesmo número.

O número de nós de um grafo visitados é um bom indicativo de eficiência do algoritmo, podendo responder tanto sobre a sua eficiência de uso de memória quanto sobre o tempo de execução. Afinal, um algoritmo que tende a visitar mais nós para encontrar a solução também tende a demorar mais a encontrá-la e a utilizar mais memória para armazenar os vértices.

Em relação ao custo do caminho do vértice inicial até o final, o algoritmo de Busca em Profundidade é bem menos eficiente que o algoritmo de Busca de Custo Uniforme, tendo um custo médio por caminho de 43 e 19, respectivamente.

Este resultado é uma das métricas que permitem demonstrar a característica do algoritmo de Busca de Custo Uniforme em sempre encontrar o menor caminho em grafos direcionados com custos, mantendo-se completo e ótimo.

Em relação ao algoritmo extra, mesmo este tendo um desempenho em relação à caminhos de menor custo semelhante ao Busca de Custo Uniforme, essa situação se dá por puro acaso do mapa de testes, já que o algoritmo não leva em consideração os custos dos caminhos.

Sugere-se que os resultados das métricas de desempenho, excetuando-se o menor custo entre um vértice inicial e final, podem ter sido específicos do grafo de teste, não representando os valores mais gerais. Grafos maiores e/ou diversos outros casos de configurações de grafos podem ser utilizados a fim de validar os resultados apresentados.

Conclusão

Com base nos resultados obtidos, pode-se afirmar que o algoritmo de Busca de Custo Uniforme demonstrou ser ótimo e completo, diferente do Busca em Profundidade, em gerar um caminho de custo mínimo entre dois vértices do grafo.

Pode-se também concluir que, para o caso de teste específico, o algoritmo de Busca em Profundidade teve um menor tempo de execução que o algoritmo de Busca de Custo Uniforme e que ambos visitaram, na média, o mesmo número de salas.

Conclui-se, por fim, que a melhor abordagem para gerar um caminho entre uma sala inicial e uma sala final em um jogo de *The Backrooms* é o algoritmo de Busca de Custo Uniforme.

Referências

Cormen, T.H.; Leiserson, C.E.; Rivest, R.L. & Stein, C. (2009). Introduction to algorithms. Cambridge: MIT press.

Eclipse Foundation (2022). https://www.eclipse.org/downloads/. Acesso em 13 de fevereiro de 2022.

Material didático Prof. Rômulo César Silva – UNIOESTE.

Material didático Prof. Huei Diana Lee – UNIOESTE.

Oracle (2022). https://docs.oracle.com/en/java/. Acesso em 13 de fevereiro de 2022.

Russel, S. e Norvig, P. Artificial Intelligence: A modern approach, Prentice Hall, 2010.

Stanford University (2016).

https://web.stanford.edu/class/archive/cs/cs161/cs161.1168/lecture10.pdf. Acesso em 9 de fevereiro de 2022.