Relatório De Estruturas De Dados

Aluno: Marcos Dalessandro Cavalcante Lima (MATRÍCULA: 20209016090)
Campus Senador Helvídio Nunes de Barros - Picos
Professor(a): Prof(a) Juliana Oliveira Carvalho
Disciplina: Estruturas de Dados 2 - 2023.1
C.H.: 60 h Créditos: 2.2.0 Período: 2023.1

16 de agosto de 2023

1 Resumo

O presente trabalho apresenta de forma breve a estrutura de dados do tipo Grafo estudadas durante a disciplina de estruturas de dados 2. Compreende também o trabalho proposto pela professora Juliana Oliveira Carvalho, de forma detalhada as funções desenvolvidas.

2 Introdução

A Teoria dos Grafos é um campo intrínseco e fundamental da matemática discreta, que se dedica ao estudo das relações e conexões entre objetos abstratos denominados vértices ou nós, por meio de segmentos denominados arestas. A abstração de grafos tem origens históricas profundas, remontando ao século XVIII, mas seu desenvolvimento moderno e estruturado ganhou destaque ao longo do século XX, impulsionado por notáveis matemáticos e cientistas da computação.

Neste contexto, os grafos constituem uma representação versátil e poderosa de uma ampla gama de fenômenos e sistemas complexos, permeando diversas áreas do conhecimento, como ciência da computação, matemática aplicada, engenharia, biologia, redes sociais e logística. A análise e manipulação de grafos possibilitam a elucidação de propriedades estruturais subjacentes, permitindo a modelagem de interações entre entidades e a resolução de desafios práticos.

A presente investigação almeja explorar os elementos basilares da Teoria dos Grafos, compreendendo a terminologia essencial, os tipos de grafos mais comuns - como grafos direcionados, não-direcionados, ponderados e bipartidos - e as principais métricas e conceitos que governam sua análise, tais como caminhos, ciclos, conectividade e árvores. Além disso, serão

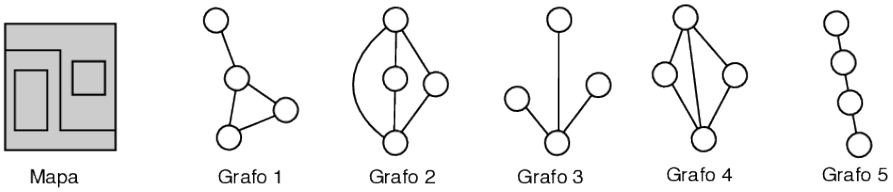

Figura 1 – Grafos Comuns

discutidas aplicações concretas da Teoria dos Grafos em cenários do mundo real, evidenciando sua relevância e eficácia na solução de problemas práticos, como a otimização de rotas, a análise de redes sociais e a resolução de quebra-cabeças.

Por meio desta exploração, espera-se fornecer um embasamento sólido no âmbito da Teoria dos Grafos, capacitando os leitores a compreender a sua natureza intrínseca, explorar suas aplicações transdisciplinares e reconhecer a sua importância como ferramenta analítica e modelagem em contextos acadêmicos e profissionais. Assim como é de suma importância ter uma algoritmo performático, é importante ter uma máquina que possa executa-los com eficiência, os algoritmos foram executados em um notebook Lenovo Ideapad Gaming 3i, uma maquina capaz de processar a base de dados utilizada sem tanto esforço.

3 Grafos

Com o decorrer da seção será apresentados os diversos Grafos relevantes para o presente trabalho, dentre eles temos os Grafos Valorados, Grafos Direcionais, Matriz de Adjacência e Listas de Adjacência.

3.1 Grafos Valorados

Na Teoria dos Grafos, um importante ramo da matemática discreta, um grafo valorado é uma estrutura que combina as características de grafos convencionais com a atribuição de valores numéricos às arestas que conectam seus vértices. Esses valores podem representar diversas informações relevantes, como distâncias, custos, capacidades, tempos ou quaisquer outras quantidades associadas à relação entre os vértices interconectados.

Um grafo valorado G é formalmente definido como G = (V, E, w), onde V é o conjunto de vértices, E é o conjunto de arestas e w é a função de valoração das arestas, que associa um valor numérico a cada aresta em E. Em outras palavras, cada aresta e E possui um valor w(e), que reflete a medida da relação específica que ela representa.

A introdução de valores nas arestas enriquece substancialmente a capacidade de modelagem dos grafos, permitindo a representação precisa de situações da vida real. Essa característica é particularmente relevante em contextos de otimização, planejamento e tomada de decisões, onde as informações quantitativas são cruciais para determinar soluções eficientes e eficazes.

Em suma, os grafos valorados representam uma extensão significativa da Teoria dos Grafos, incorporando a dimensão quantitativa nas relações entre vértices por meio da valoração das arestas. Essa abordagem fornece uma estrutura robusta para modelagem e resolução de problemas complexos, onde a consideração precisa de valores numéricos é essencial para a tomada de decisões informadas e eficientes.

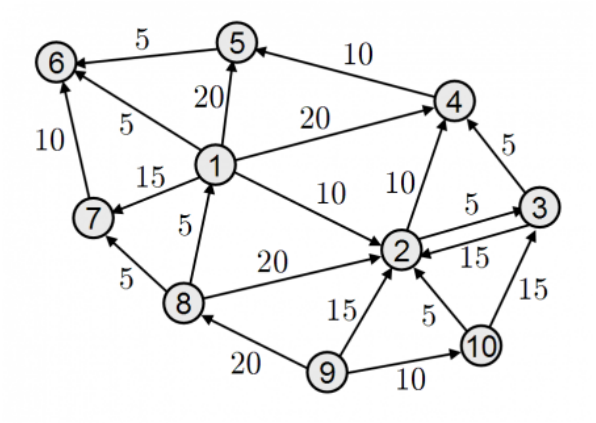

Figura 2 – Exemplo de Grafo Valorado

3.2 Grafos Direcionais

Grafos direcionais, também conhecidos como dígrafos, são estruturas que representam relações entre elementos através de arestas direcionadas. Cada aresta tem uma direção específica, indicando uma relação unidirecional entre os vértices conectados. Essa representação é útil quando a relação entre os elementos possui uma natureza assimétrica, como em fluxos de informação, relações hierárquicas ou redes de transporte unidirecional.

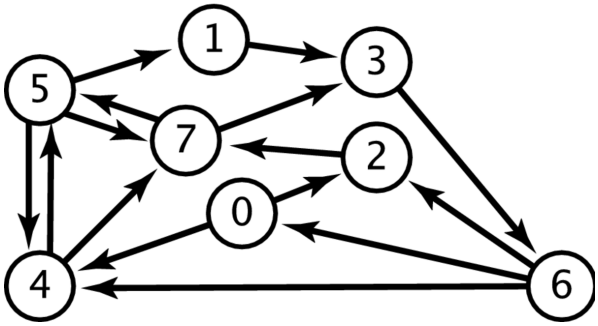

Figura 3 – Exemplo de Grafo Direcional

3.3 Matriz de Adjacência

A matriz de adjacência é uma forma de representar grafos, tanto direcionais quanto não-direcionais, usando uma matriz bidimensional. Cada linha e coluna da matriz corresponde a um vértice do grafo, e os elementos da matriz indicam a presença ou ausência de arestas

entre os vértices. No caso de um grafo direcional, os valores na matriz podem representar a existência de arestas direcionadas e suas direções. Matrizes de adjacência são eficazes para grafos densos, onde muitas conexões estão presentes.

	V1	V2	V3	V4	V5
V1	0	1	1	0	1
V2	1	0	1	1	0
V3	1	1	0	1	1
V4	0	1	1	0	0
V5	1	0	1	0	0

Figura 4 – Exemplo de Matriz de Adjacência

3.4 Listas de Adjacência

As listas de adjacência são uma alternativa à matriz de adjacência, especialmente úteis para grafos esparsos, nos quais há poucas conexões. Nesse método, cada vértice possui uma lista que enumera seus vértices adjacentes, ou seja, os vértices aos quais ele está diretamente conectado por uma aresta. Em grafos direcionais, as listas podem indicar a direção das arestas, fornecendo uma visão clara das conexões unidirecionais.

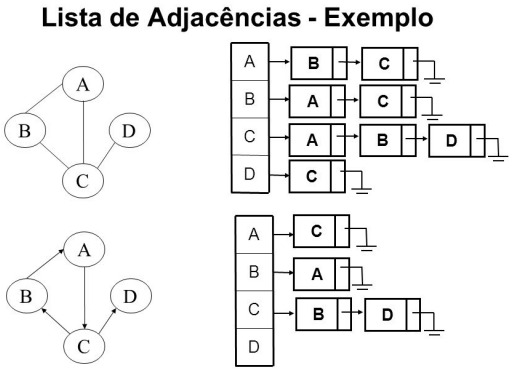

Figura 5 – Exemplo de Lista de Adjacência

3.5 Problema do Caminho Mínimo

Dentre as aplicações práticas dos grafos valorados, destacam-se problemas como o "Problema do Caminho Mínimo", que visa encontrar o trajeto de menor custo entre dois vértices em um grafo, e o "Problema do Fluxo Máximo", que aborda a otimização da transferência de recursos em uma rede. Além disso, os grafos valorados também são utilizados na análise de redes de transporte, sistemas de telecomunicações, logística e roteamento.

4 Questões 01 e 02

O problema da Torre de Hanói pode ser abordado através da modelagem em forma de grafo, onde cada vértice representa uma configuração dos discos e as arestas indicam movimentos legais entre as configurações. O uso da matriz de adjacência e do algoritmo de Dijkstra e permite encontrar o menor caminho para a solução, com o valor 1 em cada aresta indicando o movimento de um disco. Esta abordagem proporciona uma maneira elegante e eficiente de resolver o problema, permitindo a análise do desempenho do algoritmo em termos de tempo de execução.

4.1 Dijkstra

O algoritmo de Dijkstra é uma técnica utilizada para encontrar o caminho mais curto entre um vértice inicial e todos os outros vértices em um grafo ponderado não direcionado ou direcionado. A ideia central é explorar os vértices em ordem crescente de distância estimada a partir do vértice inicial. Conforme avança, o algoritmo relaxa as arestas, ajustando as distâncias estimadas, a fim de determinar o caminho mais curto.

1 - Começamos com uma estimativa infinita de distância para todos os vértices, exceto o vértice inicial, que tem distância zero.
2 - Selecionamos o vértice com a menor distância estimada e o marcamos como visitado.
3 - Para os vértices não visitados adjacentes, calculamos as distâncias estimadas através das arestas e as atualizamos, se necessário.
4 - Repetimos o passo 2 e 3 até todos os vértices estarem visitados ou até que o vértice destino tenha sido visitado.
O resultado final é um conjunto de distâncias mínimas do vértice inicial para todos os outros vértices. O algoritmo de Dijkstra é eficiente em grafos com arestas não negativas, proporcionando uma solução rápida e precisa para problemas de caminho mais curto.

4.2 Ford-Moore-Bellman

O algoritmo de Ford-Moore-Bellman é outro método para encontrar os caminhos mais curtos em um grafo ponderado, sendo também adequado para grafos com arestas de peso negativo, contudo, possivelmente menos eficiente do que o algoritmo de Dijkstra em alguns cenários.

1 - Começamos com distâncias estimadas infinitas para todos os vértices, exceto o vértice inicial, que tem distância zero.
2 - Relaxamos repetidamente todas as arestas (ou seja, ajustamos as estimativas de distância) para cada vértice, buscando melhorar as estimativas.
3 - Repetimos o passo 2 por um número de vezes igual ao número total de vértices menos um.

Se houver um caminho mais curto, após (n - 1) iterações, as distâncias estimadas não mudarão mais. A última iteração detectará se existem ciclos de peso negativo, que podem

resultar em distâncias negativas e indicam que não há solução viável.

4.3 Tempo de Busca de menor caminho

Execução

Digite o vertice de inicio: 0 4
O vertice de destino será: 80
Menor caminho de 0 para 80: 15
Dijkstra: 0.192000 ms Bellman: 0.192000 ms

4.4 Funções:

Esta seção tem como objetivo apresentar as funções que foram criadas e usadas pra resolver os problemas:

- void preencherVertices(Vertice ListaAdjVertices[]);

  Descrição: Essa função preenche a lista de adjacência dos vértices do grafo com as informações das arestas que os conectam a outros vértices. Parâmetros: Um vetor de ponteiros para "Vertice"chamado "ListaAdjVertices".

- int temArco(Vertice vertice, int destino);

  Descrição: Verifica se existe um arco (aresta direcionada) entre um vértice específico e um destino dado. Parâmetros: Um ponteiro para "Vertice"chamado "vertice"e um inteiro "destino"representando o vértice de destino.

- void imprimeVetor(Vertice ListaAdjVertices[]);

  Descrição: Imprime o conteúdo da lista de adjacência dos vértices, exibindo as conexões de cada vértice. Parâmetros: Um vetor de ponteiros para "Vertice"chamado "ListaAdjVertices".

- void preencherMatriz(Vertice ListaAdjVertices[], int MatrizAdj[][QntVertices]); Descrição: Preenche uma matriz de adjacência com informações das conexões entre os vértices, considerando a estrutura de lista de adjacência. Parâmetros: Um vetor de ponteiros para "Vertice"chamado "ListaAdjVertices"e uma matriz de inteiros "MatrizAdj"representando a matriz de adjacência.

- void imprimeMatriz(int MatrizAdj[][QntVertices]);

  Descrição: Imprime o conteúdo de uma matriz de adjacência, visualizando as relações entre os vértices. Parâmetros: Uma matriz de inteiros "MatrizAdj"representando a matriz de adjacência.

- void encontrarMenorCaminho(int MatrizAdj[][QntVertices]);

  Descrição: Função de controle que chama os algoritmos de busca de menor caminho, como Dijkstra e Bellman-Ford. Parâmetros: Uma matriz de inteiros "MatrizAdj"representando a matriz de adjacência.

- void algoDijkstra(int MatrizAdj[][QntVertices], int inicio, int destino);

  Descrição: Implementa o algoritmo de Dijkstra para encontrar o menor caminho entre dois vértices, considerando pesos nas arestas. Parâmetros: Uma matriz de inteiros "MatrizAdj"representando a matriz de adjacência, e dois inteiros "inicio"e "destino"indicando os vértices de partida e chegada.

- void algoBellmanFord(int MatrizAdj[][QntVertices], int inicio, int destino);

  Descrição: Implementa o algoritmo de Bellman-Ford para encontrar o menor caminho entre dois vértices, lidando com arestas de peso negativo. Parâmetros: Uma matriz de inteiros "MatrizAdj"representando a matriz de adjacência, e dois inteiros "inicio"e "destino"indicando os vértices de partida e chegada.

5 Questão 03

Neste problema, temos um grafo orientado com arestas que têm valores representando a confiabilidade do canal de comunicação entre os vértices. O objetivo é criar um programa eficiente para encontrar o caminho entre dois vértices com a maior confiabilidade total.

A abordagem envolve adaptar o algoritmo de Dijkstra para considerar a confiabilidade das arestas. Isso é feito calculando o produto das confiabilidades ao invés da soma dos pesos das arestas percorridas. O algoritmo encontra o caminho entre os vértices de origem e destino que maximiza o produto das confiabilidades.

A implementação cuidadosa e eficiente é fundamental para lidar com estruturas de dados otimizadas, como min-heaps, para melhorar o desempenho do algoritmo.

O resultado final é o caminho entre os vértices de origem e destino com a maior confiabilidade total, juntamente com o valor calculado dessa confiabilidade.

5.1 Funções:

Para o presente problema usamos apenas uma única função. A função "dijkstra"implementa o algoritmo de Dijkstra para encontrar o caminho mais curto entre um vértice de origem e um vértice de destino em um grafo. O algoritmo considera um grafo representado pela estrutura "Grafo"e é aplicado a um total de "numVertices"vértices. Ele busca determinar o caminho mais curto de "origem"para "destino"utilizando os pesos das arestas no grafo.

Parâmetros:

- const Grafo *grafo: Um ponteiro para a estrutura de dados "Grafo", que contém informações sobre os vértices e as arestas do grafo.

- int origem: O índice do vértice de origem a partir do qual o caminho mais curto será calculado.

- int destino: O índice do vértice de destino para o qual se deseja encontrar o caminho mais curto.

- int numVertices: O número total de vértices no grafo.

A função "dijkstra"utiliza o algoritmo de Dijkstra para calcular as distâncias mínimas entre os vértices e determinar o caminho mais curto de "origem"para "destino". Ele considera os pesos das arestas do grafo para determinar as distâncias mínimas ao longo do caminho.

Nota: A função "dijkstra"é uma ferramenta essencial para resolver problemas de caminho mais curto em grafos ponderados, e sua implementação busca encontrar uma solução eficiente para determinar o caminho mais curto entre os vértices de origem e destino em um grafo dado.

5.2 Tempo de Busca do caminho mais confiável

Execução:

Digite o vértice de origem e o vértice de destino: 0 4

O caminho mais confiável entre 0 e 4 tem confiabilidade 0.4500

6 Conclusão

A Teoria dos Grafos é um campo fundamental da matemática discreta, cujo estudo das relações e conexões entre objetos abstratos por meio de vértices e arestas tem aplicações disseminadas em diversas áreas, incluindo ciência da computação, matemática aplicada, engenharia, biologia, redes sociais e logística. Ao longo deste relatório, exploramos os conceitos-chave da Teoria dos Grafos e suas aplicações, proporcionando uma compreensão aprofundada de suas implicações e relevância.

Iniciamos nossa exploração com uma introdução histórica e contemporânea sobre a importância da Teoria dos Grafos. Essa disciplina oferece uma representação versátil e poderosa de fenômenos complexos e sistemas, permitindo a modelagem de interações entre entidades e a resolução de desafios práticos.

Delineamos os principais tipos de grafos, como grafos valorados, direcionados e bipartidos, e examinamos as representações de matriz de adjacência e listas de adjacência, cada uma delas útil para contextos específicos. Além disso, destacamos o "Problema do Caminho Mínimo"como uma aplicação prática dos grafos valorados, discutindo os algoritmos de Dijkstra e Ford-Moore-Bellman para encontrar trajetos mais curtos em grafos ponderados.

Aplicamos os conceitos aprendidos na solução do desafio da Torre de Hanói, modelando-o como um grafo direcionado e adaptando o algoritmo de Dijkstra para encontrar o caminho mais confiável. Também exploramos a resolução do problema de encontrar o caminho com a maior confiabilidade total em um grafo orientado, empregando o algoritmo de Dijkstra com considerações específicas.

Em resumo, a Teoria dos Grafos emerge como uma ferramenta poderosa e versátil para a modelagem e resolução de problemas em diversos campos. Através desta exploração abrangente, esperamos ter fornecido aos leitores um embasamento sólido para compreender a natureza intrínseca dessa teoria, explorar suas aplicações transdisciplinares e reconhecer sua

importância como uma ferramenta analítica e de modelagem em contextos acadêmicos e profissionais.