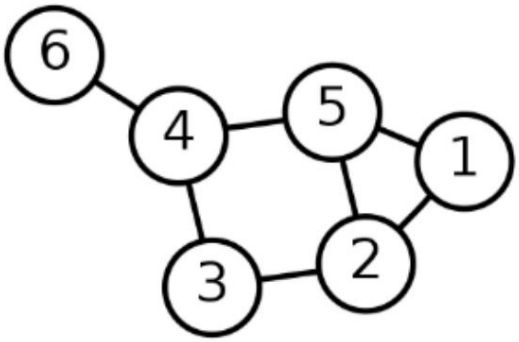
Teoria dos Grafos (Estudo Dirigido)

Aluno: Lucas Barbosa Guimaraes

1 - Defina os seguintes conceitos básicos sobre grafos (vide LAFORE, 2004 – Cap. 13, p. 562): Adjacência; Caminhos; Grafos conectados; Grafos ponderados.

Primeiramente um grafo é uma estrutura de dados não linear constituída de um conjunto de vértices ou nós, V, e um conjunto de arestas ou arcos, A, conectando pares de vértices. Cada arco é especificado por um par de nós.

Adjacência: Dois vértices são adjacentes um ao outro se estiverem conectados por uma única aresta.



Como exemplo acima, o vértice 5 é adjacente aos vértices 4, 1 e 2 e não é adjacente aos vértices 6 e 3.

Caminhos: É uma sequência de arestas para se chegar há um local, como no exemplo acima, o caminho Percorrido do vértice 6 para o 5, ele tem que passar pelo 4.

Grafos conectados: um grafo está conectado se houver pelo menos um caminho de cada vértice para outro vértice. O exemplo acima é um exemplo de grafo conectado.

Grafos ponderados: é quando há uma questão de direção entre os vértices, no caso acima não há essa indicação, portanto entende-se que pode ser percorrido em qualquer direção.

2- Seguindo a ideia de definições de conceitos tratada anteriormente, vide Drozdek (2016 – Cap. 8, p. 340-341) e defina as formas de grafos: Simples; Multigrafo; Ponderado; Subgrafo.

Um **grafo simples** G = (V, E) consiste em um conjunto não vazio V de vértices e um possivelmente conjunto vazio E de arestas, cada aresta sendo um conjunto de dois vértices de V, ou seja, é um grafo que não tem arestas paralelas nem laços.

Um **multigrafo** é um grafo no qual dois vértices podem ser unidos por arestas múltiplas.

Um grafo é chamado de **grafo ponderado** se cada aresta tiver um número atribuído. Dependendo do contexto em que tais grafos são usados, o número atribuído a uma aresta é chamado de peso, custo, distância, comprimento ou algum outro nome

Um **subgrafo** G′ do grafo G = (V,E) é um grafo (V′, E′) tal que V′ contenha V e E ′ contenha E, ou seja, um **subgrafo** de um grafo *G* é um grafo cujo conjunto de vértices é um

subconjunto do conjunto de vértices *G* e o conjunto de arestas é um subconjunto do conjunto de arestas de *G*, ou seja, cuja relação de adjacência é um subconjunto de *G* restrita a esse subconjunto.

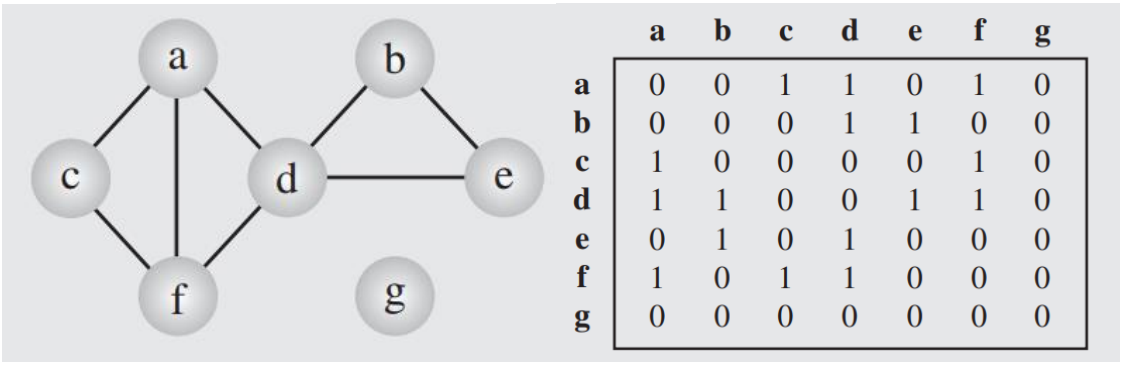
3) Com base nas referências citadas, grafos têm arestas de conexões entre nós controlados por uma matriz, o que permite a implementação destes. Logo, explique como funcionam: Matriz de adjacência; Matriz de incidência.

A **matriz de adjacências** de um grafo é uma matriz booleana com colunas e linhas indexadas pelos vértices. Se adj[][] é uma tal matriz então, para cada vértice v e cada vértice w,

adj[v][w] = 1 se v-w é um arco ;

adj[v][w] = 0 em caso contrário.

Assim, a linha v da matriz adj[][] representa o leque de saída do vértice v e a coluna w da matriz representa o leque de entrada do vértice w. Por exemplo, veja a matriz de adjacências do grafo cujos arcos são:



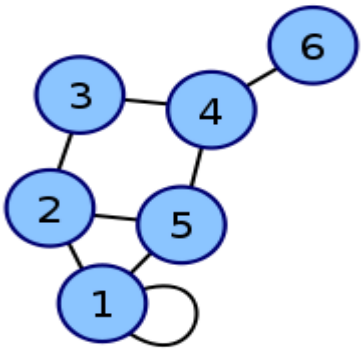
	a	b	c	d	e	f	g
a	0	0	1	1	0	1	0
b	0	0	0	1	1	0	0
c	1	0	0	0	0	1	0
d	1	1	0	0	1	1	0
e	0	1	0	1	0	0	0
f	1	0	1	1	0	0	0
g	0	0	0	0	0	0	0

Uma Matriz de Incidência representa computacionalmente um **grafo** através de uma **Matriz Bidimensional**, onde uma das dimensões são **vértices** e a outra dimensão são **arestas**.

Dado um grafo **G** com **n** vértices e **m** arestas, podemos representá-lo em uma **Matriz n x m M**. A definição precisa das **ENTRADAS** da matriz varia de acordo com as propriedades do grafo que se deseja representar, porém de forma geral **guarda informações sobre como os vértices se relacionam com cada aresta (isto é, informações sobre a incidência de um vértice em uma aresta)**.

Para representar um **Grafo Sem Pesos nas Arestas e Não-Direcionado**, basta que as entradas da matriz **M** contenham **1** se o vértice incide na aresta, **2** caso seja um **Laço** (incide duas vezes) e **0** caso o vértice não incida na aresta.

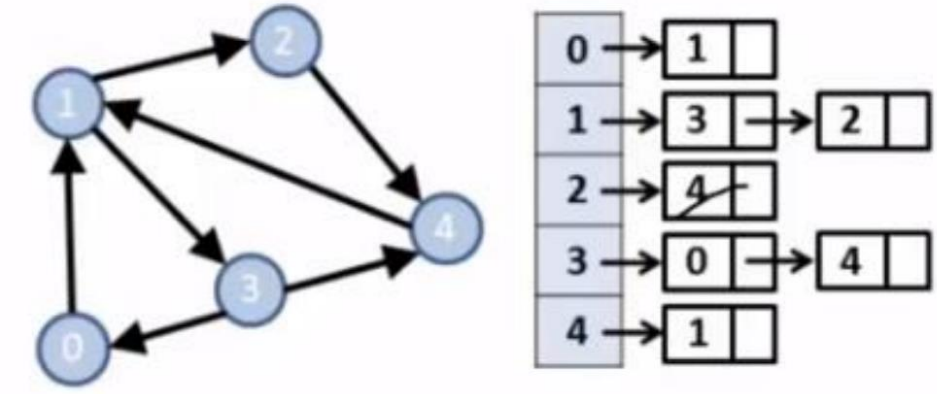
Por exemplo, a matriz de incidência do grafo ao lado é:



$$\begin{bmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

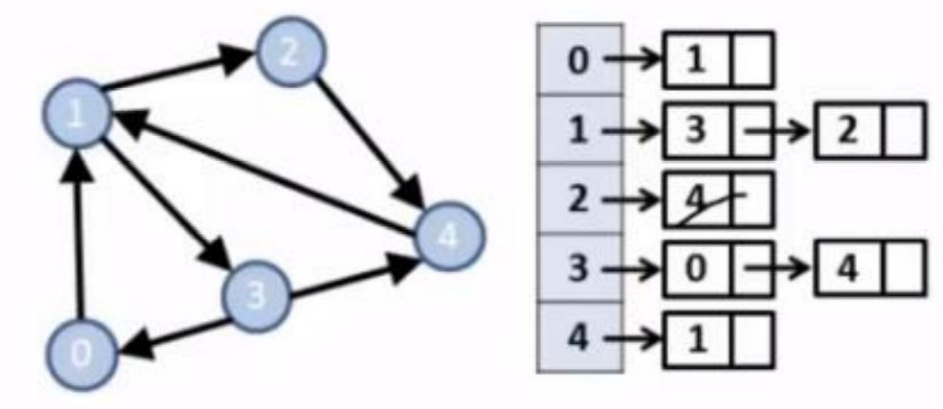
4- Uma das operações mais fundamentais para executar em um grafo é localizar quais nós podem ser alcançados a partir de um nó especificado. Neste contexto, algoritmos de busca são propostos para grafos, muitos deles similares aos aplicados em árvores, como os tipos: Busca em Profundidade e Busca em Largura. Explique, por meio de um exemplo prático, ambas as técnicas de busca em um mesmo grafo. Vide Lafore (2004, Cap. 13, p. 579-586) e/ou Drozdek (2016, Cap. 8, p. 343-354).

Busca em Largura: Dados um grafo G = (V, E) e um vértice s, chamado de fonte, a busca em largura sistematicamente explora as arestas de G de maneira a visitar todos os vértices alcançáveis a partir de s.



Este exemplo a cima mostra como é a visitação de todos os vértices a partir do vértice 0, ele começa do 0 visitando o 1, depois do um ele visita os vértices 2 e 3, no 2 ele consegue visitar o vértice 4, no vértice 3 ele consegue visitar os vértices 0 e 4 e no vértice 4 ele visita o vértice 1.

Busca em Profundidade: O algoritmo de busca DFS visita todos os vértices e todos os arcos do grafo numa determinada ordem e atribui um número a cada vértice: o k-ésimo vértice descoberto recebe o número k.



Seguindo o mesmo exemplo acima, o vértice 0 visita o 1, o vértice 1 visita o 3, o vértice 3 visitas o 4. Ainda falta visitar um vértice, então ele volta a recursividade para o vértice 3, como todos os vizinhos já foram visitados, ele volta ao vértice 1, como ainda falta visitar o vértice 2, é esse caminho que ele toma e finaliza a visita de todos os vértices.

5- Percorrendo um caminho mais curto: suponha que um grafo foi usado para representar caminhos e distâncias entre cidades para um determinado Caixeiro Viajante (exemplo clássico da Teoria de Grafos). Explique como o algoritmo de Dijkstra poderia ser aplicado para escolha de uma rota ótima entre cidades? Vide exemplo e teoria em Drozdek (2016 – Cap. 8, p. 346-350).

O algoritmo de Dijkstra é uma solução para o problema do caminho mínimo de origem única. Funciona em grafos orientados e não orientados, no entanto, todas as arestas devem ter custos não negativos. Se houver custos negativos, usa-se o algoritmo de Bellman-Ford. Entrada: Grafo ponderado G=(N,E) e nó origem O N, de modo que todos os custos das arestas sejam não negativos.

Saída: Comprimentos de caminhos mais curtos (ou os caminhos mais curtos em si) de um determinado nó origem O N para todos os outros nós.

O algoritmo de Dijkstra identifica, a partir do nó O, qual é o custo mínimo entre esse nó e todos os outros do grafo. No início, o conjunto S contém somente esse nó, chamado origem. A cada passo, selecionamos no conjunto de nós sobrando, o que está mais perto da origem. Depois atualizamos, para cada nó que está sobrando, a sua distância em relação à origem. Se passando pelo novo nó acrescentado, a distância ficar menor, é essa nova distância que será memorizada.

Escolhido um nó como origem da busca, este algoritmo calcula, então, o custo mínimo deste nó para todos os demais nós do grafo. O procedimento é iterativo, determinando, na iteração 1, o nó mais próximo do nó O, na segunda iteração, o segundo nó mais próximo do nó O, e assim sucessivamente, até que em alguma iteração todos os n sós sejam atingidos. Seja G=(N,E) um grafo orientado e s um nó de G:

- Atribua valor zero à estimativa do custo mínimo do nó O (a origem da busca) e infinito às demais estimativas;
- Atribua um valor qualquer aos precedentes (o precedente de um nó t é o nó que precede t no caminho de custo mínimo de s para t);
- Enquanto houver nó aberto:

- seja k um nó ainda aberto cuja estimativa seja a menor dentre todos os nós abertos;

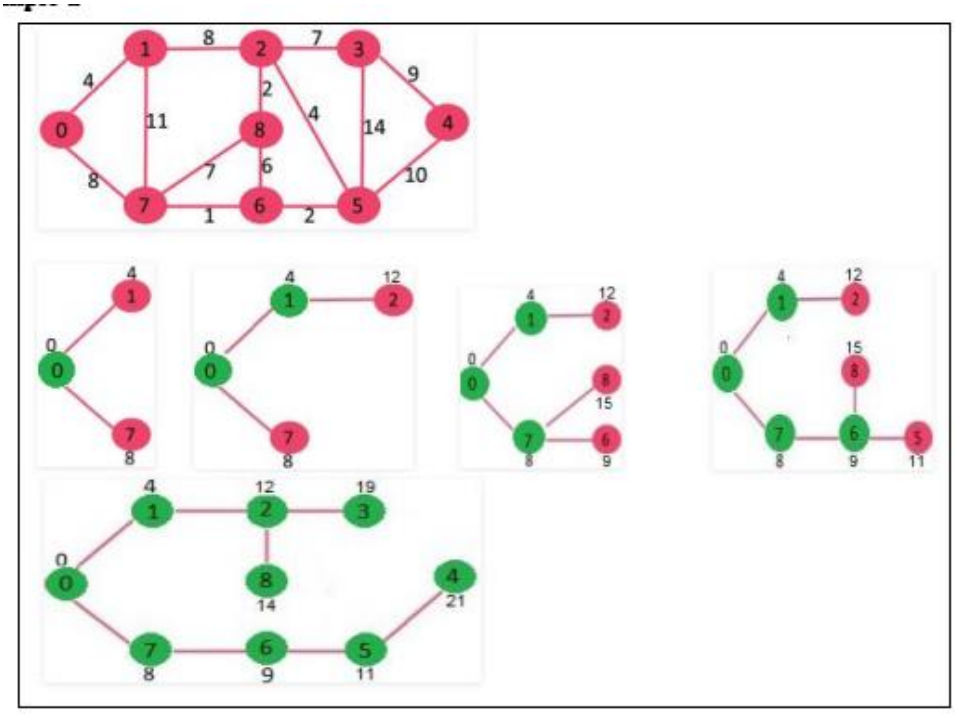
- feche o nó k;

- Para todo nó j ainda aberto que seja sucessor de k faça:

some a estimativa do nó k com o custo do arco que une k a j;

caso esta soma seja melhor que a estimativa anterior para o nó j, substitua-a e anote k como precedente de j.

Exemplo:



Em suma, ele percorreria o os caminhos e verificaria o que fosse de menor custo, sendo o de menor custo ele altera a rota para o menor valor fazendo assim até percorrer todo o vetor e no final mostraria o menor caminho do ponto inicial escolhido até o ponto final.

6- Existem situações-problema em que um grafo tem que ser reduzido a um número mínimo de conexões ou arestas entre os nós. Neste caso, o grafo é traduzido em uma Árvore de Espalhamento (ou chamada de Árvore geradora mínima). Assim, leia as referências Drozdek (2016, Cap. 8, p. 356 – seção 8.5) e Lafore (2004, Cap. 13, p. 587), e explique em detalhes como uma árvore de espalhamento (ou geradora mínima) pode ser gerada a partir de um grafo. Explique na forma de um exemplo teórico.

Dado um grafo não orientado conectado, uma árvore de extensão deste grafo é um subgrafo o qual é uma árvore que conecta todos os vértices. Um único grafo pode ter diferentes árvores de extensão. Nós podemos assinalar um peso a cada aresta, que é um número que representa quão desfavorável ela é, e atribuir um peso a árvore de extensão calculado pela soma dos pesos das arestas que a compõem. Uma árvore de extensão mínima (também conhecida como árvore de extensão de peso mínimo ou árvore geradora mínima) é então uma árvore de extensão com peso menor ou igual a cada uma das outras árvores de extensão possíveis. Generalizando mais, qualquer grafo não direcional (não

necessariamente conectado) tem uma floresta de árvores mínimas, que é uma união de árvores de extensão mínimas de cada uma de suas componentes conexas.

Um exemplo de uso de uma árvore de extensão mínima seria a instalação de fibras óticas num campus de uma faculdade. Cada trecho de fibra ótica entre os prédios possui um custo associado (isto é, o custo da fibra, somado ao custo da instalação da fibra, mão de obra, etc). Com esses dados em mãos (os prédios e os custos de cada trecho de fibra ótica entre todos os prédios), podemos construir uma árvore de extensão que nos diria um jeito de conectarmos todos os prédios sem redundância. Uma árvore geradora mínima desse grafo nos daria uma árvore com o menor custo para fazer essa ligação.