Estrutura de dados - Aula IV

...

- Grafos
- O que são
- Como funcionam

Profª Letícia Pieper

1 - Grafos

Grafos são uma das estruturas de dados mais fundamentais e versáteis da ciência da computação e da matemática discreta. Eles são usados para representar relações e conexões entre objetos, sendo uma forma poderosa de modelar uma ampla variedade de problemas.

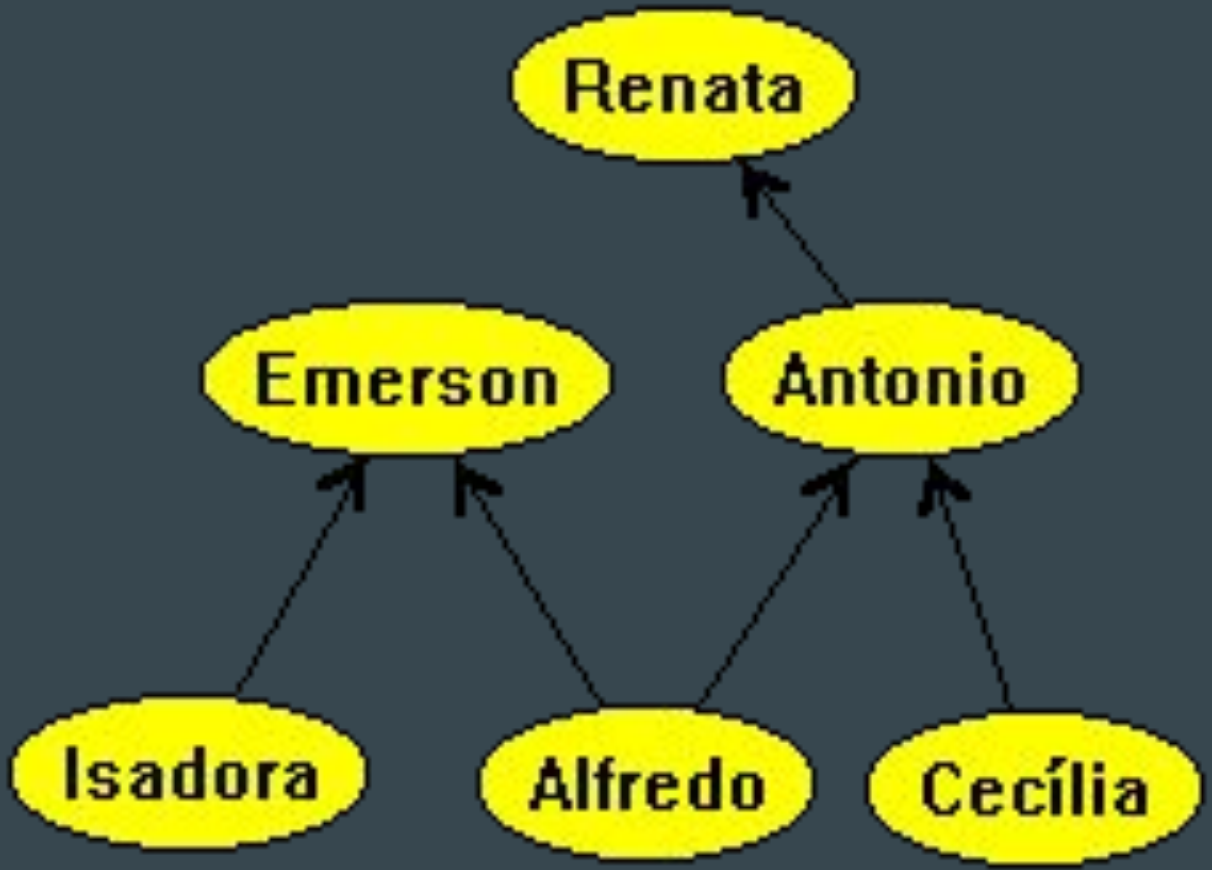
Um grafo consiste em dois componentes principais: vértices (ou nós) e arestas (ou arcos), que conectam esses vértices. Aqui estão os principais conceitos relacionados a grafos:

Vértices (Nós):

Os vértices são os pontos fundamentais do grafo e representam objetos ou entidades. Em um grafo social, por exemplo, cada pessoa pode ser representada por um vértice.

Arestas (Arcos):

As arestas são as conexões entre os vértices e representam as relações entre os objetos. Em um grafo social, uma aresta pode indicar que duas pessoas são amigas.


Renata
Emerson
Antonio
Isadora
Alfredo
Cecília

Grafo Direcionado vs. Grafo Não Direcionado:

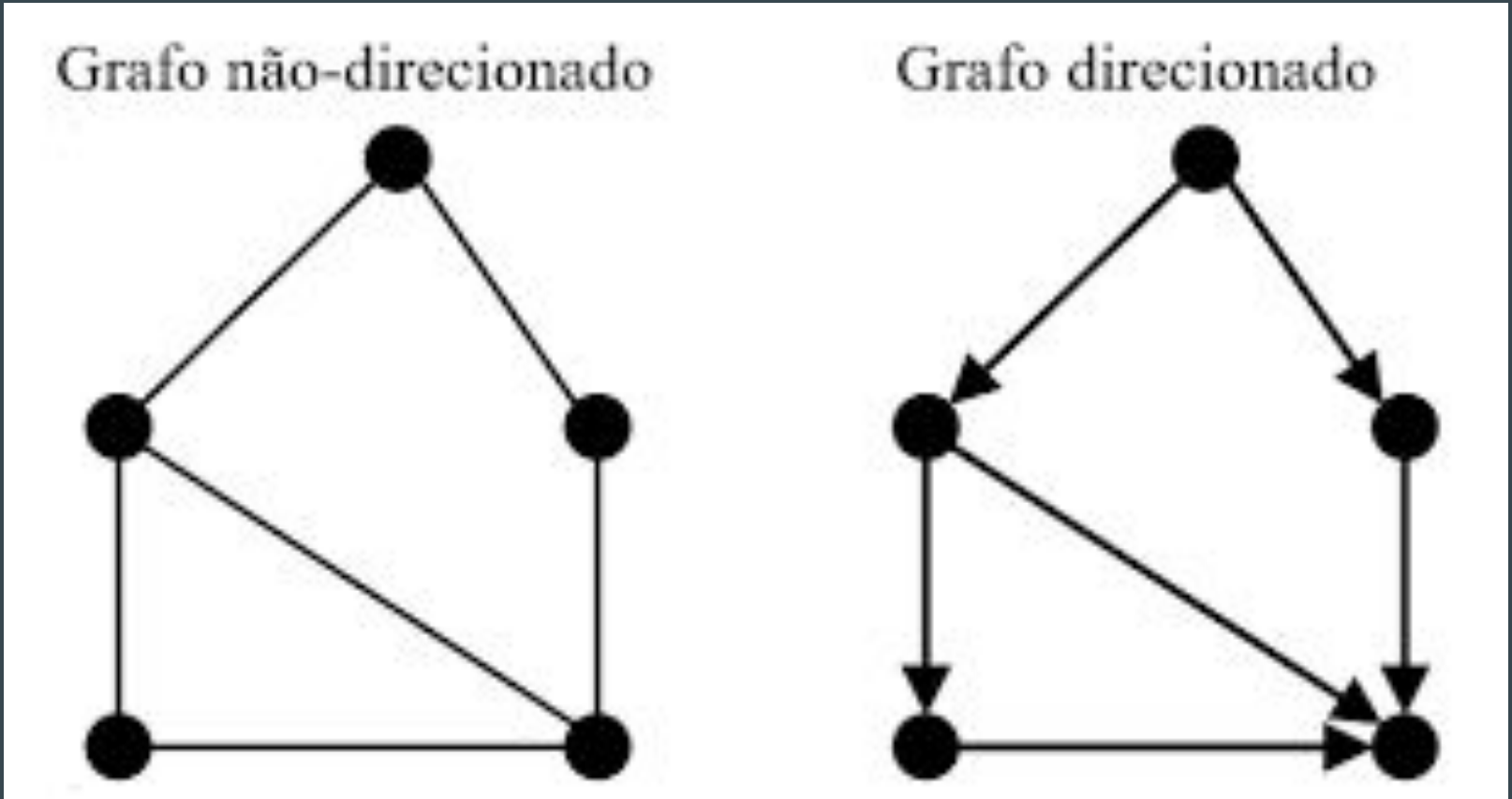
Grafo Direcionado:

Em um grafo direcionado, as arestas têm uma direção específica, indicando que a relação entre dois vértices é unidirecional. Por exemplo, em um grafo de seguidores no Twitter, um usuário segue outro, mas essa ação não implica que o segundo usuário também segue o primeiro.

Grafo Não Direcionado:

Em um grafo não direcionado, as arestas não têm uma direção específica e indicam uma relação bidirecional. Se dois vértices estão conectados por uma aresta, isso significa que a relação é mútua. Por exemplo, em uma rede de amizades, se uma pessoa é amiga de outra, a amizade é recíproca.

Grafo Direcionado vs. Grafo Não Direcionado:



Grafo Ponderado vs. Grafo Não Ponderado:

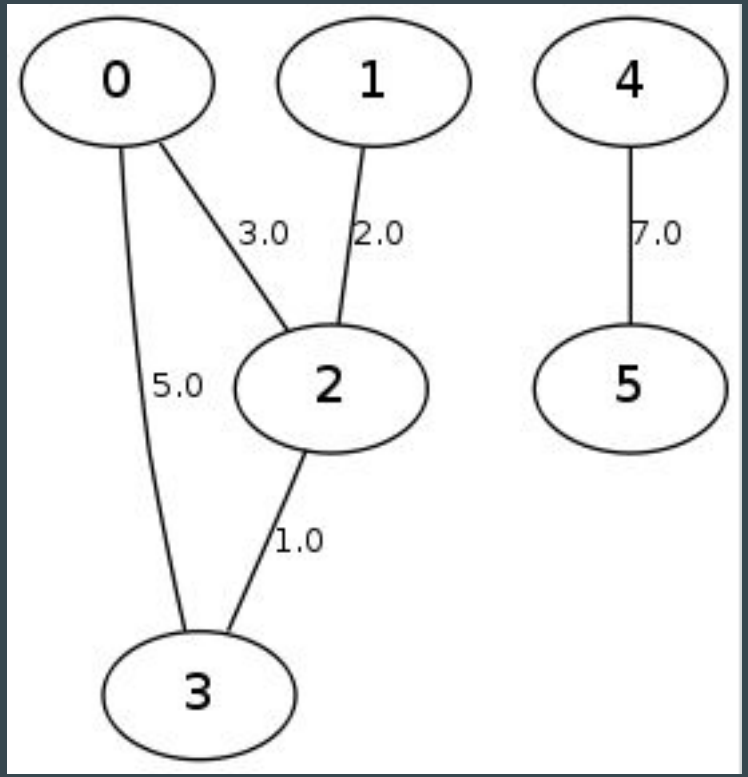
Grafo Ponderado:

Em um grafo ponderado, cada aresta tem um valor associado, chamado de peso, que representa uma medida de custo, distância, tempo, ou qualquer outra métrica relevante. Por exemplo, em um grafo de rotas de voo, as arestas podem ser ponderadas com os custos dos bilhetes.

Grafo Não Ponderado:

Em um grafo não ponderado, todas as arestas têm o mesmo valor ou são consideradas igualmente importantes. Por exemplo, em um grafo de redes sociais, as amizades podem ser representadas como arestas não ponderadas.

Grafo Ponderado:



Utilização de Grafos:

Grafos são amplamente usados para modelar sistemas e problemas do mundo real, tais como redes de computadores, mapas de rotas, redes sociais, circuitos elétricos, estruturas de dados como árvores e muitos outros.

Algoritmos de grafos são usados para resolver problemas como a busca de caminhos mais curtos, encontrar ciclos, descobrir componentes fortemente conectados, entre outros.

O conceito de "caminho mínimo" em um grafo envolve mostrar como encontrar o caminho mais curto entre dois vértices em um grafo ponderado, onde as arestas têm pesos associados.

Objetivo do Caminho Mínimo:

Encontrar o "caminho mínimo" em um grafo é descobrir a rota mais curta entre dois vértices específicos. Isso é útil em muitos cenários, como encontrar a rota mais curta em um sistema de transporte, o caminho de menor custo em uma rede de computadores, ou o menor tempo para percorrer uma sequência de cidades em um mapa.

Existem várias variantes para problemas de caminho mínimo, cada uma adequada a um conjunto de problemas diferente:

- Problema de único destino: consiste em determinar o menor caminho entre cada um dos nós do grafo e um nó de destino dado.
- Problema de único origem: determinar o menor caminho entre um nó dado e todos os demais nós do grafo.
- Problema de origem-destino: determinar o menor caminho entre nós dados.
- Problemas de todos os pares: determinar o menor caminho entre cada par de nós presentes no grafo.

A técnica de relaxamento é um conceito **fundamental** em algoritmos de busca de caminho mais curto, como o algoritmo de Dijkstra e o algoritmo de Bellman-Ford. Ela é usada para atualizar as distâncias mínimas de um vértice de origem para todos os outros vértices à medida que o algoritmo progride.

A técnica de relaxamento realiza sucessivas aproximações das distâncias até finalmente chegar na solução. A principal diferença entre Dijkstra e Bellman-Ford é que no algoritmo de Dijkstra é utilizada uma fila de prioridades para selecionar os vértices a serem relaxados, enquanto o algoritmo de Bellman-Ford simplesmente relaxa todas as arestas.

Algoritmos de Caminho Mínimo: Algoritmo de Dijkstra.

O algoritmo de Dijkstra é um algoritmo de busca em grafos usado para encontrar o caminho mais curto entre um vértice de origem e todos os outros vértices em **um grafo ponderado**. Ele funciona apenas em grafos **com pesos não negativos**, o que significa que os valores associados às arestas devem ser números não menores que zero.

Passos para executar o algoritmo de Dijkstra:

Entrada:

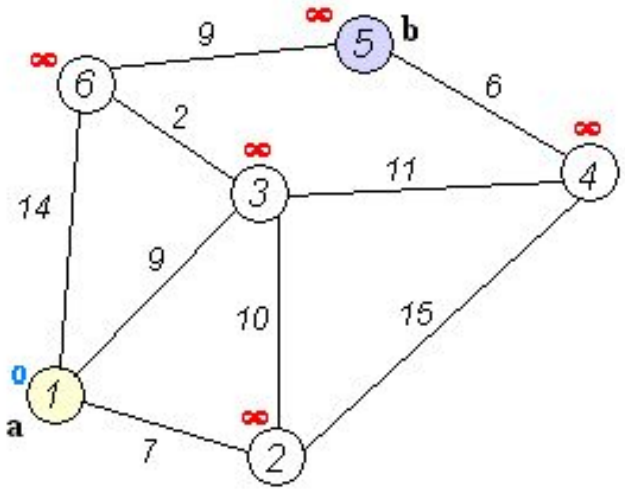
- Um grafo direcionado ou não direcionado com vértices, arestas e pesos não negativos associados a cada aresta.
- Um vértice de origem a partir do qual você deseja encontrar os caminhos mais curtos para todos os outros vértices no grafo.

Algoritmos de Caminho Mínimo: Algoritmo de Dijkstra.

Formula: O(E + V log(V)) - onde V é o número de vértices e E é o número de arestas.

O algoritmo considera um conjunto S de menores caminhos, iniciado com um vértice inicial I. A cada passo do algoritmo busca-se nas adjacências dos vértices pertencentes a S aquele vértice com menor distância relativa a I e adiciona-o a S e, então, repetindo os passos até que todos os vértices alcançáveis por I estejam em S. Arestas que ligam vértices já pertencentes a S são desconsideradas.

Um exemplo prático de problema que pode ser resolvido pelo algoritmo de Dijkstra é: alguém precisa se deslocar de uma cidade para outra. Para isso, ela dispõe de várias estradas, que passam por diversas cidades. Qual delas oferece uma trajetória de menor caminho?



O **Algoritmo de Bellman-Ford** é um algoritmo de busca de caminho mínimo em um *grafo* (grafo orientado ou dirigido) ponderado, ou seja, cujas arestas têm peso, inclusive negativo. O Algoritmo de Dijkstra resolve o mesmo problema, num tempo menor, porém exige que todas as arestas tenham pesos positivos. Portanto, o algoritmo de Bellman-Ford é **normalmente usado apenas quando existem arestas de peso negativo**.

O algoritmo de Bellman-Ford executa em tempo O(VxE) - onde *V* é o número de vértices e *E* o número de arestas.

Assim como o algoritmo de Dijkstra, o algoritmo de Bellman-Ford utiliza a técnica de relaxamento, ou seja, realiza sucessivas aproximações das distâncias até finalmente chegar na solução. A principal diferença entre Dijkstra e Bellman-Ford é que no algoritmo de Dijkstra é utilizada uma fila de prioridades para selecionar os vértices a serem relaxados, enquanto o algoritmo de Bellman-Ford simplesmente relaxa todas as arestas.

Algoritmo de Bellman-Ford

https://www.youtube.com/watch?app=desktop&v=bJMGARLV2ZM

Algoritimo de Dijkstra

https://www.youtube.com/watch?v=IIZOWRwKa_Q