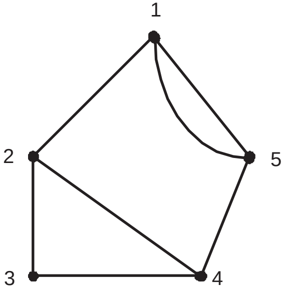
Conteúdo

Representações Computacionais

Para poder utilizar os grafos na modelagem e resolução de problemas computacionais, é necessário utilizar estruturas de dados que permitam armazená-los eficientemente em meios digitais.

Matriz de Adjacência – Grafos Não Direcionados

Seja $G = (V, E)$ um grafo, onde $V = \{1, \ldots, n\}$. A entrada (i, j) de uma matriz de adjacência indica o número de arestas que tem como extremidades i e j.



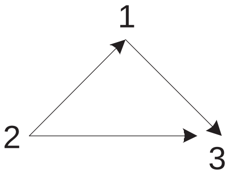
	1	2	3	4	5
1	0	1	0	0	2
2	1	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	2	0	0	1	0

Matriz de Adjacência – Grafos Não Direcionados

A matriz é simétrica, como podemos observar na Figura. Esta simetria permite guardar somente os elementos da diagonal principal e os elementos abaixo (ou acima) dela. Dessa forma economiza-se espaço no armazenamento da estrutura.

Matriz de Adjacência – Grafos Direcionados

Seja $G = (V, E)$ um dígrafo, onde $V = \{1, \ldots, n\}$. A entrada (i, j) de uma matriz de adjacência indica o número de arestas que tem i como cauda e j como cabeça. Neste caso, a matriz não é necessariamente simétrica:



	1	2	3
1	0	0	1
2	1	0	1
3	0	0	0

Matriz de Adjacência – Grafos Direcionados

Dentre as propriedades das matrizes de adjacências, destacamos:

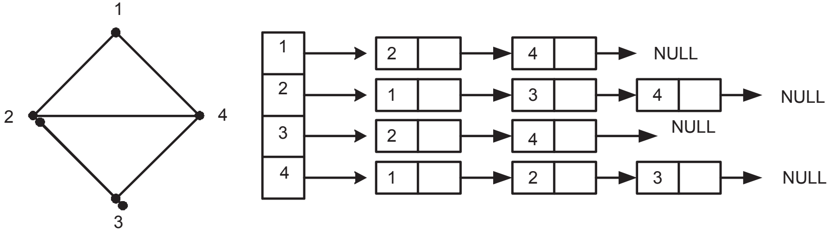
- Necessita cerca de n^2 posições de memória.
- A existência de uma aresta pode ser testada com uma única operação.
- Para listar todos os vértices e arestas do grafo precisamos de cerca de n^2 operações.

Lista de Adjacência

Em muitos casos, lidamos com grafos esparsos, ou seja, com "poucas" arestas. Nesse caso é um desperdício utilizar n^2 posições de memória. A proxima figura mostra um grafo com 5 vértices e 4 arestas, e sua matriz de adjacências. Observe que a maioria das entradas são nulas.

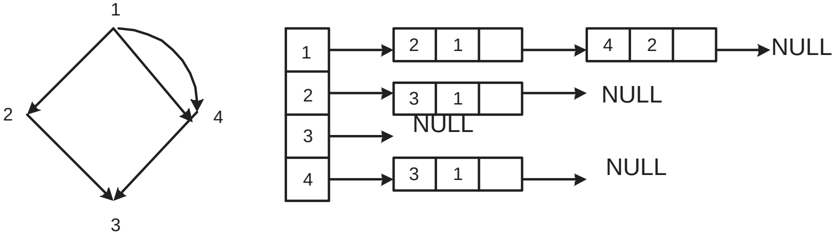
	1	2	3	4	5
1	0	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	0	0	1
5	0	0	0	0	0

Uma alternativa para evitar este desperdício, é utilizar um vetor de listas encadeadas, onde a lista correspondente a *i-ésima* posição guarda os elementos adjacentes ao vértice *i*.



Lista de Adjacência

Para representar arestas paralelas em listas de adjacências, podemos utilizar um campo extra para guardar a multiplicidade da aresta.



Lista de Adjacência

Dentre as características das listas de adjacências destacamos:

- cerca de $n + |E|$ posições de memória são necessárias.
- Para descobrir se uma aresta pertence ao grafo, pode ser necessário percorrer uma lista encadeada inteira.
- O grafo pode ser percorrido em um tempo proporcional ao número de arestas.

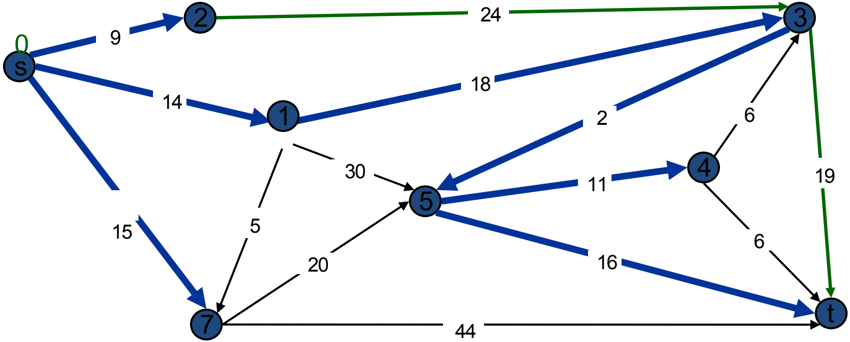
Conteúdo

Seja G um grafo direcionado com pesos **positivos** nas arestas. Um problema importante, que surge em diversas aplicações, é o de determinar o caminho mais curto entre dois vértices do grafo:

Entrada. Um grafo direcionado $G = (V, E)$, uma origem s, um destino t e uma função w que associa cada aresta de E a um peso(valor positivo).

Saída. O caminho de menor peso entre s e t, onde o peso de um caminho é a soma dos pesos das arestas do caminho.

No grafo seguinte, as arestas dos caminhos de menor peso entre s e os demais vértices do grafo estão em azul. O caminho de menor peso entre s e t é $(s, 1, 3, 5, 4, t)$ e tem peso 50.



Se tentarmos resolver este problema utilizando força bruta, teremos que gerar todos os caminhos entre s e t e selecionar o de menor peso.

Esta abordagem é muito cara computacionalmente já que o número de caminhos pode ser exponencial no número de vértices do grafo.

Lema

Seja P o caminho de menor peso entre dois vértice u e v e seja w um vértice de P. Portanto, o subcaminho de P que começa em u e termina em w é o caminho mais curto entre u e w.

Lema

Seja P o caminho de menor peso entre dois vértice u e v e seja w um vértice de P. Portanto, o subcaminho de P que começa em u e termina em w é o caminho mais curto entre u e w.

Prova. Se isso não fosse verdade poderíamos obter um caminho de peso menor que P substituindo o subcaminho de P que começa em u e termina em w por um caminho de menor peso.

Vamos definir a distância de s a um vértice v como o peso do caminho de menor peso entre s e v. Dizemos que um vértice u é mais próximo a s do que v se a distância entre s e u é menor que a distância entre s e v.

Em linhas gerais a nossa abordagem será encontrar o caminho de menor peso até o vértice mais próximo de s, depois o caminho de menor peso até o segundo vértice mais próximo a s e assim por diante...

Seja S_k o conjunto dos k vértices mais próximos de s. Assuma que já conhecemos o caminho mais curto de s até cada um dos vértices em S_{k-1}. Vamos tentar encontrar então P_{sk}, o caminho mais curto de s até v_k, onde v_k é o k-ésimo vértice mais próximo de s.

Apesar de não conhecermos quem é v_k, tampouco P_{sk}, podemos tirar algumas conclusões. A primeira delas diz respeito ao predecessor de v_k em P_{sk}:

Lema

Seja u o predecessor de v_k no caminho de menor peso entre s e v_k. Temos que $u \in S_{k-1}$

Lema

Seja u o predecessor de v_k no caminho de menor peso entre s e v_k. Temos que $u \in S_{k-1}$

Prova. O peso do caminho P_{su} é menor que o peso de do caminho P_{sv}. Portanto, se u não pertencesse a S_{k-1}, v_k não poderia ser o k-ésimo vértice mais próximo a s, o que contradiz a definição v_k. Logo, $u \in S_{k-1}$.

Uma outra conclusão é que o subcaminho de P_{sk} que começa em s e termina em u é o caminho de menor peso entre s e u. Esta é uma consequência imediata do Lema.

A partir destas conclusões deduzimos que P_{sk} é composto de um caminho de peso mínimo entre s e algum vértice u pertencente a S_{k-1} e de uma aresta deste vértice u até v_k.

Portanto, para encontrar P_{sk} podemos considerar todos os caminhos que contém a estrutura descrita e ficar com aquele de menor peso.

`EncontraCaminhoPesoMínimo`(s, t)

$S \leftarrow \{s\}$;

Enquanto $t \notin S$

Esvazie a lista $\mathcal{L}$ dos caminhos candidatos

Para todo vértice $u \in S$

Para todo vértice v adjacente a u tal que $v \notin S$

Inclua o caminho $P_{su} \rightarrow v$ na lista dos caminhos candidatos.

Fim Para

Fim Para

$P \leftarrow$ o caminho de menor peso na lista $\mathcal{L}$ dos caminhos candidatos

$v \leftarrow$ último vértice do caminho P; $P_{sv} \leftarrow P$

Adicione v a S

Fim Enquanto