



Grafos na Ciência da Computação — Teoria Formal



Definição Formal

Na matemática e ciência da computação, um **grafo** é definido como:

\$

$$G = (V, E)$$

\$

Onde:

- V é um conjunto finito de **vértices** (ou *nós*).
- $E \subseteq V \times V$ é um conjunto de **arestas** (ou *ligações*), que conectam pares de vértices.



Tipos de Grafos

1. Grafo não direcionado (ou simples)

As arestas não têm direção:

\$

$$E = \{ \{u, v\} \mid u, v \in V \}$$

\$

2. Grafo direcionado (ou dígrafo)

Cada aresta aponta de um vértice para outro:

\$

$$E = \{ (u, v) \mid u, v \in V \}$$

\$

3. Multigrafo

Permite múltiplas arestas entre o mesmo par de vértices.

4. Grafo ponderado

Cada aresta tem um peso $w(u, v) \in \mathbb{R}$, comum em problemas como caminhos mínimos (e.g., Dijkstra).



Propriedades Importantes

Conceito	Descrição
----------	-----------

Conceito	Descrição
Grau do vértice	Número de arestas conectadas a ele. Em grafos direcionados: grau de entrada/saída.
Caminho	Sequência de vértices conectados por arestas.
Ciclo	Caminho que começa e termina no mesmo vértice.
Conectividade	Grafo é conectado se existe um caminho entre qualquer par de vértices.
Árvore	Grafo acíclico e conectado.
Grafo bipartido	Os vértices podem ser divididos em dois conjuntos disjuntos com arestas só entre conjuntos.
Subgrafo	Um grafo formado a partir de subconjuntos de vértices e arestas do grafo original.

Representações em Computação

1. Matriz de Adjacência

- Tamanho $|V| \times |V|$
- Útil para grafos densos.
- Teste de adjacência é $O(1)$.

2. Lista de Adjacência

- Lista onde cada vértice aponta para seus vizinhos.
- Eficiência espacial $O(|V| + |E|)$
- Ideal para grafos esparsos.

3. Lista de Arestas

- Lista de pares (u, v) ou trios (u, v, w) se ponderado.
- Boa para algoritmos de ordenação de arestas (e.g., Kruskal).

Algoritmos Fundamentais

Algoritmo	Aplicação	Complexidade
BFS (Busca em Largura)	Conectividade, caminhos mínimos em grafos não ponderados	$O(V + E)$
DFS (Busca em Profundidade)	Ciclos, componentes, ordenação topológica	$O(V + E)$
Dijkstra	Caminho mínimo com pesos positivos	$O((V + E) \log V)$
Bellman-Ford	Caminho mínimo com pesos negativos	$O(VE)$

Algoritmo	Aplicação	Complexidade
Floyd-Warshall	Todos os caminhos mínimos	$O(V^3)$
Kruskal / Prim	Árvore Geradora Mínima (MST)	$O(E \log V)$
Kosaraju / Tarjan	Componentes fortemente conexos	$O(V + E)$

Aplicações na Ciência da Computação

- **Sistemas Operacionais:** Deadlock (grafo de espera por recursos)
- **Redes:** Roteamento de pacotes, conectividade de redes
- **Compiladores:** Dependência entre tarefas (grafo de dependência)
- **Teoria dos Jogos:** Representação de estratégias e estados
- **Inteligência Artificial:** Busca em grafos para planejamento
- **Web:** PageRank (grafo de links), recomendação

Classificação dos Grafos por Complexidade

Tipo de grafo	Denso	Esparso	Cíclico	Acíclico
Exemplo	Rede social	Árvore	Roteamento de redes	DAG (dependências de build)

Resumo Visual

```

Grafo (V, E)
├── Direcionado ou Não
├── Ponderado ou Não
├── Representações
│   ├── Matriz de Adjacência
│   ├── Lista de Adjacência
│   └── Lista de Arestas
└── Algoritmos
    ├── BFS / DFS
    ├── Dijkstra / Bellman-Ford
    └── Kruskal / Prim
  
```

Algoritmo de Dijkstra

O algoritmo de Dijkstra é usado para encontrar o caminho mais curto de um vértice de origem para todos os outros vértices em um grafo ponderado com pesos não negativos. Ele é amplamente utilizado em sistemas de roteamento e navegação.

Funcionamento

1. Inicialização:

- Defina a distância do vértice de origem como 0 e de todos os outros vértices como infinito (∞).
- Use uma fila de prioridade (geralmente implementada como uma heap) para armazenar os vértices com base em suas distâncias mínimas.

2. Processamento:

- Extraia o vértice com a menor distância da fila de prioridade.
- Para cada vizinho do vértice extraído, calcule a distância total até ele passando pelo vértice atual.
- Se a nova distância calculada for menor que a distância armazenada, atualize-a e insira o vizinho na fila de prioridade.

3. Finalização:

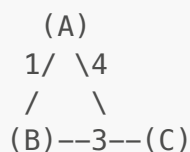
- Repita o processo até que todos os vértices tenham sido processados ou a fila de prioridade esteja vazia.

Complexidade

- **Tempo:**
 - $O(n + m \log n)$, onde n é o número de vértices e m é o número de arestas, assumindo o uso de uma fila de prioridade eficiente.
- **Espaço:**
 - $O(n)$, para armazenar as distâncias e o estado dos vértices.

Exemplo

Considere o seguinte grafo:



- Origem: A
- Passos:
 1. Inicialize: $\text{dist}(A)=0$, $\text{dist}(B)=\infty$, $\text{dist}(C)=\infty$.
 2. Atualize as distâncias dos vizinhos de A: $\text{dist}(B)=1$, $\text{dist}(C)=4$.
 3. Extraia B e atualize $\text{dist}(C)$ via B: $\text{dist}(C)=4$.
 4. Finalize: $\text{dist}(A)=0$, $\text{dist}(B)=1$, $\text{dist}(C)=4$.

Aplicações

- Sistemas de navegação GPS.

- Redes de computadores para encontrar rotas ótimas.
 - Planejamento de rotas em jogos.
-