Centro Paula Souza

Fatec Jahu

GRAFOS EM JAVASCRIPT

ESTRUTURA DE DADOS

CST em Desenvolvimento de Software Multiplataforma

PROF. Me. TIAGO A. SILVA

PARA SOBREVIVER AO JAVASCRIPT

Non-zero value

0

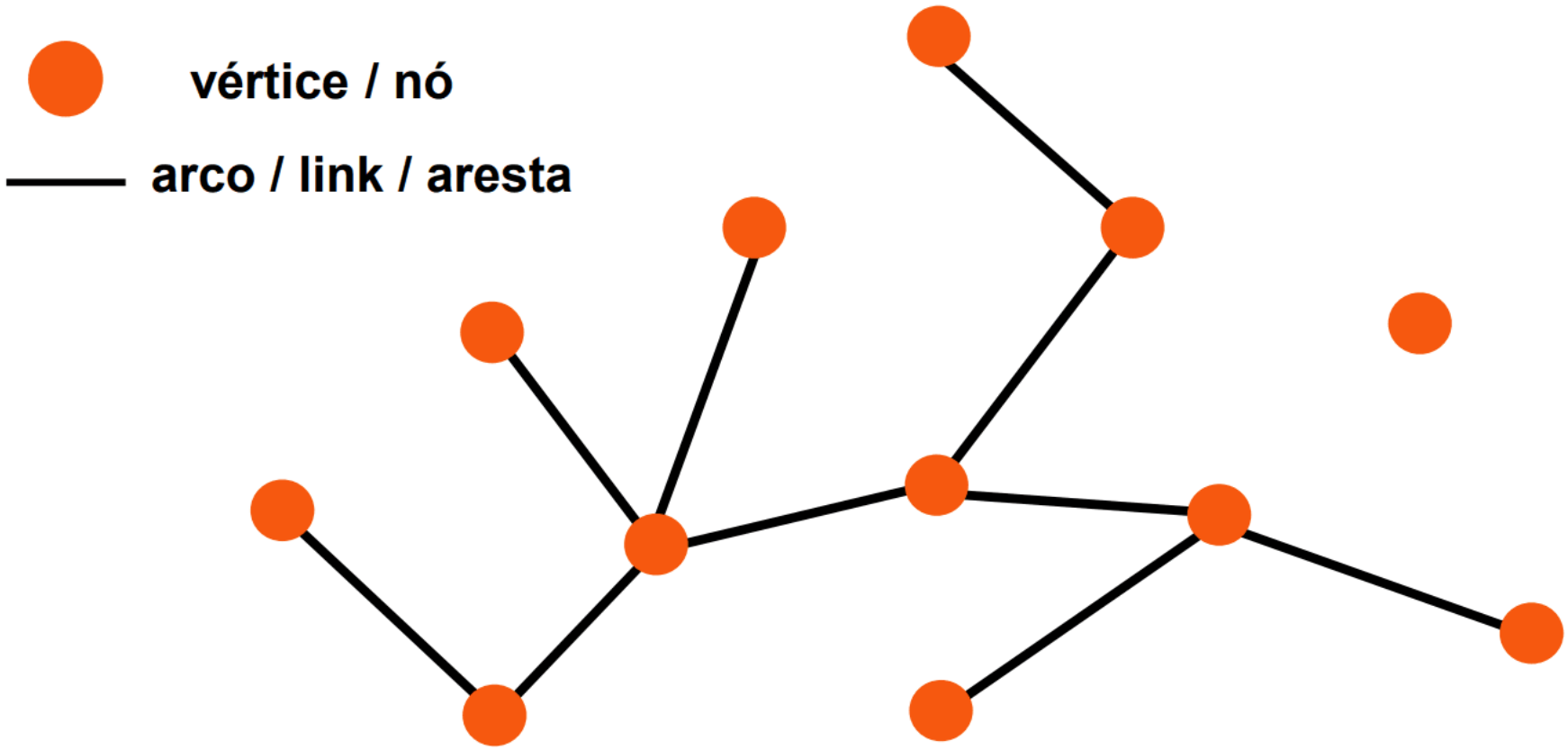
null

undefined

O QUE SÃO GRAFOS?

- Um grafo é uma estrutura de dados usada para representar relações ou conexões entre um conjunto de elementos. Ele é composto por:
  - Vértices (ou nós): os elementos individuais.
  - Arestas: as conexões entre os vértices.
- Os grafos são amplamente usados em problemas reais como redes sociais, mapas de rotas, sistemas de recomendação e muito mais. Existem diferentes tipos de grafos:
  - Dirigidos: as arestas possuem direção.
  - Não dirigidos: as arestas não têm direção.
  - Ponderados: as arestas têm peso ou custo associado.
  - Não ponderados: as arestas não possuem peso.

O QUE SÃO GRAFOS?



O QUE VAMOS IMPLEMENTAR?

```
1   class Grafo {
2   >   constructor() {···
5       }
6
7       // Adiciona um novo vértice ao grafo.
8   >   adicionarVertice(vertice) {···
12      }
13
14      // Adiciona uma aresta entre dois vértices (não dirigido).
15  >   adicionarAresta(vertice1, vertice2) {···
24      }
25
26      // Remove uma aresta entre dois vértices.
27  >   removerAresta(vertice1, vertice2) {···
30      }
31
32      // Remove um vértice e suas conexões.
33  >   removerVertice(vertice) {···
39      }
40
41      // Exibe o grafo.
42  >   imprimirGrafo() {···
46      }
47  } // Fecha classe Grafo.
```

CONSTRUTOR DA CLASSE GRAFO

- **Função**: Inicializa um grafo vazio.
- **Estrutura de dados utilizada**: Um objeto chamado adjacência é usado para armazenar os vértices e suas conexões (arestas).
- **Estado inicial**: Um grafo sem nenhum vértice ou aresta.

```
1   class Grafo {
2       constructor() {
3           // Estrutura para armazenar os vértices e suas conexões.
4           this.adjacencia = {};
5       }
```

MÉTODO PARA ADICIONAR UM VÉRTICE

- Adiciona um novo vértice ao grafo. Com parâmetro:
  - **`vertice`**: o nome ou identificador do vértice (pode ser um número, string, etc.).
- Verifica se o vértice já existe no grafo, se não existir, adiciona o vértice com uma lista vazia, que será usada para armazenar as arestas conectadas a ele.

```
7     // Adiciona um novo vértice ao grafo.
8     adicionarVertice(vertice) {
9         if (!this.adjacencia[vertice]) {
10            this.adjacencia[vertice] = [];
11        }
12    }
```

ADICIONA UMA ARESTA ENTRE DOIS VÉRTICES

- Cria uma conexão (aresta) entre dois vértices. Com parâmetros: **`vertice1`** e **`vertice2`**: os vértices que serão conectados.
- Verifica se os vértices já existem. Se não existirem, são adicionados ao grafo.
- Adiciona **`vertice2`** à lista de adjacência de **`vertice1`**.
- Adiciona **`vertice1`** à lista de adjacência de **`vertice2`** (porque o grafo é não dirigido).

ADICIONA UMA ARESTA ENTRE DOIS VÉRTICES

```
14      // Adiciona uma aresta entre dois vértices (não dirigido).
15      adicionarAresta(vertice1, vertice2) {
16          if (!this.adjacencia[vertice1]) {
17              this.adicionarVertice(vertice1);
18          }
19          if (!this.adjacencia[vertice2]) {
20              this.adicionarVertice(vertice2);
21          }
22          this.adjacencia[vertice1].push(vertice2);
23          this.adjacencia[vertice2].push(vertice1);
24      }
```

REMOVER UMA ARESTA ENTRE DOIS VÉRTICES

- Remove uma conexão (aresta) entre dois vértices. Com parâmetros: **`vertice1`** e **`vertice2`**: os vértices cujas arestas serão removidas.
- Filtra a lista de adjacência de **`vertice1`** para remover **`vertice2`**.
- Filtra a lista de adjacência de **`vertice2`** para remover **`vertice1`**.

```
26      // Remove uma aresta entre dois vértices.
27      removerAresta(vertice1, vertice2) {
28          this.adjacencia[vertice1] = this.adjacencia[vertice1].filter(v => v !== vertice2);
29          this.adjacencia[vertice2] = this.adjacencia[vertice2].filter(v => v !== vertice1);
30      }
```

REMOVER UM VÉRTICE

- Remove um vértice do grafo, junto com todas as suas conexões (arestas). Parâmetro: **`vertice`**: o vértice a ser removido.
- Enquanto o vértice tiver conexões (arestas), elas são removidas usando o método **`removerAresta`**.
- Após remover todas as conexões, o vértice é excluído do grafo.

```
32      // Remove um vértice e suas conexões.
33      removerVertice(vertice) {
34          while (this.adjacencia[vertice]?.length) {
35              const adjacente = this.adjacencia[vertice].pop();
36              this.removerAresta(vertice, adjacente);
37          }
38          delete this.adjacencia[vertice];
39      }
```

MOSTRAR COMO O GRAFO ESTÁ

- Exibe todos os vértices e suas conexões no console.
- Itera sobre cada vértice no grafo
- Exibe o vértice e sua lista de adjacência no formato vertice -> adjacente1, adjacente2.

```
41      // Exibe o grafo.
42      imprimirGrafo() {
43          for (let vertice in this.adjacencia) {
44              console.log(`${vertice} -> ${this.adjacencia[vertice].join(', ')}`);
45          }
46      }
47  } // Fecha classe Grafo.
```

EXEMPLO DE USO DA CLASSE GRAFO

```
49  // Exemplo de uso
50  const meuGrafo = new Grafo();
51  meuGrafo.adicionarVertice('A');
52  meuGrafo.adicionarAresta('A', 'B');
53  meuGrafo.adicionarAresta('A', 'C');
54  meuGrafo.adicionarAresta('B', 'D');
55  meuGrafo.imprimirGrafo();
```

CONCLUSÃO

- Os grafos são uma poderosa estrutura de dados que pode ser implementada de várias formas.
- Usando classes simples e sem funções nativas avançadas, conseguimos criar uma base sólida para manipular grafos.
- Esse conhecimento pode ser expandido para resolver problemas complexos em ciência da computação.

EXERCÍCIOS

1) Crie um grafo com os vértices 1, 2, 3, 4 e as arestas 1-2, 1-3, 2-4.
   - Imprima o grafo.
   - Remova o vértice 2 e imprima o grafo novamente.

2) Adicione pesos às arestas no grafo.
   - Implemente o algoritmo de busca em largura (BFS).
   - Crie um grafo ponderado e calcule o menor caminho entre dois vértices.

OBRIGADO!

- Encontre este **material on-line** em:
  - www.tiago.blog.br
  - Plataforma Teams
- Em caso de **dúvidas**, entre em contato:
  - **Prof. Tiago**: tiago.silva238@fatec.sp.gov.br