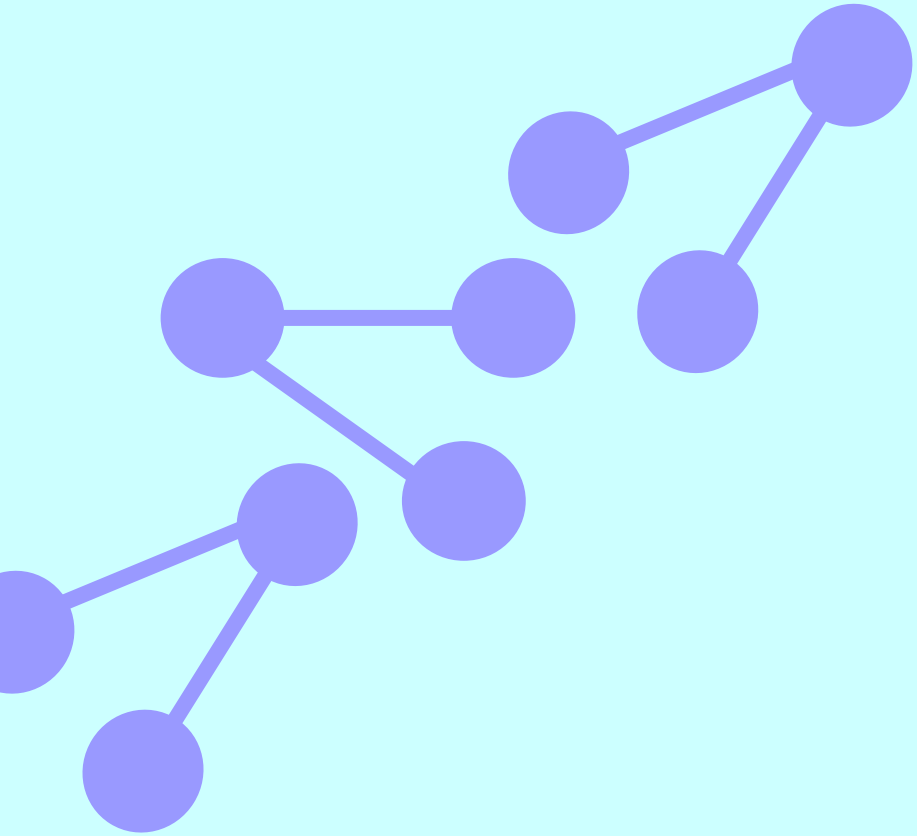
ALGORITMO DE DIJKSTRA



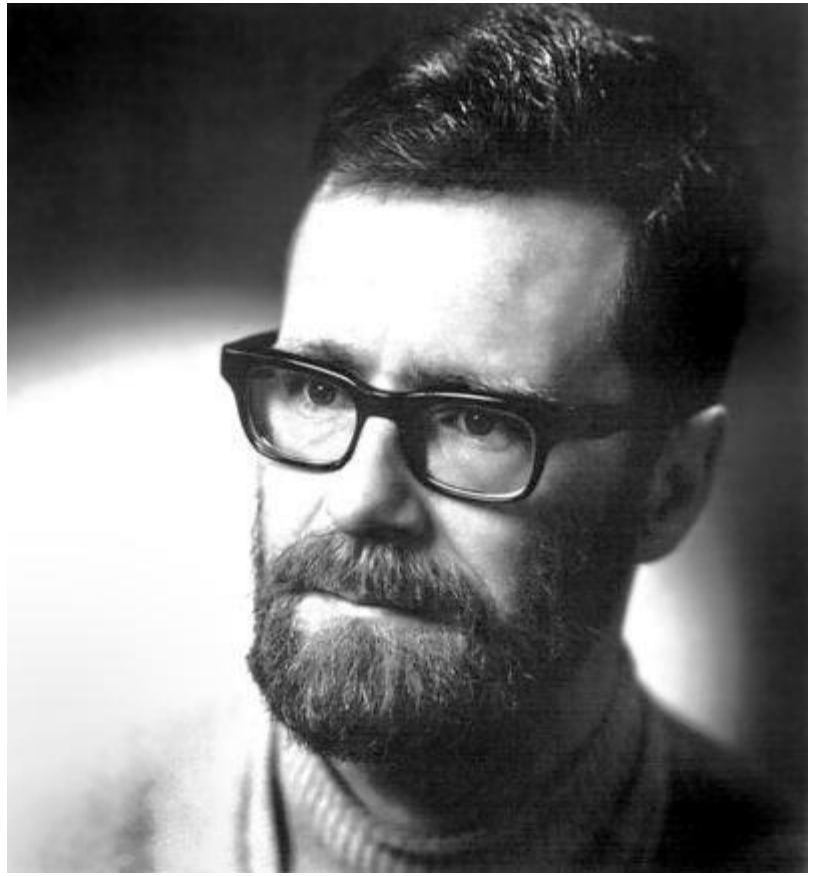
- Edsger Dijkstra
- Grafos
- O algoritmo
- Problema+Implementação

Edsger Dijkstra

- Edsger Wybe Dijkstra foi um cientista da computação holandês, conhecido por suas contribuições nas áreas de desenvolvimento de algoritmos e programas, de linguagens de programação, sistemas operacionais e processamento distribuído.

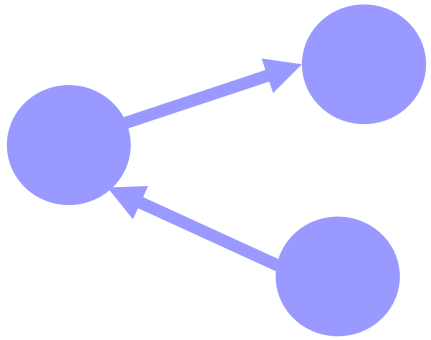
Nascimento: 11 de maio de 1930, Roterdão, Países Baixos

Falecimento: 6 de agosto de 2002, Nuenen, Países Baixos

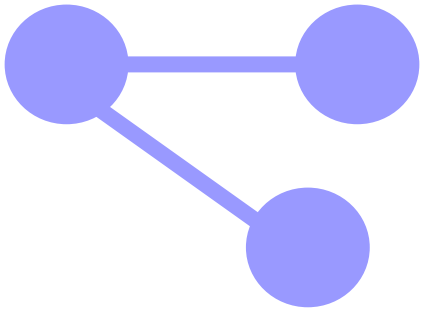


Grafos

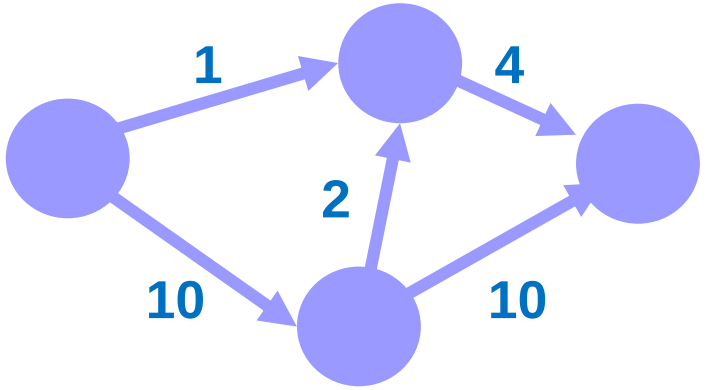
- **Um grafo é um tipo de dados abstrato, usado para modelar um conjunto de conexões.**



Grafo Direcionado



Grafo Não-Direcionado

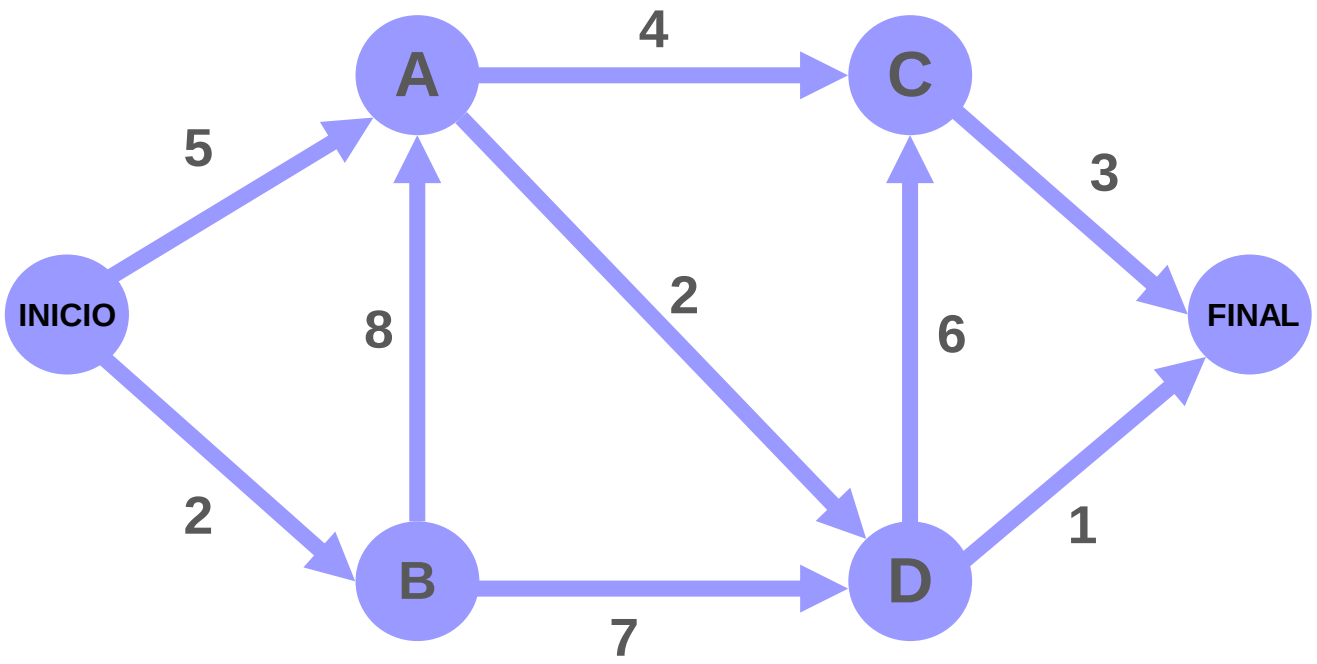


Grafo Direcionado Ponderado

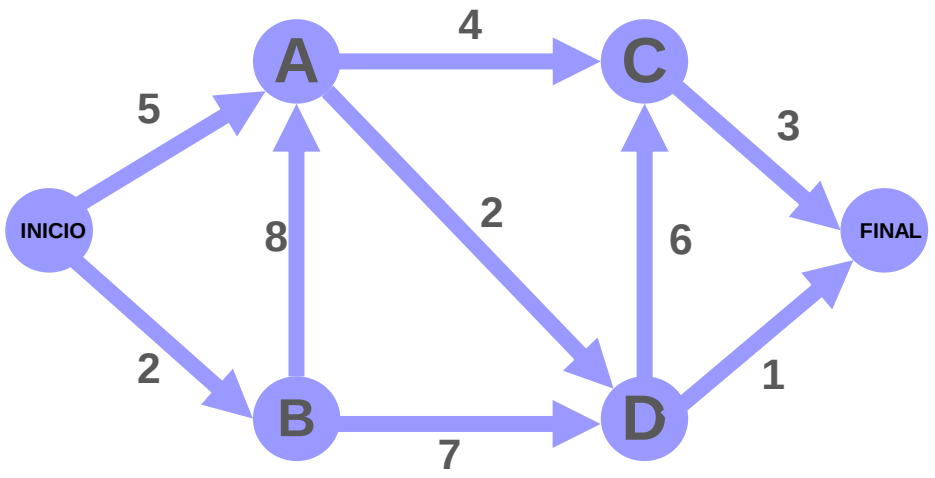
O Algoritmo

- 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.
- Um exemplo prático do 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?

Problema+Implementação



Problema+Implementação




```
const problema = {
  Inicio: {A: 5, B: 2},
  A: {C: 4, D: 2},
  B: {A: 8, D: 7},
  C: {D: 6, Final: 3},
  D: {Final: 1},
  Final: {}
};
```

Problema+Implementação

- **Descrevendo as principais etapas do algoritmo de Dijkstra.**

1 - Encontre o nó "mais barato".

2 - Atualize os custos dos vizinhos imediatos deste nó.

3 - Repita as etapas 1 e 2 até fazer isso para cada nó.

4 - Retorne o menor custo para alcançar o nó e o caminho ideal para fazê-lo.

Problema+Implementação

- **Primeiro, definiremos uma função (menorCustoNo) que, considerando os custos e os nós processados, retornará o nó mais barato que não foi processado.**

```
const menorCustoNo = (custo, processado) => {
  return Object.keys(custo).reduce((lowest, no) => {
    if (lowest === null || custo[no] < custo[lowest]) {
      if (!processado.includes(no)) {
        lowest = no;
      }
    }
    return lowest;
  }, null);
};
```

Problema+Implementação

- **Em seguida, definiremos a função principal (dijkstra) que usará o grafo inicial como parâmetro. Começaremos criando os custos, os pais e as estruturas de dados processados.**

```
const dijkstra = (graph) => {

  const custo = Object.assign({Final: Infinity}, graph.Inicio);

  const pais = {Final: null};
  for (let filhos in graph.Inicio) {
    pais[filhos] = 'Inicio';
  }

  const processado = [];

  let no = menorCustoNo(custo, processado);
```

Problema+Implementação

- **Em seguida, definiremos o valor inicial do nó que está sendo processado usando a função (menorCustoNo). E após, iniciaremos um loop while, que procurará continuamente o nó mais barato.**

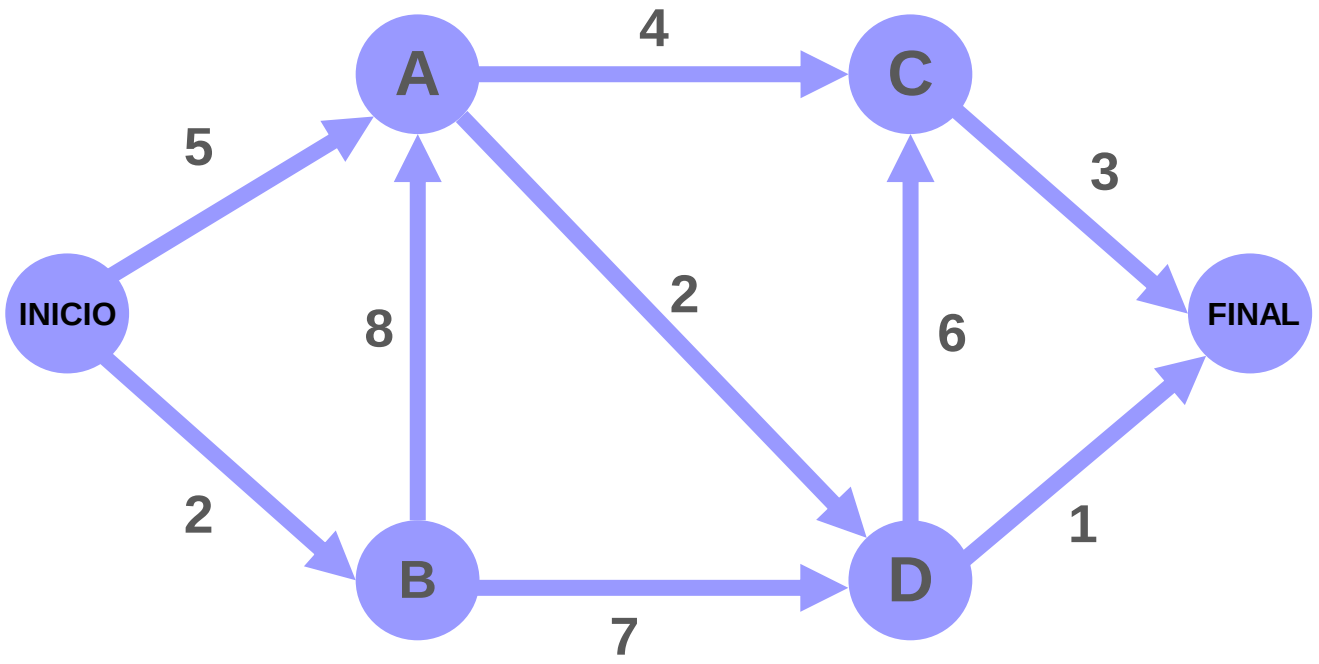
```
while (no) {
  let cost = custo[no];
  let filhoss = graph[no];
  for (let n in filhoss) {
    let novoCusto = cost + filhoss[n];
    if (!custo[n]) {
      custo[n] = novoCusto;
      pais[n] = no;
    }
    if (custo[n] > novoCusto) {
      custo[n] = novoCusto;
      pais[n] = no;
    }
  }
  processado.push(no);
  no = menorCustoNo(custo, processado);
}
```

Problema+Implementação

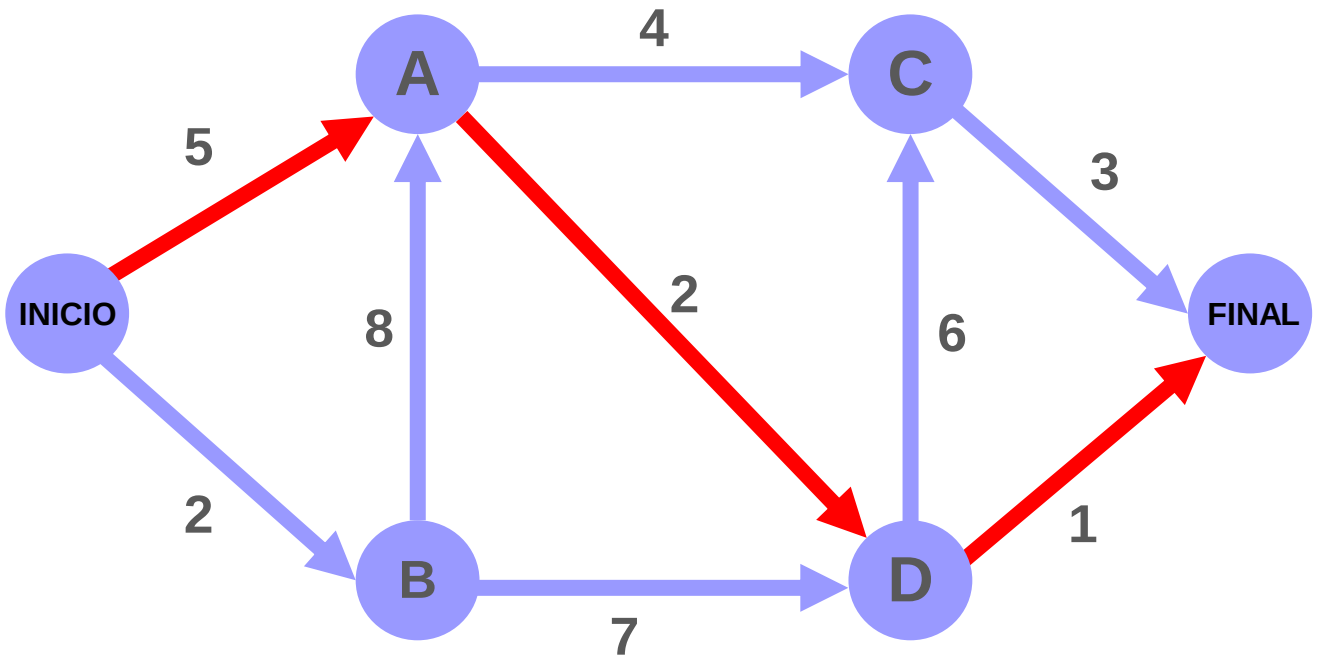
- **Por fim, quando o loop while estiver concluído, teremos o menor custo para alcançar o nó de conclusão. Agora, queremos obter o caminho para esse nó, o que podemos fazer refazendo nossas etapas com o objeto paiss.**

```
  let melhorCaminho = ['Final'];
  let paiss = pais.Final;
  while (paiss) {
    melhorCaminho.push(paiss);
    paiss = pais[paiss];
  }
  melhorCaminho.reverse();

  const resultados = {
    DistanciaCusto: custo.Final,
    Caminho: melhorCaminho
  };

  return resultados;
};
```

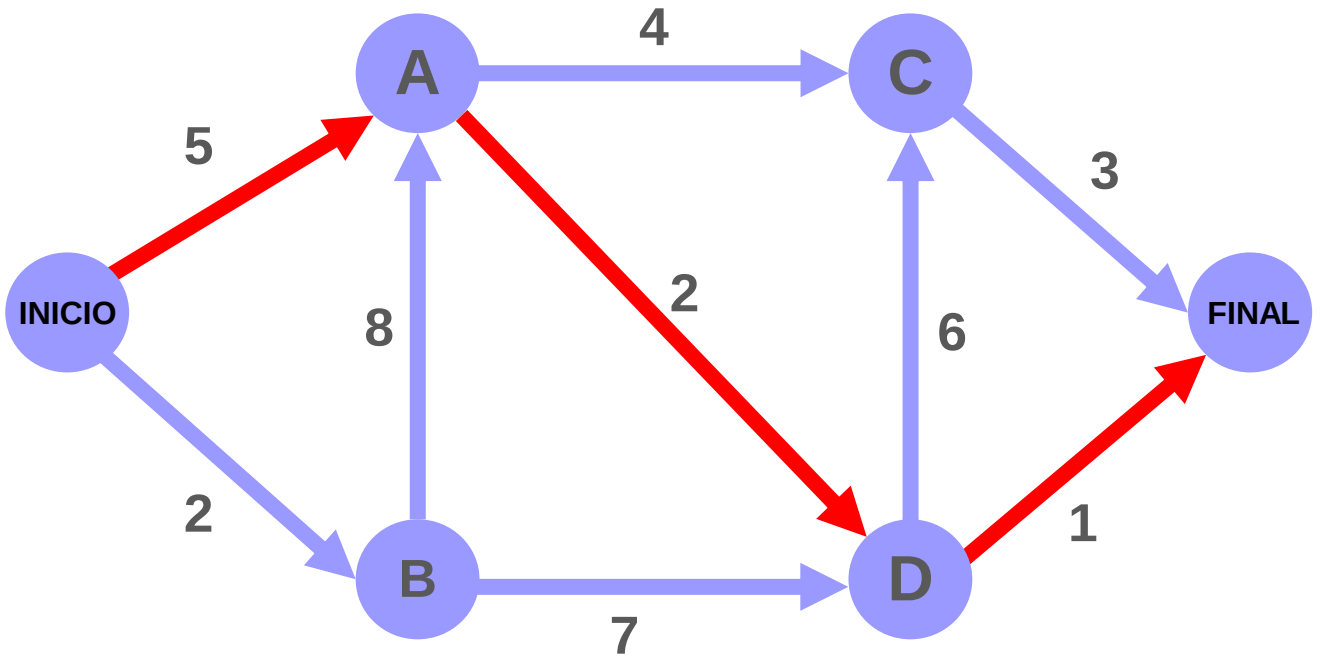
Problema+Implementação



Problema+Implementação



Problema+Implementação




```
[Running] node "c:\Users\Savyo\Documents\CCUFAL2017\IA\Dijkstra\Algoritmo.js"
{ DistanciaCusto: 8, Caminho: [ 'Inicio', 'A', 'D', 'Final' ] }
```

Referencias

- https://hackernoon.com/how-to-implement-dijkstras-algorithm-in-javascript-abdfd1702d04;
- https://www.tutorialspoint.com/Dijkstra-s-algorithm-in-Javascript;
- http://www.each.usp.br/digiampietri/SIN5013/AlgoritmoDeDijkstra