

# Grafos

*Algoritmo de Dijkstra*

**Prof. Edson Alves**

**Faculdade UnB Gama**

**Proponente**

## Proponente



**Edsger Wybe Dijkstra**  
**(1956)**

## **Características do algoritmo de Dijkstra**

## Características do algoritmo de Dijkstra

- ★ Computa o caminho mínimo de todos os vértices de  $G(V, E)$  a um dado nó  $s$

## Características do algoritmo de Dijkstra

- ★ Computa o caminho mínimo de todos os vértices de  $G(V, E)$  a um dado nó  $s$
- ★ Processa corretamente apenas grafos com arestas não-negativas

## Características do algoritmo de Dijkstra

- ★ Computa o caminho mínimo de todos os vértices de  $G(V, E)$  a um dado nó  $s$
- ★ Processa corretamente apenas grafos com arestas não-negativas
- ★ Eficiência: cada aresta é processada uma única vez

## Características do algoritmo de Dijkstra

- ★ **Computa o caminho mínimo de todos os vértices de  $G(V, E)$  a um dado nó  $s$**
- ★ **Processa corretamente apenas grafos com arestas não-negativas**
- ★ **Eficiência: cada aresta é processada uma única vez**
- ★ **Complexidade:  $O((V + E) \log V)$**



# Pseudocódigo

# Pseudocódigo

**Entrada:** um grafo  $G(V, E)$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

# Pseudocódigo

**Entrada:** um grafo  $G(V, E)$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

1. Faça  $d[s] = 0$ ,  $d[u] = \infty$  se  $u \neq s$  e seja  $U = V$

# Pseudocódigo

**Entrada:** um grafo  $G(V, E)$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

1. Faça  $d[s] = 0$ ,  $d[u] = \infty$  se  $u \neq s$  e seja  $U = V$
2. Enquanto  $U \neq \emptyset$ :
  - (a) Seja  $u \in U$  o vértice mais próximo de  $s$  em  $U$
  - (b) Relaxe as distâncias usando as arestas que partem de  $u$
  - (c) Remova  $u$  de  $U$

# Pseudocódigo

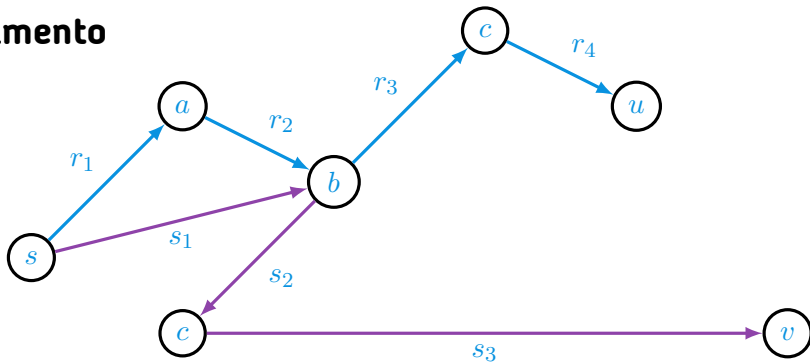
**Entrada:** um grafo  $G(V, E)$  e um vértice  $s \in V$

**Saída:** um vetor  $d$  tal que  $d[u]$  é a distância mínima em  $G$  entre  $s$  e  $u$

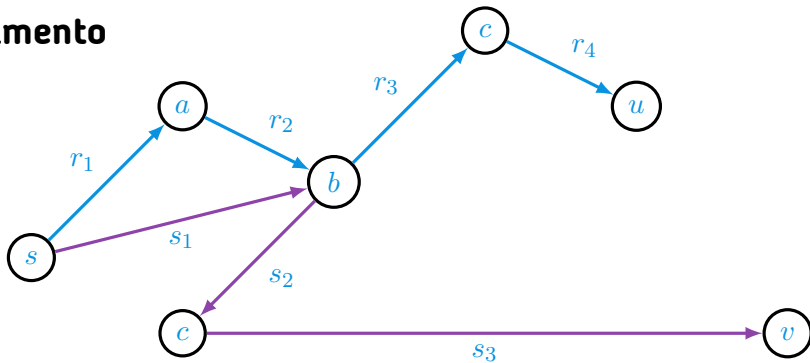
1. Faça  $d[s] = 0$ ,  $d[u] = \infty$  se  $u \neq s$  e seja  $U = V$
2. Enquanto  $U \neq \emptyset$ :
  - (a) Seja  $u \in U$  o vértice mais próximo de  $s$  em  $U$
  - (b) Relaxe as distâncias usando as arestas que partem de  $u$
  - (c) Remova  $u$  de  $U$
3. Retorne  $d$

# Relaxamento

## Relaxamento



## Relaxamento

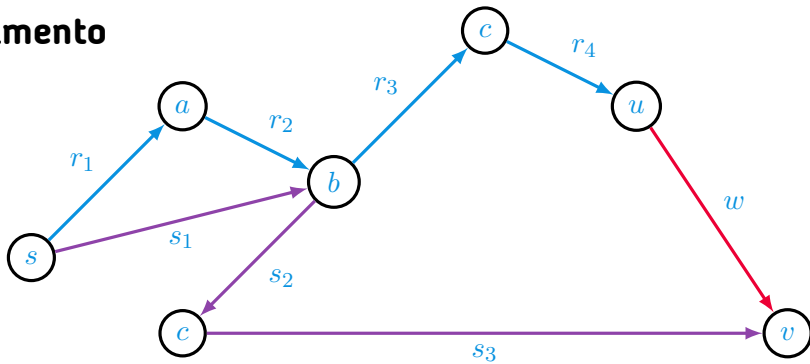


$$\text{dist}(s, u) = \sum_{i=1}^4 r_i$$

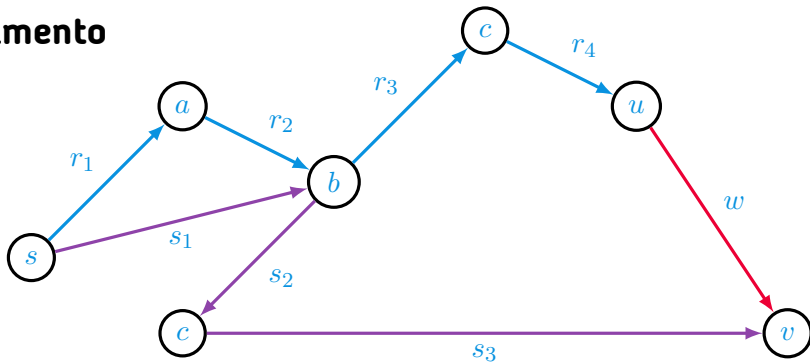
$$\text{dist}(s, v) = \sum_{j=1}^3 s_j$$



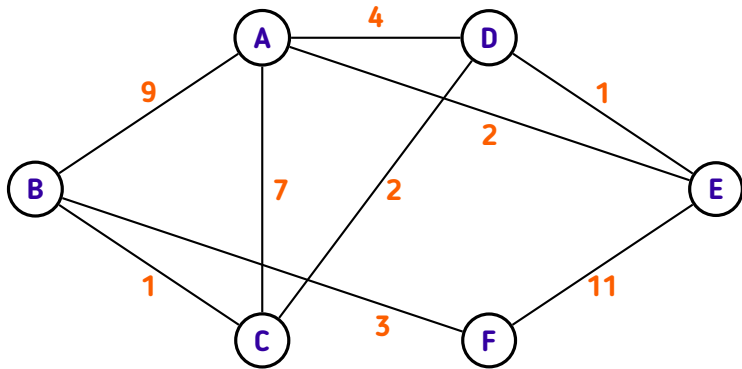
## Relaxamento

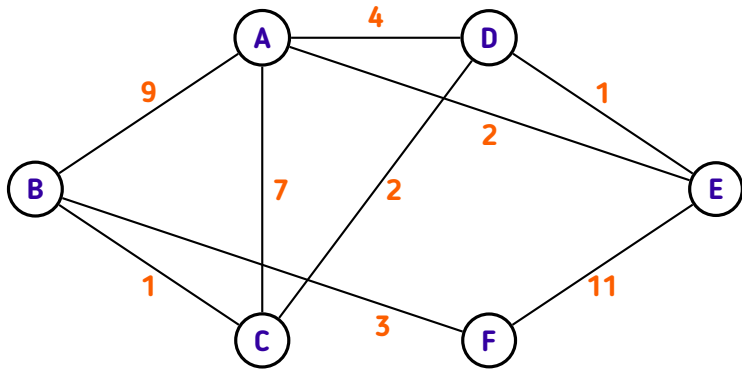


## Relaxamento



**Se**  $\text{dist}(s, u) + w < \text{dist}(s, v)$ , **faça**  $\text{dist}(s, v) = \text{dist}(s, u) + w$

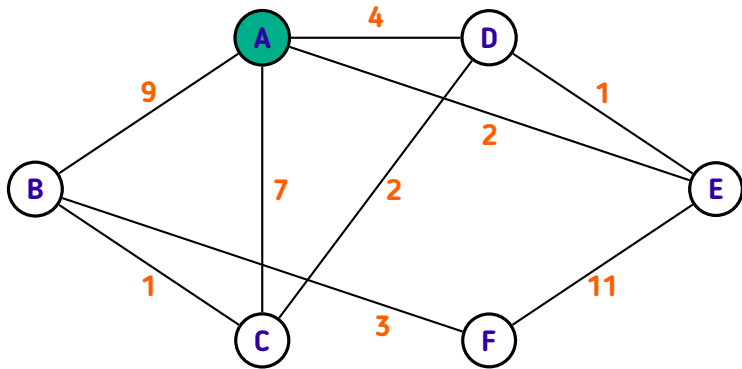




$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

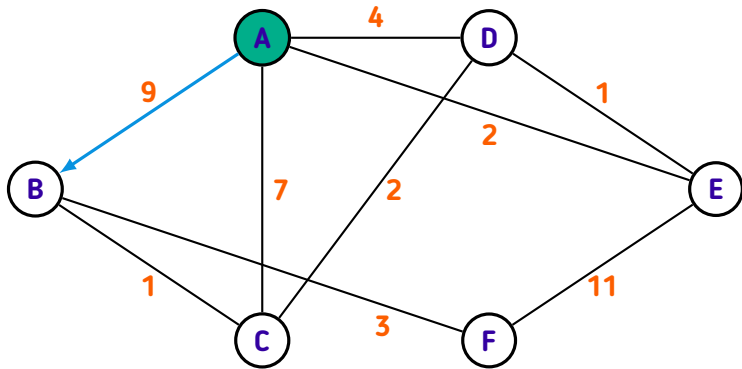
$$U = \{ \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

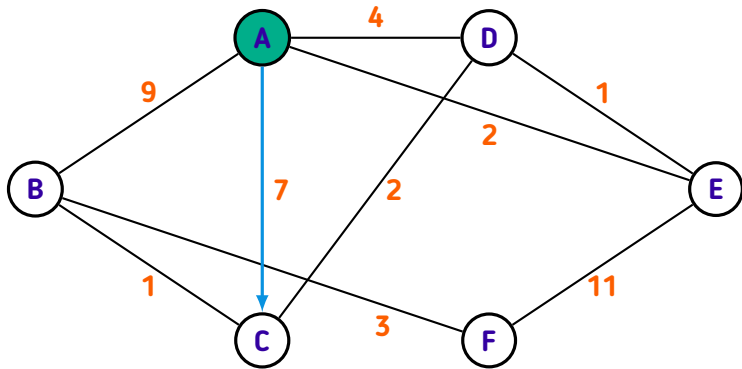
A	B	C	D	E	F
0	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$

$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F} \}$$



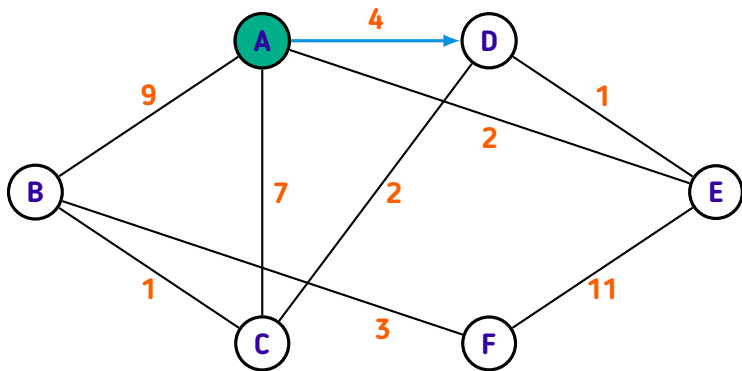
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	$\infty$	$\infty$	$\infty$	$\infty$

$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F} \}$$



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	$\infty$	$\infty$	$\infty$

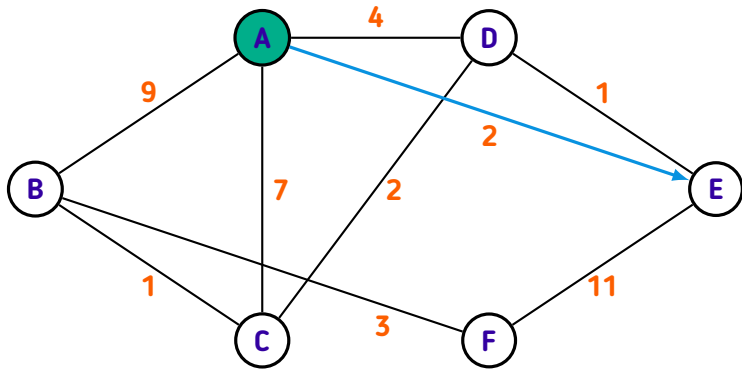
$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F} \}$$



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	$\infty$	$\infty$

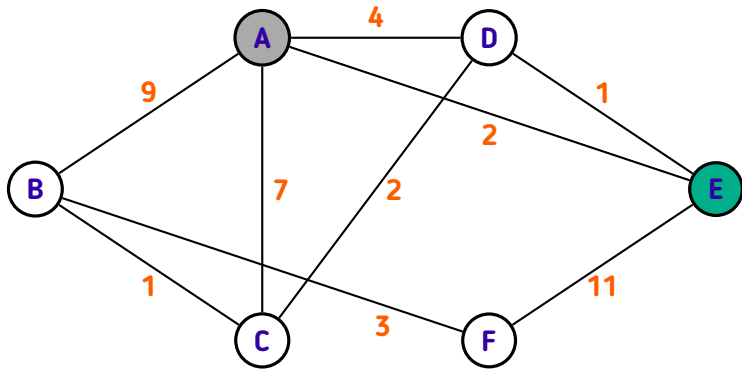
$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F} \}$$





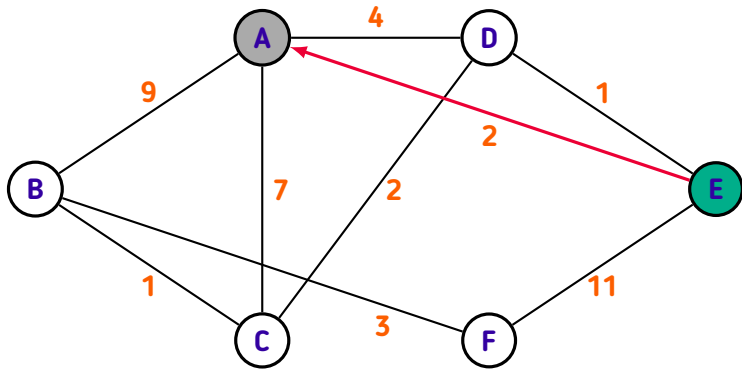
	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	2	$\infty$

$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}, \mathbf{F} \}$$



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	4	2	$\infty$

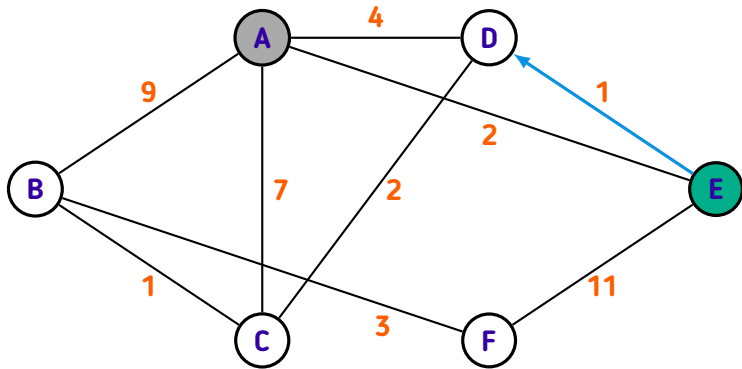
$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F
0	9	7	4	2	$\infty$

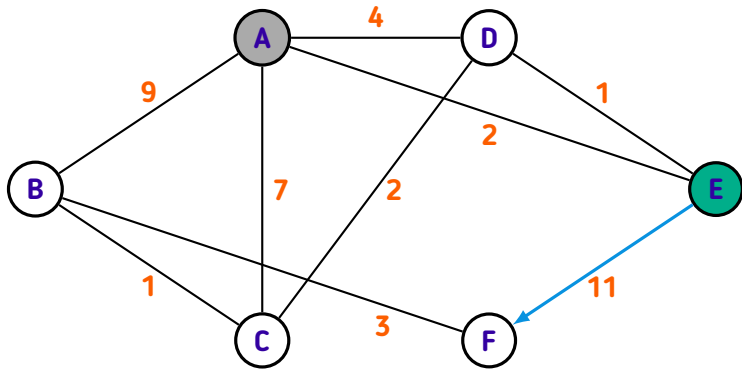
$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

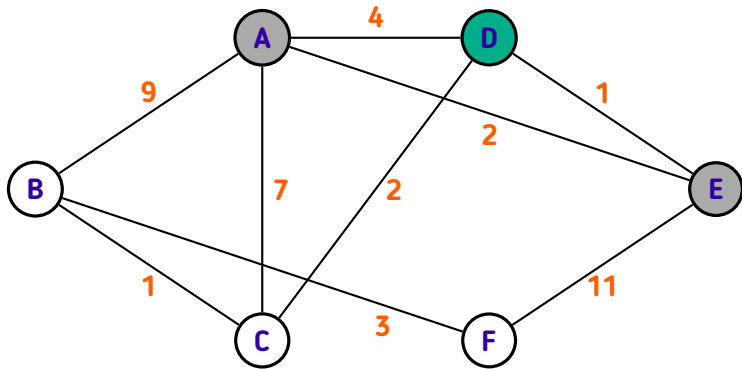
A	B	C	D	E	F
0	9	7	3	2	$\infty$

$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{F} \}$$



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	7	3	2	<b>13</b>

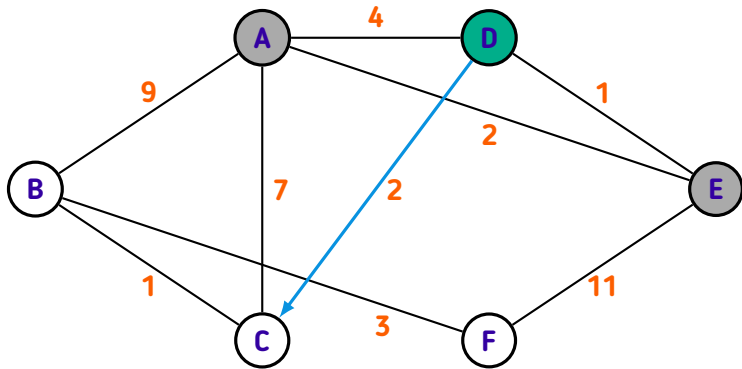
$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F
0	9	7	3	2	13

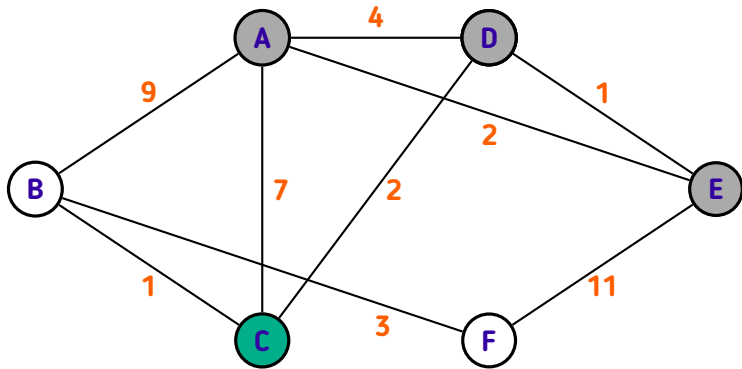
$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F
0	9	5	3	2	13

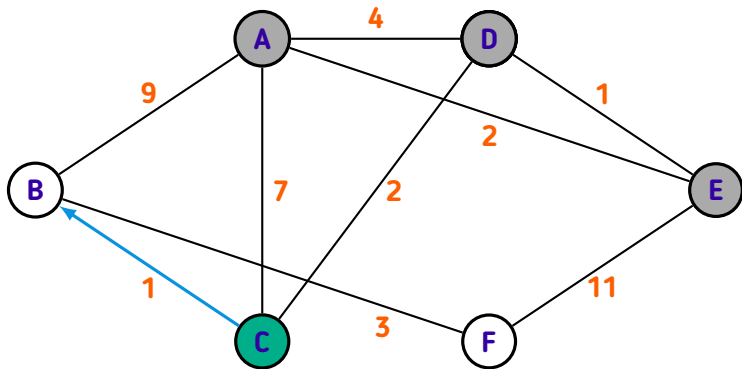
$$U = \{ \mathbf{B}, \mathbf{C}, \mathbf{F} \}$$



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	9	5	3	2	13

$$U = \{ \mathbf{B}, \mathbf{F} \}$$

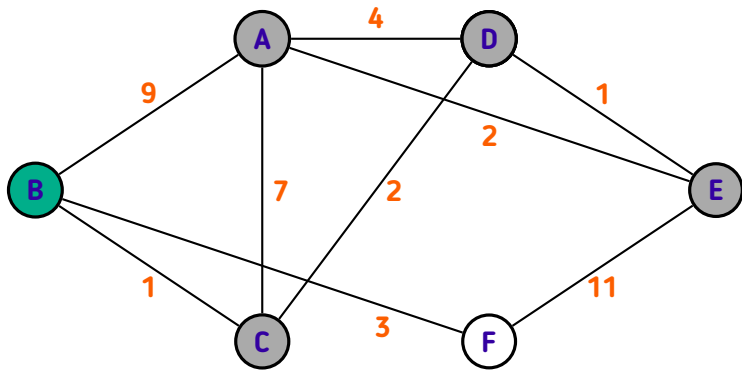




$\text{dist}(u, \mathbf{A})$

A	B	C	D	E	F
0	6	5	3	2	13

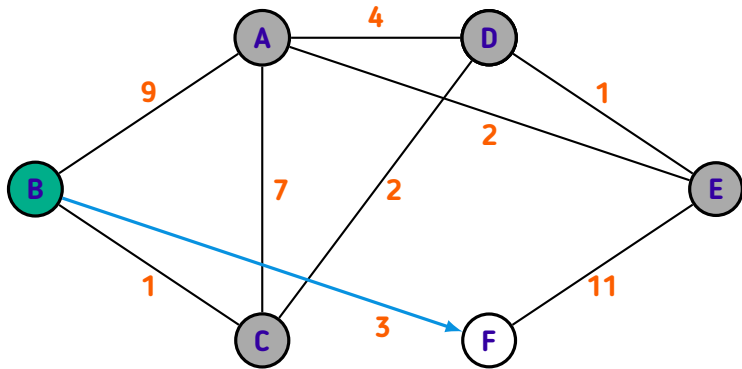
$$U = \{ \mathbf{B}, \mathbf{F} \}$$



$\text{dist}(u, \mathbf{A})$

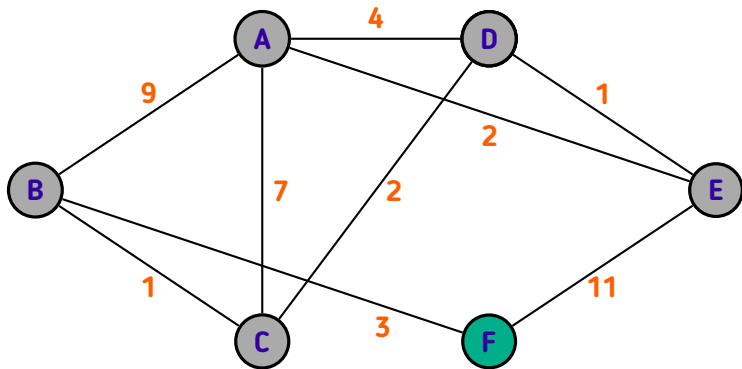
A	B	C	D	E	F
0	6	5	3	2	13

$$U = \{ \mathbf{F} \}$$



	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	6	5	3	2	<b>9</b>

$$U = \{ \mathbf{F} \}$$

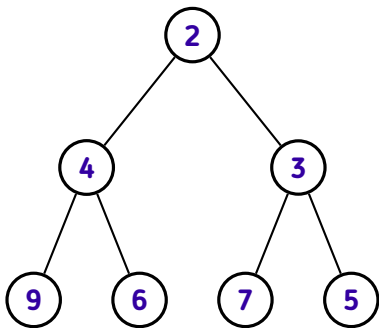


	A	B	C	D	E	F
$\text{dist}(u, \mathbf{A})$	0	6	5	3	2	9

$U = \emptyset$

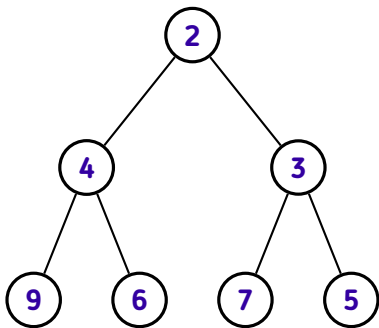
**Identificação eficiente do vértice  $u$  mais próximo de  $s$**

Identificação eficiente do vértice  $u$  mais próximo de  $s$



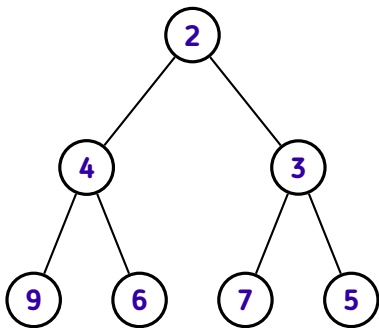
*min heap*

Identificação eficiente do vértice  $u$  mais próximo de  $s$

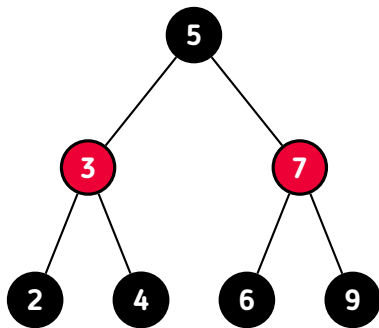


*min heap*  
(priority\_queue)

Identificação eficiente do vértice  $u$  mais próximo de  $s$



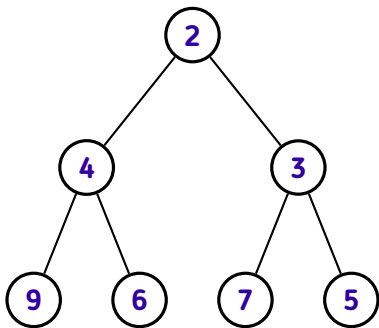
*min heap*  
(priority\_queue)



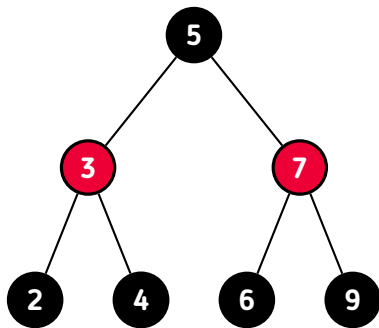
*red-black tree*



Identificação eficiente do vértice  $u$  mais próximo de  $s$



*min heap*  
(priority\_queue)



*red-black tree*  
(set)

```
vector<int> dijkstra(int s, int N)
{
    const int oo { 1000000010 };

    vector<int> dist(N + 1, oo);
    dist[s] = 0;

    set<ii> U;
    U.emplace(0, s);

    while (not U.empty())
    {
        auto [d, u] = *U.begin();
        U.erase(U.begin());
    }
}
```

```
    for (auto [v, w] : adj[u])
    {
        if (dist[v] > d + w)
        {
            if (U.count(ii(dist[v], v)))
                U.erase(ii(dist[v], v));

            dist[v] = d + w;
            U.emplace(dist[v], v);
        }
    }

    return dist;
}
```

## **Identificação do caminho mínimo**

## Identificação do caminho mínimo

- ★ O algoritmo de Dijkstra computa as distâncias mínimas, mas não os caminhos mínimos

## Identificação do caminho mínimo

- ★ O algoritmo de Dijkstra computa as distâncias mínimas, mas não os caminhos mínimos
- ★ Para determinar um caminho mínimo, é preciso definir o vetor auxiliar  $\text{pred}$ , onde  $\text{pred}[u] = \text{antecessor de } u \text{ no caminho mínimo de } s \text{ a } u$

## Identificação do caminho mínimo

- ★ O algoritmo de Dijkstra computa as distâncias mínimas, mas não os caminhos mínimos
- ★ Para determinar um caminho mínimo, é preciso definir o vetor auxiliar  $\text{pred}$ , onde  $\text{pred}[u] = \text{antecessor de } u \text{ no caminho mínimo de } s \text{ a } u$
- ★ No início do algoritmo,  $\text{pred}[s] = s$  e  $\text{pred}[u] = \text{undef}$ , se  $u \neq s$

## Identificação do caminho mínimo

★ Se  $(u, v)$  atualizar  $d[v]$ , faça  $\text{pred}[v] = u$



## Identificação do caminho mínimo

★ Se  $(u, v)$  atualizar  $d[v]$ , faça  $\text{pred}[v] = u$

★ A sequência

$$p = \{(s, \text{pred}^{k-1}[u]), \dots, (\text{pred}[\text{pred}[u]], \text{pred}[u]), (\text{pred}[u], u)\}$$

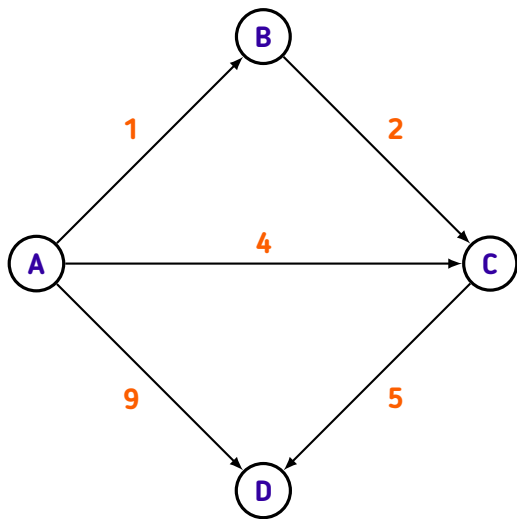
## Identificação do caminho mínimo

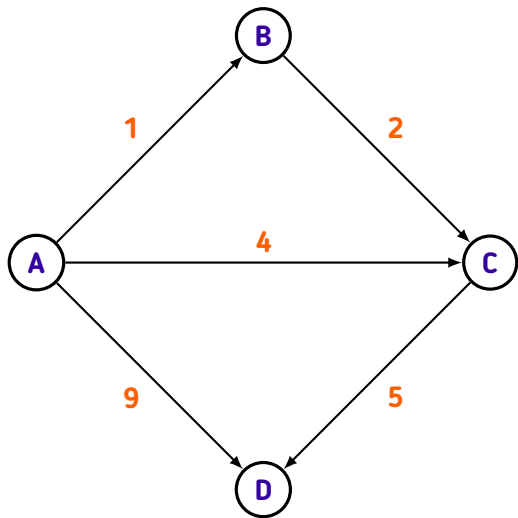
★ Se  $(u, v)$  atualizar  $d[v]$ , faça  $\text{pred}[v] = u$

★ A sequência

$$p = \{(s, \text{pred}^{k-1}[u]), \dots, (\text{pred}[\text{pred}[u]], \text{pred}[u]), (\text{pred}[u], u)\}$$

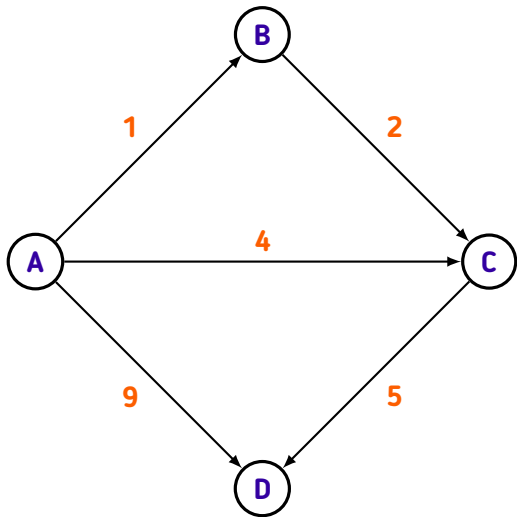
é um caminho mínimo de  $s$  a  $u$  composto de  $k$  arestas e tamanho  $d[u]$





$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	$\infty$	$\infty$	$\infty$

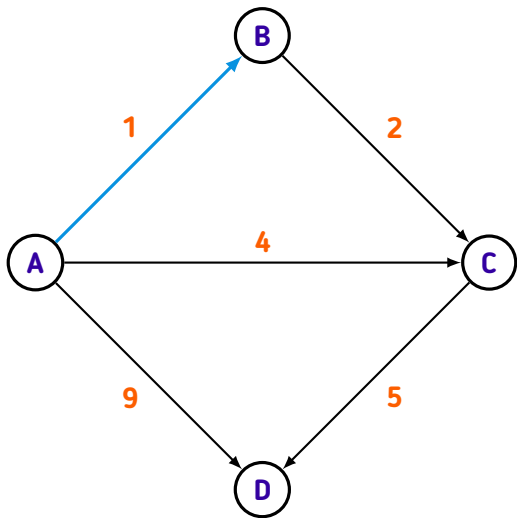


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	$\infty$	$\infty$	$\infty$

$\text{pred}(u)$

A	B	C	D
A	-	-	-

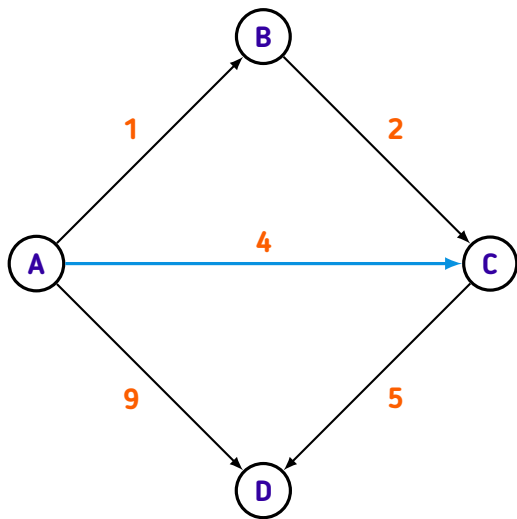


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	$\infty$	$\infty$

$\text{pred}(u)$

A	B	C	D
A	A	-	-

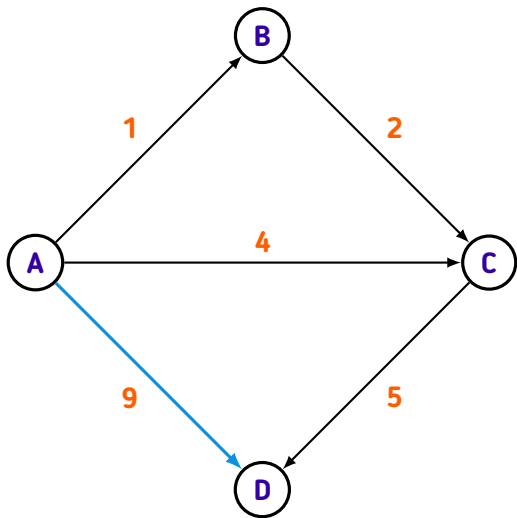


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	4	$\infty$

$\text{pred}(u)$

A	B	C	D
A	A	A	-



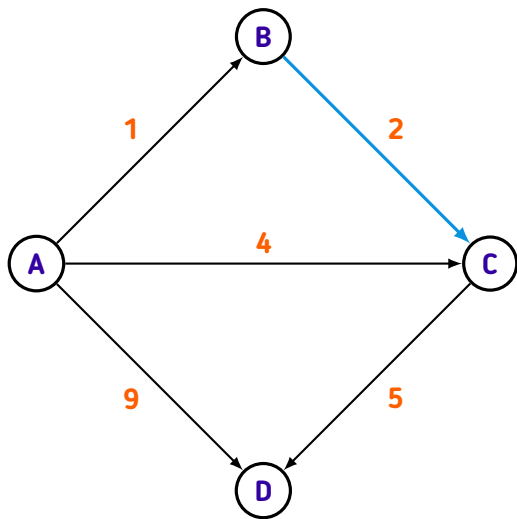
$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	4	9

$\text{pred}(u)$

A	B	C	D
A	A	A	A



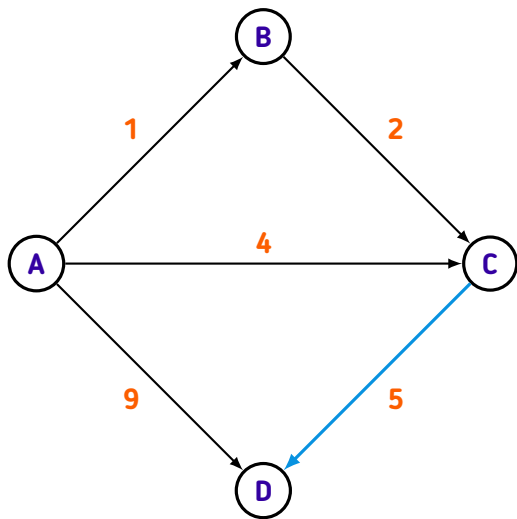


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	3	9

$\text{pred}(u)$

A	B	C	D
A	A	B	A

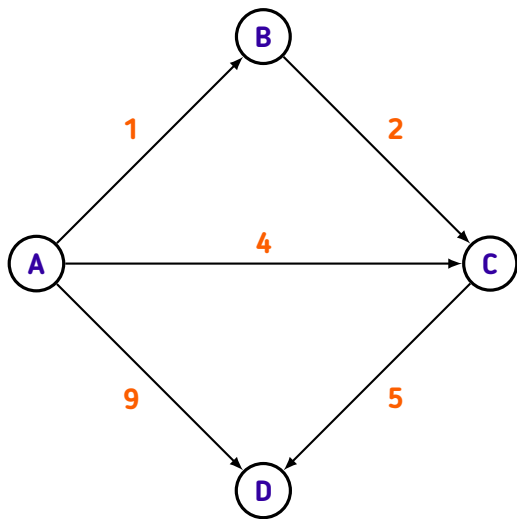


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	3	8

$\text{pred}(u)$

A	B	C	D
A	A	B	C

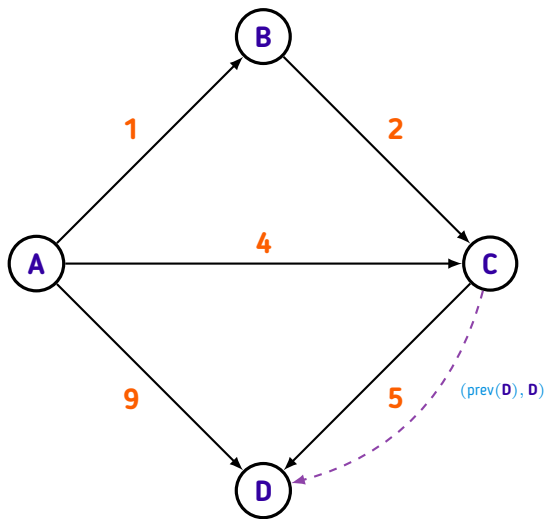


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	3	8

$\text{pred}(u)$

A	B	C	D
A	A	B	C

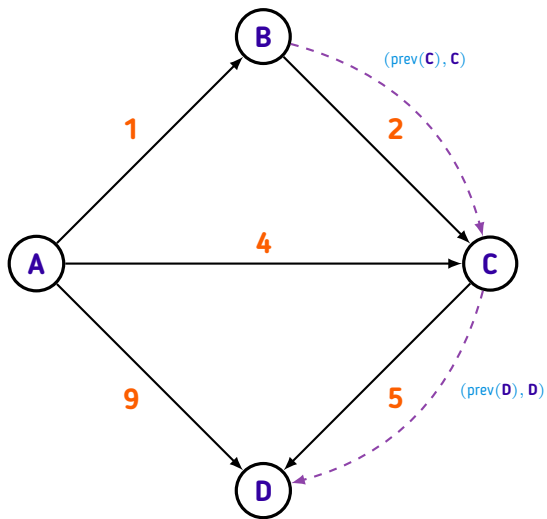


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	3	8

$\text{pred}(u)$

A	B	C	D
A	A	B	C

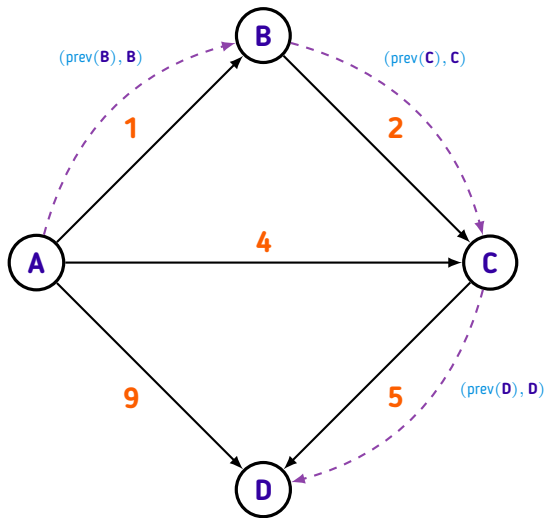


$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	3	8

$\text{pred}(u)$

A	B	C	D
A	A	B	C



$\text{dist}(u, \mathbf{A})$

A	B	C	D
0	1	3	8

$\text{pred}(u)$

A	B	C	D
A	A	B	C

```
pair<vector<int>, vector<int>> dijkstra(int s, int N)
{
    vector<int> dist(N + 1, oo), pred(N + 1, oo);
    dist[s] = 0;
    pred[s] = s;

    processed.reset();

    priority_queue<ii, vector<ii>, greater<ii>> pq;
    pq.emplace(0, s);

    while (not pq.empty())
    {
        auto [d, u] = pq.top();
        pq.pop();
```

```
    if (processed[u])
        continue;

    processed[u] = true;

    for (auto [v, w] : adj[u])
    {
        if (dist[v] > d + w) {
            dist[v] = d + w;
            pred[v] = u;
            pq.emplace(dist[v], v);
        }
    }
}

return { dist, pred };
}
```



```
vector<ii> path(int s, int u, const vector<int>& pred)
{
    vector<ii> p;
    int v = u;

    do {
        p.push_back(ii(pred[v], v));
        v = pred[v];
    } while (v != s);

    reverse(p.begin(), p.end());

    return p;
}
```

## Problemas sugeridos

1. [AtCoder Beginner Contest 143 – Problem E: Travel by Car](#)
2. [Codeforces Alpha Round #20 – Problem C: Dijkstra?](#)
3. [OJ 1112 – Mice and Maze](#)
4. [OJ 10986 – Sending email](#)

## Referências

1. HALIM, Felix; HALIM, Steve. *Competitive Programming 3*, 2010.
2. LAAKSONEN, Antti. *Competitive Programmer's Handbook*, 2018.
3. SKIENA, Steven; REVILLA, Miguel. *Programming Challenges*, 2003.
4. Wikipédia, *Dijkstra's algorithm*. Acesso em 13/07/2021.
5. Wikipédia, *Edsger W. Dijkstra*. Acesso em 13/07/2021.