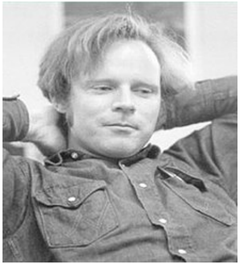
Grafos

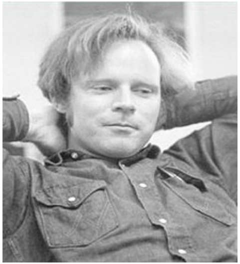
Algoritmo de Floyd-Warshall

Prof. Edson Alves

Faculdade UnB Gama

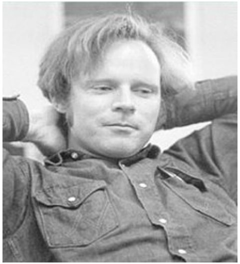


Robert W. Floyd
(1962)



Robert W. Floyd
(1962)

Stephen Warshall
(1962)



Robert W. Floyd
(1962)

Stephen Warshall
(1962)

Bernard Roy
(1959)

Características do algoritmo de Bellman-Ford

Características do algoritmo de Bellman-Ford

- $\star$ **Computa o caminho mínimo entre todos os pares de vértices de** $G(V, E)$

Características do algoritmo de Bellman-Ford

- ⋆ Computa o caminho mínimo entre todos os pares de vértices de $G(V, E)$
- ⋆ É capaz de processar arestas negativas

Características do algoritmo de Bellman-Ford

- $\star$ **Computa o caminho mínimo entre todos os pares de vértices de $G(V, E)$**
- $\star$ **É capaz de processar arestas negativas**
- $\star$ **Não processa, mas identifica ciclos negativos**

Características do algoritmo de Bellman-Ford

- $\star$ **Computa o caminho mínimo entre todos os pares de vértices de** $G(V, E)$
- $\star$ **É capaz de processar arestas negativas**
- $\star$ **Não processa, mas identifica ciclos negativos**
- $\star$ **As distâncias são reduzidas por meio do uso de vértices intermediários**

Características do algoritmo de Bellman-Ford

- $\star$ **Computa o caminho mínimo entre todos os pares de vértices de** $G(V, E)$
- $\star$ **É capaz de processar arestas negativas**
- $\star$ **Não processa, mas identifica ciclos negativos**
- $\star$ **As distâncias são reduzidas por meio do uso de vértices intermediários**
- $\star$ **Complexidade:** $O(V^3)$

Pseudocódigo

Pseudocódigo

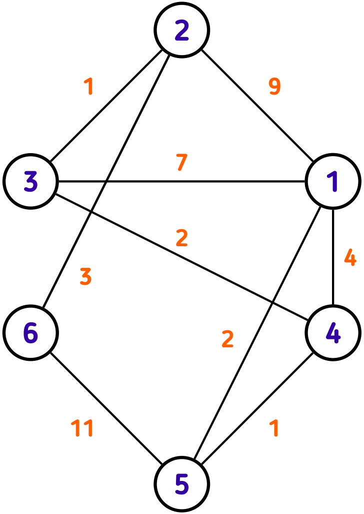
Entrada: um grafo $G(V, E)$
Saída: uma matriz d tal que $d[u][v]$ é a distância mínima em G entre u e v

1. **Faça:**
   (a) $d[u][u] = 0$, **para todos** $u \in V$
   (b) $d[u][v] = w$, **se** $(u, v, w) \in E$
   (c) $d[u][v] = \infty$, **caso contrário**

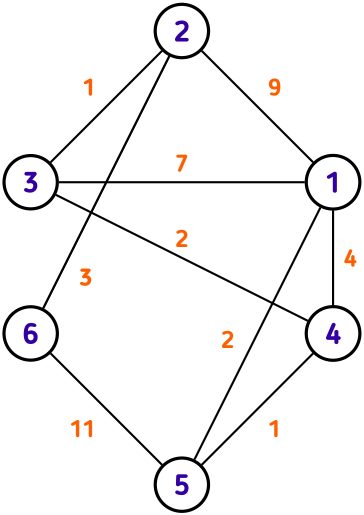
2. **Para cada vértice** k **e todos os pares** $(u, v) \in V^2$, **faça**

$$d[u][v] = \min(d[u][v], d[u][k] + d[k][v])$$

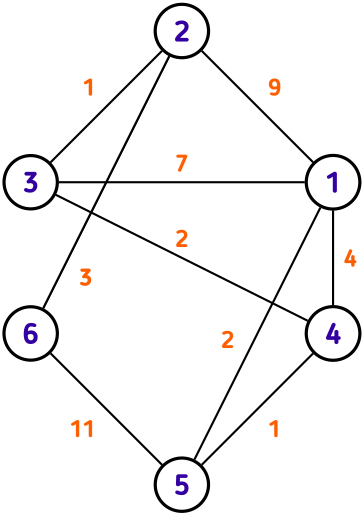
3. **Retorne** d



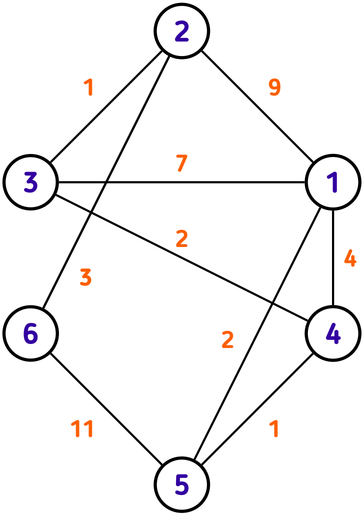
	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						



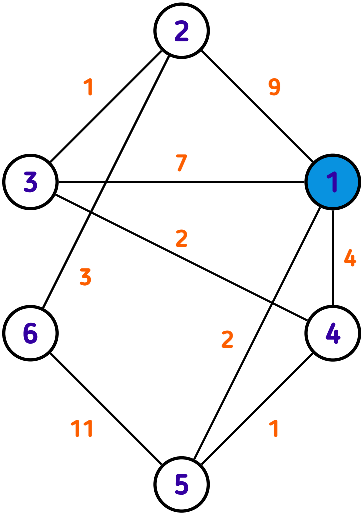
	1	2	3	4	5	6
1	0					
2		0				
3			0			
4				0		
5					0	
6						0



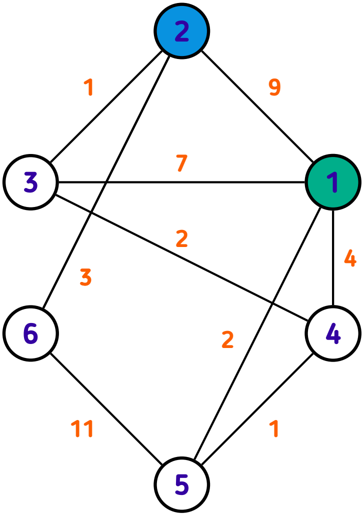
	1	2	3	4	5	6
1	0	9	7	4	2	
2	9	0	1			3
3	7	1	0	2		
4	4		2	0	1	
5	2			1	0	11
6		3			11	0



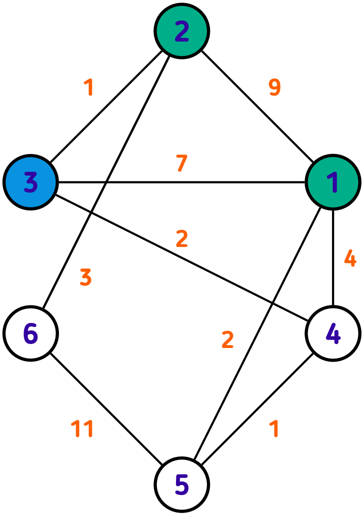
	1	2	3	4	5	6
1	0	9	7	4	2	∞
2	9	0	1	∞	∞	3
3	7	1	0	2	∞	∞
4	4	∞	2	0	1	∞
5	2	∞	∞	1	0	11
6	∞	3	∞	∞	11	0



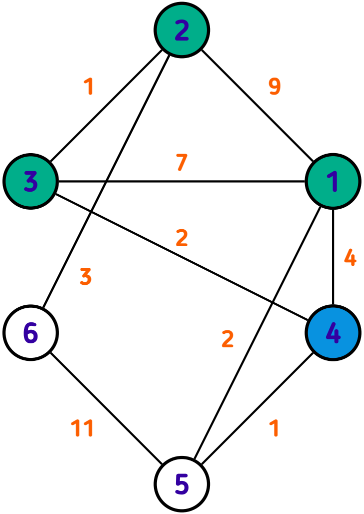
	1	2	3	4	5	6
1	0	9	7	4	2	∞
2	9	0	1	**13**	**11**	3
3	7	1	0	2	**9**	∞
4	4	**13**	2	0	1	∞
5	2	**11**	**9**	1	0	11
6	∞	3	∞	∞	11	0



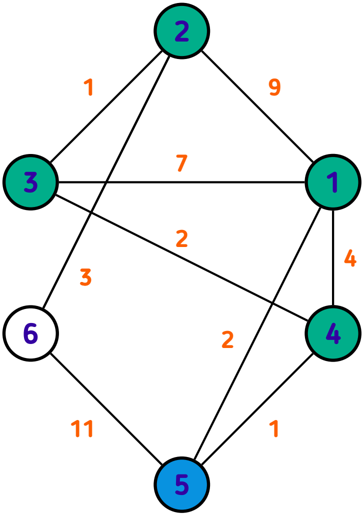
	1	2	3	4	5	6
1	0	9	7	4	2	**12**
2	9	0	1	13	11	3
3	7	1	0	2	9	**4**
4	4	13	2	0	1	**16**
5	2	11	9	1	0	11
6	**12**	3	**4**	**16**	11	0



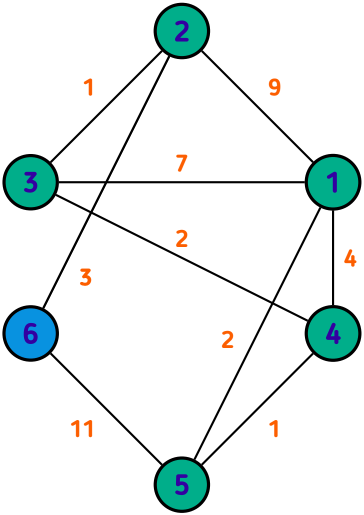
	1	2	3	4	5	6
1	0	**8**	7	4	2	**11**
2	**8**	0	1	**3**	**10**	3
3	7	1	0	2	9	4
4	4	**3**	2	0	1	**6**
5	2	**10**	9	1	0	11
6	**11**	3	4	**6**	11	0



	1	2	3	4	5	6
1	0	**7**	**6**	4	2	**10**
2	**7**	0	1	3	**4**	3
3	**6**	1	0	2	**3**	4
4	4	3	2	0	1	6
5	2	**4**	**3**	1	0	**7**
6	**10**	3	4	6	**7**	0



	1	2	3	4	5	6
1	0	**6**	**5**	**3**	2	**9**
2	**6**	0	1	3	4	3
3	**5**	1	0	2	3	4
4	**3**	3	2	0	1	6
5	2	4	3	1	0	7
6	**9**	3	4	6	7	0



	1	2	3	4	5	6
1	0	6	5	3	2	9
2	6	0	1	3	4	3
3	5	1	0	2	3	4
4	3	3	2	0	1	6
5	2	4	3	1	0	7
6	9	3	4	6	7	0

```cpp
vector<vector<int>> floyd_warshall(int N)
{
    vector<vector<int>> dist(N + 1, vector<int>(N + 1, oo));

    for (int u = 1; u <= N; ++u)
        dist[u][u] = 0;

    for (int u = 1; u <= N; ++u)
        for (auto [v, w] : adj[u])
            dist[u][v] = w;

    for (int k = 1; k <= N; ++k)
        for (int u = 1; u <= N; ++u)
            for (int v = 1; v <= N; ++v)
                if (dist[u][k] < oo and dist[k][v] < oo)
                    dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v]);

    return dist;
}
```

Identificação de um caminho mínimo

Identificação de um caminho mínimo

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

Identificação de um caminho mínimo

- $\star$ **O algoritmo de Dijkstra computa as distâncias mínimas, mas não os caminhos mínimos**

- $\star$ **Para determinar um caminho mínimo, é preciso definir uma matriz auxiliar** pred, **onde** $\text{pred}[u][v] =$ **antecessor de** v **no caminho mínimo de** u **a** v

Identificação de um caminho mínimo

$\star$ **O algoritmo de Dijkstra computa as distâncias mínimas, mas não os caminhos mínimos**

$\star$ **Para determinar um caminho mínimo, é preciso definir uma matriz auxiliar** pred, **onde** $\mathsf{pred}[u][v] =$ **antecessor de** v **no caminho mínimo de** u **a** v

$\star$ **No início do algoritmo,**

(a) $\mathsf{pred}[u][u] = u, \forall u \in V$

(b) $\mathsf{pred}[u][v] = u$, **se** $(u, v) \in E$

(c) $\mathsf{pred}[u][v] = \mathsf{undef}$, **caso contrário**

Identificação de um caminho mínimo

Identificação de um caminho mínimo

- **Se** k **atualizar** $d[u][v]$**, faça** $\text{pred}[u][v] = \text{pred}[k][v]$

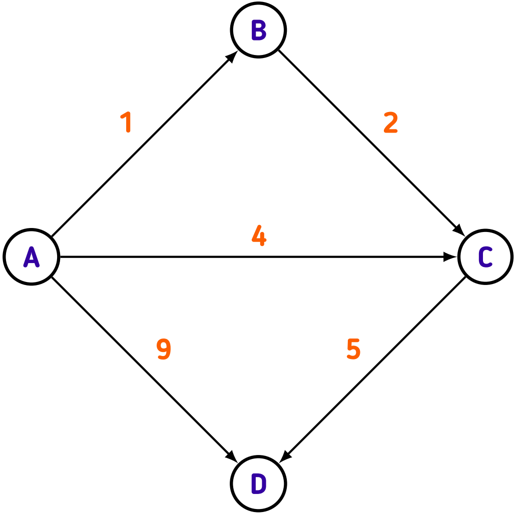
Identificação de um caminho mínimo

- $\star$ **Se k atualizar $d[u][v]$, faça $\textsf{pred}[u][v] = \textsf{pred}[k][v]$**

- $\star$ **A sequência**

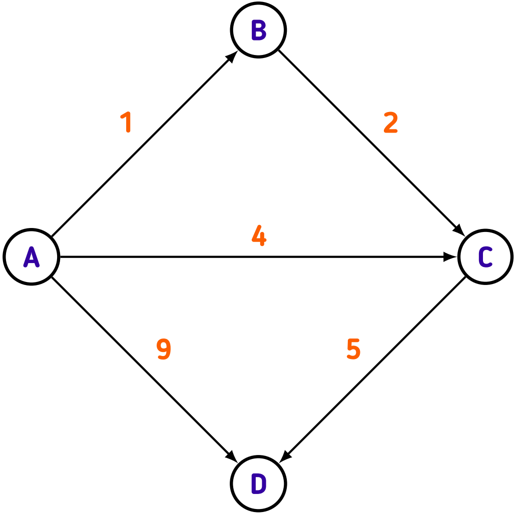
$$p = \{(u, \textsf{pred}^{k-1}[u][v]), \ldots, (\textsf{pred}[\textsf{pred}[u][v]], \textsf{pred}[u][v]), (\textsf{pred}[u][v], v)\}$$

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



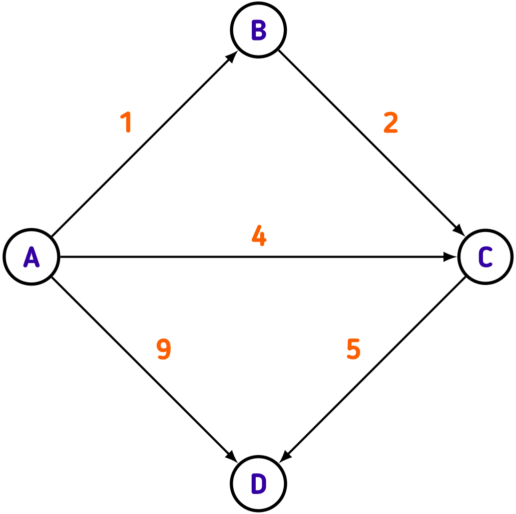
$$M = \begin{array}{cc} \begin{array}{cccc} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} \end{array} & \\ \left[\begin{array}{c|c|c|c} & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \end{array}\right] & \begin{array}{c} \mathsf{A} \\ \mathsf{B} \\ \mathsf{C} \\ \mathsf{D} \end{array} \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



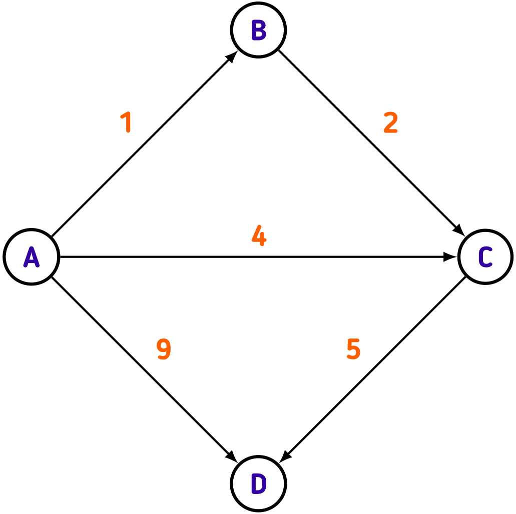
$$M = \begin{array}{cc} \begin{array}{cccc} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} \end{array} & \\ \begin{bmatrix} 0_A & & & \\ & 0_B & & \\ & & 0_C & \\ & & & 0_D \end{bmatrix} & \begin{array}{c} \mathsf{A} \\ \mathsf{B} \\ \mathsf{C} \\ \mathsf{D} \end{array} \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



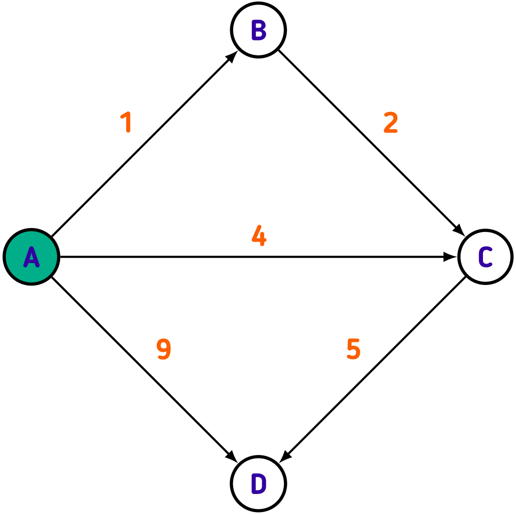
$$M = \begin{array}{cccc|c} A & B & C & D & \\ \hline 0_A & 1_A & 4_A & 9_A & A \\ & 0_B & 2_B & & B \\ & & 0_C & 5_C & C \\ & & & 0_D & D \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



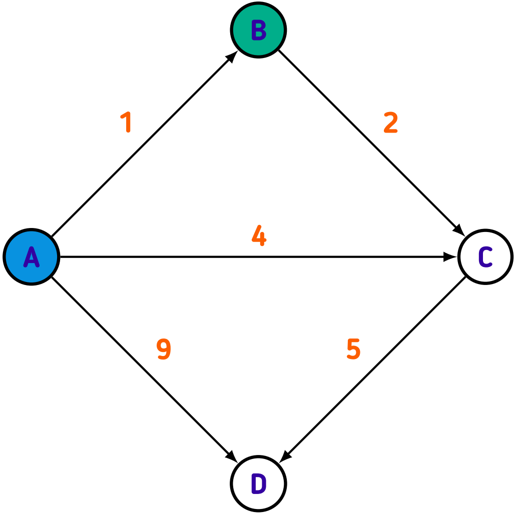
$$M = \begin{array}{|c|c|c|c|c} \multicolumn{1}{c}{\mathsf{A}} & \multicolumn{1}{c}{\mathsf{B}} & \multicolumn{1}{c}{\mathsf{C}} & \multicolumn{1}{c}{\mathsf{D}} & \\ \hline 0_{\mathsf{A}} & 1_{\mathsf{A}} & 4_{\mathsf{A}} & 9_{\mathsf{A}} & \mathsf{A} \\ \hline \infty_{-} & 0_{\mathsf{B}} & 2_{\mathsf{B}} & \infty_{-} & \mathsf{B} \\ \hline \infty_{-} & \infty_{-} & 0_{\mathsf{C}} & 5_{\mathsf{C}} & \mathsf{C} \\ \hline \infty_{-} & \infty_{-} & \infty_{-} & 0_{\mathsf{D}} & \mathsf{D} \\ \hline \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



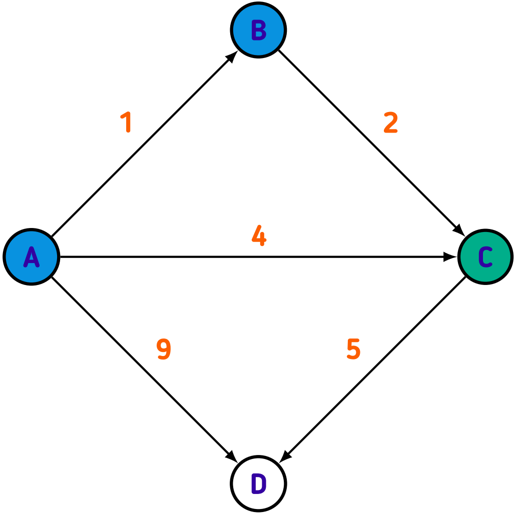
$$M = \begin{array}{cccc|c} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} & \\ \hline 0_{\mathsf{A}} & 1_{\mathsf{A}} & 4_{\mathsf{A}} & 9_{\mathsf{A}} & \mathsf{A} \\ \infty_{-} & 0_{\mathsf{B}} & 2_{\mathsf{B}} & \infty_{-} & \mathsf{B} \\ \infty_{-} & \infty_{-} & 0_{\mathsf{C}} & 5_{\mathsf{C}} & \mathsf{C} \\ \infty_{-} & \infty_{-} & \infty_{-} & 0_{\mathsf{D}} & \mathsf{D} \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



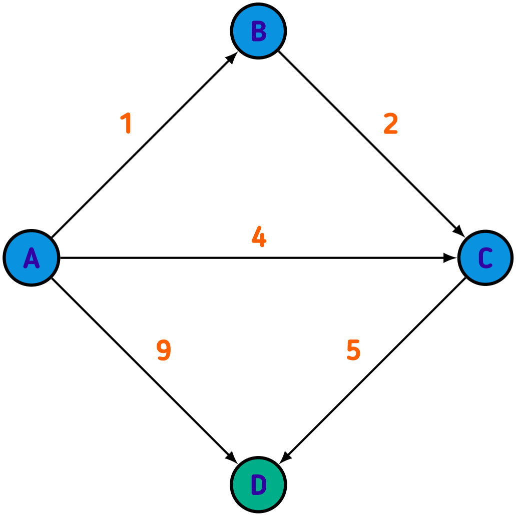
$$M = \begin{array}{c|c|c|c|c}
 & \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} \\
\hline
\mathsf{A} & 0_{\mathsf{A}} & 1_{\mathsf{A}} & \mathbf{3}_{\mathsf{B}} & 9_{\mathsf{A}} \\
\hline
\mathsf{B} & \infty_{-} & 0_{\mathsf{B}} & 2_{\mathsf{B}} & \infty_{-} \\
\hline
\mathsf{C} & \infty_{-} & \infty_{-} & 0_{\mathsf{C}} & 5_{\mathsf{C}} \\
\hline
\mathsf{D} & \infty_{-} & \infty_{-} & \infty_{-} & 0_{\mathsf{D}}
\end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



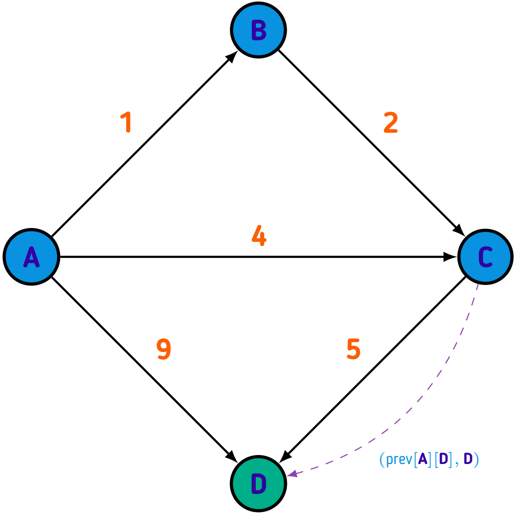
$$M = \begin{array}{cccccc} & \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} & \\ \left[\begin{array}{c} \\ \\ \\ \\ \end{array}\right. & \begin{array}{|c|}\hline 0_{\mathsf{A}} \\\hline \infty_{-} \\\hline \infty_{-} \\\hline \infty_{-} \\\hline \end{array} & \begin{array}{c|}\hline 1_{\mathsf{A}} \\\hline 0_{\mathsf{B}} \\\hline \infty_{-} \\\hline \infty_{-} \\\hline \end{array} & \begin{array}{c|}\hline 3_{\mathsf{B}} \\\hline 2_{\mathsf{B}} \\\hline 0_{\mathsf{C}} \\\hline \infty_{-} \\\hline \end{array} & \begin{array}{c|}\hline \mathbf{8}_{\mathsf{C}} \\\hline \mathbf{7}_{\mathsf{C}} \\\hline 5_{\mathsf{C}} \\\hline 0_{\mathsf{D}} \\\hline \end{array} & \begin{array}{c} \mathsf{A} \\ \mathsf{B} \\ \mathsf{C} \\ \mathsf{D} \end{array} \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



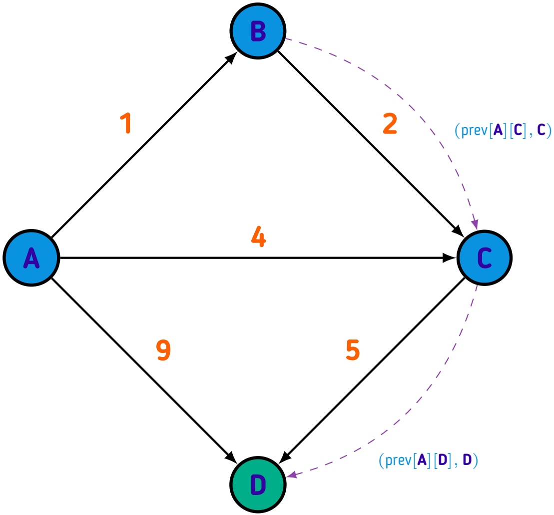
$$
M = \begin{array}{c c}
\begin{array}{cccc} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} \end{array} & \\
\left[\begin{array}{cccc}
0_{\mathsf{A}} & 1_{\mathsf{A}} & 3_{\mathsf{B}} & 8_{\mathsf{C}} \\
\infty_{-} & 0_{\mathsf{B}} & 2_{\mathsf{B}} & 7_{\mathsf{C}} \\
\infty_{-} & \infty_{-} & 0_{\mathsf{C}} & 5_{\mathsf{C}} \\
\infty_{-} & \infty_{-} & \infty_{-} & 0_{\mathsf{D}}
\end{array}\right] &
\begin{array}{c} \mathsf{A} \\ \mathsf{B} \\ \mathsf{C} \\ \mathsf{D} \end{array}
\end{array}
$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



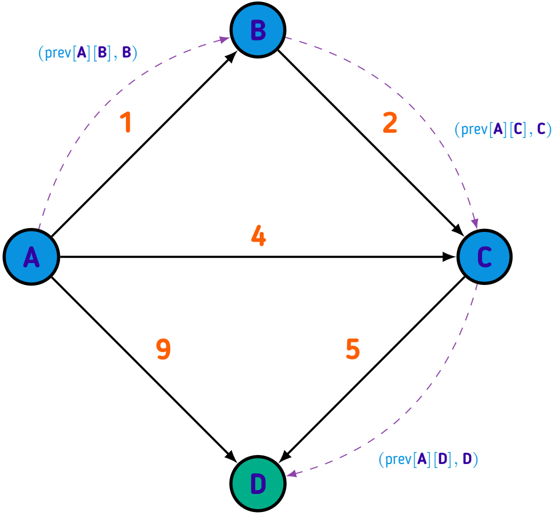
$$M = \begin{array}{cccc|c} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} & \\ \hline 0_{\mathsf{A}} & 1_{\mathsf{A}} & 3_{\mathsf{B}} & 8_{\mathsf{C}} & \mathsf{A} \\ \infty_{-} & 0_{\mathsf{B}} & 2_{\mathsf{B}} & 7_{\mathsf{C}} & \mathsf{B} \\ \infty_{-} & \infty_{-} & 0_{\mathsf{C}} & 5_{\mathsf{C}} & \mathsf{C} \\ \infty_{-} & \infty_{-} & \infty_{-} & 0_{\mathsf{D}} & \mathsf{D} \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



$$M = \begin{array}{cccc|c} \mathsf{A} & \mathsf{B} & \mathsf{C} & \mathsf{D} & \\ 0_{\mathsf{A}} & 1_{\mathsf{A}} & 3_{\mathsf{B}} & 8_{\mathsf{C}} & \mathsf{A} \\ \infty_{-} & 0_{\mathsf{B}} & 2_{\mathsf{B}} & 7_{\mathsf{C}} & \mathsf{B} \\ \infty_{-} & \infty_{-} & 0_{\mathsf{C}} & 5_{\mathsf{C}} & \mathsf{C} \\ \infty_{-} & \infty_{-} & \infty_{-} & 0_{\mathsf{D}} & \mathsf{D} \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$



$$M = \begin{array}{|c|c|c|c|l} \multicolumn{1}{c}{\mathsf{A}} & \multicolumn{1}{c}{\mathsf{B}} & \multicolumn{1}{c}{\mathsf{C}} & \multicolumn{1}{c}{\mathsf{D}} & \\ \hline 0_{\mathsf{A}} & 1_{\mathsf{A}} & 3_{\mathsf{B}} & 8_{\mathsf{C}} & \mathsf{A} \\ \hline \infty_{-} & 0_{\mathsf{B}} & 2_{\mathsf{B}} & 7_{\mathsf{C}} & \mathsf{B} \\ \hline \infty_{-} & \infty_{-} & 0_{\mathsf{C}} & 5_{\mathsf{C}} & \mathsf{C} \\ \hline \infty_{-} & \infty_{-} & \infty_{-} & 0_{\mathsf{D}} & \mathsf{D} \\ \hline \end{array}$$

$$m_{ij} = \mathsf{dist}[i][j]_{\mathsf{pred}[i][j]}$$

```cpp
pair<vector<vector<int>>, vector<vector<int>>>
floyd_warshall(int N)
{
    vector<vector<int>> dist(N + 1, vector<int>(N + 1, oo));
    vector<vector<int>> pred(N + 1, vector<int>(N + 1, oo));

    for (int u = 1; u <= N; ++u)
    {
        dist[u][u] = 0;
        pred[u][u] = u;
    }

    for (int u = 1; u <= N; ++u)
        for (auto [v, w] : adj[u]) {
            dist[u][v] = w;
            pred[u][v] = u;
        }
```

```
    for (int k = 1; k <= N; ++k)
    {
        for (int u = 1; u <= N; ++u)
        {
            for (int v = 1; v <= N; ++v)
            {
                if (dist[u][k] < oo and dist[k][v] < oo
                    and dist[u][v] > dist[u][k] + dist[k][v]) {
                        dist[u][v] = dist[u][k] + dist[k][v];
                        pred[u][v] = pred[k][v];
                }
            }
        }
    }

    return { dist, pred };
}
```

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

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

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

    return p;
}
```

Identificação de ciclos negativos

Identificação de ciclos negativos

- O algoritmo de Floyd-Warshall é capaz de detectar ciclos negativos

Identificação de ciclos negativos

- **O algoritmo de Floyd-Warshall é capaz de detectar ciclos negativos**
- **Inicialmente** $\text{dist}[u][u] = 0, \forall u \in V$**, se** G **não tem *autoloops***

Identificação de ciclos negativos

- $\star$ **O algoritmo de Floyd-Warshall é capaz de detectar ciclos negativos**
- $\star$ **Inicialmente** $\textsf{dist}[u][u] = 0, \forall u \in V$**, se** G **não tem** ***autoloops***
- $\star$ **Caso exista um ciclo negativo que passe por** u**, seguir este ciclo de** u **a** u **torna** $\textsf{dist}[u][u] < 0$

Identificação de ciclos negativos

- ⋆ **O algoritmo de Floyd-Warshall é capaz de detectar ciclos negativos**

- ⋆ **Inicialmente** $\text{dist}[u][u] = 0, \forall u \in V$**, se** G **não tem** ***autoloops***

- ⋆ **Caso exista um ciclo negativo que passe por** u**, seguir este ciclo de** u **a** u **torna** $\text{dist}[u][u] < 0$

- ⋆ **Assim,** G **terá um ciclo negativo se, ao final do algoritmo,** $\text{dist}[i][i] < 0$ **para algum** $i \in V$

```cpp
bool has_negative_cycle(int N) {
    for (int u = 1; u <= N; ++u)
        for (int v = 1; v <= N; ++v)
            dist[u][v] = u == v ? 0 : oo;

    for (int u = 1; u <= N; ++u)
        for (auto [v, w] : adj[u])
            dist[u][v] = w;

    for (int k = 1; k <= N; ++k)
        for (int u = 1; u <= N; ++u)
            for (int v = 1; v <= N; ++v)
                if (dist[u][k] < oo and dist[k][v] < oo)
                    dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v]);

    for (int i = 1; i <= N; ++i)
        if (dist[i][i] < 0) return true;

    return false;
}
```

Problemas sugeridos

1. Codeforces Round #179 (Div. 1) – Problem B: Greg and Graph
2. LightOJ – Travel Company
3. OJ 104 – Arbitrage
4. OJ 10171 – Meeting Prof. Miguel...

Referências

1. CP-Algorithms. *Floyd-Warshall Algorithm*, acesso em 03/08/2021.
2. HALIM, Felix; HALIM, Steve. *Competitive Programming 3*, 2010.
3. LAAKSONEN, Antti. *Competitive Programmer's Handbook*, 2018.
4. SKIENA, Steven; REVILLA, Miguel. *Programming Challenges*, 2003.