INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

Ciência da Computação

ESTRUTURA DE DADOS II

Prof. Adilso Nunes de Souza

BELLMAN-FORD

- Publicado em 1956 por Ford, em 1958 por Bellman e em 1957 por Edward F. Moore.
- Também conhecido como algoritmo de Bellman-Ford-Moore.
- Menos eficiente do que Dijkstra, mas trata arestas com pesos negativos. É capaz de detectar ciclos negativos.

BELLMAN-FORD

Lester Randolph Ford Jr.

Richard Ernest Bellman

BELLMAN-FORD

- O algoritmo de Bellman-Ford resolve o problema de caminhos mínimos de fonte única no caso geral no qual os pesos das arestas podem ser negativos.
- O algoritmo devolve um valor booleano que indica se existe ou não um ciclo de peso negativo que pode ser alcançado da fonte.
- Se tal ciclo existe, o algoritmo indica que não há nenhuma solução. Caso contrário produz os caminhos mínimos e seus pesos.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

BELLMAN-FORD

- Exemplo do algoritmo:

Bellman-Ford(G, w, s)

Initialize-Single-Source(G, s)
for $i \leftarrow 1$ **to** $|V[G]| - 1$
    do for each $(u, v) \in E[G]$
        do Relax(u, v, w)
for each $(u, v) \in E[G]$
    do if $d[v] > d[u] + w(u, v)$
        then return *FALSE* $\triangleright$ Ciclo negativo
return *TRUE* $\triangleright$ Sem ciclos negativos

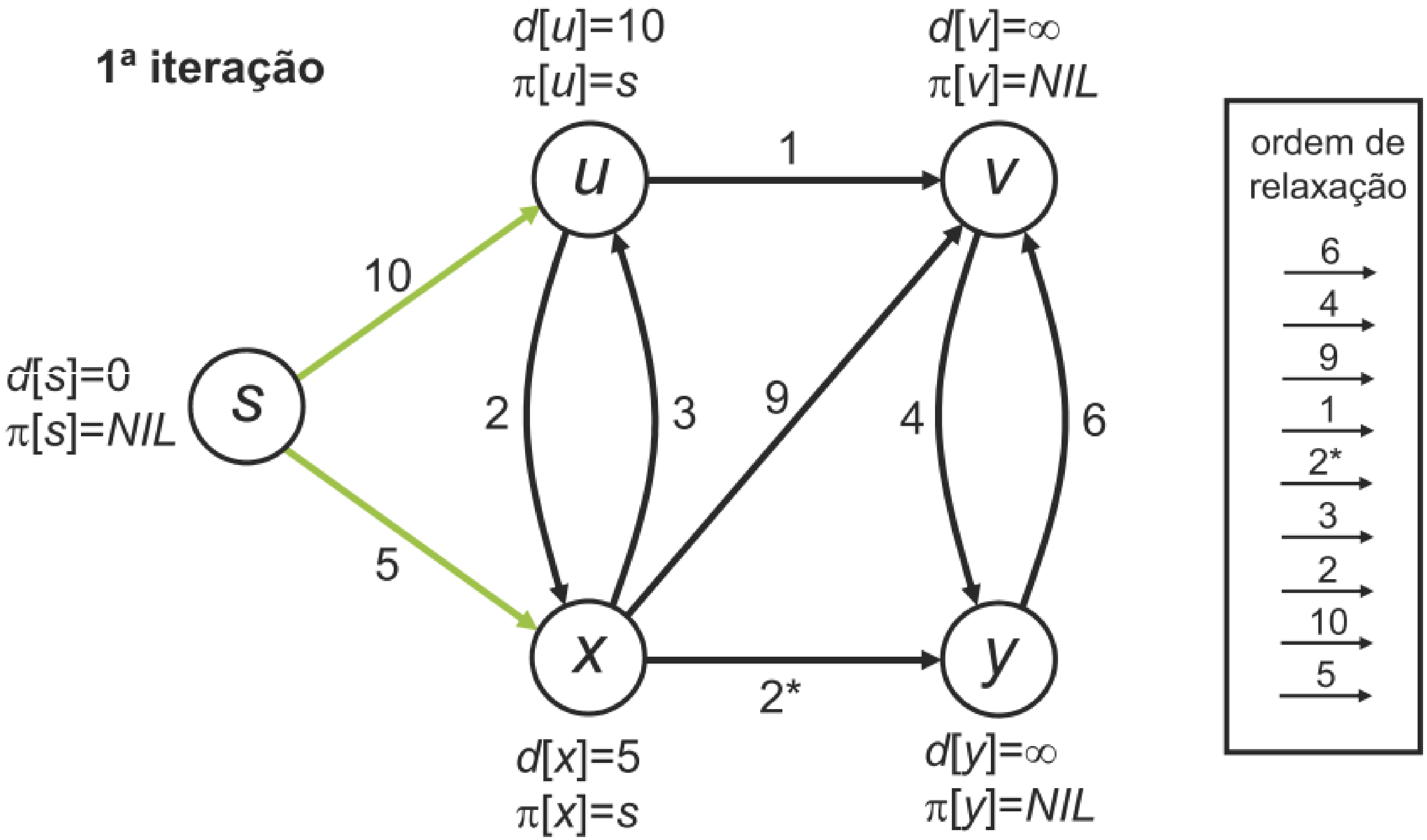
BELLMAN-FORD

- Relax(u,v,w)

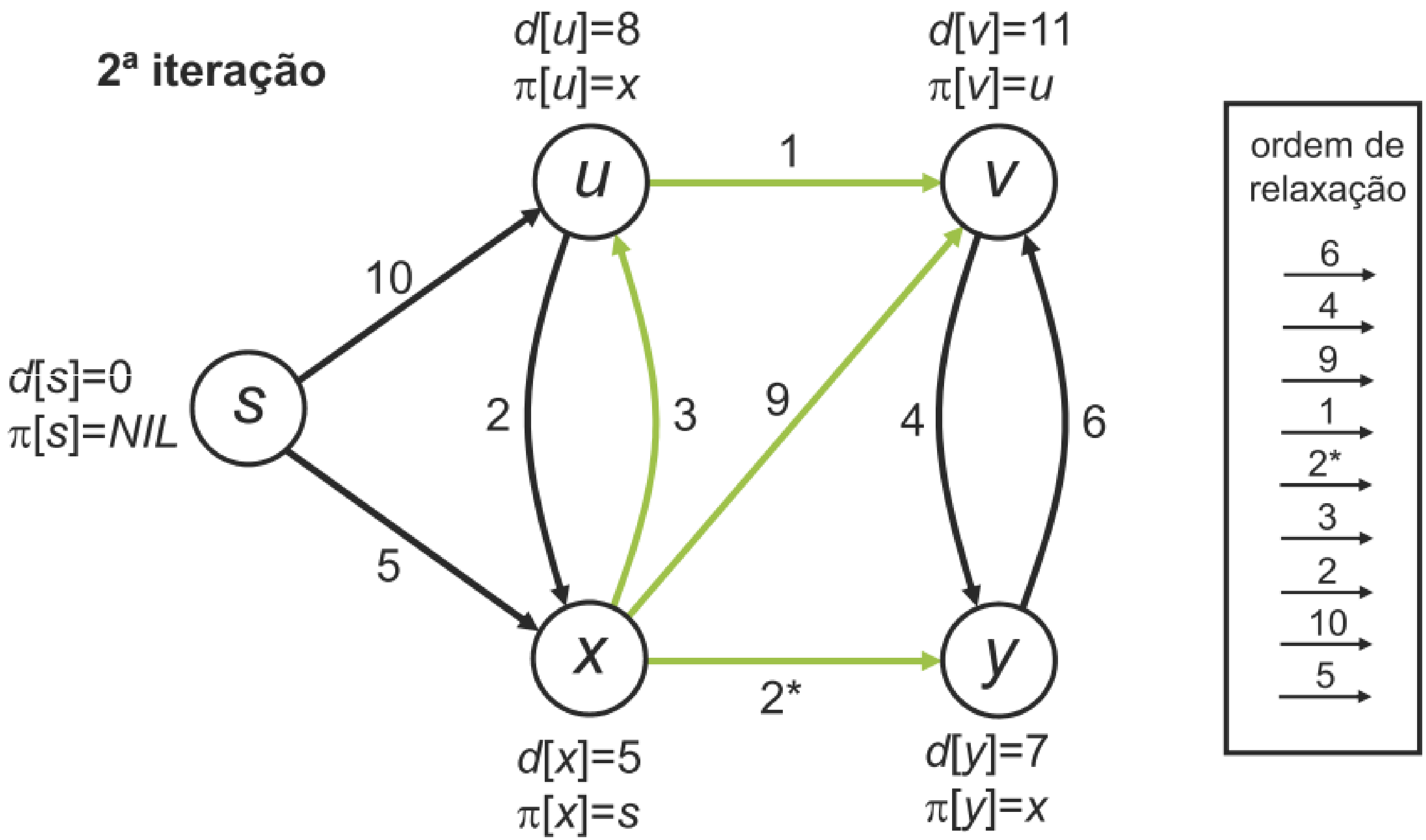
soma d[u] ao peso da aresta (u,v) (w(u,v))

Caso a soma seja menor que d[v], atualize d[v] e faça p[v] = u

BELLMAN-FORD

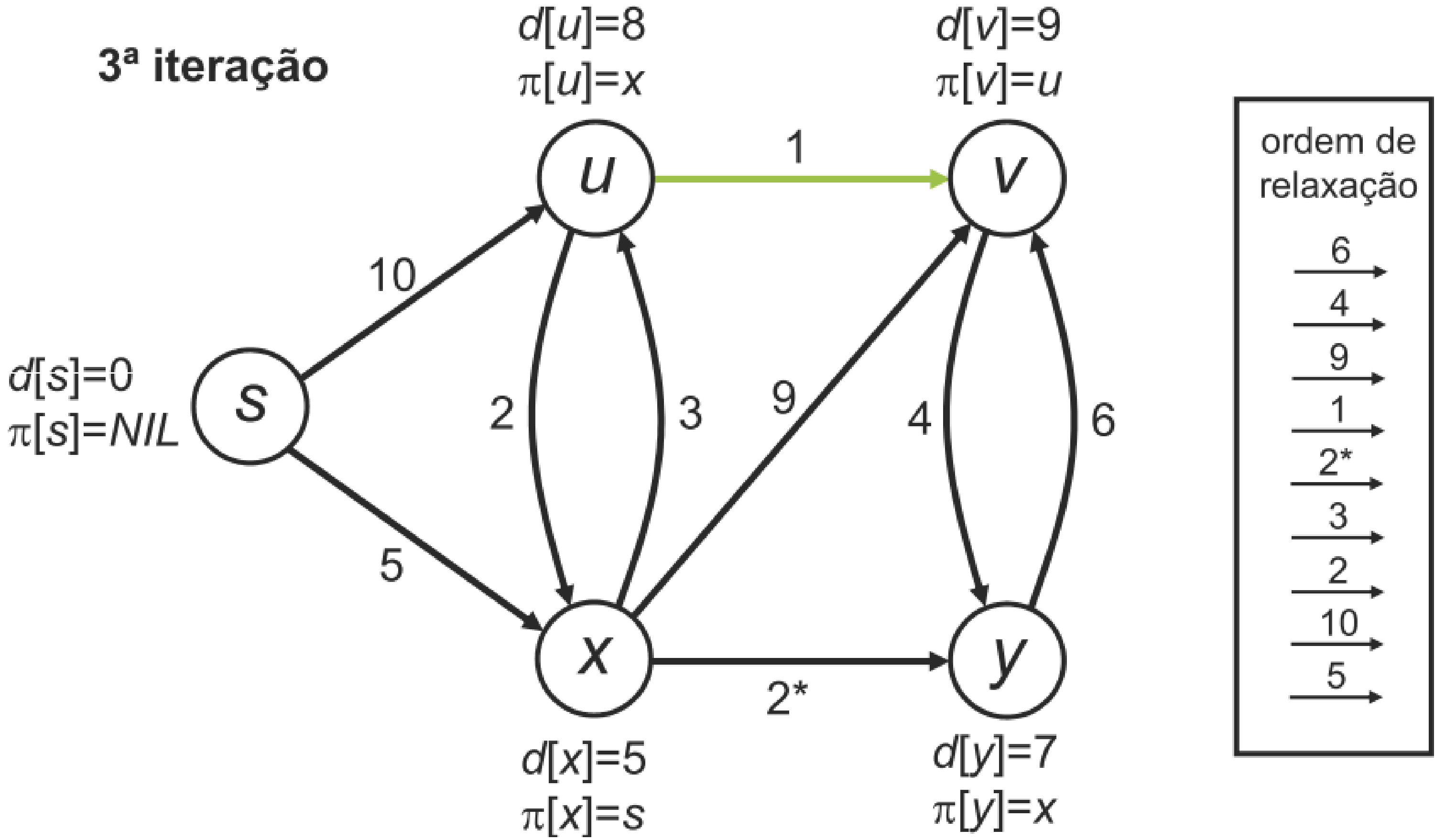


BELLMAN-FORD



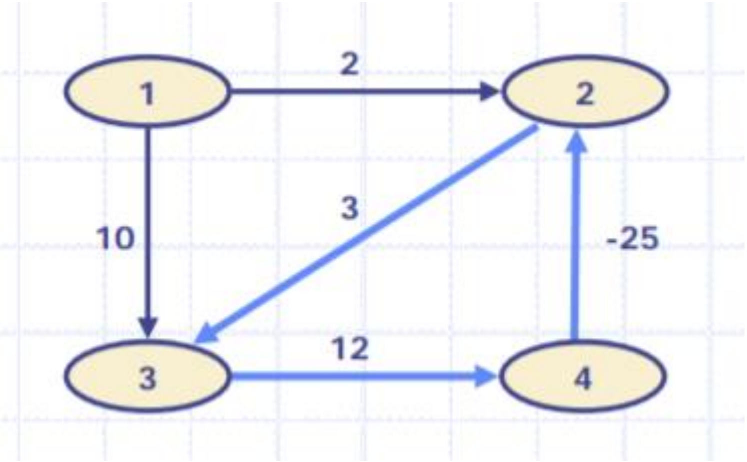
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

BELLMAN-FORD



BELLMAN-FORD

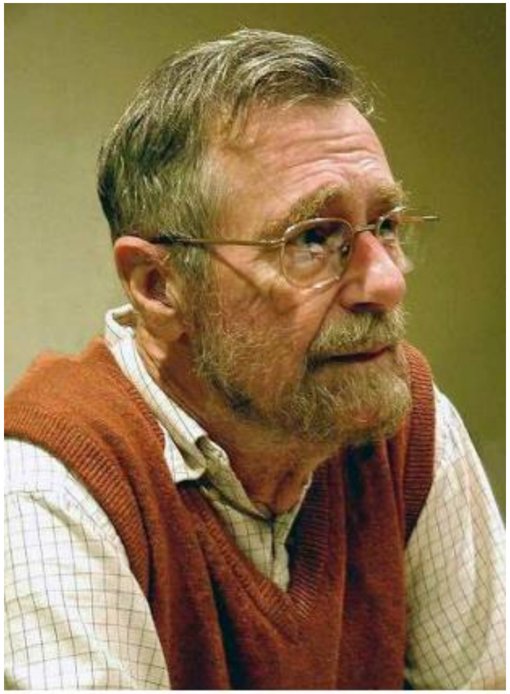
- Presença de ciclo negativo:



DIJKSTRA

- Em 1959 Dijkstra (1930–2002) sugeriu um algoritmo de rotulação para caminhos em grafos com arcos não negativos, utilizando indução e ajuste, eficiente e de fácil implementação computacional.
- Grafo deve ser conexo.
- Funciona em grafos direcionados e não direcionados.

DIJKSTRA



Edsger Wybe Dijkstra.

"Assim como Prim está para a árvore geradora mínima, Dijkstra está para árvore de caminhos mínimos".

DIJKSTRA

- O algoritmo de Dijkstra determina a árvore de caminhos mínimos a partir de s para qualquer grafo ponderado com pesos não negativos nas arestas.
- Mantém um conjunto de S vértices cujos pesos finais de caminho mínimo que parte da fonte S já foram determinados.
- O tempo de execução do algoritmo Dijkstra é inferior ao algoritmo de Bellman-Ford.

DIJKSTRA

- Coloca-se o vértice fonte na raiz da árvore e constrói-se a árvore uma aresta de cada vez.
- Toma-se sempre a aresta que estabelece o caminho mais curto do vértice fonte a algum vértice que ainda não esteja na SPT.
- Adicionam-se vértices à SPT por ordem crescente da sua distância (através da SPT) ao vértice de partida.

DIJKSTRA

- Este algoritmo pode ser visto como um método de procura generalizada, diferente da DFS, da BFS e do algoritmo de Prim apenas pela regra de entrada de arestas na árvore.

DIJKSTRA

Faça as inicializações

Enquanto houver vértice aberto:

Escolha u cuja estimativa seja a menor dentre os abertos

Feche u

Para todo nó aberto v na adjacência de u:

Relaxe a aresta(u,v)

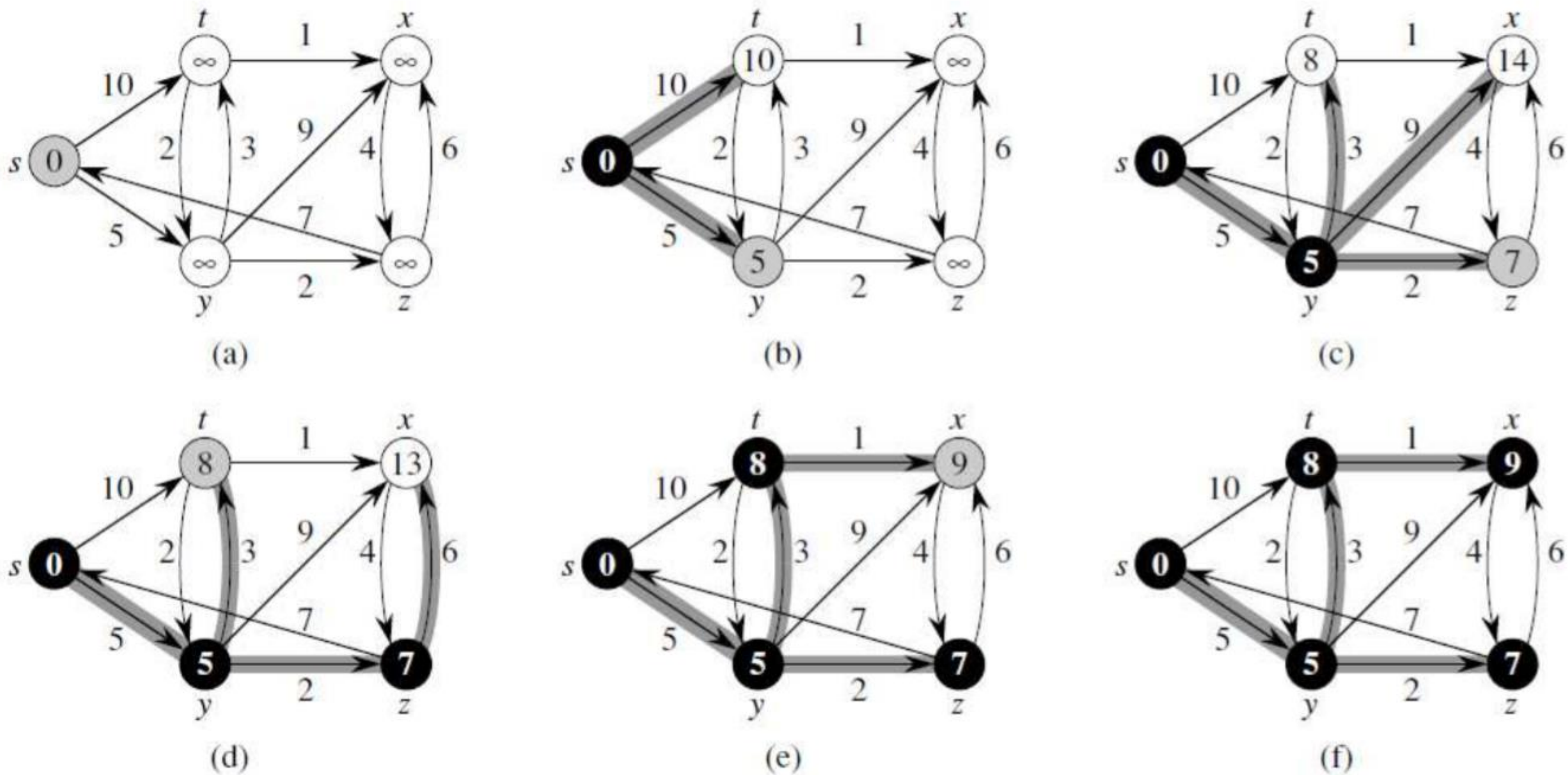
Relaxar:

soma d[u] ao peso da aresta (u,v)

Caso a soma seja menor que d[v], atualize d[v] e faça [p]v = u

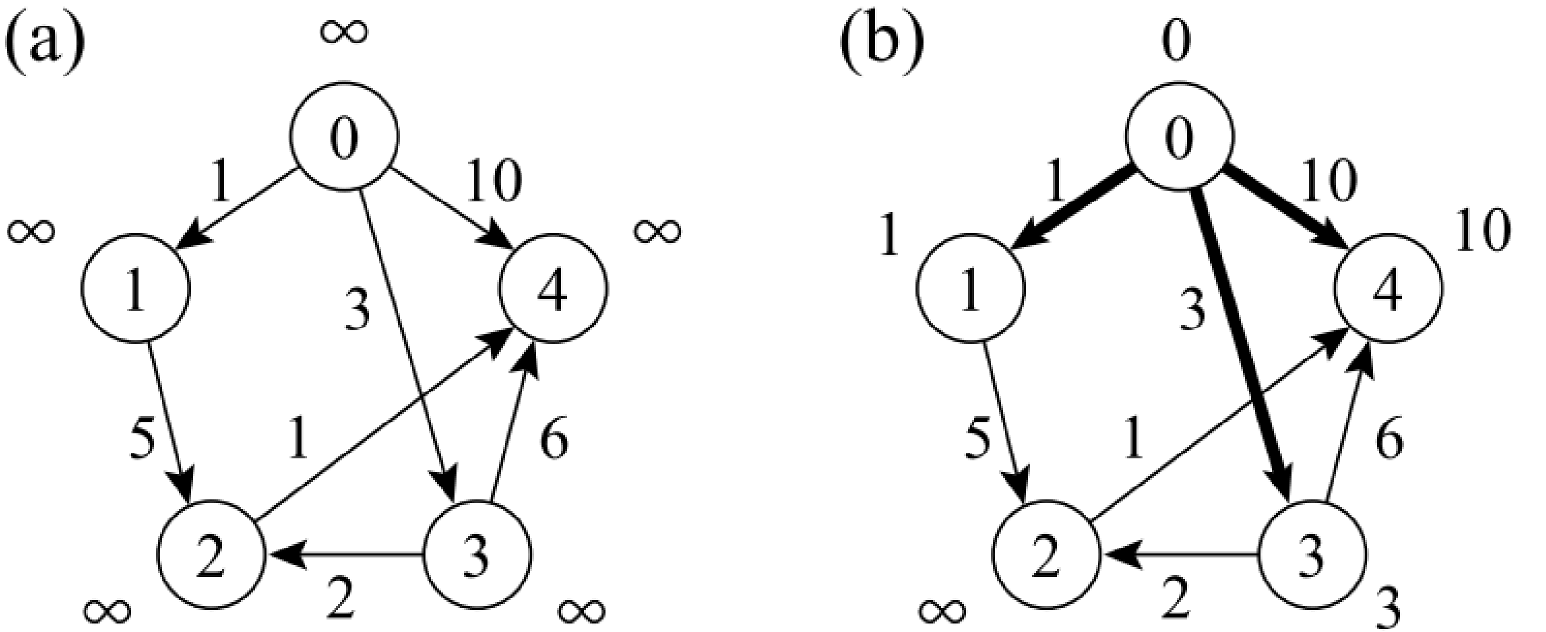
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

DIJKSTRA



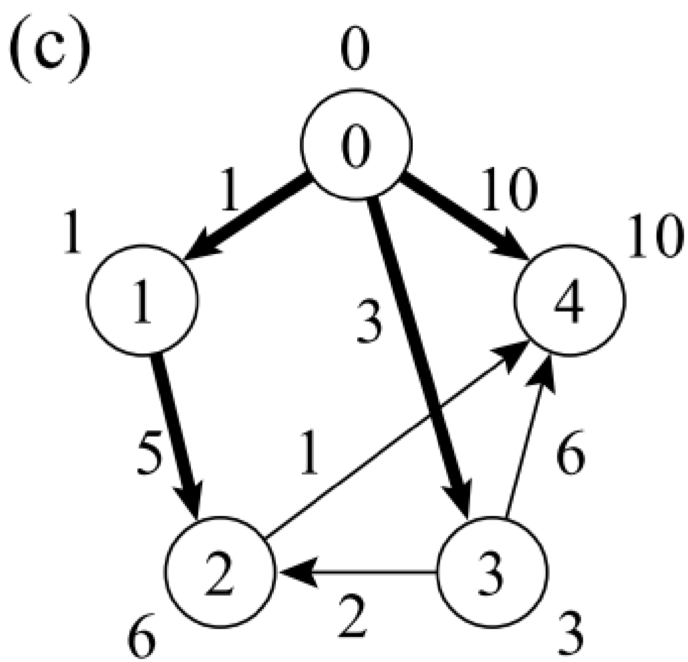
DIJKSTRA

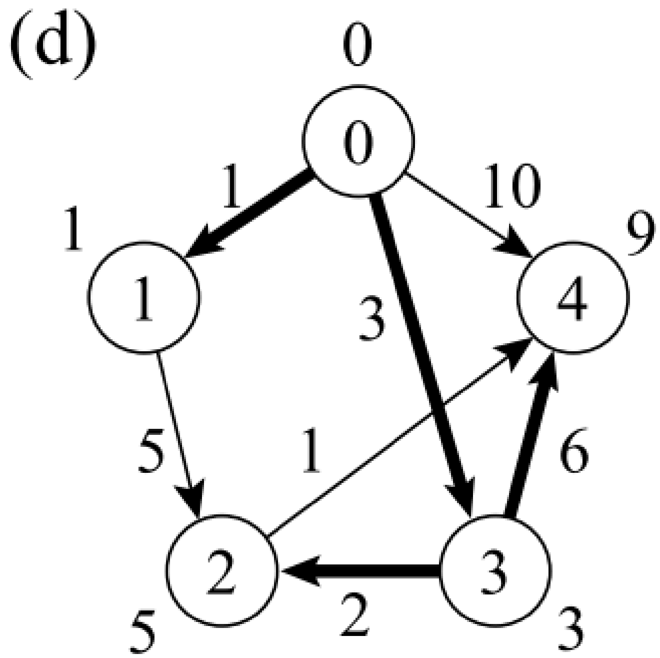
- Exemplo



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

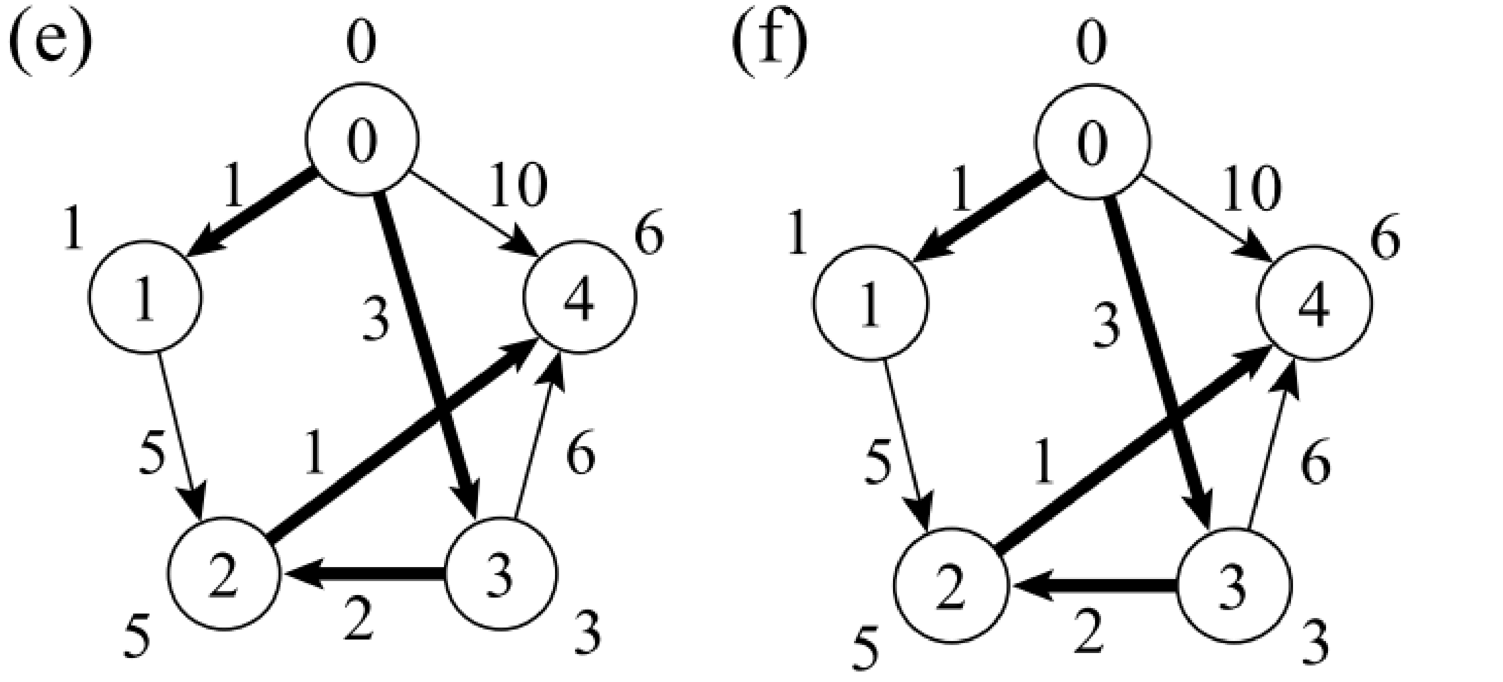
DIJKSTRA





INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

DIJKSTRA



DIJKSTRA

```
caminhos_mínimos (grafo g, vértice v) {
    iniciar o grafo;  iniciar a fila de prioridades pQ;
    enquanto  pQ não vazia {
        remover w de pQ; /* dist(v,w) é mínima */
        para todo x adjacente a w {
            nd = dist(v,w)+dist(w,x);
            se (nd < dist(v,x) {
                dist(v,x) = nd;
                rearranjar pQ;  /* dist(v,x) foi alterada */
            }
        }
    }
}
```

DIJKSTRA

iniciar o grafo( ){

para todo vértice w {

se E contém a aresta (v,w)

então dist(v,w) = distância 'direta' de v a w;

senão dist(v,w) = ∞ ;

}

iniciar fila de prioridades pQ( ) {

para todo vértice w

inserir w em pQ de acordo com dist(v,w);

}

FLOYD-WARSHALL

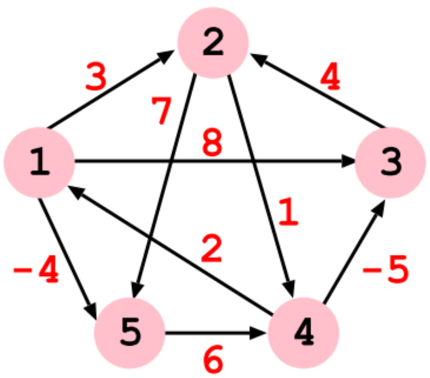
- O algoritmo de Floyd-Warshall é um algoritmo que resolve o problema de calcular o caminho mais curto entre todos os pares de vértices em um grafo orientado e valorado.
- O algoritmo Floyd-Warshall foi publicado por Robert Floyd em 1962.
- Conhecido como: Floyd's Algorithm, Roy-Warshall Algorithm, Roy-Floyd Algorithm.

FLOYD-WARSHALL

- O algoritmo de Floyd-Warshall considera os vértices intermediários de um caminho mínimo, onde um vértice intermediário de um caminho simples p = (v1, v2, ..., vi) é qualquer vértice de p exceto v1 ou vi, isto é qualquer vértice no conjunto(v2, v3,..., vi-1)
- Existe uma variedade de métodos diferentes para construir caminhos mínimos no algoritmo de Floyd-Warshall

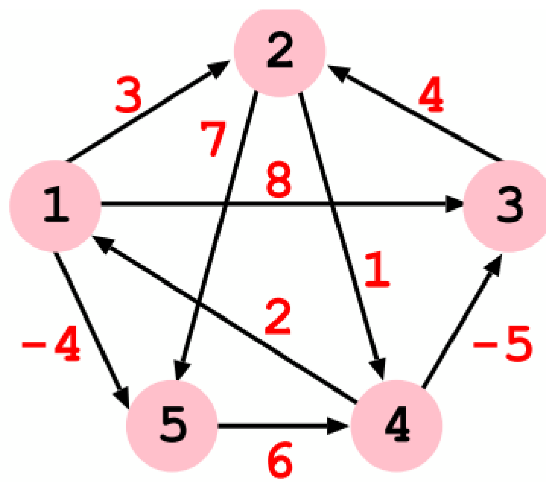
FLOYD-WARSHALL

- Um dos métodos é calcular a matriz de pesos de caminhos mínimos.



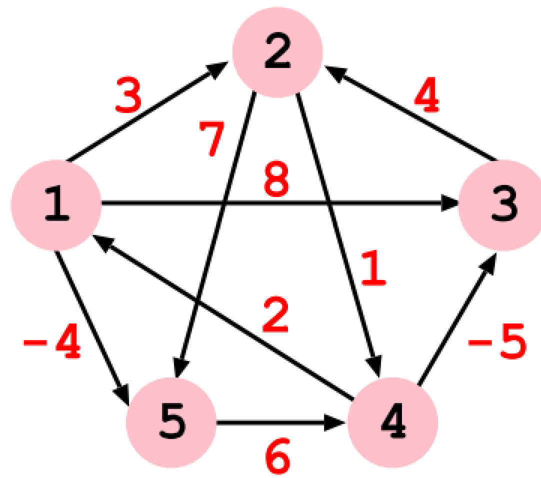
$$dist = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \quad pai = \begin{bmatrix} null & 1 & 1 & null & 1 \\ null & null & null & 2 & 2 \\ null & 3 & null & null & null \\ 4 & null & 4 & null & null \\ null & null & null & 5 & null \end{bmatrix}$$

FLOYD-WARSHALL



$$dist = \begin{bmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \quad pai = \begin{bmatrix} null & 1 & 1 & null & 1 \\ null & null & null & 2 & 2 \\ null & 3 & null & null & null \\ 4 & 1 & 4 & null & 1 \\ null & null & null & 5 & null \end{bmatrix}$$

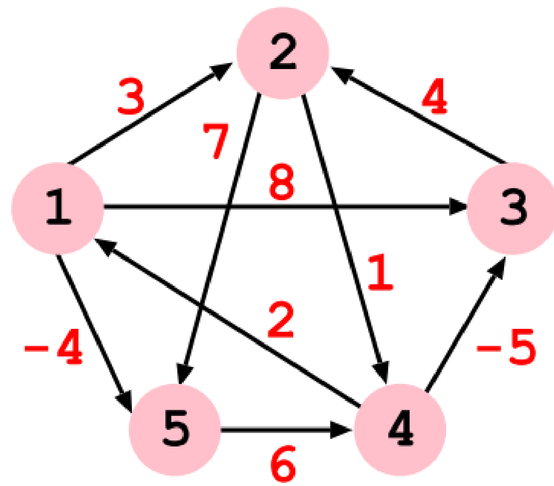
FLOYD-WARSHALL



$$dist = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \qquad pai = \begin{bmatrix} null & 1 & 1 & 2 & 1 \\ null & null & null & 2 & 2 \\ null & 3 & null & 2 & 2 \\ 4 & 1 & 4 & null & 1 \\ null & null & null & 5 & null \end{bmatrix}$$

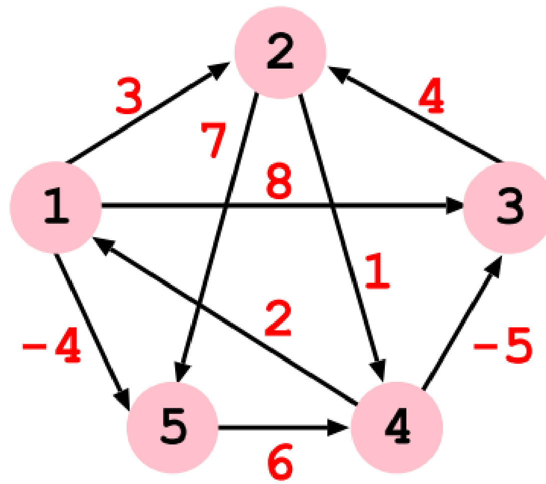
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

FLOYD-WARSHALL



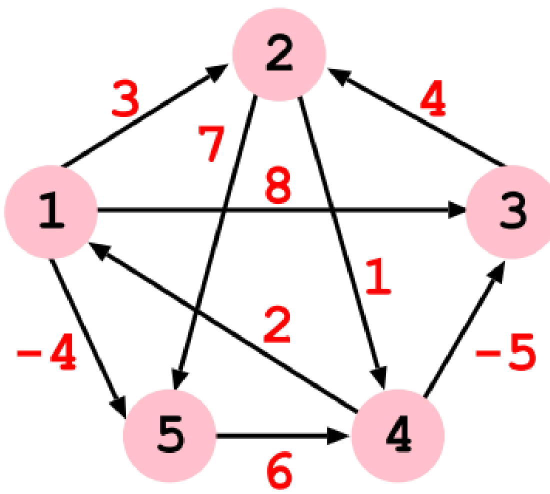
$$dist = \begin{bmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{bmatrix} \qquad pai = \begin{bmatrix} null & 1 & 1 & 2 & 1 \\ null & null & null & 2 & 2 \\ null & 3 & null & 2 & 2 \\ 4 & 3 & 4 & null & 1 \\ null & null & null & 5 & null \end{bmatrix}$$

FLOYD-WARSHALL



$$dist = \begin{bmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{bmatrix} \qquad pai = \begin{bmatrix} null & 1 & 4 & 2 & 1 \\ 4 & null & 4 & 2 & 1 \\ 4 & 3 & null & 2 & 1 \\ 4 & 3 & 4 & null & 1 \\ 4 & 3 & 4 & 5 & null \end{bmatrix}$$

FLOYD-WARSHALL



$$dist = \begin{bmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{bmatrix} \qquad pai = \begin{bmatrix} null & 3 & 4 & 5 & 1 \\ 4 & null & 4 & 2 & 1 \\ 4 & 3 & null & 2 & 1 \\ 4 & 3 & 4 & null & 1 \\ 4 & 3 & 4 & 5 & null \end{bmatrix}$$

FLOYD-WARSHALL

```
floyd-warshall(W):
    dist = W
    para k=1 a n faça
        para i=1 a n faça
            para j=1 a n faça
                dist[i,j]=min(d[i,j],d[i,k]+d[k,j])
    retorne dist;
```

CONSIDERAÇÕES

- Floyd-Warshall e Bellman Ford trabalha com arestas de peso negativo enquanto Dijkstra não.
- No Dijkstra, é possível reproduzir o caminho, enquanto que o Floyd-Warshall fornece apenas o caminho mais curto, e não a sequência das arestas.

REFERÊNCIAS

- PEREIRA, Silvio do Lago. Estrutura de Dados Fundamentais: Conceitos e Aplicações, 12. Ed. São Paulo, Érica, 2008.
- BACKES, André Ricardo, Estrutura de dados descomplicada: em linguagem C, 1 Ed. – Rio de Janeiro: Elsevier, 2016.
- ROCHA, Anderson, Grafos – Representações, buscas e Aplicações.
- SENGER, H., Notas de Aula, Universidade de São Judas Tadeu, 1999.
- WALDEMAR Celes, Renato Cerqueira, José Lucas Rangel, Introdução a Estruturas de Dados, Editora Campus (2004).
- VELOSO, Paulo. SANTOS, Celso dos. AZEVEDO, Paulo. FURTADO, Antonio. Estrutura de dados. Rio de Janeiro: Ed. Elsevier, 1983 27ª reimpressão.
- https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
- ZIVIANI, Nivio. Projeto de Algoritmos com implementações em Java e C++, 2007.