Problemas em Grafos

Caminho Mínimo - Algoritmo de Bellman-Ford

Gabriel Ramalho
Túlio Lemes
Vinicius Rodrigues

1. Motivação

1.1 Definição do Problema:

O algoritmo de Bellman-Ford resolve o problema do caminho mais curto de única origem para o caso mais geral. Diferentemente do algoritmo de Dijkstra, o algoritmo de Bellman-Ford não impõe nenhuma restrição sobre o sinal do peso das arestas, o que o torna uma solução mais genérica.

Podemos tomar como exemplo eventos da vida real:

- Redes de computadores: Protocolos de roteamento do vetor de distância
- Economia: Problema “Triangular Arbitrage”

1.2 Descrição Informal:

- O problema do caminho mínimo consiste na obtenção do menor custo possível entre dois vértices em um grafo onde suas arestas possuem pesos.

- Dado um grafo com pesos nas arestas e sabendo qual vértice será a origem podemos calcular o valor mínimo e o caminho necessário para chegar a qualquer outro nó.

1.3 Descrição Formal:

- ***Questão***: Dado um grafo definido pelos seus vértices e suas arestas temos que calcular a soma mínima dos pesos de suas arestas para chegar do nó origem a qualquer outro nó.

- ***Entrada***: Vértices, arestas e origem.

- ***Saída***: Grafo com a distância mínima calculada em cada nó para se chegar ao nó origem e também o nó antecessor para efetuar o menor caminho.

1.4 Ideia

- O algoritmo está divido em três etapas (inicialização, relaxamento e verificação de ciclos negativos). A primeira, a inicialização, é responsável por padronizar as distâncias antes do inicio da resolução. O relaxamento fica responsável pelo cálculo do caminho mínimo e a última etapa se responsabiliza em verificar se é possível ou não calcular o caminho mínimo partindo do princípio que não se pode ter um ciclo negativo.

- A existência e cálculo do caminho mais curto são garantidos caso não haja a presença de ciclos negativos durante o caminho da origem até um nó v do grafo. Se há um ciclo negativo, em princípio, o problema não tem solução, pois o “caminho” pode passar ao longo do ciclo infinitas vezes obtendo caminhos cada vez menores.

1.4 Inicialização

A inicialização é uma etapa simples, onde se padroniza os valores de distância mínima para cada nó. Iremos percorrer todos os vértices e iremos definir que a sua distância mínima no momento é infinito, enquanto na origem iremos colocar 0.

1.4 Relaxamento

- A técnica do relaxamento consiste em verificar se pode ser encontrado um caminho mais curto para v (do que aquele encontrado até o momento) passando pelo vértice u. Ou seja, verificamos se caminho passando pelo vértice u é menor do que a distância anteriormente calculada.

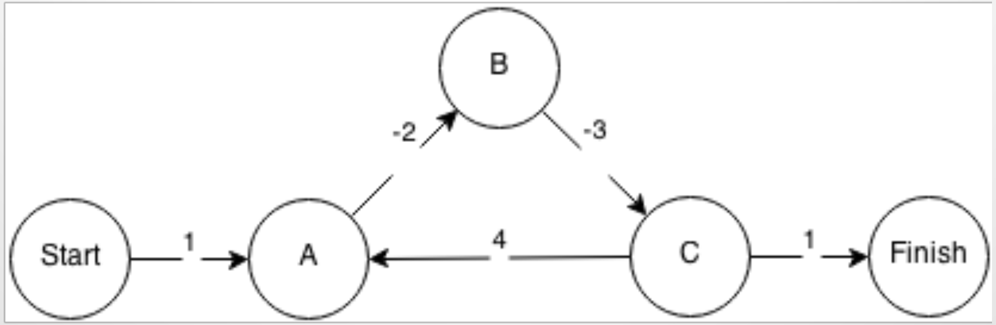
- Se distância(origem,u) + peso(u,v) < distancia(origem,v) então:

    - Distância(origem, v) (origem, u) + peso(u,v)
    - Nó predecessor u

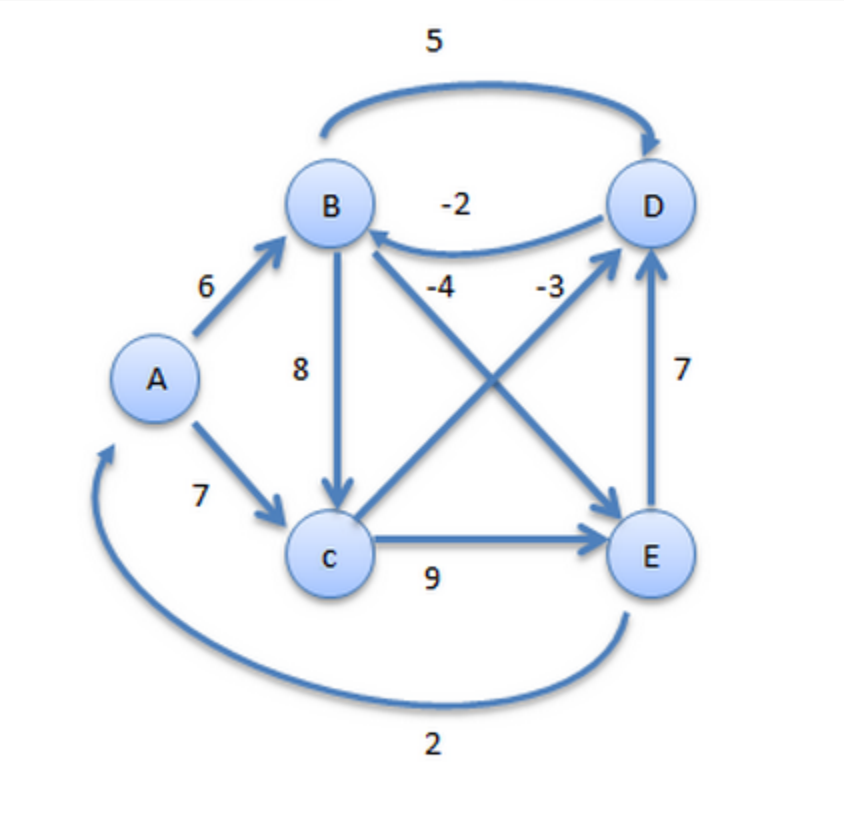
1.4 Checagem de Ciclos Negativos

- Depois de executado (V-1) a técnica do relaxamento, precisamos verificar se o grafo não contém um ciclo negativo.

- Para verificar se o grafo não contém um ciclo negativo executaremos a técnica do relaxamento mais uma vez e se conseguirmos minimizar a distância mínima para qualquer nó provaremos que existe um ciclo negativo.



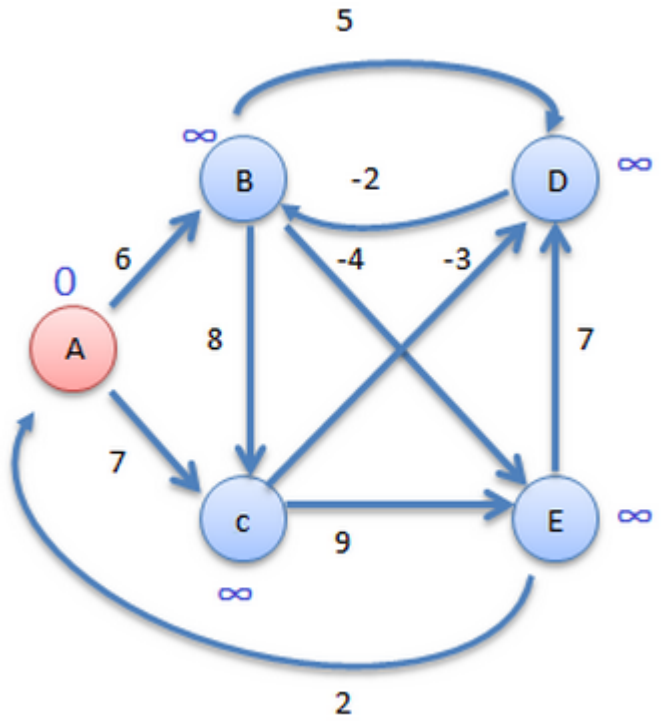
1.4 Exemplo:



1.4 Exemplo:

Passo 1:
Definimos o vértice A como fonte:

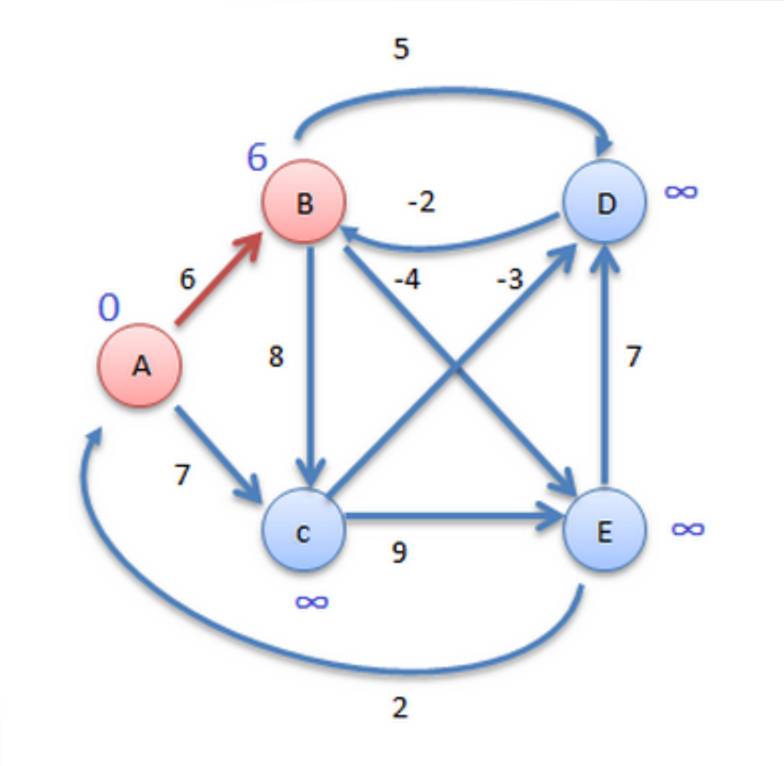
Vértices:	A	B	C	D	E
Distância	0	6	7	∞	∞
Distância de:	A	A	A		



1.4 Exemplo:

Passo 2:
Agora o vértice B como fonte:

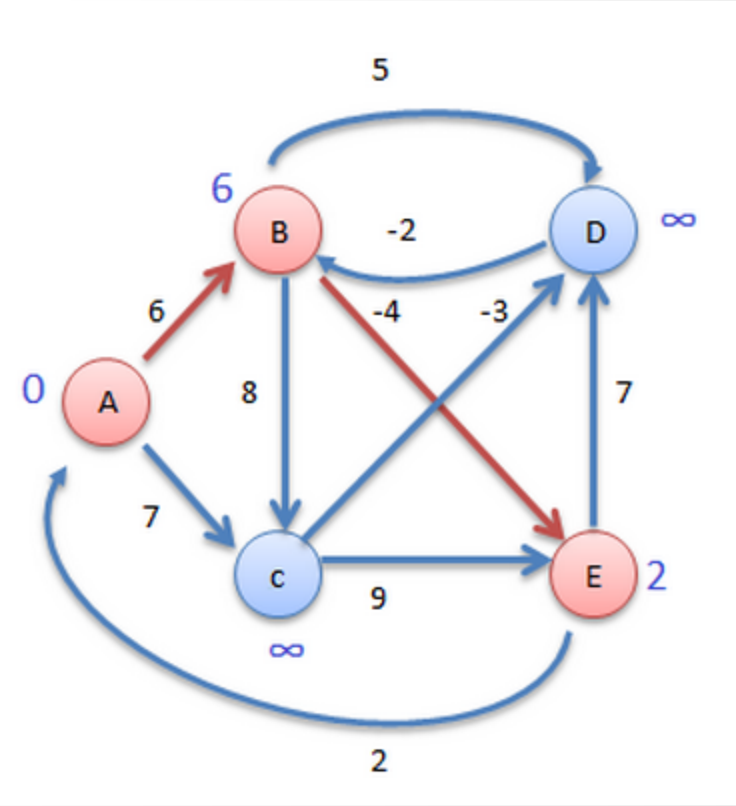
Vértices:	A	B	C	D	E
Distância	0	6	7	11	2
Distância de:	A	A	A	B	B



1.4 Exemplo:

Passo 3:
Agora o vértice E como fonte:

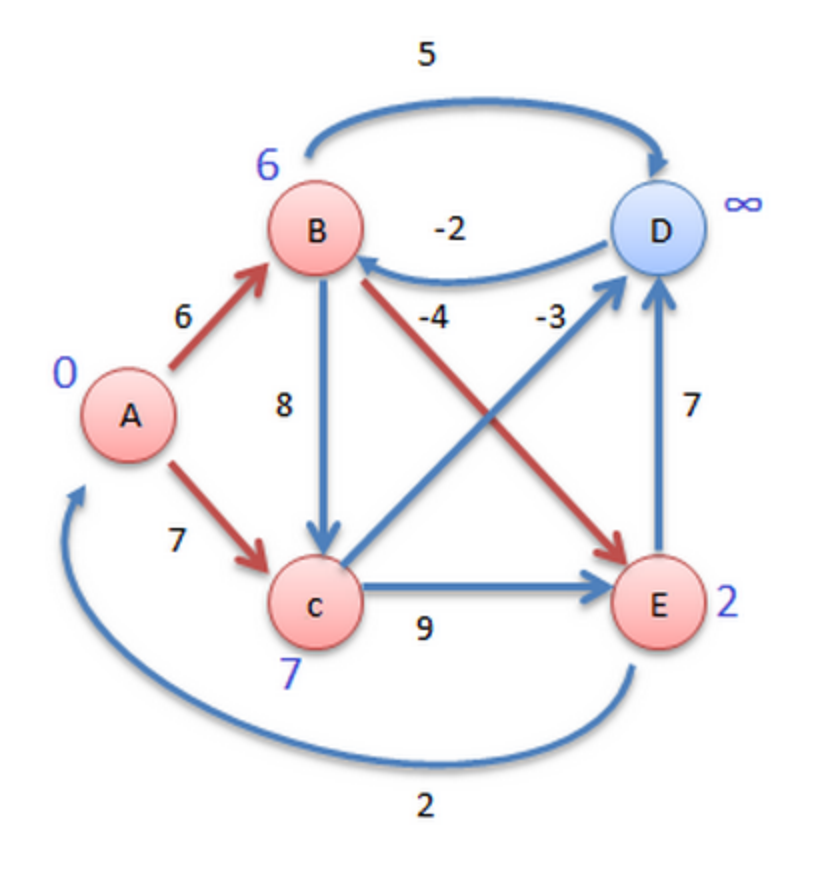
Vértices:	A	B	C	D	E
Distância	0	6	7	9	2
Distância de:	A	A	A	E	B



1.4 Exemplo:

Passo 4:
Agora o vértice C como fonte:

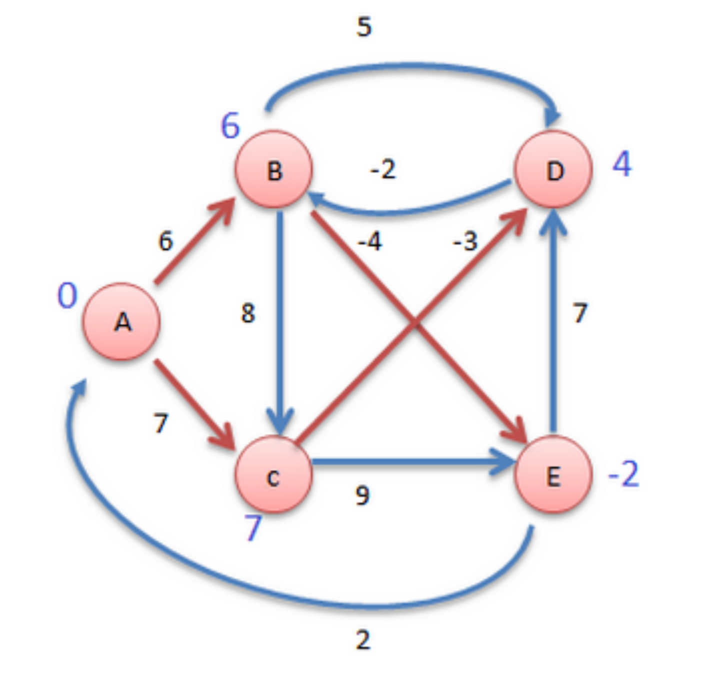
Vértices:	A	B	C	D	E
Distância	0	6	7	4	2
Distância de:	A	A	A	C	B



1.4 Exemplo:

Passo 5:
Agora o vértice D como fonte:

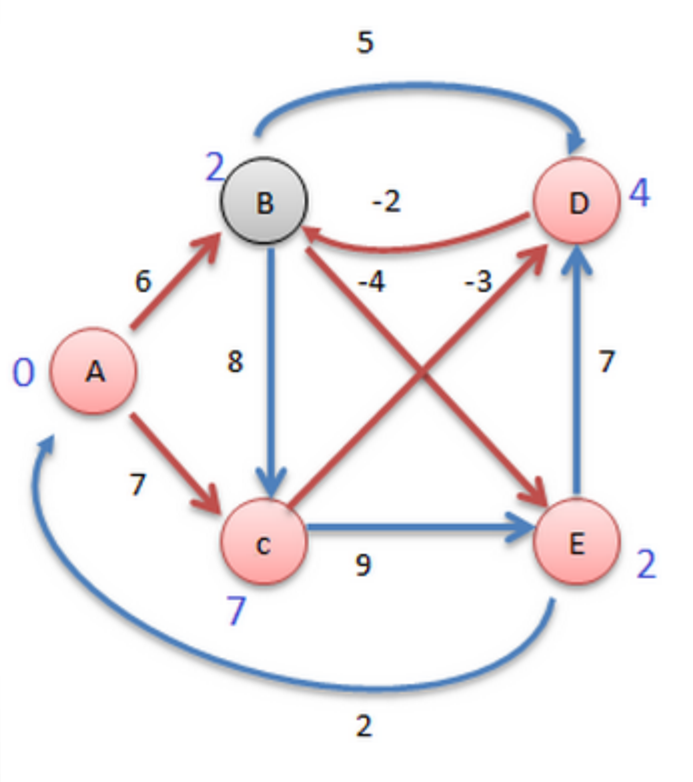
Vértices:	A	B	C	D	E
Distância	0	2	7	4	2
Distância de:	A	D	A	C	B



1.4 Exemplo:

Passo 6:
Agora o vértice B como fonte:

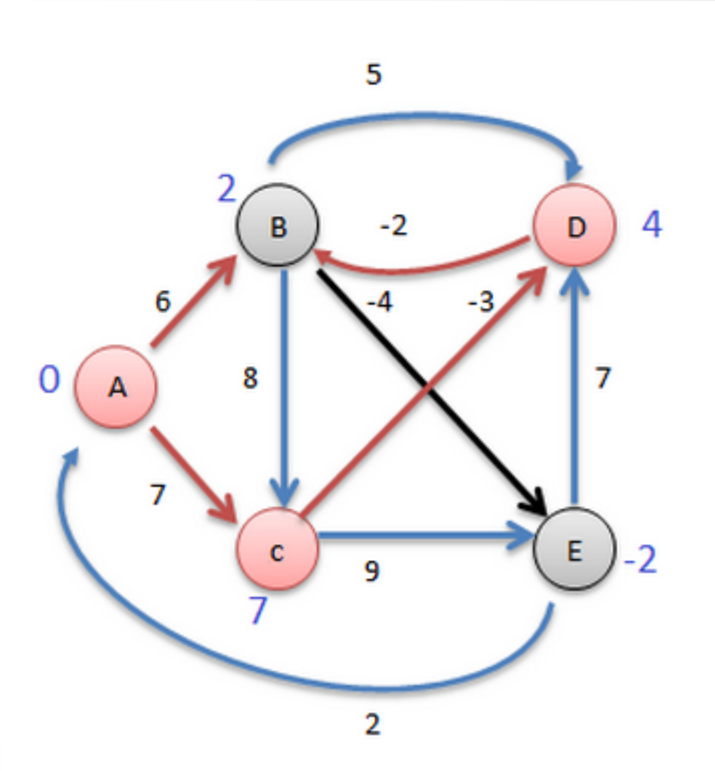
Vértices:	A	B	C	D	E
Distância	0	2	7	4	-2
Distância de:	A	D	A	C	B



1.4 Exemplo:

Passo 7:
Agora o vértice E como fonte:

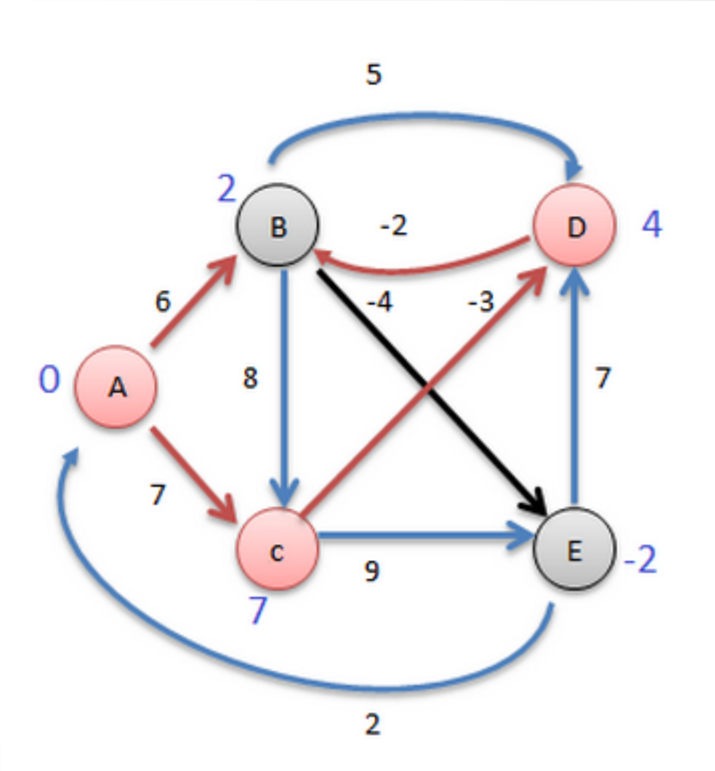
Vértices:	A	B	C	D	E
Distância	0	2	7	4	-2
Distância de:	A	D	A	C	B



1.4 Exemplo:

Conclusão:

Vértice	Distância de A
A	0
B	2
C	7
D	4
E	-2

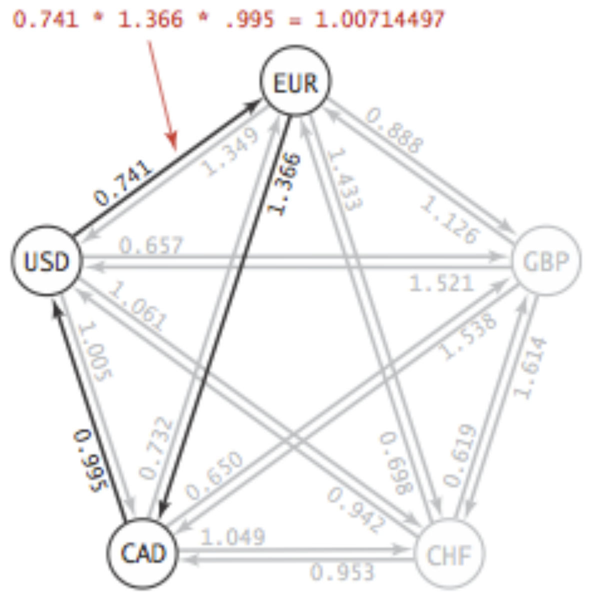


1.4 Exemplo (Triangular Arbitrage):

- Com 1.000 dólares americanos podemos comprar 1.000 × 0,741 = 741 euros, então podemos comprar 741 × 1,366 = 1.012,206 dólares canadenses, e finalmente, 1.012,206 × 0,995 = 1.007,14497 dólares americanos com nossos dólares canadenses, obtendo 7,14497 de lucro

- Se trocarmos os pesos das arestas pelo seu respectivo log, verificamos a existência de um ciclo negativo

- Portanto o problema pode ser modelado pela busca de um ciclo negativo no grafo e podemos utilizar o algorimo de Bellman-Ford para isso



An arbitrage opportunity

2. Algoritmo

```
BellmanFord (vértices, arestas, origem)
    para cada vértice v em vértices faça: // Inicialização
        se v é origem então:
            v.distância = 0
        senão:
            v.distância = infinito
            v.anterior = nulo

    para i = 1 até tamanho(vértices) - 1: // Relaxamento
        para cada aresta uv em arestas faça:
            u = uv.origem
            v = uv.destino
            se v.distância > u.distância + uv.peso então:
                v.distância = u.distância + uv.peso
                v.anterior = u

    para cada aresta uv em arestas faça: // Checagem de ciclos negativos
        u = uv.origem
        v = uv.destino
        se v.distância < u.distância + uv.peso então:
            erro "Ciclo de peso negativo"
```

3. Análise do Algoritmo

3.1 Prova de Corretude:

Supondo que o grafo orientado ponderado G=(V, E) não possui ciclos negativos atingíveis por **s** (origem). Provamos que o algortimo está correto para o grafo G com base na propriedade de relaxamento de caminho.

Considere qualquer vértice **v** que seja acessível a partir de **s**, e seja **p** = (v_0, v_1, … v_k), onde v_0 = **s** e v_k = **v**, qualquer acíclico caminho mais curto de **s** para **v.** O caminho **p** tem no máximo |V| - 1 arestas, e assim k <= |V| - 1. Cada uma das |V| - 1 iterações do loop no algortimo relaxa todas as |E| arestas.

Entre as arestas relaxadas na i-ésima iteração, para i = 1, 2, … k, encontra-se (v_{i-1}, v_i). Então pela propriedade de relaxamento de caminho, d [v] = d[v_k] = dist(s, v_k) = dist(s, v).

3. Análise do Algoritmo

3.2 Complexidade:

```
para cada vértice v em vértices faça: // Inicialização
    se v é origem então:
        v.distância = 0
    senão:
        v.distância = infinito
        v.anterior = nulo
```

- Inicialização: V

3. Análise do Algoritmo

3.2 Complexidade:

```
para i = 1 até tamanho(vértices) - 1: // Relaxamento
    para cada aresta uv em arestas faça:
        u = uv.origem
        v = uv.destino
        se v.distância > u.distância + uv.peso então:
            v.distância = u.distância + uv.peso
            v.anterior = u
```

- Relaxamento: A * (V - 1)

3. Análise do Algoritmo

3.2 Complexidade:

```
para cada aresta uv em arestas faça: // Checagem de ciclos negativos
    u = uv.origem
    v = uv.destino

    se v.distância < u.distância + uv.peso então:
    erro "Ciclo de peso negativo"
```

- Checagem de ciclos negativos: A

3. Análise do Algoritmo

3.2 Complexidade:

- **Inicialização**: V
- **Relaxamento**: A * (V - 1)
- **Checagem de ciclos negativos**: A

Total = V + (A*V-1) + A = **O(A*V)**

4. Conclusão

Vantagens:

- Simples implementação
- Permite arestas com peso negativo

Desvantagens:

- Tempo maior de execução em comparação com o algoritmo de Dijkstra
- Alto custo em relação aos casos onde o algoritmo guloso retorna uma solução ótima

4. Conclusão

- Complexidade: O(A*V)
- Não permite ciclos negativos
- Menos eficiente que o algoritmo de Dijkstra

Bibliografia

- http://www.inf.ufrgs.br/~cgdaudt/inf05515/art2.pdf
- http://scanftree.com/Data_Structure/bellman-ford-algorithm
- http://www.ic.unicamp.br/~rezende/ensino/mo417/2010s2/Slides/Aula23.pdf
- http://algs4.cs.princeton.edu/44sp/
- http://professor.unisinos.br/pjaques/material/est2/15-grafos-caminhos-minimos-pesos-negativos.pdf