



# Algoritmo de Bellman Ford

O algoritmo Bellman Ford nos ajuda a encontrar o caminho mais curto de um vértice para todos os outros vértices de um gráfico ponderado.

É semelhante ao [algoritmo de Dijkstra, \(/dsa/dijkstra-algorithm\)](/dsa/dijkstra-algorithm), mas pode funcionar com gráficos nos quais as arestas podem ter pesos negativos.

---

## Por que alguém teria arestas com pesos negativos na vida real?

Bordas de peso negativo podem parecer inúteis no início, mas podem explicar muitos fenômenos como fluxo de caixa, o calor liberado / absorvido em uma reação química, etc.

Por exemplo, se houver maneiras diferentes de ir de uma substância química A a outra substância química B, cada método terá sub-reações envolvendo tanto a dissipação quanto a absorção de calor.

Se quisermos encontrar o conjunto de reações em que a energia mínima é necessária, precisaremos ser capazes de fatorar a absorção de calor como pesos negativos e a dissipação de calor como pesos positivos.

---

## Por que precisamos ter cuidado com pesos negativos?

Arestas de peso negativo podem criar ciclos de peso negativo, ou seja, um ciclo que reduzirá a distância total do caminho ao voltar ao mesmo ponto.



Procure tutoriais e exemplos

[www.domain-name.com](http://www.domain-name.com)



Os ciclos de peso negativo podem dar um resultado incorreto ao tentar descobrir o caminho mais curto

Algoritmos de caminho mais curto, como o algoritmo de Dijkstra, que não são capazes de detectar tal ciclo, podem fornecer um resultado incorreto porque podem passar por um ciclo de peso negativo e reduzir o comprimento do caminho.

## Como funciona o algoritmo de Bellman Ford

O algoritmo Bellman Ford superestima o comprimento do caminho do vértice inicial a todos os outros vértices. Em seguida, ele relaxa iterativamente essas estimativas, encontrando novos caminhos que são mais curtos do que os caminhos anteriormente superestimados.

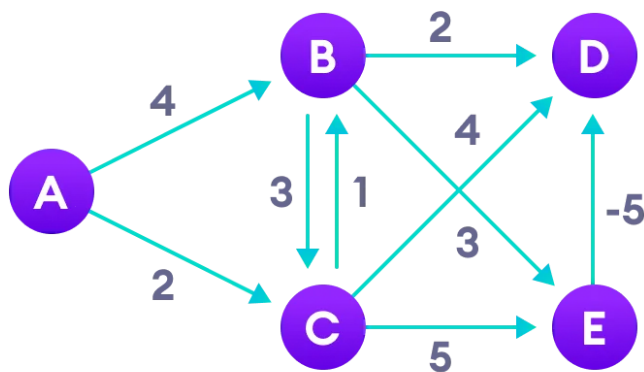
ADVERTISEMENTS



Procure tutoriais e exemplos

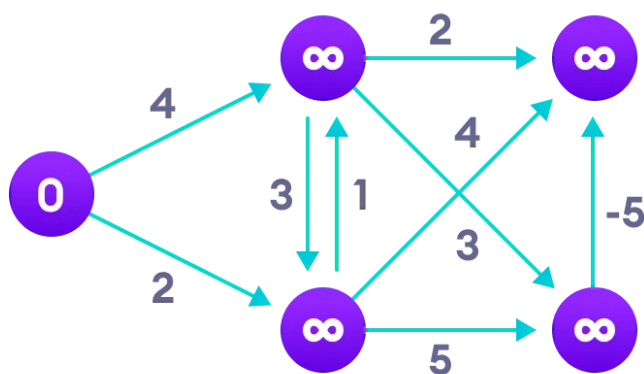
[www.domain-name.com](http://www.domain-name.com)

### Step 1: Start with the weighted graph



Etapa 1 para o algoritmo de Bellman Ford

### Step 2: Choose a starting vertex and assign infinity path values to all other vertices

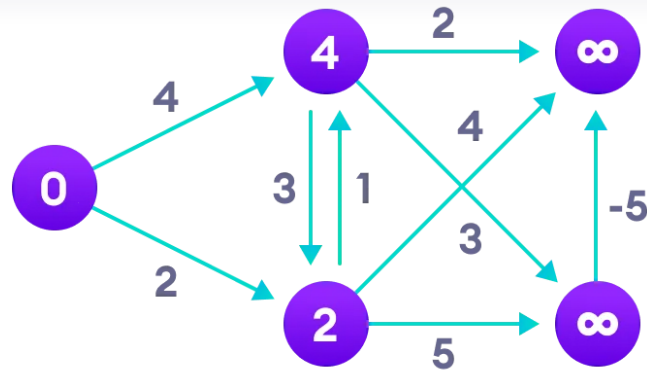


Etapa 2 para o algoritmo de Bellman Ford



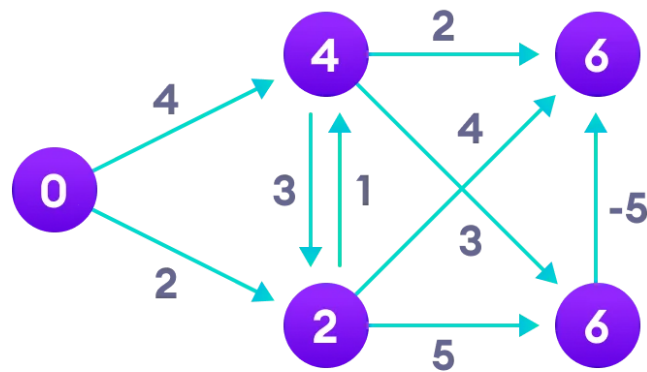
Procure tutoriais e exemplos

[www.domain-name.com](http://www.domain-name.com)



Etapa 3 para o algoritmo de Bellman Ford

**Step 4: We need to do this V times because in the worst case, a vertex's path length might need to be readjusted V times**

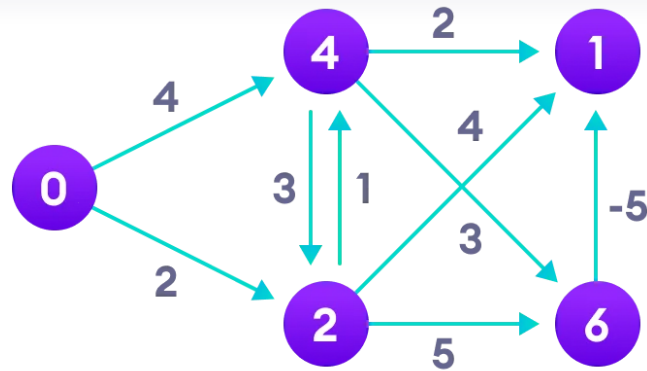


Etapa 4 para o algoritmo de Bellman Ford



Procure tutoriais e exemplos

[www.domain-name.com](http://www.domain-name.com)



Etapa 5 para o algoritmo de Bellman Ford

**Step 6: After all the vertices have their path lengths, we check if a negative cycle is present**

	B	C	D	E
0	$\infty$	$\infty$	$\infty$	$\infty$
0	4	2	$\infty$	$\infty$
0	3	2	6	6
0	3	2	1	6
0	3	2	1	6

Etapa 6 para o algoritmo de Bellman Ford

## Pseudocódigo Bellman Ford



Procure tutoriais e exemplos

[www.domain-name.com](http://www.domain-name.com)

extensão do caminho mais curto. Para isso, mapeamos cada vértice para o vértice que atualizou seu comprimento de caminho pela última vez.

Assim que o algoritmo terminar, podemos retroceder do vértice de destino ao vértice de origem para encontrar o caminho.

```
função bellmanFord (G, S)
  para cada vértice V em G
    distância [V] <- infinito
    anterior [V] <- NULL
  distância [S] <- 0

  para cada vértice V em G
    para cada aresta (U, V) em G
      tempDistance <- distância [U] + edge_weight (U, V)
      se tempDistance < distância [V]
        distância [V] <- tempDistance
        anterior [V] <- U

  para cada aresta (U, V) em G
    Se distância [U] + edge_weight (U, V) < distância [V]
      Erro: Existe Ciclo Negativo

  distância de retorno [], anterior []
```

## Bellman Ford vs Dijkstra

O algoritmo de Bellman Ford e o algoritmo de Dijkstra são muito semelhantes em estrutura. Enquanto Dijkstra olha apenas para os vizinhos imediatos de um vértice, Bellman passa por cada aresta em cada iteração.



Procure tutoriais e exemplos

[www.domain-name.com](http://www.domain-name.com)

<pre>for each edge (U,V) in G   tempDistance &lt;- distance[U] + edge_weight(U, V)   if tempDistance &lt; distance[V]     distance[V] &lt;- tempDistance     previous[V] &lt;- U  for each edge (U,V) in G   If distance[U] + edge_weight(U, V) &lt; distance[V]     Error: Negative Cycle Exists  return distance[], previous[]</pre>	<pre>for each unvisited neighbour V of U   tempDistance &lt;- distance[U] + edge_weight(U, V)   if tempDistance &lt; distance[V]     distance[V] &lt;- tempDistance     previous[V] &lt;- U  return distance[], previous[]</pre>
--	--

Algoritmo de Bellman Ford vs Algoritmo de Dijkstra

## Exemplos de Python, Java e C / C ++

Python

Java

C

C ++



Procure tutoriais e exemplos

[www.domain-name.com](http://www.domain-name.com)

```
// Struct for the edges of the graph
struct Edge {
    int u; //start vertex of the edge
    int v; //end vertex of the edge
    int w; //w of the edge (u,v)
};

// Graph - it consists of edges
struct Graph {
    int V; // Total number of vertices in the graph
    int E; // Total number of edges in the graph
    struct Edge* edge; // Array of edges
};

// Creates a graph with V vertices and E edges
struct Graph* createGraph(int V, int E) {
    struct Graph* graph = new Graph;
    graph->V = V; // Total Vertices
    graph->E = E; // Total edges

    // Array of edges for graph
    graph->edge = new Edge[E];
    return graph;
}
```

## Complexidade de Bellman Ford

### Complexidade de tempo

Melhor caso de complexidade	$O(E)$
Complexidade média do caso	$O(VE)$
Pior Caso Complexidade	$O(VE)$

### Complexidade do Espaço

E, a complexidade do espaço é  $O(V)$ .





Procure tutoriais e exemplos


[www.domain-name.com](http://www.domain-name.com)

Próximo tutorial:  
**Tipo de bolha** (</dsa/bubble-sort>)

Tutorial anterior:  
**Pesquisa abrangente** (</dsa/graph-bfs>)

Compartilhar no:

 (<https://www.facebook.com/sharer/sharer.php?u=https://www.programiz.com/dsa/bellman-ford-algorithm>)

 (<https://twitter.com/intent/tweet?text=Check%20this%20amazing%20ford-algorithm>)

Você achou este artigo útil?



ADVERTISEMENTS



Procure tutoriais e exemplos

[www.domain-name.com](http://www.domain-name.com)

[\(/dsa/graph\)](#)

[DS e Algoritmos](#)

**[Algoritmo de Prim](#)**

[\(/dsa/prim-algorithm\)](#)

[DS e Algoritmos](#)

**[Algoritmo de Kruskal](#)**

[\(/dsa/kruskal-algorithm\)](#)

[DS e Algoritmos](#)

**[Algoritmo de Dijkstra](#)**

[\(/dsa/dijkstra-algorithm\)](#)