DSA Shortest Path

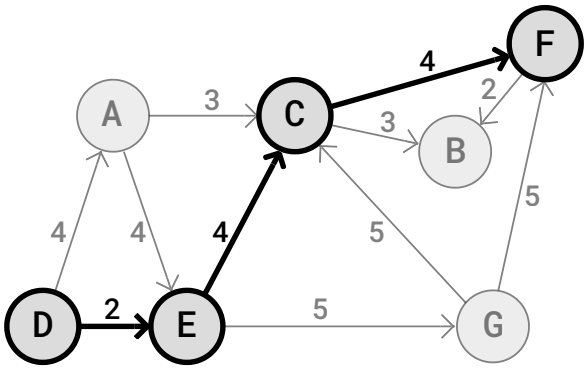
Previous Next

The Shortest Path Problem

The shortest path problem is famous in the field of computer science.

To solve the shortest path problem means to find the shortest possible route or path between two vertices (or nodes) in a Graph.

In the shortest path problem, a Graph can represent anything from a road network to a communication network, where the vertices can be intersections, cities, or routers, and the edges can be roads, flight paths, or data links.



Solutions to The Shortest Path Problem

Dijkstra's algorithm and the Bellman-Ford algorithm find the shortest path from one start vertex, to all other vertices.

To solve the shortest path problem means to check the edges inside the Graph until we find a path where we can move from one vertex to another using the lowest possible combined weight along the edges.

This sum of weights along the edges that make up a path is called a **path cost** or a **path weight**.

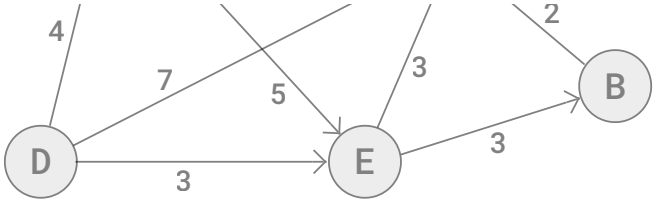
Algorithms that find the shortest paths, like Dijkstra's algorithm or the Bellman-Ford algorithm, find the shortest paths from one start vertex to all other vertices.

To begin with, the algorithms set the distance from the start vertex to all vertices to be infinitely long. And as the algorithms run, edges between the vertices are checked over and over, and shorter paths might be found many times until the shortest paths are found at the end.

Every time an edge is checked and it leads to a shorter distance to a vertex being found and updated, it is called a **relaxation**, or **relaxing** an edge.

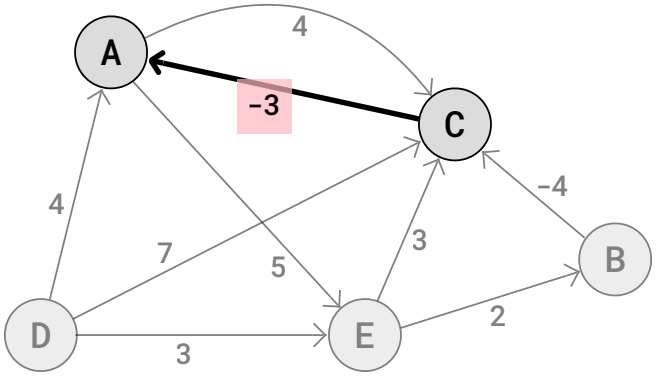
Positive and Negative Edge Weights

Some algorithms that find the shortest paths, like Dijkstra's algorithm, can only find the shortest paths in graphs where all the edges are positive. Such graphs with positive distances are also the easiest to understand because we can think of the edges between vertices as distances between locations.



If we interpret the edge weights as money lost by going from one vertex to another, a positive edge weight of 4 from vertex A to C in the graph above means that we must spend $4 to go from A to C.

But graphs can also have negative edges, and for such graphs the Bellman-Ford algorithm can be used to find the shortest paths.



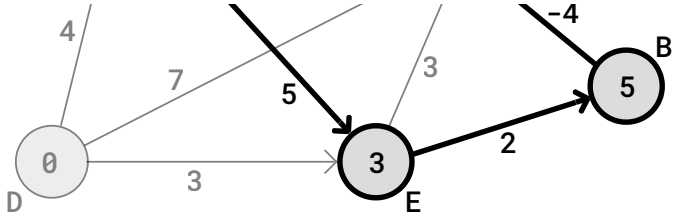
And similarly, if the edge weights represent money lost, the negative edge weight -3 from vertex C to A in the graph above can be understood as an edge where there is more money to be made than money lost by going from C to A. So if for example the cost of fuel is $5 going from C to A, and we get paid $8 for picking up packages in C and delivering them in A, money lost is -3, meaning we are actually earning $3 in total.

Negative Cycles in Shortest Path Problems

Finding the shortest paths becomes impossible if a graph has negative cycles.

Having a negative cycle means that there is a path where you can go in circles, and the edges that make up this circle have a total path weight that is negative.

In the graph below, the path A->E->B->C->A is a negative cycle because the total path weight is 5+2-4-4=-1.



The reason why it is impossible to find the shortest paths in a graph with negative cycles is that it will always be possible to continue running an algorithm to find even shorter paths.

Let's say for example that we are looking for the shortest distance from vertex D in graph above, to all other vertices. At first we find the distance from D to E to be 3, by just walking the edge D->E. But after this, if we walk one round in the negative cycle E->B->C->A->E, then the distance to E becomes 2. After walking one more round the distance becomes 1, which is even shorter, and so on. We can always walk one more round in the negative cycle to find a shorter distance to E, which means the shortest distance can never be found.

Luckily, the the Bellman-Ford algorithm, that runs on graphs with negative edges, can be implemented with detection for negative cycles.

‹ Previous Sign in to track progress Next ›

COLOR PICKER

PHP Tutorial
Java Tutorial
C++ Tutorial
jQuery Tutorial

Top References

HTML Reference
CSS Reference
JavaScript Reference
SQL Reference
Python Reference
W3.CSS Reference
Bootstrap Reference
PHP Reference
HTML Colors
Java Reference
AngularJS Reference
jQuery Reference

Top Examples

HTML Examples
CSS Examples
JavaScript Examples
How To Examples
SQL Examples
Python Examples
W3.CSS Examples
Bootstrap Examples
PHP Examples
Java Examples
XML Examples
jQuery Examples

Get Certified

HTML Certificate
CSS Certificate
JavaScript Certificate
Front End Certificate
SQL Certificate
Python Certificate
PHP Certificate
jQuery Certificate
Java Certificate
C++ Certificate
C# Certificate
XML Certificate