

**Algoritme dhe Struktura të të Dhënave**

**"Shortest path"**

**Studente**: Prof.:

Edonë Haziri Edmond Jajaga

Ariola Cerkini

# Përmbajtja

[Përmbajtja 2](#_Toc9880)

[Cfarë janë “Shortest Path” Algoritmet ? 3](#_Toc9881)

[Llojet e “Shortest Path” Algoritmeve 3](#_Toc9881)

[A. Single-source 3](#_Toc9883)

1. [Algoritmi i Dijkstra's 4](#_Toc9884)

[b) Algoritmi i Bellman-Ford 5](#_Toc9885)

[B. All-pairs 5](#_Toc9886)

[a) Algoritmi i Floyd-Warshall 5](#_Toc9887)

[b) Algoritmi i Johnson 5](#_Toc9888)

[c) Algoritmi Topological Sort 6](#_Toc9889)

[Real world applications 6](#_Toc9890)

[A. Digital Mapping Services in Google Maps 6](#_Toc9891)

[B. Social networking applications 7](#_Toc9892)

[C. IP routing to find Open shortest Path First 7](#_Toc9893)

[D. Flight agenda 7](#_Toc9894)

[Python Implementation 8](#_Toc9895)

# What are Shortest Path Algorithms

A set of algorithms known as shortest path algorithms is created to address the shortest path issue. Most people have some intuitive knowledge of the shortest path issue, which asks: given two points A and B, what is the shortest path between them? However, the shortest path problem can take various forms in computer science, necessitating the use of various methods to be able to solve them all.

Shortest path algorithms often operate on an input graph, G, for simplicity and generality. There are a number of vertices (V) and edges (E) connecting them in this graph. The graph is referred to as a weighted graph if the edges have weights. The graph is referred to as undirected when these edges are bidirectional. Cycles may occasionally even appear in the graph. For a certain sort of graph, each of these minute variations enables one method to perform better than the other. Below is a graph illustration.

Shortest path algorithms come in a variety of forms. The shortest route between points A and B may not always be the same as the shortest route between A and every other point in the graph.

A picture containing text, pool ball, vector graphics

Description automatically generated

Numerous applications exist for shortest path algorithms. As was already mentioned, shortest path algorithms are used in mapping programs like Google or Apple maps. They are crucial for study on the road network, operations, and logistics. For computer networks like the Internet, shortest path algorithms are also crucial.

# Types of Shortest Path Algorithms

Single-source and all-pairs are the two primary categories of shortest path algorithms. There are algorithms for each type that work well in a particular way. All-pairs methods require more processing time due to their increased complexity. Even if the return values differ in type or format from algorithm to algorithm, all shortest path algorithms produce results that can be utilized to determine the shortest path.

## A. Single-source Shortest Path Algorithms

Given a graph *G*, with vertices *V*, edges *E* with weight function w(u,v)=wu,and a single source vertex, *s*, return the shortest paths from s*s* to all other vertices in *V*.

The process can simply be terminated once the shortest path between the two specified vertices, s and t, has been identified if that is the algorithm's only task. All algorithms that find the shortest path between two given vertices have the same worst-case asymptotic complexity as single-source shortest path algorithms since there is no mechanism to choose which vertices to "complete" first.

The single-destination shortest path problem can also be solved using this approach. The singledestination problem can be converted into the single-source problem by flipping every edge in a graph. This algorithm will therefore discover the shortest pathways beginning at all other vertices and terminating at the provided destination vertex - t.

### a) Dijkstra's Shortest Path Algorithm

You may determine the shortest path between two nodes in a graph using Dijkstra's Algorithm. In particular, you may create a shortest-path tree by determining the shortest path from a node (referred to as the "source node") to every other node in the graph. To determine the shortest route between the present location and the destination, GPS devices employ this method. It has several industrial uses, particularly in fields where modeling networks is necessary.

Brilliant Dutch computer scientist and software engineer Dr. Edsger W. Dijkstra developed and published this approach. His novel technique was described in a three-page piece he wrote in 1959 titled "A comment on two difficulties in connexon with graphs". In essence, Dijkstra's Algorithm begins at the node you select (the source node) and examines the graph to determine the shortest path from that node to every other node in the graph. The algorithm keeps track of the shortest paths that are currently known to exist between each node and the source node, and it updates these values when a shorter path is discovered. The other node is added to the path and designated as "visited" once the algorithm has determined the shortest path between the source node and that other node. Up until every node in the graph has been added to the path, the process is repeated. In this manner, a path that connects the source node to every other node and takes the shortest route to each node is created.

Chart, radar chart

Description automatically generated

*Click for moving image:*

[*https://www3.cs.stonybrook.edu/~skiena/combinatorica/animations/anim/dijkstra.gif*](https://www3.cs.stonybrook.edu/~skiena/combinatorica/animations/anim/dijkstra.gif)

### b) Bellman-Ford Shortest Path Algorithm

The Bellman-Ford algorithm uses a graph with V vertices and E edges called G as its input. Given that the Bellman-Ford algorithm is a single-source shortest path algorithm, s must be provided as well. Bellman-Ford, however, determines the shortest path from the source vertex to all vertices in the graph, hence no destination vertex is required.

Similar to Dijkstra's algorithm, the Bellman-Ford algorithm finds increasing amounts of path length accuracy by applying the relaxation principle. However, Bellman-Ford addresses two key issues with this procedure:

· In the event that there are negative weight cycles, finding the shortest route will be fruitless.

· Exponential relaxations result from selecting the incorrect ordering for relaxations.

Although the detection of negative cycles is significant, the algorithm's greatest value is in the relaxations it orders. The nearest unprocessed vertex is chosen by Dijkstra's algorithm, which is greedy. On the other hand, Bellman-Ford softens every edge. For a graph G, Bellman-Ford names its edges as e1,e2,...,e*n*, and that set of edges is relaxed exactly V−1 times, where V is the number of vertices in the graph.

## B. All-pairs Shortest Path Algorithms

Given a graph *G*, with vertices *V*, edges *E* with weight function w(u,v)=wu,return the shortest path from u*u* to *v for all (u,v) in V.*

The floyd-warshall algorithm is the most popular solution to the all-pairs problem. The output of this procedure is a matrix of values M, each cell of which represents the shortest path from vertex *i* to vertex *j. It is feasible to rebuild the path that was really followed to arrive at the shortest path, although this is not a feature of the core algorithm.*

### a) Floyd-Warshall Shortest Path Algorithm

The all-pairs shortest path problem is resolved by the Floyd-Warshall method. To do this, it employs a dynamic programming strategy. For Floyd-Warshall, negative edge weight might be present.

Floyd-Warshall makes use of the observation that the shortest route from point A to point C is either the shortest route from point A to point B plus the shortest route from point B to point C, or it is the shortest route from point A to point C that has previously been discovered. This may seem unimportant, but it is the key to Floyd-ability Warshall's to use dynamic programming to generate longer shortest paths from shorter ones.

### b) Johnson's Shortest Path Algorithm

Johnson's approach performs best for sparse graphs, whereas Floyd-Warshall performs better for dense graphs (defined as having many edges) (meaning few edges). Johnson's technique outperforms FloydWarshall in terms of asymptotic running time on sparse graphs.

Johnson's technique makes use of the idea of reweighting and, after reweighting the edges, performs Dijkstra's algorithm on numerous vertices to determine the shortest path.

### c) Topological Sort Shortest Path Algorithm

Finding the shortest paths in directed acyclic graphs (DAGs) is a highly helpful tool. The shortest path problem can be solved in linear time by topologically sorting the graph's vertices.

For each edge (u,v) in E, the vertices are arranged in a topological sort such that u comes before v. Because there can never be a negative weight cycle in a DAG, even if there are negative weight edges, shortest paths are always clearly characterized.

# Real world applications

Finding the shortest path through a network from a starting point to an ending point is one of the most important tasks in the highly developed information systems of today. Computing, geography, management, transportation, and other scientific disciplines and professions all use the concepts of developing, testing, and successfully implementing algorithms for finding the shortest paths in networks. The right algorithm will be chosen based on the network's given parameters and the algorithm's execution time when it comes to choosing one that attempts to solve the problem of the shortest path within the network.

## A. Digital Mapping Services in Google Maps

People have frequently attempted to calculate the distance in G-Maps between cities or between your location and the closest desired location. The Shortest Path Algorithm is used because there are numerous routes and paths that connect them, but it must display the shortest distance. Dijkstra's Algorithm is used to determine the shortest distance between any two points along the path. The shortest routes between any two cities or places in India, as well as from one city or place to another, can be determined using this algorithm. Imagine India as a graph, with each city or place represented as a vertex, and the route between two cities or places as an edge.

|  |
| --- |
| function initMap() {  var directionsService = new google.maps.DirectionsService(); var directionsRenderer = new google.maps.DirectionsRenderer(); var chicago = new google.maps.LatLng(41.850033, -87.6500523); var mapOptions = { zoom:7,  center: chicago  }  var map = new google.maps.Map(document.getElementById('map'), mapOptions); directionsRenderer.setMap(map);  }    function calcRoute() {  var start = document.getElementById('start').value; var end = document.getElementById('end').value; var request = { |
| origin: start, destination: end, travelMode: 'DRIVING'  };  directionsService.route(request, function(result, status) { if (status == 'OK') { directionsRenderer.setDirections(result);  }  });  } |

*An example taken from Google’s map documentation*

*To view this example:*

[*https://developers.google.com/maps/documentation/javascript/examples/directions-simple*](https://developers.google.com/maps/documentation/javascript/examples/directions-simple)

*To view the implementation and more examples:* [*https://developers.google.com/maps/documentation/javascript/directions*](https://developers.google.com/maps/documentation/javascript/directions)

## B. Social networking applications

You may have noticed that many of these applications offer a list of friends that a specific user might know. Given that the system has more than a billion users, how effectively do you think many social media companies will implement this feature? The shortest path between users, as determined by handshakes or connections between them, can be used to apply the standard Dijkstra algorithm. The standard Dijkstra's algorithm is used in conjunction with other features to find the shortest paths when the social networking graph is small. However, as the graph grows larger and larger, the standard algorithm becomes slower and slower, necessitating the use of alternative advanced algorithms.

## C. IP routing to find Open shortest Path First

Bellman-algorithm Ford's is typically only applied when the nodes' edge weights are negative. Many protocols are used on the Web that employ this algorithm. The fundamental network routing information protocol is the best illustration. One of the earliest Internet protocols, it prevents loops by capping the number of hops each packet can make before reaching its destination. The interior gateway routing protocol is a further illustration. Within a system, machines can exchange routing information with the aid of this secure protocol.

## D. Flight agenda

As previously stated in our ‘War stories’ presentation, the shortest path algorithm is also used in finding the cheapest and shortest flights. For instance, suppose someone needs software to create a schedule of flights for clients. A database with all airports and flights is available to the agent. The flights also include the departure and arrival times in addition to the flight number, origin airport, and destination. The agent wants to know the earliest arrival time at the destination given the airport of origin and the start time. This algorithm is put to use there.

# Python Implementation

|  |
| --- |
| class Graph():    def \_\_init\_\_(self, vertices):  self.V = vertices  self.graph = [[0 for column in range(vertices)] for row in range(vertices)]    def printSolution(self, dist):  print("Vertex \t Distance from Source") for node in range(self.V):  print(node, "\t\t", dist[node])    # A utility function to find the vertex with  # minimum distance value, from the set of vertices  # not yet included in shortest path tree def minDistance(self, dist, sptSet):    # Initialize minimum distance for next node min = 1e7    # Search not nearest vertex not in the  # shortest path tree for v in range(self.V): if dist[v] < min and sptSet[v] == False:  min = dist[v]  min\_index = v    return min\_index    # Function that implements Dijkstra's single source  # shortest path algorithm for a graph represented  # using adjacency matrix representation def dijkstra(self, src):    dist = [1e7] \* self.V  dist[src] = 0  sptSet = [False] \* self.V    for cout in range(self.V):    # Pick the minimum distance vertex from # the set of vertices not yet processed.  # u is always equal to src in first iteration |
| u = self.minDistance(dist, sptSet)    # Put the minimum distance vertex in the  # shortest path tree  sptSet[u] = True    # Update dist value of the adjacent vertices  # of the picked vertex only if the current  # distance is greater than new distance and # the vertex in not in the shortest path tree for v in range(self.V):  if (self.graph[u][v] > 0 and sptSet[v] == False and dist[v] > dist[u] + self.graph[u][v]): dist[v] = dist[u] + self.graph[u][v]    self.printSolution(dist)    # Driver program g = Graph(9)  g.graph = [[0, 4, 0, 0, 0, 0, 0, 8, 0],  [4, 0, 8, 0, 0, 0, 0, 11, 0],  [0, 8, 0, 7, 0, 4, 0, 0, 2],  [0, 0, 7, 0, 9, 14, 0, 0, 0],  [0, 0, 0, 9, 0, 10, 0, 0, 0],  [0, 0, 4, 14, 10, 0, 2, 0, 0],  [0, 0, 0, 0, 0, 2, 0, 1, 6],  [8, 11, 0, 0, 0, 0, 1, 0, 7],  [0, 0, 2, 0, 0, 0, 6, 7, 0]  ]    g.dijkstra(0) |