Dijkstra's Algorithm to find Shortest Paths from a Source to all

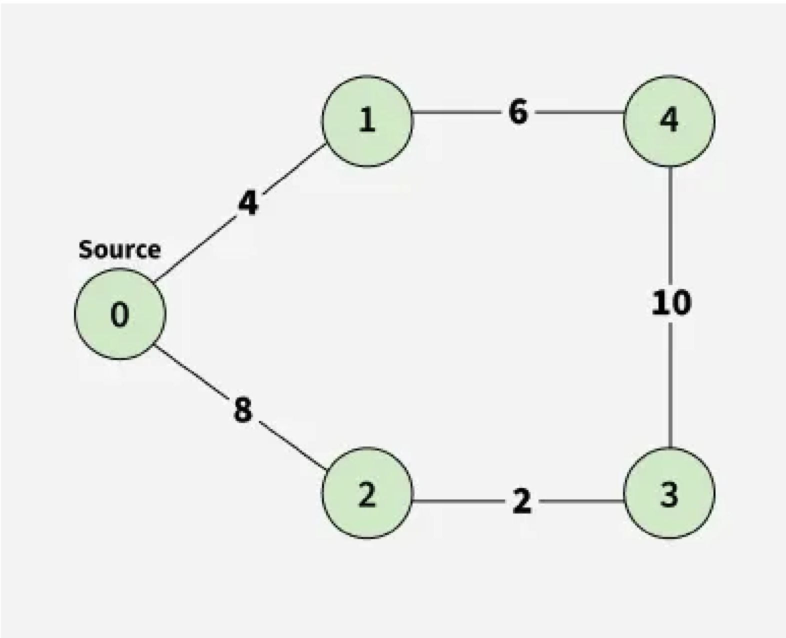
Last Updated : 23 Jul, 2025

Given a weighted undirected graph represented as an **edge** list and a source vertex `src`, find the shortest path distances from the source vertex to all other vertices in the graph. The graph contains **V** vertices, numbered from `0` to `V - 1`.

Note: The given graph does not contain any negative edge.

Examples:

> ***Input:*** *src = 0, V = 5, edges[][] = [[0, 1, 4], [0, 2, 8], [1, 4, 6], [2, 3, 2], [3, 4, 10]]*



Graph with 5 node

> ***Output:*** *0 4 8 10 10*
> ***Explanation:*** *Shortest Paths:*
> *0 to 1 = 4. 0 → 1*
> *0 to 2 = 8. 0 → 2*
> *0 to 3 = 10. 0 → 2 → 3*
> *0 to 4 = 10. 0 → 1 → 4*

Dijkstra's Algorithm using Min Heap - O(E*logV) Time and O(V) Space

> *In Dijkstra's Algorithm, the goal is to find the shortest distance from a given source node to all other nodes in the graph. As the source node is the starting point, its distance is initialized to zero. From there, we iteratively pick the unprocessed node with the minimum distance from the source, this is where a min-heap (priority queue) or a set is typically used for efficiency. For each picked node u, we update the distance to its neighbors v using the formula:* ***dist[v] = dist[u] + weight[u][v],*** *but only if this new path offers a shorter distance than the current known one. This process continues until all nodes have been processed.*

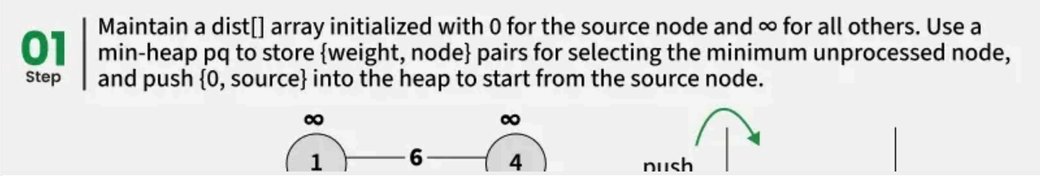
Step-by-Step Implementation

1. Set dist[source]=0 and all other distances as infinity.
2. Push the source node into the min heap as a pair <distance, node> → i.e., <0, source>.
3. Pop the top element (node with the smallest distance) from the min heap.
    1. For each adjacent neighbor of the current node:
    2. Calculate the distance using the formula:

       dist[v] = dist[u] + weight[u][v]

       If this new distance is shorter than the current dist[v], update it.
       Push the updated pair <dist[v], v> into the min heap
4. **Repeat** step 3 until the min heap is empty.
5. Return the distance array, which holds the shortest distance from the source to all nodes.

Illustration:



1 / 7

C++ Java Python C# JavaScript

```python
# Returns shortest distances from src to all other vertices
def dijkstra(V, edges, src):
    # Create adjacency list
    adj = constructAdj(edges, V)

    # Create a priority queue to store vertices that
    # are being preprocessed.
    pq = []

    # Create a list for distances and initialize all
    # distances as infinite
    dist = [sys.maxsize] * V

    # Insert source itself in priority queue and initialize
    # its distance as 0.
    heapq.heappush(pq, [0, src])
    dist[src] = 0

    # Looping till priority queue becomes empty (or all
    # distances are not finalized)
    while pq:
        # The first vertex in pair is the minimum distance
        # vertex, extract it from priority queue.
        u = heapq.heappop(pq)[1]

        # Get all adjacent of u.
        for x in adj[u]:
            # Get vertex label and weight of current
            # adjacent of u.
            v, weight = x[0], x[1]

            # If there is shorter path to v through u.
            if dist[v] > dist[u] + weight:
                # Updating distance of v
```

```
                dist[v] = dist[u] + weight
                heapq.heappush(pq, [dist[v], v])

    # Return the shortest distance array
    return dist
```

Output

```
0 4 8 10 10
```

Time Complexity: O(E*logV), Where E is the number of edges and V is the number of vertices.
Auxiliary Space: O(V), Where V is the number of vertices, We do not count the adjacency list in auxiliary space as it is necessary for representing the input graph.

Problems based on Shortest Path

- Shortest Path in Directed Acyclic Graph
- Shortest path with one curved edge in an undirected Graph
- Minimum Cost Path
- Path with smallest difference between consecutive cells
- Print negative weight cycle in a Directed Graph
- 1st to Kth shortest path lengths in given Graph
- Shortest path in a Binary Maze
- Minimum steps to reach target by a Knight
- Number of ways to reach at destination in shortest time
- Snake and Ladder Problem
- Word Ladder

Dijkstra's Shortest Path Algorithm

Visit Course

Comment

More info

Campus Training Program

Next Article

Analysis of Algorithms

Corporate & Communications Address:
A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305)

Registered Address:
K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305

GET IT ON Google Play
Download on the App Store

Advertise with us

Company
About Us
Legal
Privacy Policy
Careers
In Media
Contact Us
Corporate Solution
Campus Training Program

Explore
Job-A-Thon
Offline Classroom Program
DSA in JAVA/C++
Master System Design
Master CP
Videos

Tutorials
Python
Java
C++
PHP
GoLang
SQL
R Language
Android

DSA
DSA Tutorial
Problem Of The Day
GfG 160
DSA 360
DSA Roadmap
DSA Interview Questions
Competitive Programming

Data Science & ML
Data Science With Python
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies
HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
NodeJs
Bootstrap
Tailwind CSS

Python Tutorial

Python Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar

Databases

SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB

Preparation Corner

Company-Wise Recruitment Process
Aptitude Preparation
Puzzles
Company-Wise Preparation

More Tutorials

Software Development
Software Testing
Product Management
Project Management
Linux
Excel
All Cheat Sheets

Courses

IBM Certification Courses
DSA and Placements
Web Development
Data Science
Programming Languages
DevOps & Cloud

Programming Languages

C Programming with Data Structures
C++ Programming Course
Java Programming Course
Python Full Course

Clouds/Devops

DevOps Engineering
AWS Solutions Architect Certification
Salesforce Certified Administrator Course

GATE 2026

GATE CS Rank Booster
GATE DA Rank Booster
GATE CS & IT Course - 2026
GATE DA Course 2026

GATE Rank Predictor

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved