

# Programming Challenge: Implementing Dijkstra's

In this lesson, you'll be implementing Dijkstra's Shortest Path Algorithm.

## WE'LL COVER THE FOLLOWING ^

- Problem Statement
  - Input
  - Output
  - Sample Input
  - Sample Output
- Coding Exercise

## Problem Statement #

Given an adjacency matrix in a 2D array, solve the **Single Source Shortest Path** algorithm, essentially by implementing the Dijkstra's algorithm discussed in the previous lesson. We've written some skeleton code for the function.

1. The value of the weight of the link is `graph[src][dst]`.
2. The graph is undirected so `graph[src][dst]==graph[dst][src]`.
3. A link between the `src` and `dst` exists if `-1<graph[src][dst]<16`.
4. If `graph[src][dst]>=16` the weight of the link is infinite and it does not function.

## Input #

1. An adjacency matrix, i.e., a 2D array, a source node, and a destination node.

## Output #

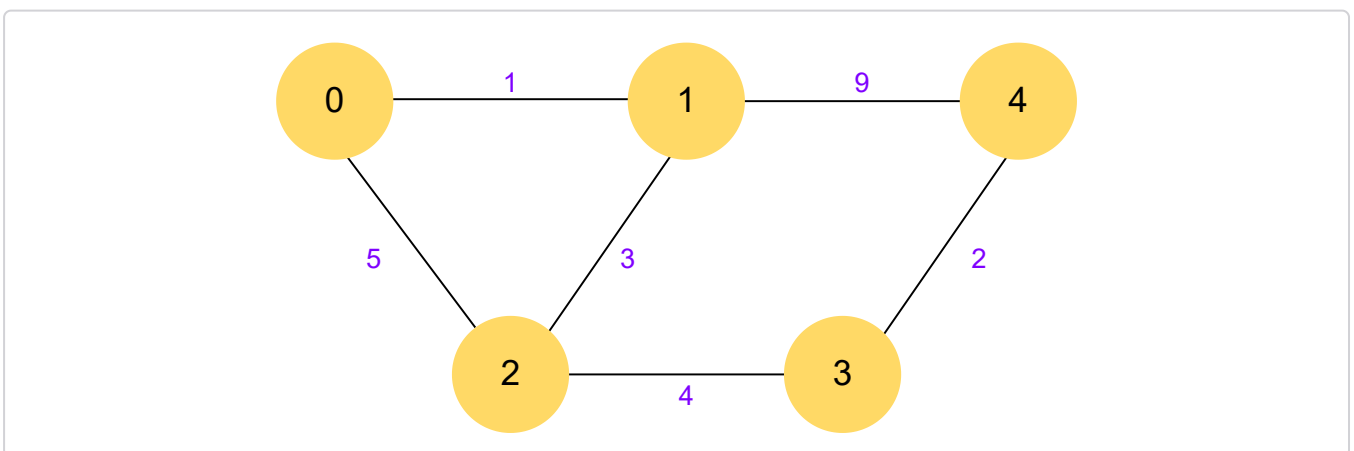
The **shortest path** between the source and destination in the form of an array

of integers where each integer represents a node and the **total weight** of the path.

## Sample Input #

```
1. graph = [  
    [0,1,5,-1,-1],  
    [1,0,3,-1,9],  
    [5,3,0,4,-1],  
    [-1,-1,4,0,2],  
    [-1,9,-1,2,0]  
]
```

This adjacency matrix represents the following graph:



2. A source and destination:

```
src = 0  
dst = 3
```

## Sample Output #

```
shortest_path = [0,1,2,3]  
cost = 8
```

## Coding Exercise #

Try it yourself below!

```
def Dijkstra(graph, src, dst):  
    pass
```



---

In the next lesson, we'll look at a solution to this problem.