

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY**

**Individual Work № 8:**

**Topic: Dijkstra’s algorithm with examples in Python**

**Subject: Obyektyönümlü proqramlaşdırma**

**Teacher: Mehemmed Shahmaliyev**

**Group: 1459i Student: Hesenova Firuze**

**Date: Signature:H.F**

**Baku 2022**

**Python Program for Dijkstra’s shortest path algorithm**

Given a graph and a source vertex in the graph, find the shortest paths from source to all vertices in the given graph.  
Dijkstra’s algorithm is very similar to [Prim’s algorithm for minimum spanning tree](https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/). Like Prim’s MST, we generate an SPT (shortest path tree) with a given source as root. We maintain two sets, one set contains vertices included in the shortest-path tree, another set includes vertices not yet included in the shortest-path tree. At every step of the algorithm, we find a vertex that is in the other set (set of not yet included) and has a minimum distance from the source.  
Below are the detailed steps used in Dijkstra’s algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.   
Algorithm   
1) Create a set sptSet (shortest path tree set) that keeps track of vertices included in shortest path tree, i.e., whose minimum distance from source is calculated and finalized. Initially, this set is empty.   
2) Assign a distance value to all vertices in the input graph. Initialize all distance values as INFINITE. Assign distance value as 0 for the source vertex so that it is picked first.   
3) While sptSet doesn’t include all vertices:

Pick a vertex u which is not there in sptSet and has minimum distance value.

Include u to sptSet.

Update distance value of all adjacent vertices of u. To update the distance values, iterate through all adjacent vertices. For every adjacent vertex v, if the sum of a distance value of u (from source) and weight of edge u-v, is less than the distance value of v, then update the distance value of v.

*# Python program for Dijkstra's single*

*# source shortest path algorithm. The program is*

*# for adjacency matrix representation of the graph*

*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)*

*# This code is contributed by Divyanshu Mehta*

Output

Vertex Distance from Source

0 0

1 4

2 12

3 19

4 21

5 11

6 9

7 8

8 14