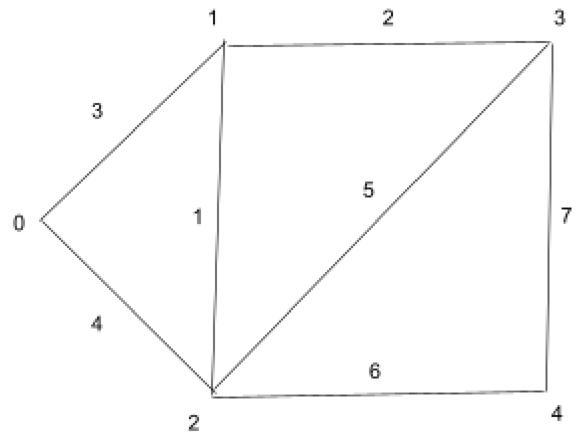
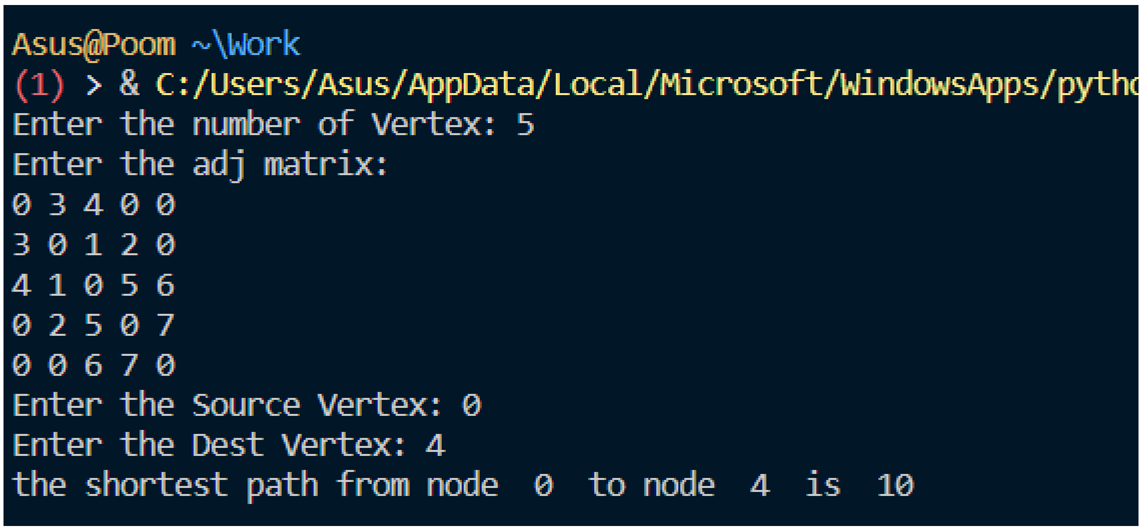
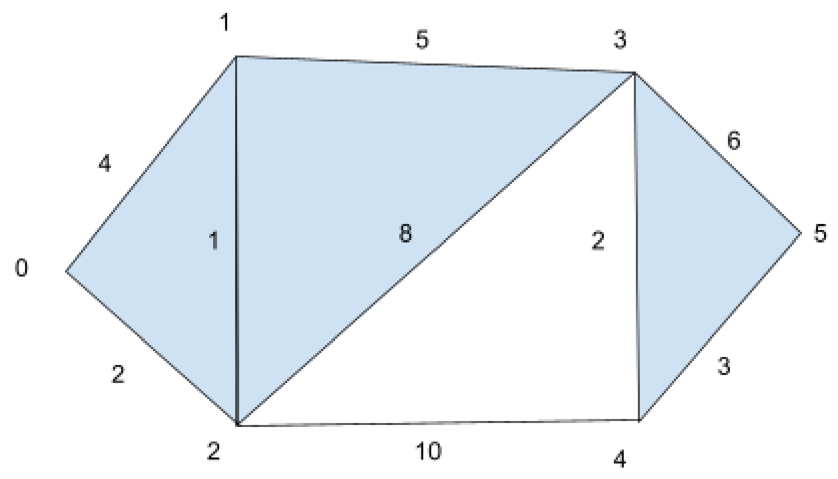


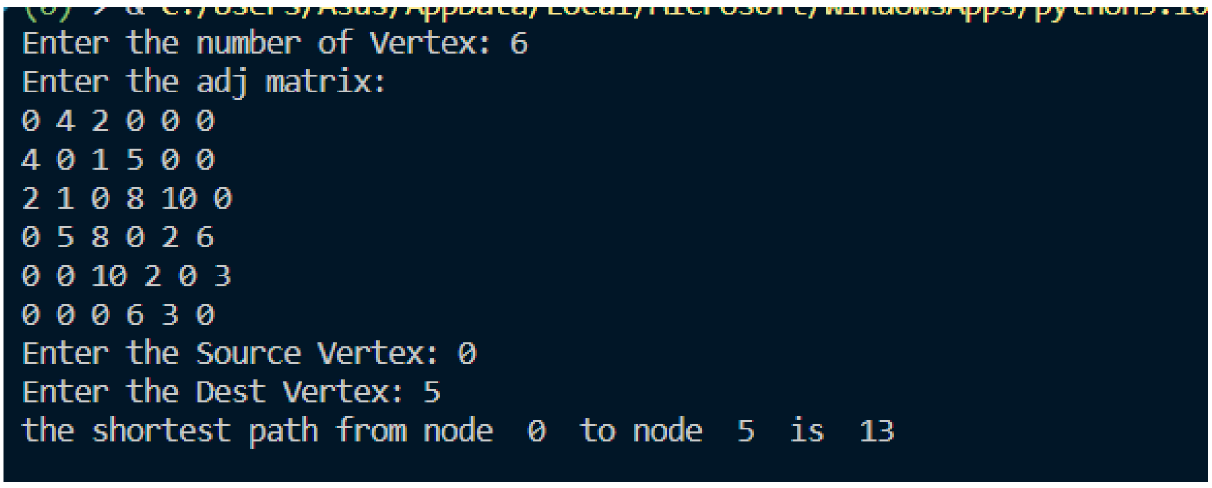
5
0 3 4 0 0
3 0 1 2 0
4 1 0 5 6
0 2 5 0 7
0 0 6 7 0



```
Asus@Poom ~\Work
(1) > & C:/Users/Asus/AppData/Local/Microsoft/WindowsApps/pytho
Enter the number of Vertex: 5
Enter the adj matrix:
0 3 4 0 0
3 0 1 2 0
4 1 0 5 6
0 2 5 0 7
0 0 6 7 0
Enter the Source Vertex: 0
Enter the Dest Vertex: 4
the shortest path from node  0  to node  4  is  10
```



6
0 4 2 0 0 0
4 0 1 5 0 0
2 1 0 8 10 0
0 5 8 0 2 6
0 0 10 2 0 3
0 0 0 6 3 0



```
Enter the number of Vertex: 6
Enter the adj matrix:
0 4 2 0 0 0
4 0 1 5 0 0
2 1 0 8 10 0
0 5 8 0 2 6
0 0 10 2 0 3
0 0 0 6 3 0
Enter the Source Vertex: 0
Enter the Dest Vertex: 5
the shortest path from node  0  to node  5  is  13
```

Code

```
import sys
# define infinity
INF = sys.maxsize
# input graph with the number of Vertex, adj_matrix, Source Vertex and Dest Vertex
n=int(input("Enter the number of Vertex: "))
adj_matrix = [[int(i) for i in input().split()] for k in range(n)]
source=int(input("Enter the Source Vertex: "))
dest=int(input("Enter the Dest Vertex: "))

def dijkstra(adj_matrix, source, dest):

  # initialize distances and visited arrays
  distances = [INF for i in range(len(adj_matrix))]
  visited = [False for i in range(len(adj_matrix))]

  # set distance of source to 0
  distances[source] = 0

  # while there are unvisited nodes
  while False in visited:

    # find the unvisited node with the smallest distance
    min_distance = INF
    min_index = None
    for i in range(len(adj_matrix)):
      if not visited[i] and distances[i] < min_distance:
        min_distance = distances[i]
        min_index = i
    if min_index is None:
      break

    # update distances of neighbors
    for i in range(len(adj_matrix)):
      if adj_matrix[min_index][i] != 0 and distances[i] > distances[min_index] +
adj_matrix[min_index][i]:
        distances[i] = distances[min_index] + adj_matrix[min_index][i]

    # mark the current node as visited

    visited[min_index] = True

    # if destination has been visited, break
    if visited[dest]:
      break
  return distances[dest]

# find the shortest distance from node s to node d
distance = dijkstra(adj_matrix, source, dest)

# print the shortest distance from node s to node d
print(distance)
```