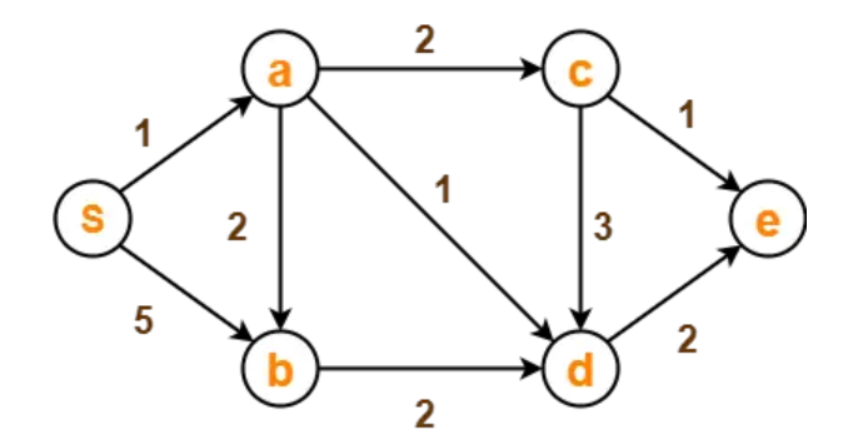
**22AIE203 – DATA STRUCTURES & ALGORITHMS 2**

**ASSIGNMENT**

Name : Anuvind MP

Roll no.: AM.EN.U4AIE22010

----------------------------------------------------------------------------------------------------------



Using Dijkstra’s Algorithm, find the shortest distance from source vertex ‘S’ to the remaining vertices in the given graph.

**Code :**

matrix = {

'A':{'B':2, 'C':2, 'D':1},

'B':{'D':2},

'C':{'E':1, 'D':3},

'D':{'E':2},

'E':{},

'S':{'A':1, 'B':5},

}

class Graph:

     def \_\_init\_\_(self, adj\_matrix):

          self.adj\_matrix = adj\_matrix

     def Edge(self, u, v):

          if v in self.adj\_matrix[u]:

               return self.adj\_matrix[u][v]

          return None

     def childNode(self, s):

          return self.adj\_matrix[s]

     def vertices(self):

          return list(self.adj\_matrix.keys())

graph = Graph(matrix)

def Dijkstra(Graph, source):

     min\_dist = {source:0}

     dist = {}

     for vertex in Graph.vertices():

          dist[vertex] = float('inf')

     dist.pop(source)

     Node=source

     while dist!={}:

          for vertex in Graph.childNode(Node):

               if vertex in min\_dist:

                    continue

               if min\_dist[Node] + Graph.Edge(Node, vertex) < dist[vertex]:

                    dist[vertex] = min\_dist[Node] + Graph.Edge(Node, vertex)

          Node = min(dist, key= lambda k: dist[k])

          min\_dist[Node] = dist.pop(Node)

     return min\_dist

print(Dijkstra(graph, 'S'))

**Output :**

****