CODE:

```
G = []
P = []
const = 10
#-------------------------------------------------------------------------#
def InitOpen(Open):
   for i in range(const):
       Open.append([])
       for j in range(2):
           Open[i].append(0)
#-------------------------------------------------------------------------#
def EmptyOpen(Open):
     return len(Open) - Open.count(Open[0]) == 0
#-------------------------------------------------------------------------#
def AddQ(Open,n,value,index):
    n = n + 1
    Open[n][0] = value
    Open[n][1] = index
    i = n
    while i > 1:
        j = int(i / 2)
        if Open[i][0] < Open[j][0]:
            temp = Open[i]
            Open[i] = Open[j]
            Open[j] = temp
        i = j
    return n
#-------------------------------------------------------------------------#
def RemoveQ(Open):
    value = Open[1][0]
    index = Open[1][1]
    n = len(Open) - Open.count(Open[0])
```

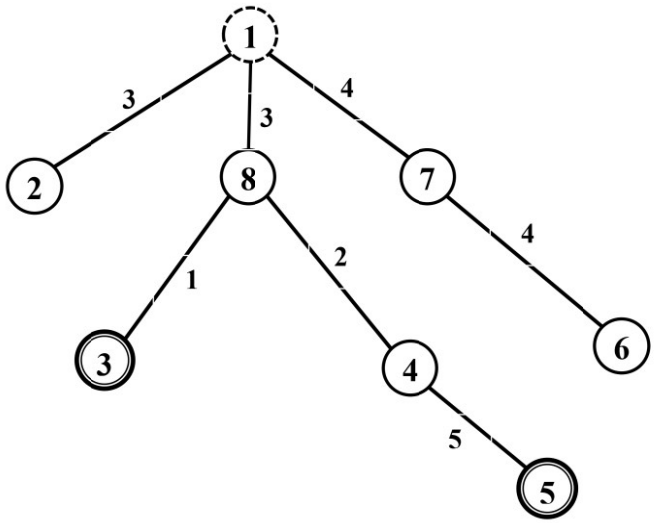
```
    Open[1][0] = Open[n][0]
    Open[1][1] = Open[n][1]
    Open[n][0] = 0
    Open[n][1] = 0
    n = n - 1
    i = 1
    while i <= int(n / 2):
        j = i * 2;
        if j < n:
            if Open[j][0] > Open[j + 1][0]:
                j = j + 1
                if Open[i][0] > Open[j][0]:
                    Open[i], Open[j] = Open[j], Open[i]
        else:
            if Open[i][0] > Open[j][0]:
                    Open[i], Open[j] = Open[j], Open[i]
        i = i + 1
    return value, index, n
#-------------------------------------------------------------------------#
def Split(string):
    k = string.index(' ')
    str = string[0:k]
    a = int(str,base = 10)
    m = string.index(' ',k + 1,-1)
    str = string[k + 1:m]
    b = int(str,base = 10)
    str = string[m + 1:len(string)]
    c = int(str,base = 10)
    return a, b, c
#-------------------------------------------------------------------------#
def Init(path,G):
    f = open(path)
    string = f.readline()
```

```
    string = string.replace('\t',' ')
    n, a, z = Split(string)
    for i in range(n + 1):
        G.append([])
        for j in range(n + 1):
            G[i].append(0)
    while True:
        string = f.readline()
        if not string:
            break
        string = string.replace('\t',' ')
        i, j, x = Split(string)
        G[i][j] = G[j][i] = int(x)
    f.close()
    return int(n), int(a), int(z)
#---------------------------------------------------------------------------#
def ViewMatrix(G,n):
    for i in range(1,n + 1):
        for j in range(1,n + 1):
            print("%d" % G[i][j], end = ' ')
        print()
#---------------------------------------------------------------------------#
def Algorithm_for_Tree(G, P, n, s, g):
    resul = 0
    Close = []
    O = []
    for i in range(const):
       Close.append(0)
       O.append(0)
    Open = []
    InitOpen(Open)
    m = 0
    m = AddQ(Open,m,resul,s)
```

```
    O[s] = 1
    P[s] = s
    while not EmptyOpen(Open):
        value, u, m = RemoveQ(Open)
        if u == g:
            resul = value
        for v in range(1, n + 1):
            if G[u][v] != 0 and O[v] == 0 and Close[v] == 0:
                x = value + G[u][v]
                m = AddQ(Open,m,x,v)
                O[v] = 1
                P[v] = u
        Close[u] = 1
        O[u] = 0
    return resul
#-------------------------------------------------------------------------#
def Print(P, n, s, g):
    Path = []
    for i in range(0,n + 1):
        Path.append(0)
    print("\nDuong di ngan nhat tu %d" % s, "den %d" % g,"la\nPath:", end = ' ');
    Path[0] = g
    i = P[g]
    k = 1
    while i != s:
        Path[k] = i
        k = k + 1
        i = P[i]
    Path[k] = s
    for j in range(0, k + 1):
        i = k - j
        if i > 0:
            print("%d => "% Path[i],end = ' ')
```

```
        else:
            print("%d" % Path[i],end = ' ')
#-------------------------------------------------------------------------#
def main():
    n,s,g = Init("data\Graph1.inp",G)
    print("n: %d" %n, end = '\n')
    ViewMatrix(G,n)
    for i in range(0,n + 1):
        P.append(0)
    resul = Algorithm_for_Tree(G,P, n, s, g)
    Print(P, n, s, g)
    print("\nresul: %d" %resul, end = '\n')
if __name__=="__main__":
    main()
```

DATA:



Graph.inp

8	1	5
1	2	3
1	7	4
1	8	3
3	8	1
4	5	5
4	8	2
6	7	4