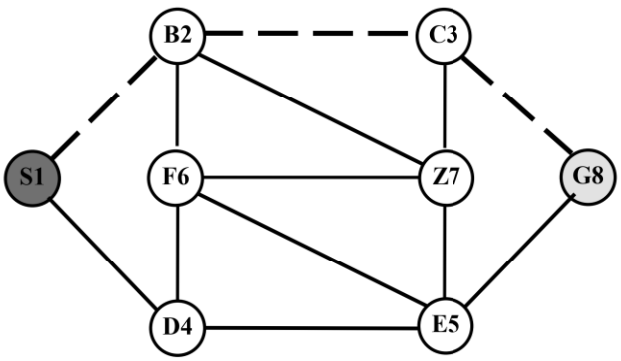
CHƯƠNG TRÌNH: Breadth First Search

```
G = []
P = []
n = 0
#---------------------------------------------------------------------------#
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] = x
    f.close()
    return n, a, z
#---------------------------------------------------------------------------#
def Check(M, n, u):
    for i in range(1, n + 1):
        if M[i] == u:
            return True
    return False
#---------------------------------------------------------------------------#
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 Breadth_First_Search(G, P, n, s, g):
    Open = []
    Close = []
    for j in range(n + 3):
        Open.append(0)
        Close.append(0)
        P.append(0)
    front = 1
```

```
    rear = 1;
    Open[rear] = s
    P[s] = s;
    while(front <= rear):
        u = Open[front]
        front = front + 1
        if u == g:
            return 1
        for v in range(1, n + 1):
            if G[u][v] != 0 and not Check(Open,n,v) and not Check(Close,n,v):
                rear = rear + 1
                Open[rear] = v;
                P[v] = u;
        Close[u] = u;
    return 0
#---------------------------------------------------------------------------#
def Print(P, n, s, g):
    Path = []
    for i in range(0,n + 1):
        Path.append(0)
    print("\nDuong di tu %d" % s, "den %d" % g,"la\n", 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\Graph.inp",G)
    Breadth_First_Search(G, P, n, s, g)
    Print(P, n, s, g)
if __name__=="__main__":
    main()
```

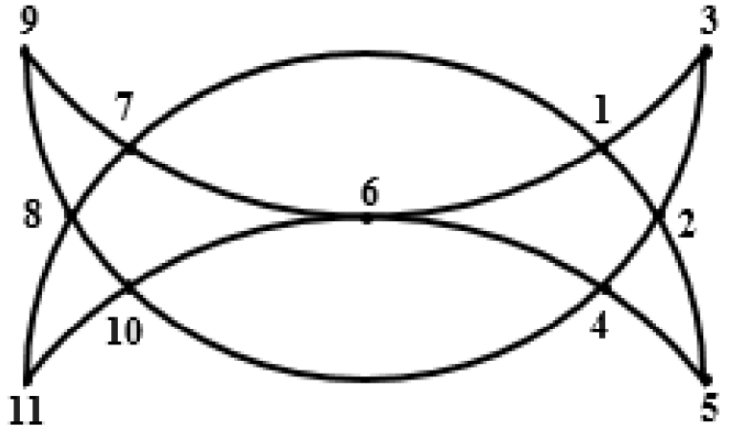


Thứ tự đỉnh s = 1, b = 2,c = 3,…
DATA:
Tạo file **`Graph1.inp`** để lưu trữ đồ thị và tính bậc đỉnh của đồ thị, đỉnh bắt đầu 1, đỉnh kết thúc 8.

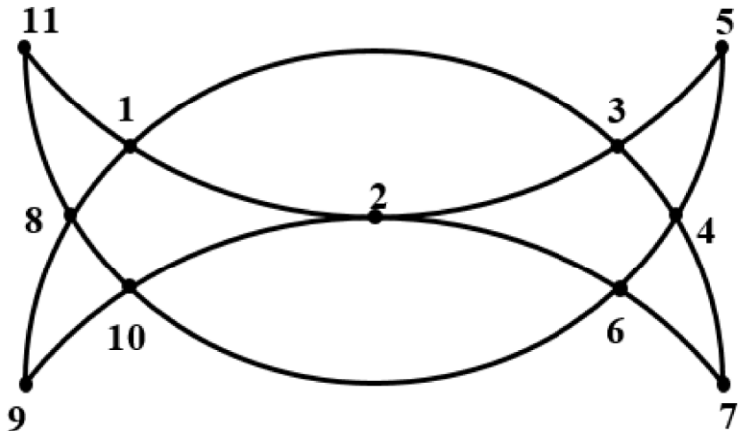
```
8    1    8
1    2    1
1    4    1
2    3    1
2    6    1
2    7    1
3    7    1
3    8    1
4    5    1
4    6    1
5    6    1
5    7    1
5    8    1
6    7    1
```

Tìm đường đi giữa hai đỉnh *u*, *v* trên các đồ thị sau:

ĐỒ THỊ G_1



ĐỒ THỊ G_2



ĐỒ THỊ G_3