CODE:

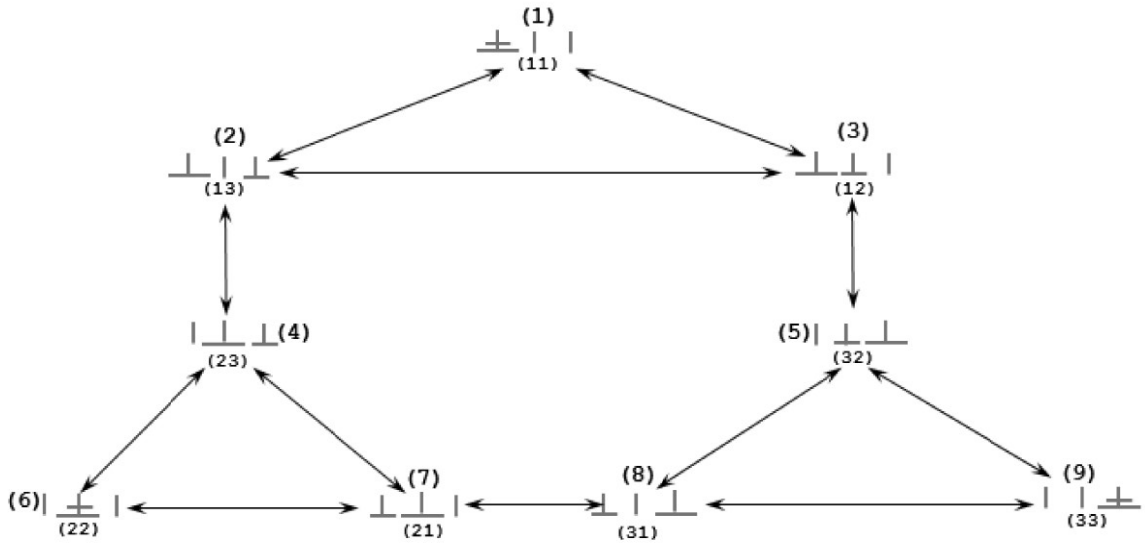
```
G = []
P = []
V = []
n = 0
#---------------------------------------------------------------------------#
def Split(string):
    k = string.index(' ')
    str = string[0:k]
    a = int(str,base = 10)
    m = string.index(' ',k + 1,-1)
    str = string[k + 1:m]
    b = int(str,base = 10)
    str = string[m + 1:len(string)]
    c = int(str,base = 10)
    return a, b, c
#---------------------------------------------------------------------------#
def Init(path_g,G,path_h,V):
    f = open(path_g)
    string = f.readline()
    string = string.replace('\t',' ')
    n, a, z = Split(string)
    for i in range(n + 1):
        V.append([])
        G.append([])
        for j in range(n + 1):
            G[i].append(0)
        for j in range(4):
            V[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()
```

```
    f = open(path_h)
    while True:
        string = f.readline()
        if not string:
            break
        string = string.replace('\t',' ')
        i, first, last = Split(string)
        V[i][0] = i
        V[i][1] = first
        V[i][2] = last
    return int(n), int(a), int(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("Cac buoc chuyen bai toan Thap Ha Noi, 2 dia")
    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:
            u = Path[i]
            v = Path[i + 1]
            if u == 1:
                print("Trang thai dau:\ndia lon o coc %d"% V[u][1],"\ndia nho o coc %d"% V[u][2],end = ' ')
            else:
                if V[v][1] == V[u][1]:
                    print("\nchuyen tu coc %d, "% V[v][2],"sang coc %d, "% V[u][2],end = ' ')
                elif V[v][2] == V[u][2]:
                    print("\nchuyen tu coc %d, "% V[v][1],"sang coc %d, "% V[u][1],end = ' ')
            if i == 0:
```

```
            print("\nTrang thai cuoi:\ndia lon o coc %d"% V[u][1],"\ndia nho o coc %d"% V[u][2],end = '\n')
#-------------------------------------------------------------------------#
def main():
    n,s,g = Init("data\ThapHaNoi_2.inp",G,"data\HaNoi_dinh.inp",V)
    Breadth_First_Search(G, P, n, s, g)
    Print(P, n, s, g)
if __name__=="__main__":
    main()
```

DATA:



ThapHaNoi_2.inp			HaNoi_dinh.inp		
9	1	9	1	1	1
1	2	1	2	1	3
1	3	1	3	1	2
2	3	1	4	2	3
2	4	1	5	3	2
3	5	1	6	2	2
4	6	1	7	2	1
4	7	1	8	3	1
5	8	1	9	3	3
5	9	1			
6	7	1			
7	8	1			
8	9	1			