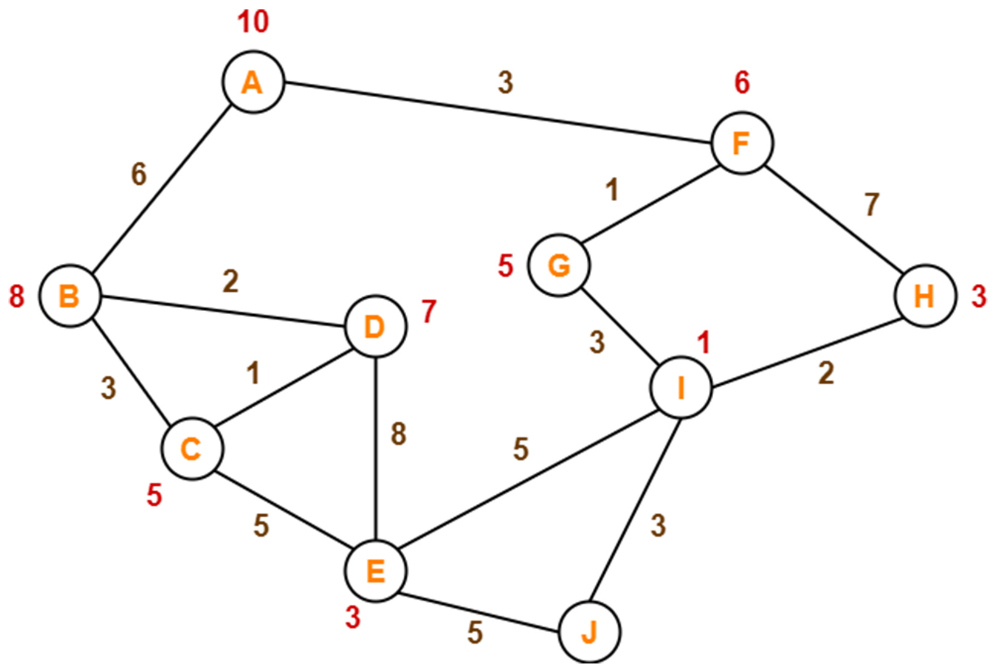
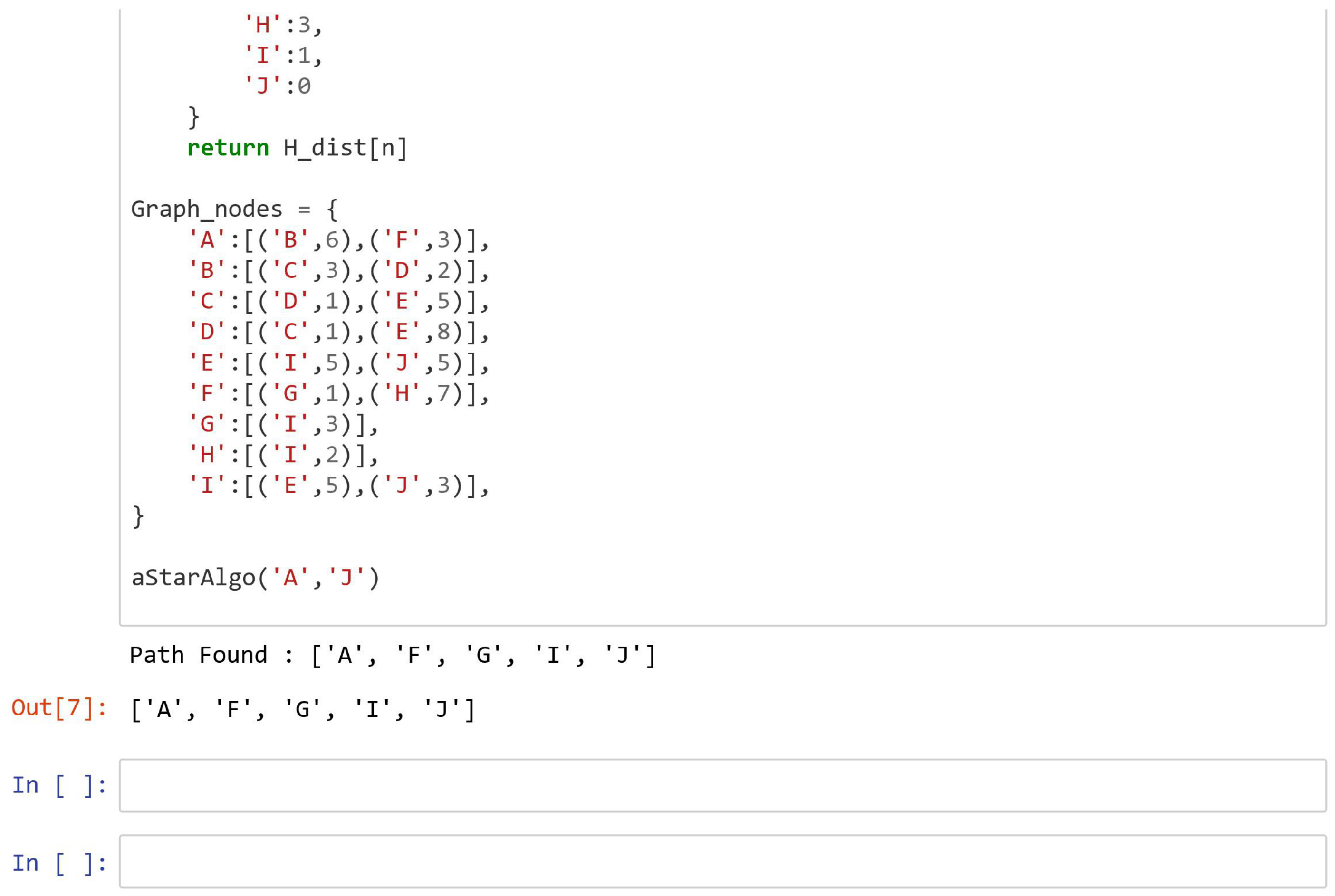
PROGRAM 1- AStarSearch Algorithm

Input Graph



In [7]:
```python
def aStarAlgo(start_node,stop_node):
    open_set=set(start_node)
    closed_set=set()
    g={}
    parents={}

    g[start_node]=0
    parents[start_node]=start_node
    while len(open_set)>0:
        n=None
        for v in open_set:
            if n==None or g[v]+heuristic(v)<g[n]+heuristic(n):
                n=v
        if n== stop_node or Graph_nodes[n] == None:
            pass
        else:
            for (m,weight) in get_neighbors(n):
                if m not in open_set and m not in closed_set:
                    open_set.add(m)
                    parents[m]=n
                    g[m]=g[n]+weight
                else:
                    if g[m]>g[n]+weight:
                        g[m]=g[n]+weight
                        parents[m]=n

                        if m in closed_set:
                            closed_set.remove(m)
                            open_set.add(m)
        if n == None:
            print('Path Does not exist')
            return None

        if n == stop_node:
            path=[]
            while parents[n]!= n:
                path.append(n)
                n=parents[n]
            path.append(start_node)
            path.reverse()
            print('Path Found : {}'.format(path))
            return path
        open_set.remove(n)
        closed_set.add(n)
    print('path does not exist')
    return None

def get_neighbors(v):
    if v in Graph_nodes:
        return Graph_nodes[v]
    else:
        return None

def heuristic(n):
    H_dist = {
        'A':10,
        'B':8,
        'C':5,
        'D':7,
        'E':3,
        'F':6,
        'G':5,
```



```
            'H':3,
            'I':1,
            'J':0
        }
        return H_dist[n]

Graph_nodes = {
        'A':[('B',6),('F',3)],
        'B':[('C',3),('D',2)],
        'C':[('D',1),('E',5)],
        'D':[('C',1),('E',8)],
        'E':[('I',5),('J',5)],
        'F':[('G',1),('H',7)],
        'G':[('I',3)],
        'H':[('I',2)],
        'I':[('E',5),('J',3)],
}

aStarAlgo('A','J')
```

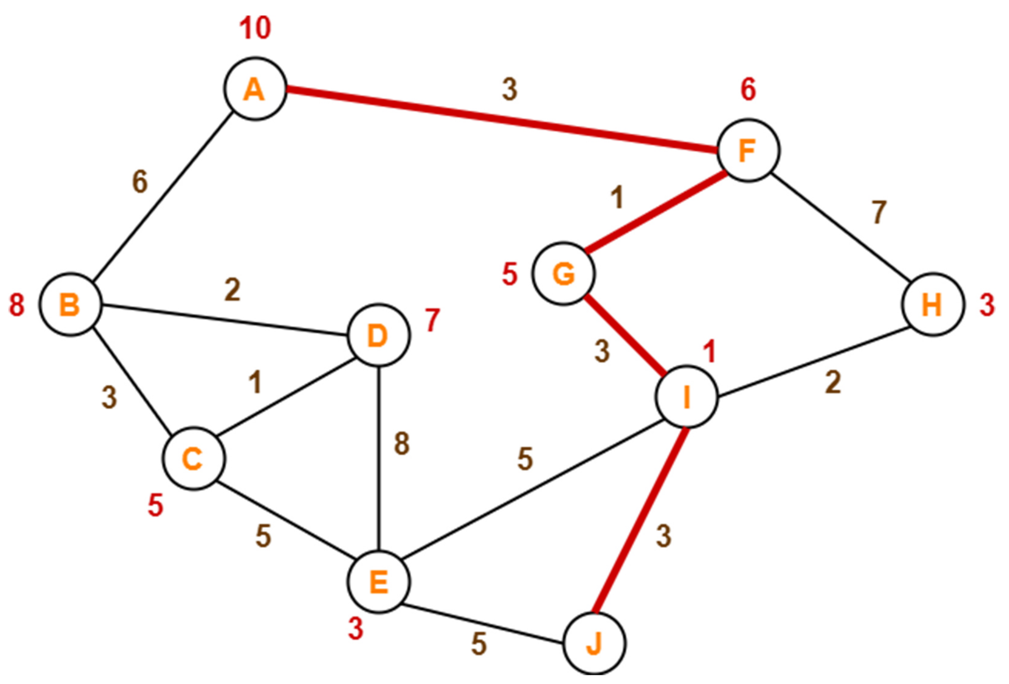
```
Path Found : ['A', 'F', 'G', 'I', 'J']
```

```
Out[7]: ['A', 'F', 'G', 'I', 'J']
```

```
In [ ]:
```

```
In [ ]:
```

Shortest path for input graph:



10
A
3
6
F
6
1
7
8
B
2
D
7
5
G
H
3
3
1
3
1
I
2
C
8
5
5
5
3
E
3
5
J