```
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':11,
        'B':6,
        'C':99,
        'D':1,
        'E':7,
        'G':0,
    }
    return h_dist[n]

graph_nodes={
    'A':[('B',2),('E',3)],
    'B':[('C',1),('G',9)],
    'C':None,
    'E':[('D',6)],
    'D':[('G',1)],
}
aStarAlgo('A','G')
```