```
graph = {
    'A':[('B', 2), ('E', 3)],
    'B':[('C', 1), ('G', 9)],
    'C':None,
    'D':[('G', 1)],
    'E':[('D', 6)],
    'G': None
}
openList = []
closedList = []
parent = {}
g = {} #f(n) = g(n) + h(n)
heuristicValues = {
    'A': 11,
    'B': 6,
    'C': 99,
    'D': 1,
    'E': 7,
    'G': 0
}
def h(n):
    return heuristicValues.get(n, 0)
def getChildren(n):
    return graph.get(n, None)
def AStar(startNode, stopNode):
    openList.append(startNode)
    g[startNode] = 0
    parent[startNode] = startNode
    #ol = B, E
    while(len(openList) > 0):
        node = None
        for v in openList:
            if node == None or g[v] + h(v) < g[node] + h(node):
                node = v
        if node is None:
            return None
        if node == stopNode:
            path = []
            while parent[node] != node:
                path.append(node)
                node = parent[node]
            path.append(startNode)
            path.reverse()
            return path
        for child, weight in getChildren(node):
            parent[child] = node
            if child not in [openList, closedList]: # if child not in openlist or child not in closedlist
                openList.append(child)
                g[child] = g[node] + weight
            else:
                if g[child] > g[node] + weight:
                    g[child] = g[node] + weight
                    if child in closedList:
                        closedList.remove(child)
                        openList.append(child)
        openList.remove(node)
        closedList.append(node)
res = AStar('A', 'G')
if res is None:
    print("No path exists")
else:
    print(res)
```