A* Search

In [6]:

```
def aStar(start, stop):
    open_set = set(start)
    closed_set = set()
    g = {}
    parents = {}

    g[start] = 0
    parents[start] = start

    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 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:
            path = []

            while parents[n] != n:
                path.append(n)
                n = parents[n]
            path.append(start)
            path.reverse()

            print("Path found: {}".format(path))
            return path

        open_set.remove(n)
        closed_set.add(n)
    print("Path does not exist")
    return None
```

In [7]:

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

In [8]:

```
def heuristic(n):
    h_dist = {

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

In [9]:

```
graph_nodes = {

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

In [10]:

```
aStar('A', 'G')
```

```
Path found: ['A', 'E', 'D', 'G']
```

Out[10]:

```
['A', 'E', 'D', 'G']
```

In [ ]: