

```

In [3]: from collections import deque
class Graph:
    def __init__(self,adj_list):
        self.adj_list=adj_list

    def get_neighbours(self,v):
        return self.adj_list[v]

    def h(self,n):
        H={
            'A':1,
            'B':1,
            'C':1,
            'D':1
        }
        return H[n]

    def a_star_alg(self,start_node,stop_node):
        open_list=set([start_node])
        close_list=set([])

        g={}
        g[start_node]=0

        parents={}
        parents[start_node]=start_node

        while len(open_list) >0:
            n=None

            for v in open_list:
                if n == None or g[v] +self.h(v) <g[n] +self.h(n):
                    n=v
            if n == None:
                print("Path does not exist")
                return None
            if n==stop_node:
                recons_path=[]
                while parents[n] != n:
                    recons_path.append(n)
                    n=parents[n]
                recons_path.append(start_node)
                recons_path.reverse()
                print("Path found!",recons_path)
                return recons_path
            for(m,weight) in self.get_neighbours(n):
                if m not in open_list and m not in close_list:
                    open_list.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 close_list:
            close_list.remove(m)
            open_list.add(m)
        open_list.remove(n)
        close_list.add(n)
        print("Path does not exist")
        return None
adj_list={
    'A': [('B',1), ('C',3), ('D',7)],
    'B': [('D',5)],
    'C': [('D',12)]
}
graph= Graph(adj_list)
graph.a_star_alg('A','D')
```

Path found!: ['A', 'B', 'D']

Out[3]: ['A', 'B', 'D']