A*search algorithm

```
from collections import deque

class Graph:
def __init__(self, adjac_lis):
self.adjac_lis = adjac_lis

def get_neighbors(self, v):
return self.adjac_lis.get(v, [])

def h(self, n):
H = {'A': 1, 'B': 1, 'C': 1, 'D': 1}
return H.get(n, 0)

def a_star_algorithm(self, start, stop):
open_lst = set([start])
closed_lst = set()
poo = {start: 0}
par = {start: start}

while open_lst:
n = min(open_lst, key=lambda v: poo[v] + self.h(v))
open_lst.remove(n)

if n == stop:
reconst_path = []
while n != start:
reconst_path.append(n)
n = par[n]
reconst_path.append(start)
reconst_path.reverse()
print(f"Path found: {reconst_path}")
return reconst_path

for m, weight in self.get_neighbors(n):
if m not in open_lst and m not in closed_lst:
open_lst.add(m)
par[m] = n
poo[m] = poo[n] + weight
elif poo[m] > poo[n] + weight:
poo[m] = poo[n] + weight
par[m] = n

print("Path does not exist!")
return None

adjac_lis = {
'A': [('B', 1), ('C', 3), ('D', 7)],
'B': [('D', 5)],
'C': [('D', 12)]
}

graph1 = Graph(adjac_lis)
graph1.a_star_algorithm('A', 'D')
```

```
OUTPUT:
Path found: ['A', 'B', 'D']
```