1. **Implement A\* Search algorithm.**

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 = {

'S': 5,

'A': 4,

'B': 5,

'E': 0,

}

return H\_dist[n]

Graph\_nodes = {

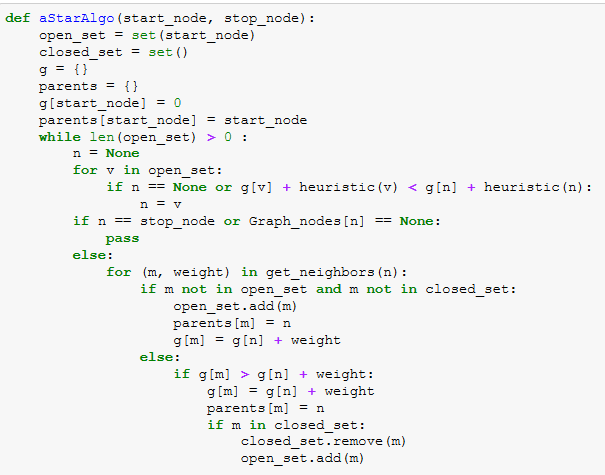
'S': [('A', 1), ('B', 2)],

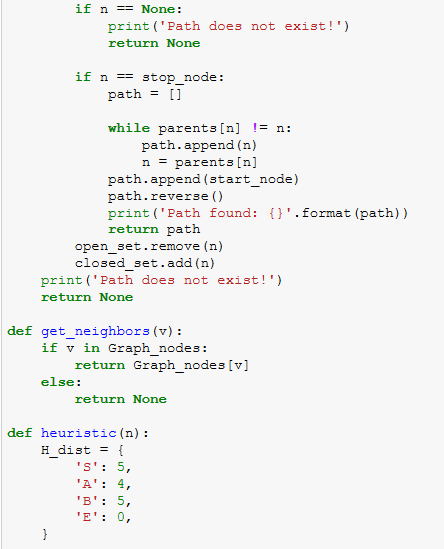
'A': [('E', 13),],

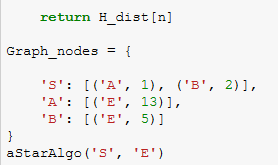
'B': [('E', 5)]

}

aStarAlgo('S', 'E')







**Algorithm: A\***

1. Start with OPEN containing only the initial node
2. Until a goal node is found, repeat the following procedure:
3. If no node on OPEN, report failure. Otherwise pick the node on OPEN with lowest f’ value. This node is called BESTNODE. Remove it from OPEN and place it in CLOSED.
4. If BESTNODE is a goal node, then exit and report a solution. Otherwise generate the successors of BESTNODE
5. For each SUCCESSOR, do the following:
6. Set SUCCESSOR to point to BESTNODE. These backwards links will make it possible to recover the path once a solution is found.
7. Compute g(SUCCESSOR)=g(BESTNODE)+the cost of getting from BESTNODE to SUCCESSOR.
8. If SUCCESSOR is same as the node on OPEN, then take this OLD node to the BESTNODE and update f’(OLD)
9. If SUCCESSOR was not on OPEN. If it is on CLOSED, call the node on CLOSED and add to BESTNODE.
10. If SUCCESSOR was not already on either OPEN or CLOSED, then put it on OPEN and compute

f’(SUCCESSOR)=g(SUCCESSOR)+h’(SUCCESSOR)