A* SEARCH ALGORITHM

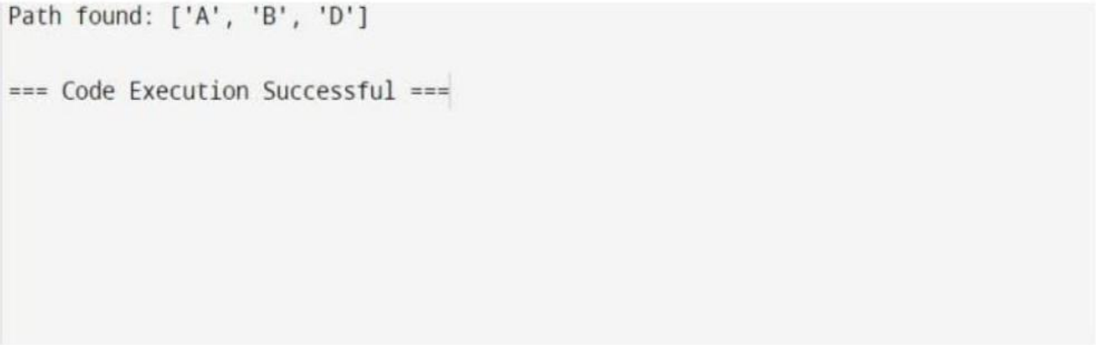
Name: SARANYA V **Reg No.:231801155**

PROGRAM:

```
 from collections import deque
class Graph:
               def __init__(self, adjacency_list):
                             self.adjacency_list = adjacency_list
               def get_neighbors(self, v):
                            return self.adjacency_list[v]
              def h(self, n):
                             H = { 'A': 1, 'B': 1, 'C': 1, 'D': 1 }
                             return H[n]
              def a_star_algorithm(self, start_node, stop_node):
                            open_list = set([start_node])
                            closed_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:
```

```
reconst_path = []

 while parents[n] != n:

             reconst_path.append(n)

              n = parents[n]

 reconst_path.append(start_node)

reconst_path.reverse()

print('Path found: {}'.format(reconst_path))

 return reconst_path

 for (m, weight) in self.get_neighbors(n):

             if m not in open_list and m not in closed_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 closed_list:

                                     closed_list.remove(m)

                                     open_list.add(m)

    open_list.remove(n)

   closed_list.add(n)

   print('Path does not exist!')

   return None
```

OUTPUT:



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

=== Code Execution Successful ===
```