PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

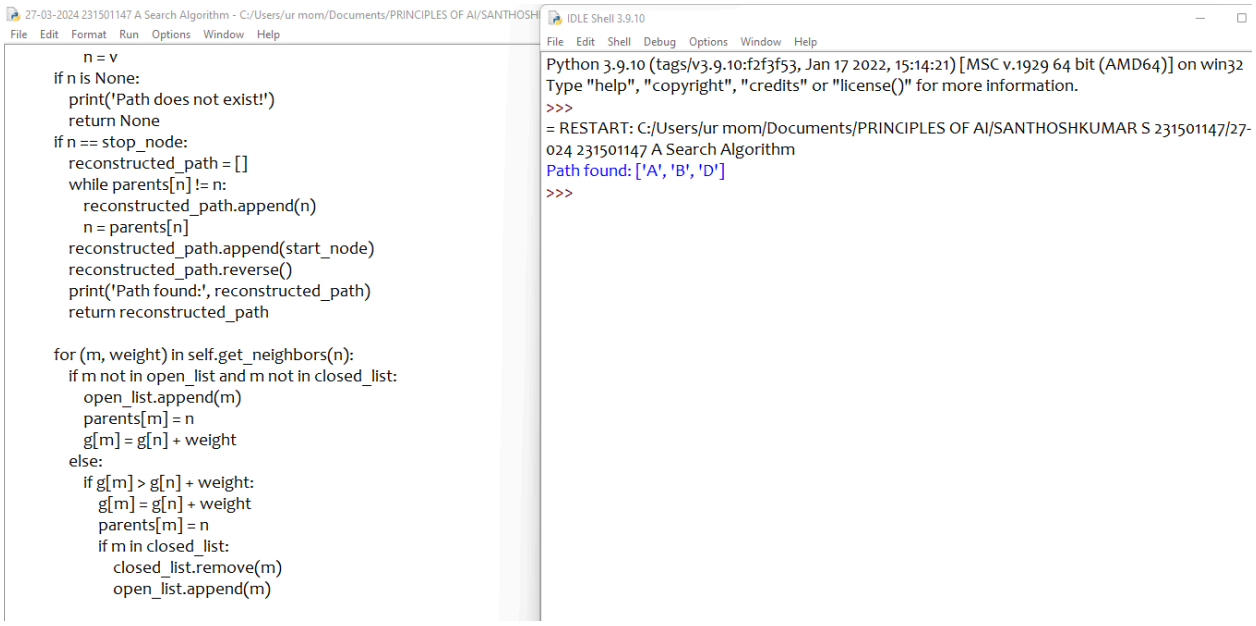
A* SEARCH ALGORITHM PYTHON PROGRAM

SOURCE CODE:

```
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):
        # Replace this with your specific heuristic function
        H = {
            'A': 1,
            'B': 1,
            'C': 1,
            'D': 1
        }
        return H[n]

    def a_star_algorithm(self, start_node, stop_node):
        open_list = deque([start_node])  # Use deque for efficient insertions/removals
        closed_list = set()
        g = {start_node: 0}  # Cost from start to each node
        parents = {start_node: start_node}

        while open_list:
            n = None
            for v in open_list:
                if n is None or g[v] + self.h(v) < g[n] + self.h(n):
                    n = v
            if n is None:
```

```
                print('Path does not exist!')
                return None
            if n == stop_node:
                reconstructed_path = []
                while parents[n] != n:
                    reconstructed_path.append(n)
                    n = parents[n]
                reconstructed_path.append(start_node)
                reconstructed_path.reverse()
                print('Path found:', reconstructed_path)
                return reconstructed_path

            for (m, weight) in self.get_neighbors(n):
                if m not in open_list and m not in closed_list:
                    open_list.append(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.append(m)
            open_list.remove(n)
            closed_list.add(n)

        print('Path does not exist!')
        return None
# Example usage (assuming the adjacency list is defined as before)
adjacency_list = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('D', 5)],
    'C': [('D', 12)]
}
graph1 = Graph(adjacency_list)
graph1.a_star_algorithm('A', 'D')
```

OUTPUT:



```
27-03-2024 231501147 A Search Algorithm - C:/Users/ur mom/Documents/PRINCIPLES OF AI/SANTHOSH
File  Edit  Format  Run  Options  Window  Help
            n = v
        if n is None:
            print('Path does not exist!')
            return None
        if n == stop_node:
            reconstructed_path = []
            while parents[n] != n:
                reconstructed_path.append(n)
                n = parents[n]
            reconstructed_path.append(start_node)
            reconstructed_path.reverse()
            print('Path found:', reconstructed_path)
            return reconstructed_path

        for (m, weight) in self.get_neighbors(n):
            if m not in open_list and m not in closed_list:
                open_list.append(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.append(m)
```

```
IDLE Shell 3.9.10
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ur mom/Documents/PRINCIPLES OF AI/SANTHOSHKUMAR S 231501147/27-024 231501147 A Search Algorithm
Path found: ['A', 'B', 'D']
>>>
```