A\* Search Algorithm

Aim:

To implement the **A\*** (**A-star**) search algorithm in Python to find the **shortest path** from a start node to a goal node in a weighted graph, using both path cost and a heuristic function to guide the search.

Code:

from queue import PriorityQueue

def a\_star(start, goal, graph, h):

open\_list = PriorityQueue()

open\_list.put((0, start))

came\_from = {}

g\_score = {node: float('inf') for node in graph}

g\_score[start] = 0

while not open\_list.empty():

\_, current = open\_list.get()

if current == goal:

path = []

while current in came\_from:

path.append(current)

current = came\_from[current]

path.append(start)

path.reverse()

return path

for neighbor, cost in graph[current]:

tentative\_g\_score = g\_score[current] + cost

if tentative\_g\_score < g\_score[neighbor]:

came\_from[neighbor] = current

g\_score[neighbor] = tentative\_g\_score

f\_score = tentative\_g\_score + h[neighbor]

open\_list.put((f\_score, neighbor))

return None

graph = {

'A': [('B', 1), ('C', 3)],

'B': [('D', 3), ('E', 1)],

'C': [('F', 5)],

'D': [],

'E': [('F', 2)],

'F': []

}

heuristic = {'A': 6, 'B': 4, 'C': 4, 'D': 2, 'E': 2, 'F': 0}

path = a\_star('A', 'F', graph, heuristic)

print("Path found by A\*:", path)

Result:

Path found by A\*: ['A', 'B', 'E', 'F']