LAB 3

Aim: Heuristic Search Techniqus

Theoretical Description/ Algorithm : Program To Implement A* Algorithm

Source Code:

```
import networkx as nx
import matplotlib.pyplot as plt
import heapq

# A* Algorithm function
def a_star_algorithm(tree, start, goal, heuristic):
  open_set = []
  heapq.heappush(open_set, (0, start))

  g_cost = {node: float('inf') for node in tree}
  g_cost[start] = 0

  f_cost = {node: float('inf') for node in tree}
  f_cost[start] = heuristic[start]

  parent = {node: None for node in tree}

  explored_nodes = []

  while open_set:
    current_f_cost, current_node = heapq.heappop(open_set)
    explored_nodes.append(current_node)

    if current_node == goal:
      path = []
      while current_node is not None:
        path.append(current_node)
        current_node = parent[current_node]
      return path[::-1], explored_nodes  # Return reversed path and explored nodes
```

```
        for neighbor, weight in tree[current_node]:
            tentative_g_cost = g_cost[current_node] + weight
            if tentative_g_cost < g_cost[neighbor]:
                parent[neighbor] = current_node
                g_cost[neighbor] = tentative_g_cost
                f_cost[neighbor] = g_cost[neighbor] + heuristic[neighbor]
                heapq.heappush(open_set, (f_cost[neighbor], neighbor))

    return [], explored_nodes  # No path found

# Tree structure (graph) - Undirected edges
tree = {
    'A': [('B', 6), ('F', 3)],
    'B': [('A', 6), ('C', 3), ('D', 2)],
    'C': [('B', 3), ('E', 5)],
    'D': [('B', 2), ('E', 8)],
    'E': [('C', 5), ('D', 8), ('I', 5), ('J', 5)],
    'F': [('A', 3), ('G', 1), ('H', 7)],
    'G': [('F', 1), ('I', 3)],
    'H': [('F', 7), ('I', 2)],
    'I': [('G', 3), ('H', 2), ('E', 5), ('J', 3)],
    'J': [('E', 5), ('I', 3)]
}

# Heuristic values for A* search
heuristic = {
    'A': 10,
    'B': 8,
    'C': 5,
    'D': 7,
    'E': 3,
    'F': 6,
    'G': 5,
    'H': 3,
```

```
    'I': 1,
    'J': 0  # Goal node
}

start = 'A'
goal = 'J'

# Run A* search and get the path and explored nodes
path, explored_nodes = a_star_algorithm(tree, start, goal, heuristic)

# Create the undirected graph for visualization
G = nx.Graph()

# Add edges and weights to the undirected graph
edges = [
    ('A', 'B', 6), ('A', 'F', 3),
    ('B', 'C', 3), ('B', 'D', 2),
    ('C', 'E', 5),
    ('D', 'E', 8),
    ('E', 'I', 5), ('E', 'J', 5),
    ('F', 'G', 1), ('F', 'H', 7),
    ('G', 'I', 3),
    ('H', 'I', 2),
    ('I', 'E', 5), ('I', 'J', 3)
]

for edge in edges:
    G.add_edge(edge[0], edge[1], weight=edge[2])

# Get positions for the nodes
pos = nx.spring_layout(G)

# Draw the complete graph
plt.figure(figsize=(10, 7))
```

```
# Draw nodes
nx.draw_networkx_nodes(G, pos, node_size=700, node_color='lightpink')

# Highlight start and goal nodes
nx.draw_networkx_nodes(G, pos, nodelist=[start], node_color='lightgreen', node_size=800, label="Start")
nx.draw_networkx_nodes(G, pos, nodelist=[goal], node_color='lightblue', node_size=800, label="Goal")

# Draw explored nodes in yellow
explored_nodes.remove(start)  # Remove start from explored, as it's already colored green
nx.draw_networkx_nodes(G, pos, nodelist=explored_nodes, node_color='yellow', node_size=700)

# Highlight the path found by A* in orange
path_edges = [(path[i], path[i + 1]) for i in range(len(path) - 1)]
nx.draw_networkx_edges(G, pos, edgelist=path_edges, edge_color='orange', width=3)

# Draw undirected edges with weights
nx.draw_networkx_edges(G, pos, edgelist=G.edges(), style='solid', arrows=False)
nx.draw_networkx_edge_labels(G, pos, edge_labels={(u, v): d['weight'] for u, v, d in G.edges(data=True)})

# Draw labels (node names)
nx.draw_networkx_labels(G, pos, font_size=12, font_family="sans-serif")

# Display the graph
plt.title(f"A* Path from {start} to {goal} (Undirected Graph)")
plt.axis('off')
plt.show()

# Print the final path and explored nodes
print("Path:", path)
print("Explored Nodes:", explored_nodes)
```

Output:

```
~/python ai
python -u "/Users/charuramnani/python ai/undirectedvis.py"
Path: ['A', 'F', 'G', 'I', 'J']
Explored Nodes: ['F', 'G', 'I', 'J']
```