AI LAB WEEK 5

Dr. Mubashir Ahmad

HOW TO CREATE A SIMPLE GRAPH IN PYTHON

Install Networkx with conda for visualization

Open anaconda prompt and write the command

conda install

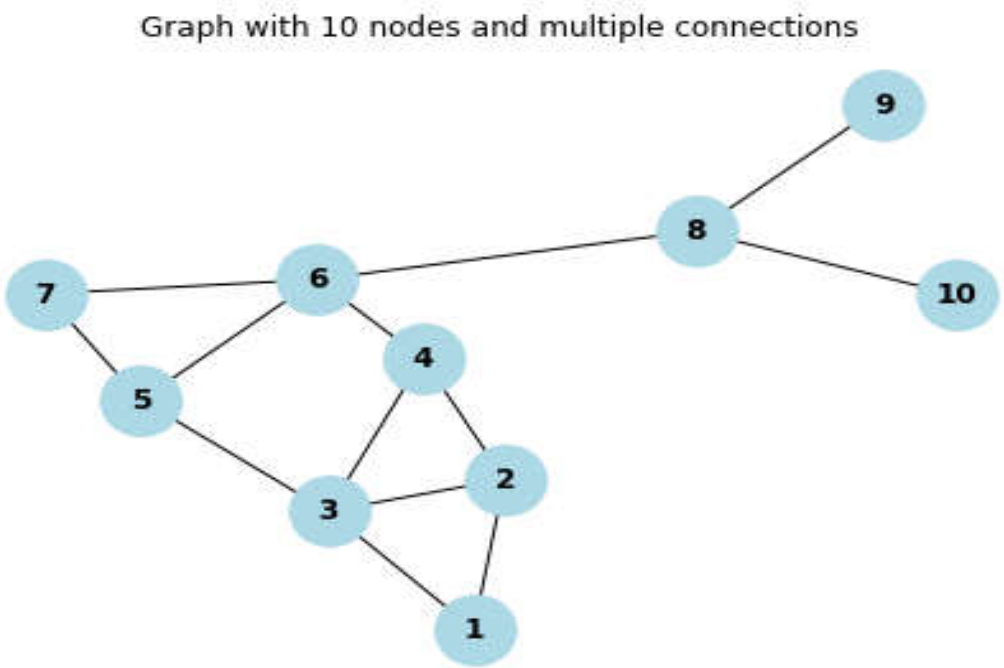
To install this package run the following:

conda install anaconda::networkx

HOW TO CREATE A SIMPLE GRAPH IN PYTHON

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

# Create a graph
G = nx.Graph()
nodes = range(1, 11)
edges = [(1, 2), (1, 3), (2, 3), (2, 4), (3, 4), (3, 5), (4, 6), (5, 6), (5, 7), (6, 7), (6, 8), (8, 9), (8, 10)]

G.add_nodes_from(nodes)
G.add_edges_from(edges)

# Visualize the graph
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=1000, node_color='lightblue', font_weight='bold')
plt.title("Graph with 10 nodes and multiple connections")
plt.show()
```

HOW TO CREATE A SIMPLE GRAPH IN PYTHON



BEST FIRST SEARCH

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

# Define the graph
G = nx.Graph()
G.add_nodes_from(range(1, 7))
G.add_edges_from([(1, 2), (1, 3), (2, 4), (2, 5), (3, 6), (4, 6), (5, 6)])

# Define the heuristic values for each node
heuristics = {1: '5', 2: '4', 3: '2', 4: '3', 5: '1', 6: '0'}
```

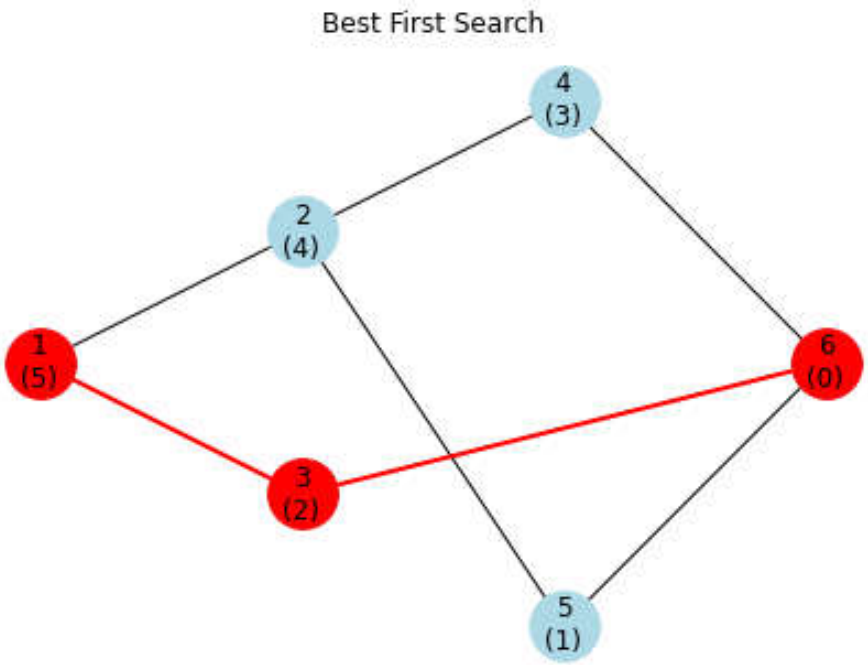
BEST FIRST SEARCH

```python
# Implement Best First Search algorithm
def best_first_search(graph, start, goal, heuristics):
    visited = []
    queue = [(start, [start])]

    while queue:
        node, path = queue.pop(0)
        visited.append(node)

        if node == goal:
            return path

        neighbors = sorted(graph.neighbors(node), key=lambda x: heuristics[x])
        for neighbor in neighbors:
            if neighbor not in visited:
                queue.append((neighbor, path + [neighbor]))
```

BEST FIRST SEARCH

```
# Find the best path
start_node = 1
goal_node = 6
best_path = best_first_search(G, start_node, goal_node, heuristics)

# Visualize the graph with the best path
pos = {1: (1, 3), 2: (2, 4), 3: (2, 2), 4: (3, 5), 5: (3, 1), 6: (4, 3)}
nx.draw(G, pos, with_labels=False, node_size=1000, node_color='lightblue')

# Draw heuristic values near nodes
nx.draw_networkx_labels(G, pos, labels={node: f"{node}\n({heuristics[node]})"
for node in G.nodes()}, font_size=12, font_color='black', verticalalignment='center')

nx.draw_networkx_nodes(G, pos, nodelist=best_path, node_color='red', node_size=1000)
nx.draw_networkx_edges(G, pos, edgelist=[(best_path[i], best_path[i+1])
for i in range(len(best_path)-1)], edge_color='red', width=2)
plt.title("Best First Search")
plt.show()
```

BEST FIRST SEARCH



A* SEARCH ALGORITHM

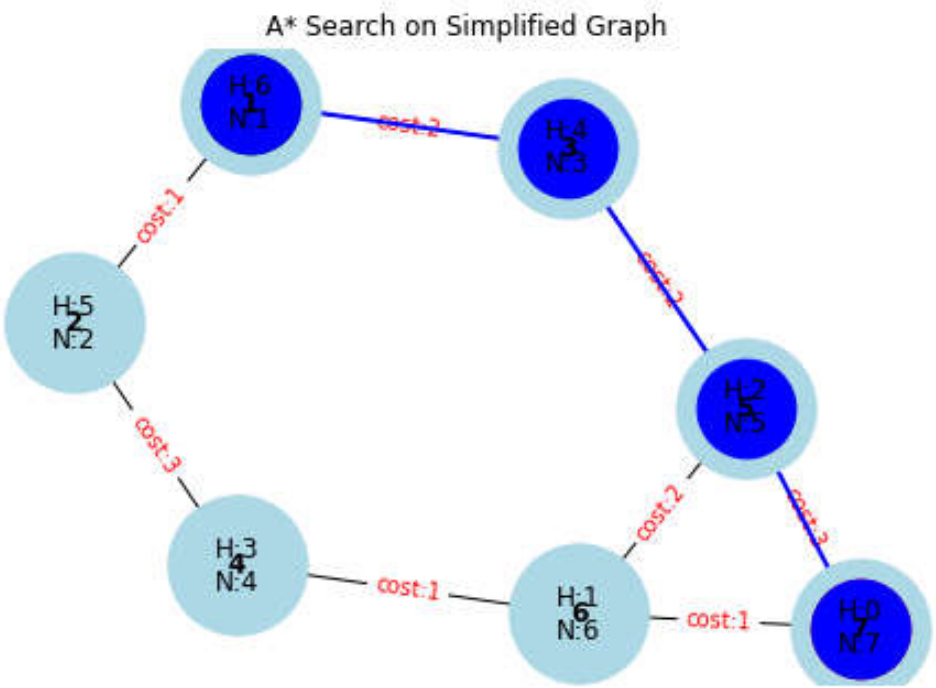
```
import networkx as nx
import matplotlib.pyplot as plt
import heapq

# Create a graph
G = nx.Graph()
nodes = range(1, 8)
edges = [(1, 2, {'cost': 1}), (1, 3, {'cost': 2}), (2, 4, {'cost': 3}),
         (3, 5, {'cost': 2}), (4, 6, {'cost': 1}), (5, 6, {'cost': 2}),
         (5, 7, {'cost': 3}), (6, 7, {'cost': 1})]

G.add_nodes_from(nodes)
G.add_edges_from(edges)

# Define the heuristic values for each node
heuristics = {1: 6, 2: 5, 3: 4, 4: 3, 5: 2, 6: 1, 7: 0}
```

A* SEARCH ALGORITHM

```python
# Implement A* search algorithm
def astar_search(graph, start, goal, heuristics):
    visited = set()
    heap = [(0 + heuristics[start], start, [start])]

    while heap:
        _, node, path = heapq.heappop(heap)

        if node in visited:
            continue

        if node == goal:
            return path

        visited.add(node)

        for neighbor in graph.neighbors(node):
            if neighbor not in visited:
                heapq.heappush(heap, (len(path) + heuristics[neighbor], neighbor, path + [neighbor]))
```

```python
start_node = 1
goal_node = 7
best_path = astar_search(G, start_node, goal_node, heuristics)

# Visualize the graph with the best path
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True, node_size=4000, node_color='lightblue', font_weight='bold')

# Draw edge costs on edges
edge_labels = {(u, v): f"cost:{d['cost']}" for u, v, d in G.edges(data=True)}
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels, font_color='red')

# Draw heuristic values near nodes
nx.draw_networkx_labels(G, pos, labels={node: f"H:{heuristics[node]}\nN:{node}"
for node in G.nodes()}, font_size=12, font_color='black', verticalalignment='center')

# Highlight start and goal nodes
nx.draw_networkx_nodes(G, pos, nodelist=[start_node], node_color='green', node_size=2000)
nx.draw_networkx_nodes(G, pos, nodelist=[goal_node], node_color='red', node_size=2000)

# Highlight best path
nx.draw_networkx_nodes(G, pos, nodelist=best_path, node_color='blue', node_size=2000)
nx.draw_networkx_edges(G, pos, edgelist=[(best_path[i], best_path[i+1])
                                         for i in range(len(best_path)-1)], edge_color='blue', width=2)

plt.title("A* Search on Simplified Graph")
plt.show()
```

A* SEARCH ALGORITHM



TASKS

#1: Please implement a graph with 10 nodes with multiple connections

#2: Write heuristic values and on the nodes and costs on the edges

#3: Visualize the graph

#4: Write the code for A* Search algorithm with 10 nodes graph