AI LAB
ASSIGNMENT 7

by

AMMAR JAMIL (01-134231-010)

Submitted to: DR ARSHAD FARHAD
Date: 14/10/2025

DEPARTMENT OF COMPUTER SCIENCE

BAHRIA UNIVERSITY ISLAMABAD E-8

Task1:

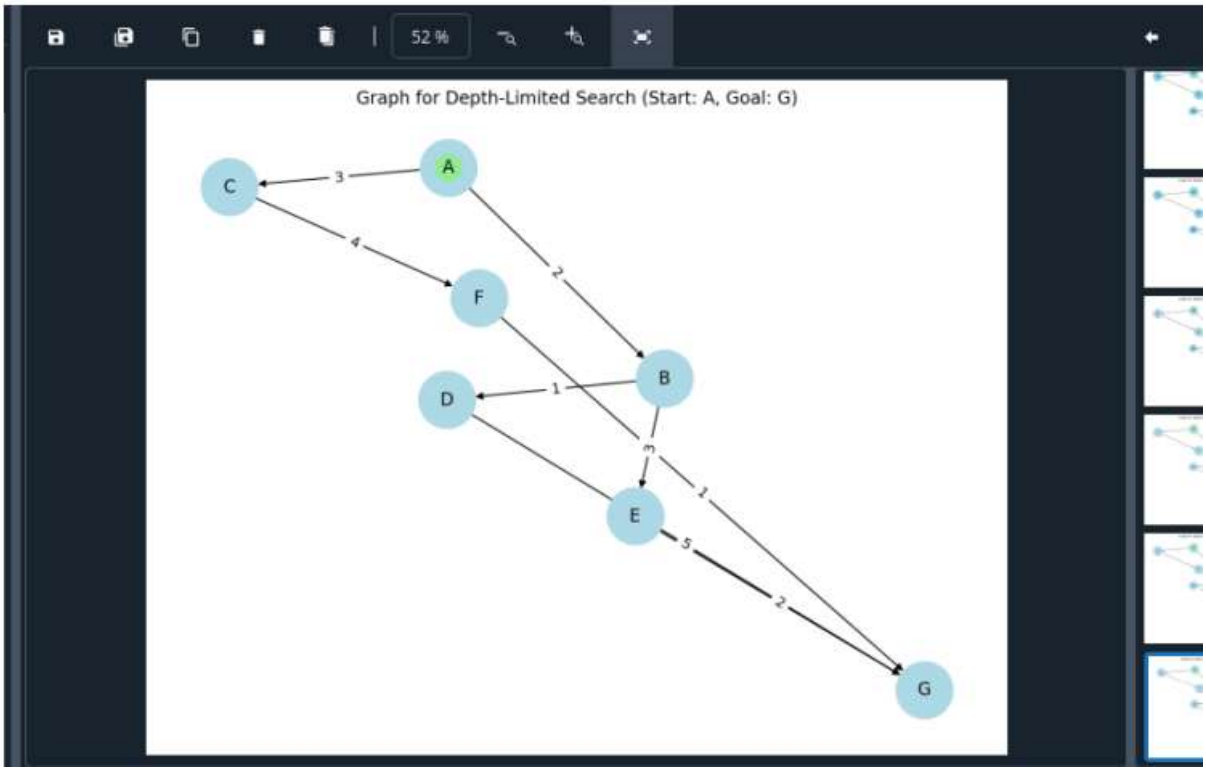
Task 1: DLS

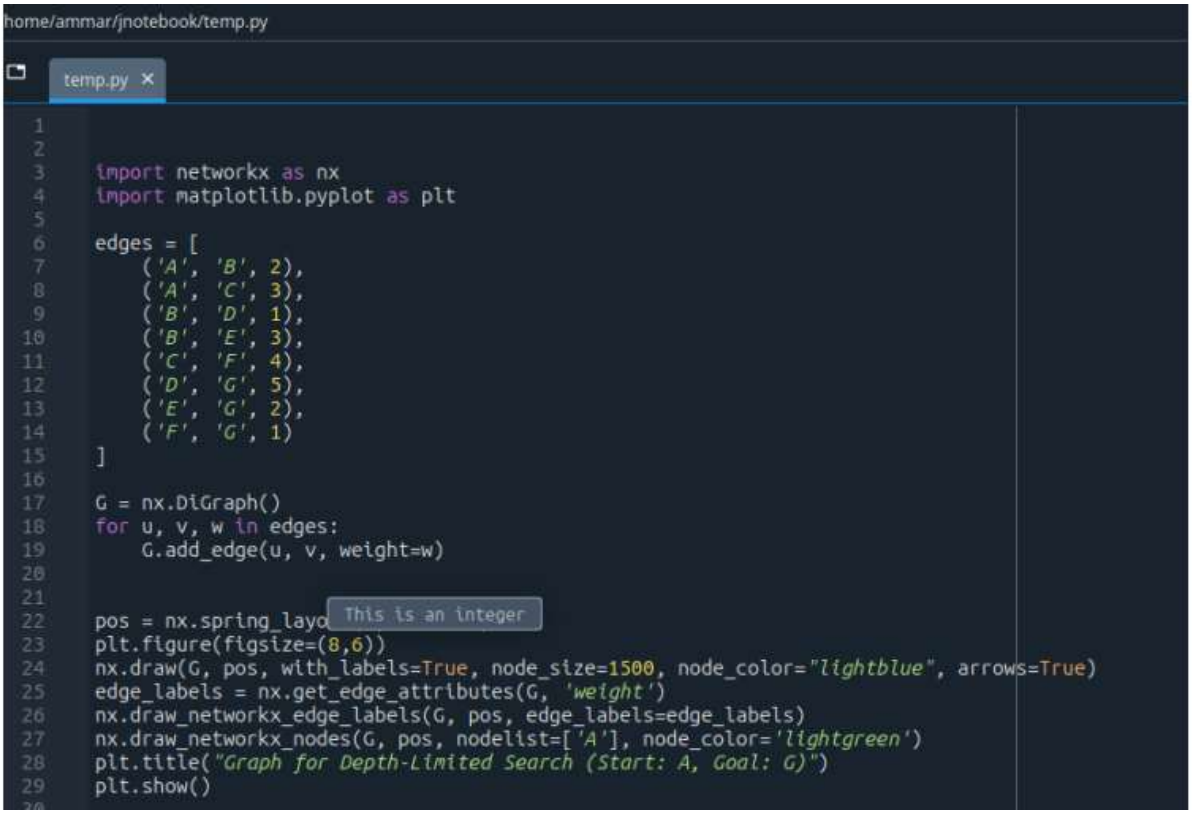
Key Constraint

- **Cannot explore beyond the specified depth limit**
- Nodes at depth = limit are treated as leaf nodes (not expanded further)
- If goal exists beyond depth limit, DLS will return failure
- **Depth Limit**: 3 levels (Root A is at depth 0)

Implementation Tasks

1. Implement DLS with strict depth limit enforcement
2. Test with goals at different depth levels
3. Demonstrate failure when goal is beyond depth limit
4. Show that nodes at depth limit are not expanded




home/ammar/jnotebook/temp.py
temp.py
import networkx as nx
import matplotlib.pyplot as plt
edges = [
('A', 'B', 2),
('A', 'C', 3),
('B', 'D', 1),
('B', 'E', 3),
('C', 'F', 4),
('D', 'G', 5),
('E', 'G', 2),
('F', 'G', 1)
]
G = nx.DiGraph()
for u, v, w in edges:
G.add_edge(u, v, weight=w)
pos = nx.spring_layo
This is an integer
plt.figure(figsize=(8,6))
nx.draw(G, pos, with_labels=True, node_size=1500, node_color="lightblue", arrows=True)
edge_labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
nx.draw_networkx_nodes(G, pos, nodelist=['A'], node_color='lightgreen')
plt.title("Graph for Depth-Limited Search (Start: A, Goal: G)")
plt.show()

IPython Console History

```
    visited = []

    def recursive_dls(node, depth):
        visited.append(node)
        print("Visiting:", node, "at depth", depth)

        if node == goal:
            return True
        if depth == limit:
            print("Reached depth limit at node", node, "- not expanding further.")
            return False

        for neighbor in graph.neighbors(node):
            if recursive_dls(neighbor, depth + 1):
                return True
        return False

    found = recursive_dls(start, 0)
    print("\nTraversal Order:", visited)
    if found:
        print("Goal", goal, "found within depth limit", limit)
    else:
        print("Goal", goal, "NOT found within depth limit", limit)


DLS(G, 'A', 'G', limit=3)


DLS(G, 'A', 'G', limit=1)

```

Console 1/A

```
In [8]: %runfile /home/ammar/jnotebook/temp.py --wdir
Visiting: A at depth 0
Visiting: B at depth 1
Visiting: D at depth 2
Visiting: G at depth 3

Traversal Order: ['A', 'B', 'D', 'G']
Goal G found within depth limit 3
Visiting: A at depth 0
Visiting: B at depth 1
Reached depth limit at node B - not expanding further.
Visiting: C at depth 1
Reached depth limit at node C - not expanding further.

Traversal Order: ['A', 'B', 'C']
Goal G NOT found within depth limit 1

In [9]:
```

Task2

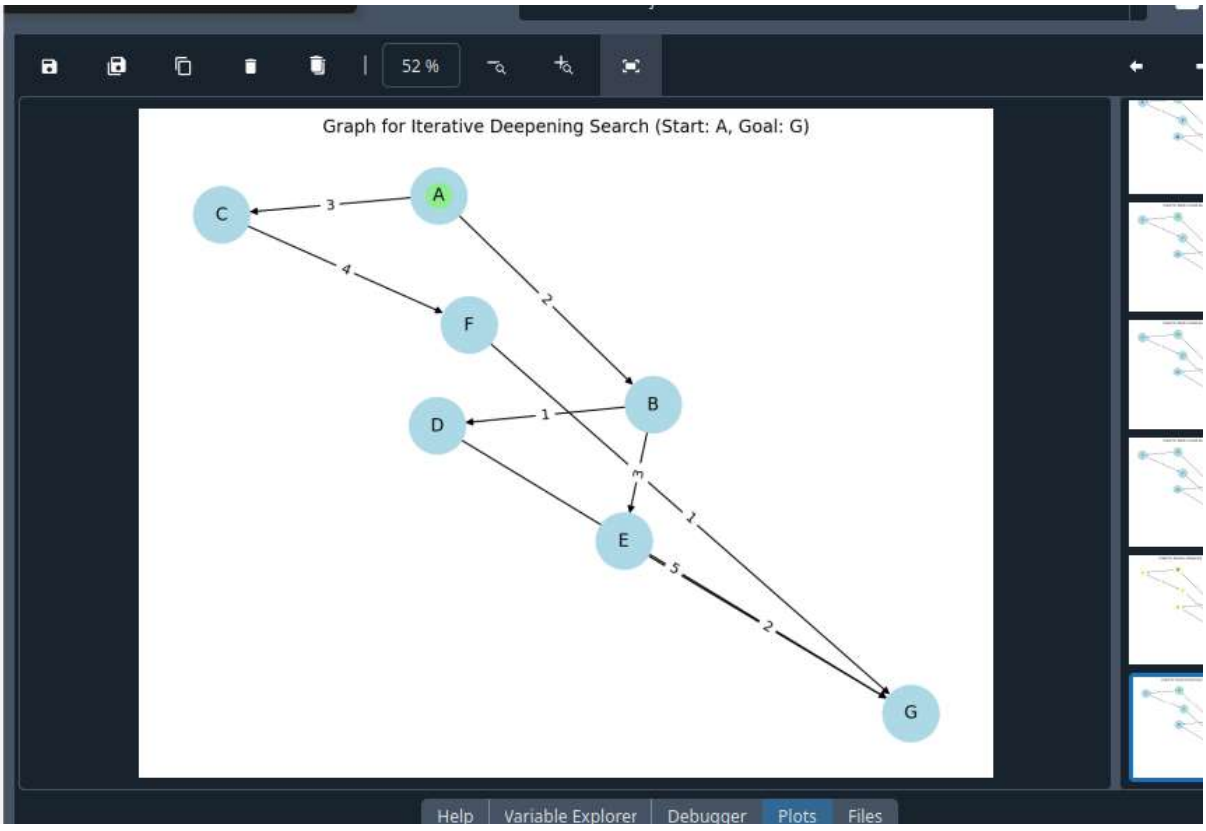
Task 2: IDS

Algorithm Requirements

- Start with depth limit 0, increment by 1 each iteration
- Reuse DLS implementation from Task 1
- Continue increasing depth until goal is found
- Track cumulative nodes expanded across all iterations

Implementation Tasks

1. Implement IDS that calls DLS with increasing depth limits
2. Show how IDS finds goals that DLS with fixed limit cannot
3. Compare nodes expanded in IDS vs theoretical DFS
4. Demonstrate completeness of IDS



```python


import networkx as nx
import matplotlib.pyplot as plt

edges = [
    ('A', 'B', 2),
    ('A', 'C', 3),
    ('B', 'D', 1),
    ('B', 'E', 3),
    ('C', 'F', 4),
    ('D', 'G', 5),
    ('E', 'G', 2),
    ('F', 'G', 1)
]

G = nx.DiGraph()
for u, v, w in edges:
    G.add_edge(u, v, weight=w)

pos = nx.spring_layout(G, seed=42)
plt.figure(figsize=(8,6))
nx.draw(G, pos, with_labels=True, node_size=1500, node_color="lightblue", arrows=True)
edge_labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=edge_labels)
nx.draw_networkx_nodes(G, pos, nodelist=['A'], node_color='lightgreen')
plt.title("Graph for Iterative Deepening Search (Start: A, Goal: G)")
plt.show()

def DLS(graph, start, goal, limit):
    visited = []

    def recursive_dls(node, depth):
        visited.append(node)
        if node == goal:
            return True
        if depth == limit:
            return False

        for neighbor in graph.neighbors(node):
            if recursive_dls(neighbor, depth + 1):
                return True
        return False

    found = recursive_dls(start, 0)
    return found, visited
```

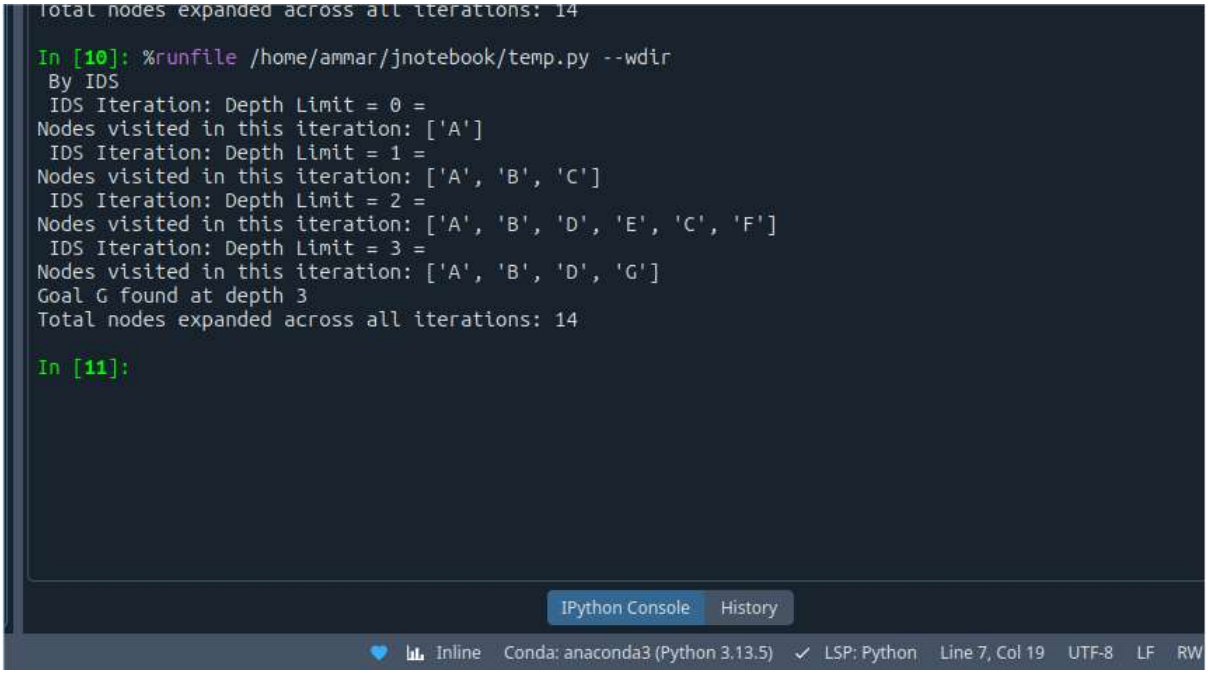
```python
def IDS(graph, start, goal):
    total_nodes_expanded = 0
    depth = 0

    while True:
        print(" IDS Iteration: Depth Limit =", depth, "=")
        found, visited = DLS(graph, start, goal, depth)
        total_nodes_expanded += len(visited)
        print("Nodes visited in this iteration:", visited)

        if found:
            print("Goal", goal, "found at depth", depth)
            print("Total nodes expanded across all iterations:", total_nodes_expanded)
            break

        depth += 1

print(" By IDS ")
IDS(G, 'A', 'G')
```


Total nodes expanded across all iterations: 14
In [10]: %runfile /home/ammar/jnotebook/temp.py --wdir
By IDS
IDS Iteration: Depth Limit = 0 =
Nodes visited in this iteration: ['A']
IDS Iteration: Depth Limit = 1 =
Nodes visited in this iteration: ['A', 'B', 'C']
IDS Iteration: Depth Limit = 2 =
Nodes visited in this iteration: ['A', 'B', 'D', 'E', 'C', 'F']
IDS Iteration: Depth Limit = 3 =
Nodes visited in this iteration: ['A', 'B', 'D', 'G']
Goal G found at depth 3
Total nodes expanded across all iterations: 14
In [11]:
IPython Console
History
Inline
Conda: anaconda3 (Python 3.13.5)
LSP: Python
Line 7, Col 19
UTF-8
LF
RW