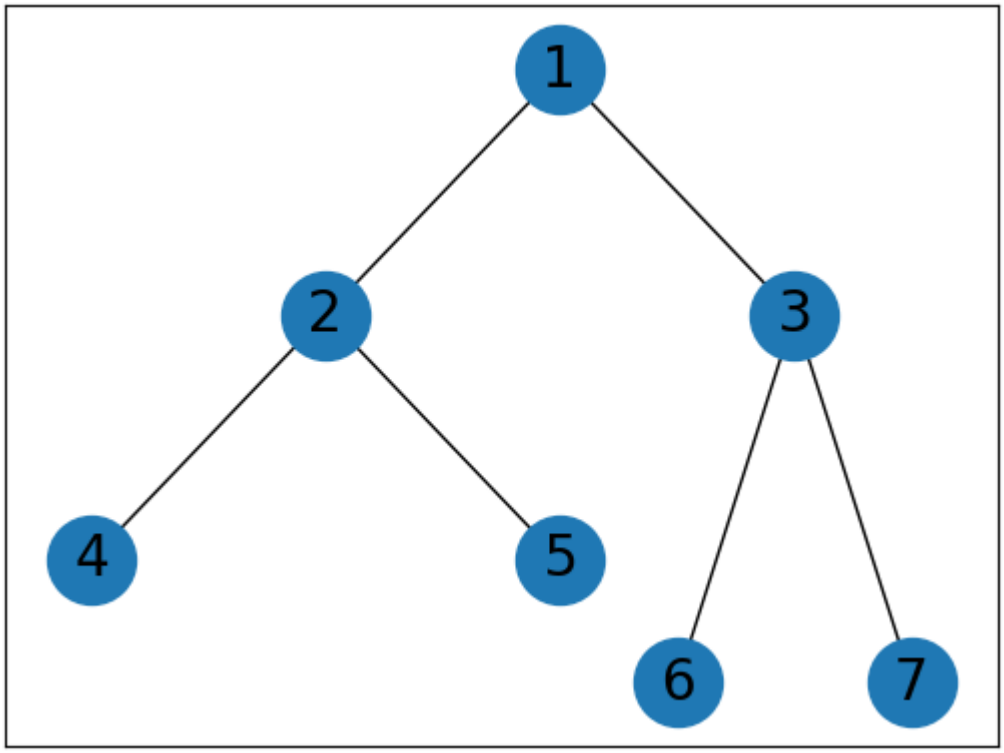
Breadth firsh search

In [ ]:
```
import networkx as nx
import matplotlib.pyplot as plt
def bfs(graph, sn):
    visited = set()
    queue = [sn]
    tp = []
    while queue:
        node = queue.pop(0)
        if node not in visited:
            tp.append(node)
            visited.add(node)
            neighbors = graph.neighbors(node)
            for i in neighbors:
                if i not in visited:
                    queue.append(i)
    return tp
```

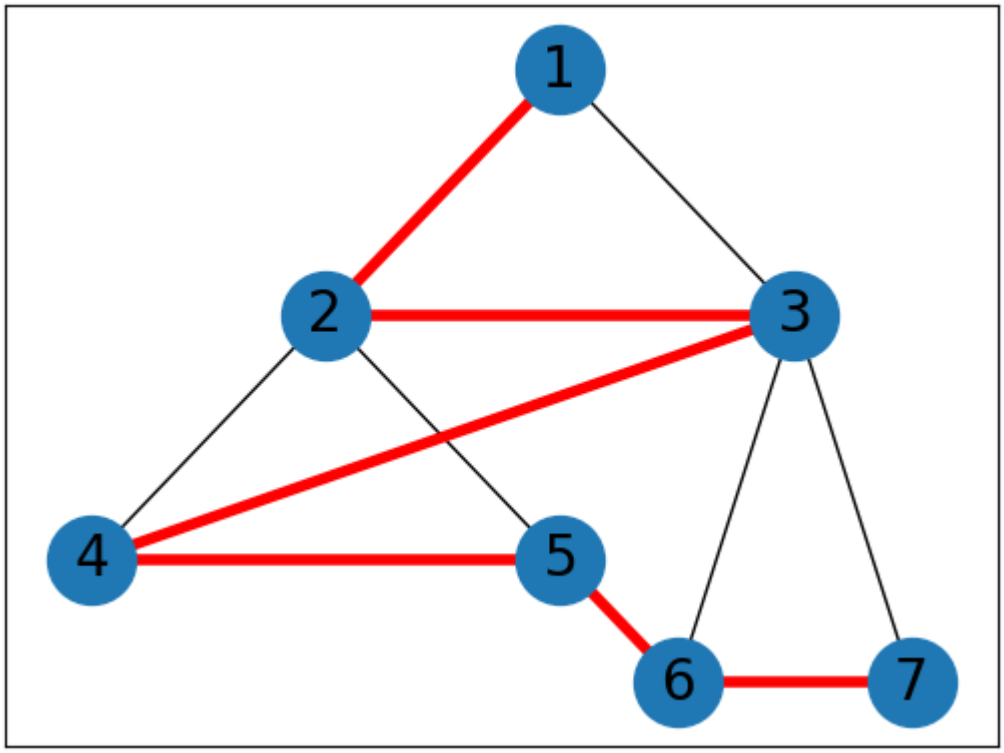
In [8]:
```
def visualize_graph(graph, tp=None):
    pos = {
        1: (0, 0),
        2: (-1, -1),
        3: (1, -1),
        4: (-2, -2),
        5: (0, -2),
        6: (0.5, -2.5),
        7: (1.5, -2.5)
    }
    nx.draw_networkx(graph, pos, with_labels=True, node_size=1000, font_size=20)

    if tp:
        edges = [(tp[i], tp[i+1]) for i in range(len(tp)-1)]
        nx.draw_networkx_edges(graph, pos, edgelist=edges, edge_color='r', width

    plt.show()
```

In [9]:
```
def input_graph():
    G = nx.Graph()
    num_edges = int(input("Enter the number of edges: "))
    print("Enter teh souce destination")
    for i in range(num_edges):
        edge = input().split()
        source, destination = int(edge[0]), int(edge[1])
        G.add_edge(source, destination)
    return G

g = input_graph()
visualize_graph(g)
sn = int(input("Enter the starting node : "))
print("BFS traversal:")
tp = bfs(g, sn)
print(tp)
visualize_graph(g, tp)
```

```
Enter teh souce destination
```




```
BFS traversal:
[1, 2, 3, 4, 5, 6, 7]
```



Depth first search

In [5]:

```python
import networkx as nx
import matplotlib.pyplot as plt
def dfs(graph, sn):
    visited = set()
    stack = [sn]
    tp = []

    while stack:
```

```
        node = stack.pop()
        if node not in visited:
            tp.append(node)
            visited.add(node)
            neighbors = graph.neighbors(node)
            for neighbor in neighbors:
                if neighbor not in visited:
                    stack.append(neighbor)
    return tp
```

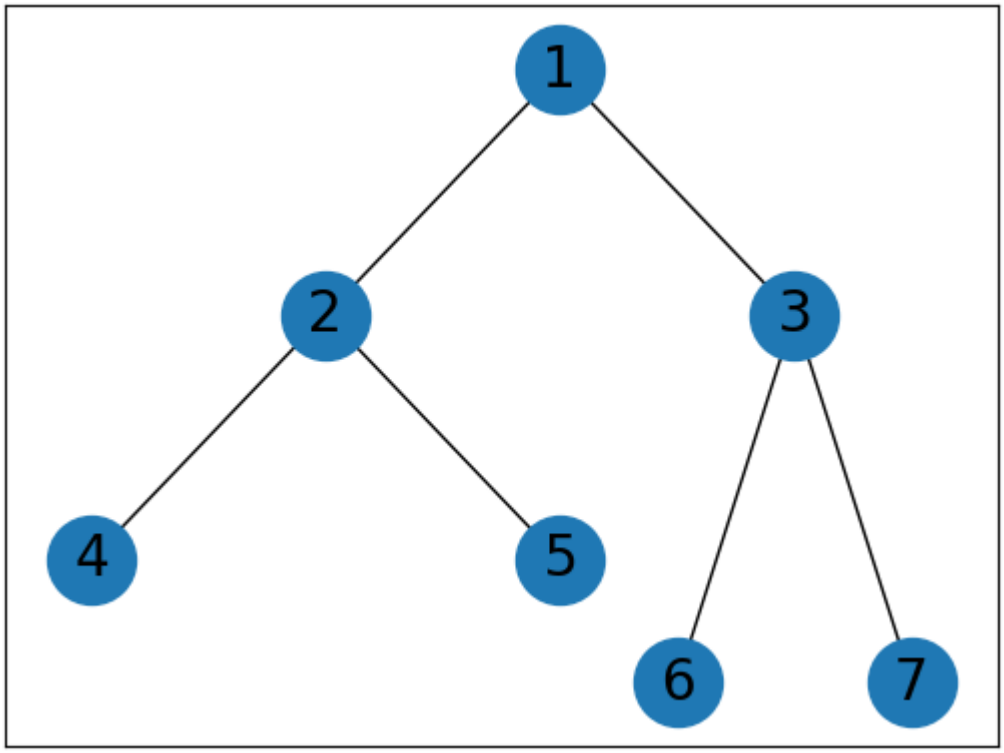
In [6]:

```
def visualize_graph(graph, tp=None):
    pos = {
        1: (0, 0),
        2: (-1, -1),
        3: (1, -1),
        4: (-2, -2),
        5: (0, -2),
        6: (0.5, -2.5),
        7: (1.5, -2.5)
    }
    nx.draw_networkx(graph, pos, with_labels=True, node_size=1000, font_size=20)

    if tp:
        edges = [(tp[i], tp[i+1]) for i in range(len(tp)-1)]
        nx.draw_networkx_edges(graph, pos, edgelist=edges, edge_color='r', width

    plt.show()
```

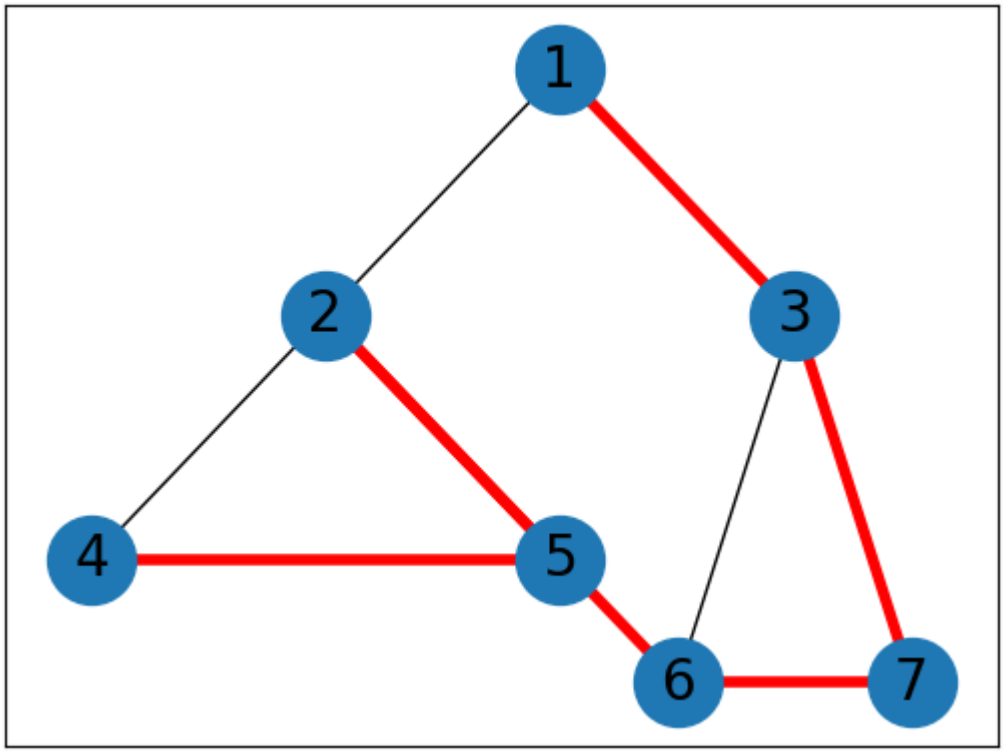
In [10]:

```
def input_graph():
    G = nx.Graph()
    num_edges = int(input("Enter the number of edges: "))
    print("Enter the source destination")
    for _ in range(num_edges):
        edge = input().split()
        source, destination = int(edge[0]), int(edge[1])
        G.add_edge(source, destination)
    return G

graph = input_graph()
visualize_graph(graph)

sn = int(input("Enter the starting node : "))
print("DFS traversal:")
tp = dfs(graph, sn)
print(tp)

visualize_graph(graph, tp)
```

```
Enter the source destination
```




```
DFS traversal:
[1, 3, 7, 6, 2, 5, 4]
```



depth limied search

In [13]:

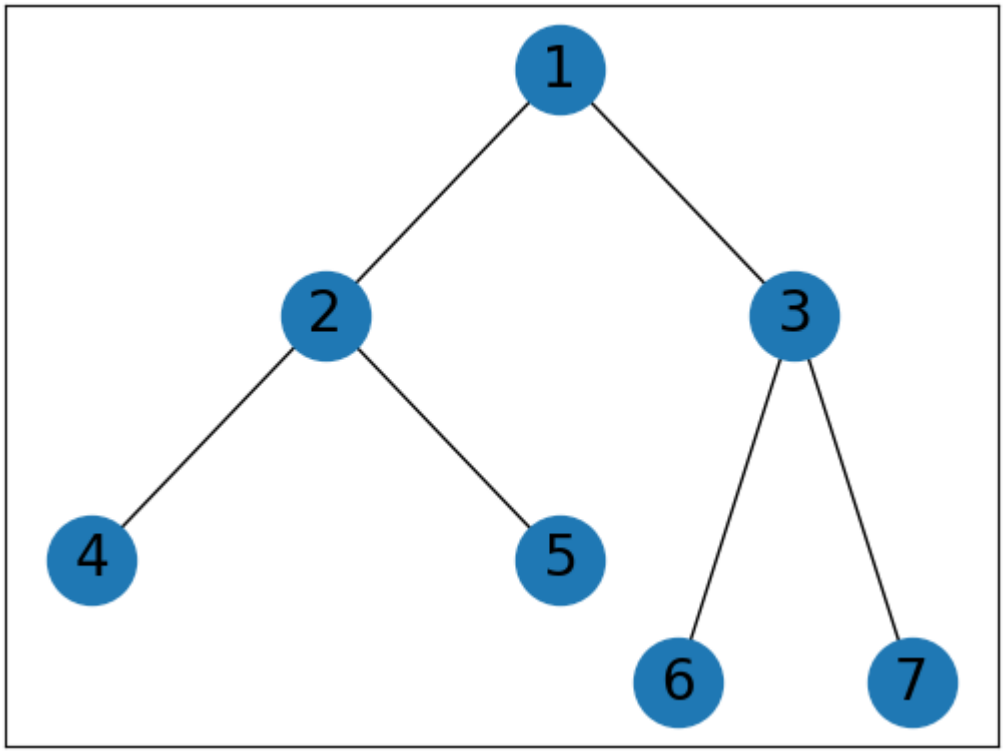
```python
import networkx as nx
import matplotlib.pyplot as plt

def dls(graph, sn, dept):
    visited = set()
    stack = [(sn, 0)]
    tp = []
```

```
    while stack:
        node, depth = stack.pop()
        if depth <= dept:
            if node not in visited:
                tp.append(node)
                visited.add(node)
                neighbors = graph.neighbors(node)
                for i in neighbors:
                    stack.append((i, depth + 1))
    return tp
```

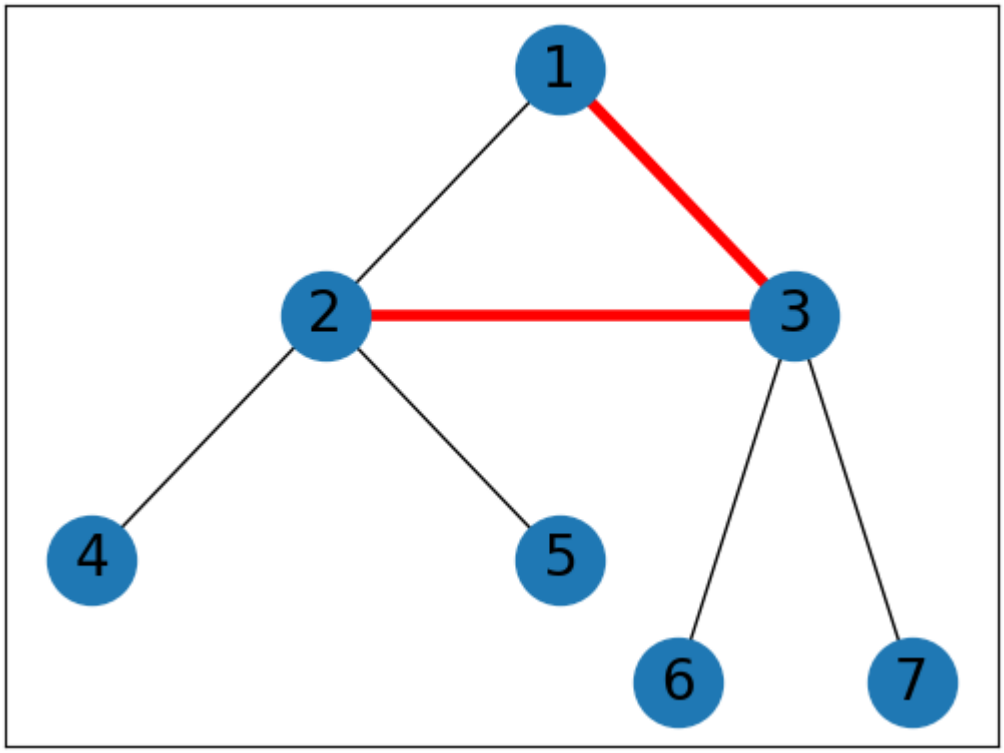
In [14]:
```
def visualize_graph(graph, tp=None):
    pos = {
        1: (0, 0),
        2: (-1, -1),
        3: (1, -1),
        4: (-2, -2),
        5: (0, -2),
        6: (0.5, -2.5),
        7: (1.5, -2.5)
    }
    nx.draw_networkx(graph, pos, with_labels=True, node_size=1000, font_size=20)

    if tp:
        edges = [(tp[i], tp[i+1]) for i in range(len(tp)-1)]
        nx.draw_networkx_edges(graph, pos, edgelist=edges, edge_color='red', wid

    plt.show()
```

In [16]:
```
def input_graph():
    G = nx.Graph()
    num_edges = int(input("Enter the number of edges: "))
    print("Enter the source destination")
    for _ in range(num_edges):
        edge = input().split()
        source, destination = int(edge[0]), int(edge[1])
        G.add_edge(source, destination)
    return G

graph = input_graph()
visualize_graph(graph)

sn = int(input("Enter the starting node : "))
dept = int(input("Enter the dept limit: "))
print("DFS traversal with depth limit:")
tp = dls(graph, sn, dept)
print(tp)
visualize_graph(graph, tp)
```

```
Enter the source destination
```




```
DFS traversal with depth limit:
[1, 3, 2]
```



In [ ]: