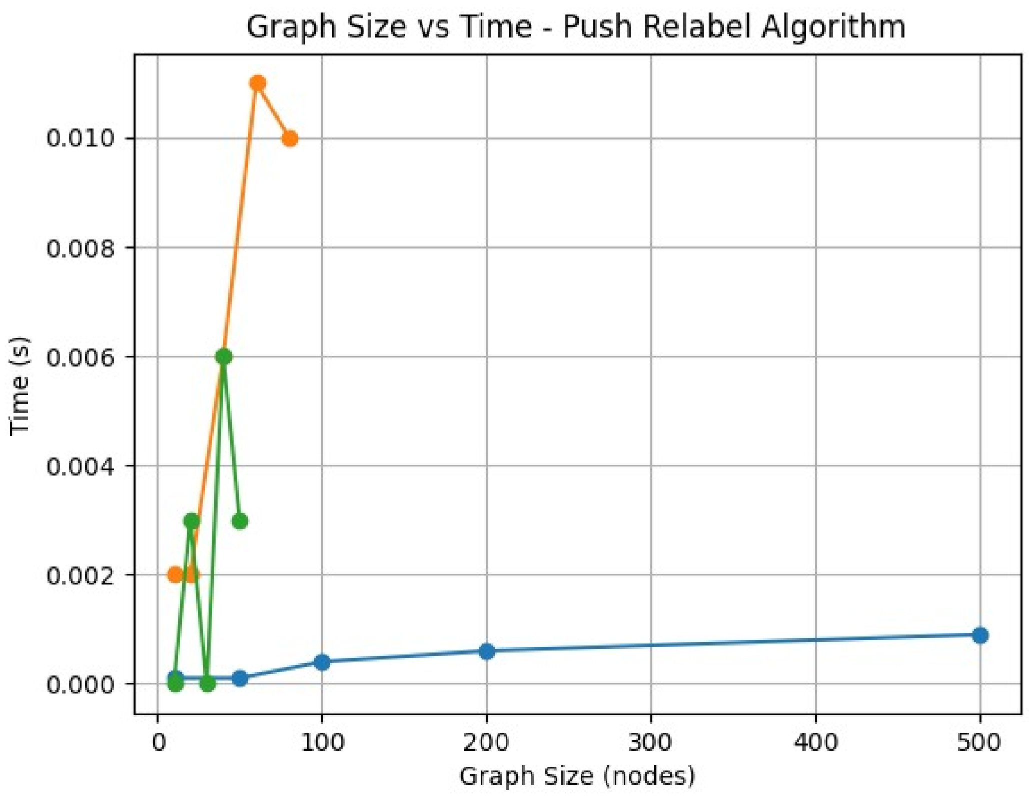
Practical Assignment No. 3 **Title:** Push Relabel Algorithm (Max Flow Problem)

```
import time
import random
import networkx as nx
import matplotlib.pyplot as plt

def push_relabel_max_flow(graph, source, sink):
    G = {u: {} for u in graph}
    for u in graph:
        for v in graph[u]:
            G[u][v] = graph[u][v]
            if v not in G:
                G[v] = {}
            if u not in G[v]:
                G[v][u] = 0  # Add reverse edge with 0 capacity

    n = len(G)
    height = {u: 0 for u in G}
    excess = {u: 0 for u in G}
    flow = {u: {v: 0 for v in G[u]} for u in G}

    height[source] = n
    for v in G[source]:
        flow[source][v] = G[source][v]
        flow[v][source] = -G[source][v]
        excess[v] = G[source][v]
        excess[source] -= G[source][v]

    def push(u, v):
        send = min(excess[u], G[u][v] - flow[u][v])
        if send > 0:
            flow[u][v] += send
            flow[v][u] -= send
            excess[u] -= send
            excess[v] += send

    def relabel(u):
        min_height = float('inf')
        for v in G[u]:
            if G[u][v] - flow[u][v] > 0:
                min_height = min(min_height, height[v])
        if min_height < float('inf'):
            height[u] = min_height + 1
```

```
def discharge(u):
        while excess[u] > 0:
            for v in G[u]:
                if G[u][v] - flow[u][v] > 0 and height[u] == height[v] + 1:
                    push(u, v)
                    if excess[u] == 0:
                        break
            else:
                relabel(u)

    vertices = [u for u in G if u != source and u != sink]
    p = 0
    while p < len(vertices):
        u = vertices[p]
        old_height = height[u]
        discharge(u)
        if height[u] > old_height:
            vertices.insert(0, vertices.pop(p))
            p = 0
        else:
            p += 1

    return sum(flow[source][v] for v in G[source])

def generate_graph(n, edge_factor=2):
    G = nx.gnm_random_graph(n, n * edge_factor, directed=True)
    graph = {}
    for u, v in G.edges():
        cap = random.randint(5, 20)
        if u not in graph:
            graph[u] = {}
        if v not in graph:
            graph[v] = {}
        graph[u][v] = cap
        if u not in graph[v]:
            graph[v][u] = 0  # Ensure reverse edge
    return graph

# Measure time
sizes = [10, 20, 30, 40, 50]
times = []
```

```
for size in sizes:
    g = generate_graph(size)
    source, sink = 0, size - 1
    start = time.time()
    max_flow = push_relabel_max_flow(g, source, sink)
    end = time.time()
    times.append(end - start)

# Plotting
plt.plot(sizes, times, marker='o')
plt.xlabel('Graph Size (nodes)')
plt.ylabel('Time (s)')
plt.title('Graph Size vs Time - Push Relabel Algorithm')
plt.grid(True)
plt.show()
```


Graph Size vs Time - Push Relabel Algorithm
Time (s)
Graph Size (nodes)