Practical Assignment No. 2 **Title:** Karger's Min-Cut Algorithm

```
import random
import copy
import time
import matplotlib.pyplot as plt

def karger_min_cut(graph):
    g = copy.deepcopy(graph)
    while len(g) > 2:
        u = random.choice(list(g.keys()))
        v = random.choice(g[u])

        # Merge v into u and remove v
        g[u].extend(g[v])
        for node in g[v]:
            g[node] = [u if x == v else x for x in g[node]]
        del g[v]
        g[u] = [x for x in g[u] if x != u]

    remaining_edges = list(g.values())[0]
    return len(remaining_edges)

# Create a sample random graph
def generate_graph(n, edges_per_node):
    graph = {i: [] for i in range(n)}
    edges = set()
    for i in range(n):
        while len(graph[i]) < edges_per_node:
            j = random.randint(0, n - 1)
            if i != j and (i, j) not in edges and (j, i) not in edges:
                graph[i].append(j)
                graph[j].append(i)
                edges.add((i, j))
    return graph

# Testing and plotting
sizes = [10, 20, 40, 60, 80]
times = []

for size in sizes:
    graph = generate_graph(size, 3)
    start = time.time()
    min_cut = min([karger_min_cut(graph) for _ in range(5)])  # Run multiple times
    end = time.time()
    times.append(end - start)
```

```
# Plotting the result
plt.plot(sizes, times, marker='o')
plt.xlabel('Graph Size (nodes)')
plt.ylabel('Time (s)')
plt.title('Graph Size vs Time for Karger’s Min-Cut Algorithm')
plt.grid(True)
plt.show()
```