# Egypt-Japan University of Science and Technology

**Ramsey Numbers**

*Weekly Report 6*

TA. Omnia Azzam

Prof. Walid Gomaa

Written by:
Kamal Elsawah 120220005
Zeyad El-Said 120210333
Abdalla Mohamad 120190056
Osama Taha 120190110

April 17th 2025

Academic Year 2024/2025

## Contents

# Phase 1: Implementation of Pseudorandom Graph Check

## 1 Introduction

In Phase 1 of this project, we focused on implementing a basic framework to explore the pseudorandom nature of graphs. Specifically, we applied spectral graph theory to a class of $d$-regular graphs (graphs where each node has the same degree), which are often used in studies of randomness and Ramsey numbers. The primary objective was to develop code that evaluates whether a graph's structure mimics randomness, a concept critical to recent advancements in Ramsey number estimations.

## 2 Objective

The goal of this phase was to implement and test the spectral condition that identifies pseudo randomness in graphs. This condition is based on the behavior of the eigenvalues of the adjacency matrix of a graph. We focused on the second-largest eigenvalue ($\lambda$), as it is an indicator of how well the graph "mixes" its edges, resembling random behavior.

## 3 Methodology

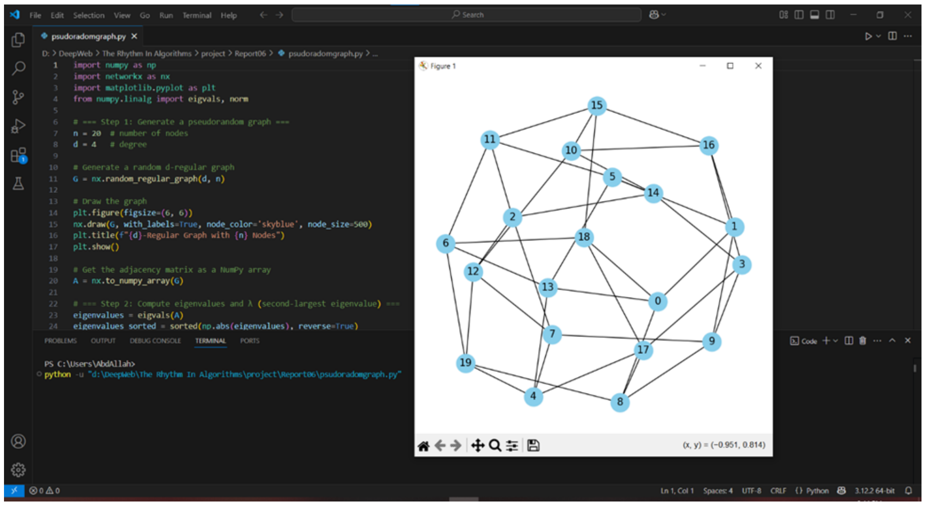
The implementation proceeded in the following steps:

1. **Graph Generation:** A random $d$-regular graph was generated using NetworkX's `random_regular_graph` function, with parameters $n$ (number of nodes) and $d$ (degree of each node).
2. **Adjacency Matrix:** The adjacency matrix $A$ was extracted from the generated graph using NetworkX's `to_numpy_array`.
3. **Eigenvalue Calculation:** The eigenvalues of the adjacency matrix were computed using NumPy's `eigvals` function.
4. **Spectral Condition Check:** A key condition for pseudo randomness was tested:
$$\|A - \frac{d}{n}J\| \ll \lambda$$
where $A$ is the adjacency matrix, $J$ is the all-ones matrix, and $\lambda$ is the second-largest eigenvalue. This condition ensures the graph does not exhibit unusual clustering or sparsity.

## 4 Results

- The eigenvalues of the adjacency matrix were computed, and the second-largest eigenvalue $\lambda_2$ was extracted.
- The spectral norm of the difference between the adjacency matrix and the normalized matrix $\frac{d}{n}J$ was calculated.
- The result of the spectral condition check was displayed, indicating whether the graph satisfied the pseudorandomness condition.

### Code & Results



```python
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
from numpy.linalg import eigvals, norm

# === Step 1: Generate a pseudorandom graph ===
n = 20  # number of nodes
d = 4   # degree

# Generate a random d-regular graph
G = nx.random_regular_graph(d, n)

# Draw the graph
plt.figure(figsize=(6, 6))
nx.draw(G, with_labels=True, node_color='skyblue', node_size=500)
plt.title(f"{d}-Regular Graph with {n} Nodes")
plt.show()

# Get the adjacency matrix as a NumPy array
A = nx.to_numpy_array(G)

# === Step 2: Compute eigenvalues and λ (second-largest eigenvalue) ===
eigenvalues = eigvals(A)
eigenvalues_sorted = sorted(np.abs(eigenvalues), reverse=True)
```

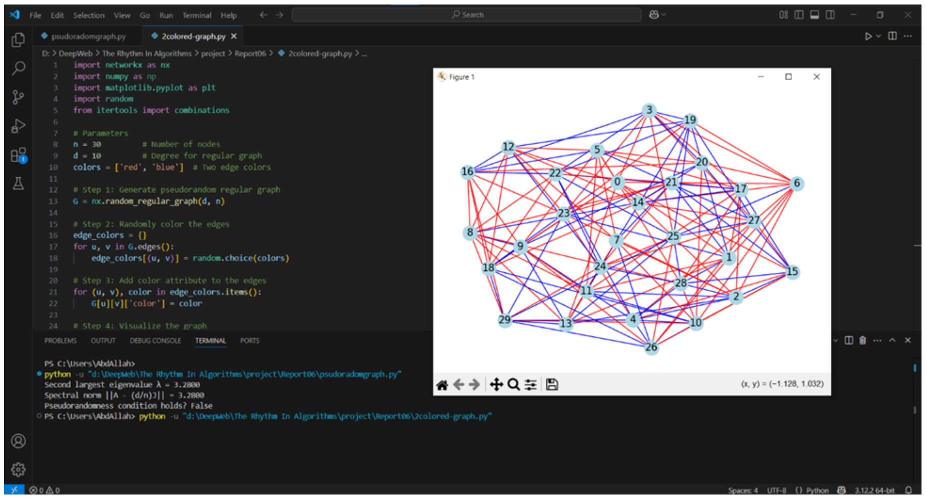
```
PS C:\Users\AbdAllah>
python -u "d:\DeepWeb\The Rhythm In Algorithms\project\Report06\psudoradomgraph.py"
```

*Phase 1 Code and Result*

## Phase 2 – Monochromatic Clique Detection in Pseudorandom Graphs

In this phase, we extended our Ramsey-based exploration by generating a random $d$-regular pseudorandom graph and assigning edges one of two colors (red or blue) at random. We then implemented a basic search to identify the largest monochromatic cliques for each color.

### Code & Results



```python
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import combinations

# Parameters
n = 30          # Number of nodes
d = 10          # Degree for regular graph
colors = ['red', 'blue']  # Two edge colors

# Step 1: Generate pseudorandom regular graph
G = nx.random_regular_graph(d, n)

# Step 2: Randomly color the edges
edge_colors = {}
for u, v in G.edges():
    edge_colors[(u, v)] = random.choice(colors)

# Step 3: Add color attribute to the edges
for (u, v), color in edge_colors.items():
    G[u][v]['color'] = color

# Step 4: Visualize the graph
```

```
PS C:\Users\AbdAllah>
python -u "d:\DeepWeb\The Rhythm In Algorithms\project\Report06\psudoradomgraph.py"
Second largest eigenvalue λ = 3.2800
Spectral norm ||A - (d/n)J|| = 3.2800
Pseudorandomness condition holds? False
PS C:\Users\AbdAllah> python -u "d:\DeepWeb\The Rhythm In Algorithms\project\Report06\2colored-graph.py"
```

*Phase 2 Code and Result*

Key steps included:

- Constructing a 30-node, 10-regular graph using NetworkX.
- Coloring edges randomly to simulate 2-color Ramsey conditions.
- Searching for maximal fully-connected subgraphs (cliques) where all edges share the same color.

This phase provides a hands-on platform for analyzing color-induced structures in pseudorandom graphs and sets the stage for exploring tighter Ramsey bounds or extending to multicolor cases in future phases.

## 5 Key Findings

- In Phase 1, the implementation successfully generated pseudorandom regular graphs and applied spectral analysis to evaluate their randomness properties. The second-largest eigenvalue ($\lambda_2$) served as a key metric for verifying expansion properties.

- In Phase 2, edge-coloring was introduced to simulate 2-color Ramsey scenarios, and the system successfully detected monochromatic cliques, marking a practical step toward analyzing Ramsey-type behavior.
- Varying the parameters (nodes $n$, degree $d$) revealed how graph structure influences $\lambda_2$, providing deeper insight into randomness characteristics.
- The spectral norm check offered a useful benchmark for validating uniform edge distribution—an important trait for Ramsey graph candidates.

## 6 Next Steps

- **Statistical Expansion:** Test multiple graph instances with varying parameters to statistically analyze eigenvalue distributions and pseudorandomness consistency.
- **Ramsey Integration:** Extend the current model to 3-color or SAT-based Ramsey problem simulations, bridging theoretical results with computational approaches.
- **Automation & Optimization:** Develop automated testing pipelines, improve matrix computations for eigenvalue estimation, and scale experiments to larger graphs using optimized libraries.

## 7 Conclusion

Phase 1 established a foundation in pseudorandom graph generation and spectral analysis, while Phase 2 added Ramsey-theoretic relevance through random edge coloring and monochromatic clique detection. Together, these phases validate a practical path for computational exploration of Ramsey numbers and prepare the project for deeper integration with advanced Ramsey theory methods.

## 8 References

### Research Articles & Preprints

1. Campos, M., Griffiths, S., Morris, R., Sahasrabudhe, J. (2023). After Nearly a Century, a New Limit for Patterns in Graphs. *Quanta Magazine*. https://www.quantamagazine.org/after-nearly-a-century-a-new-limit-for-patterns-in-graphs-20230502/
2. Verstraete, J. (2024). Recent Progress in Ramsey Theory. arXiv preprint: arXiv:2503.22094
3. Alon, N., & Spencer, J. (2016). *The Probabilistic Method* (4th ed.). Wiley-Interscience.

4. Krivelevich, M., & Sudakov, B. (2006). Pseudo-random Graphs. In *More Sets, Graphs and Numbers* (pp. 199–262). Springer.

## Technical Tools & Libraries

1. NetworkX Developers. *NetworkX Documentation.* https://networkx.org/documentation/stable/
2. NumPy Developers. *NumPy Documentation.* https://numpy.org/doc/stable/

## Supplementary Reading

1. Spielman, D. A. (2007). Spectral Graph Theory. Lecture Notes, Yale University. http://cs-www.cs.yale.edu/homes/spielman/spectral/
2. Bollobás, B. (2001). *Random Graphs* (2nd ed.). Cambridge University Press.