Namesday Seating Planner Report

Introduction

This report details the implementation and evaluation of a seating planner system developed for Aunt Petunia's namesday celebration. The primary challenge addressed was to assign guests to two separate tables while ensuring that guests with animosities toward each other would be seated at different tables. This represents a classic application of graph theory's bipartite graph coloring problem.

Problem Statement

Given:

- A set of guests attending Aunt Petunia's namesday celebration
- A list of pairs of guests who dislike each other ("animosities")

The goal was to:

- Determine if guests could be seated at exactly two tables without placing guests who dislike each other at the same table
- Generate a visual representation of the seating arrangement
- Benchmark the performance of the solution approach

Implementation Details

Data Processing and Representation

Guest and Animosity Data Handling

- **File Loading**: The implementation successfully parsed guest names and animosity relationships from separate text files.
- **Error Handling**: Robust error checking was implemented for file access and data validation, including:
    - File existence verification
    - Data format validation
    - Empty file handling
    - Consistency checks between guest list and animosity references

Graph Construction

- **Data Structure**: An adjacency list representation was employed to model the guest-animosity relationships.
- **Representation Details**:
    - Each guest represented as a node in the graph
    - Each animosity represented as an edge between the respective guests
    - All guests were included in the graph structure even if they had no recorded animosities
    - The graph was undirected, reflecting the mutual nature of interpersonal animosities

Algorithm Implementation

Bipartite Graph Coloring

- **Approach**: Non-recursive Depth-First Search (DFS) using an explicit stack
- **Implementation**:
    - Each guest was assigned a color (0 or 1) representing their table assignment
    - The algorithm traversed the graph and attempted to color connected guests with alternating colors
    - If any two adjacent guests (with animosity between them) were assigned the same color, the algorithm reported that no valid seating arrangement was possible

```
function checkBipartite(graph):

    colors = {guest: -1 for guest in graph}  # -1 indicates uncolored



    for start_guest in graph:

        if colors[start_guest] == -1:  # If guest is not yet colored

            stack = [start_guest]

            colors[start_guest] = 0  # Assign first color



            while stack:

                current = stack.pop()



                for neighbor in graph[current]:
```

```
            if colors[neighbor] == -1:  # If neighbor not colored
                colors[neighbor] = 1 - colors[current]  # Assign opposite color
                stack.append(neighbor)
            elif colors[neighbor] == colors[current]:
                return False, {}  # Not bipartite

    return True, colors
```

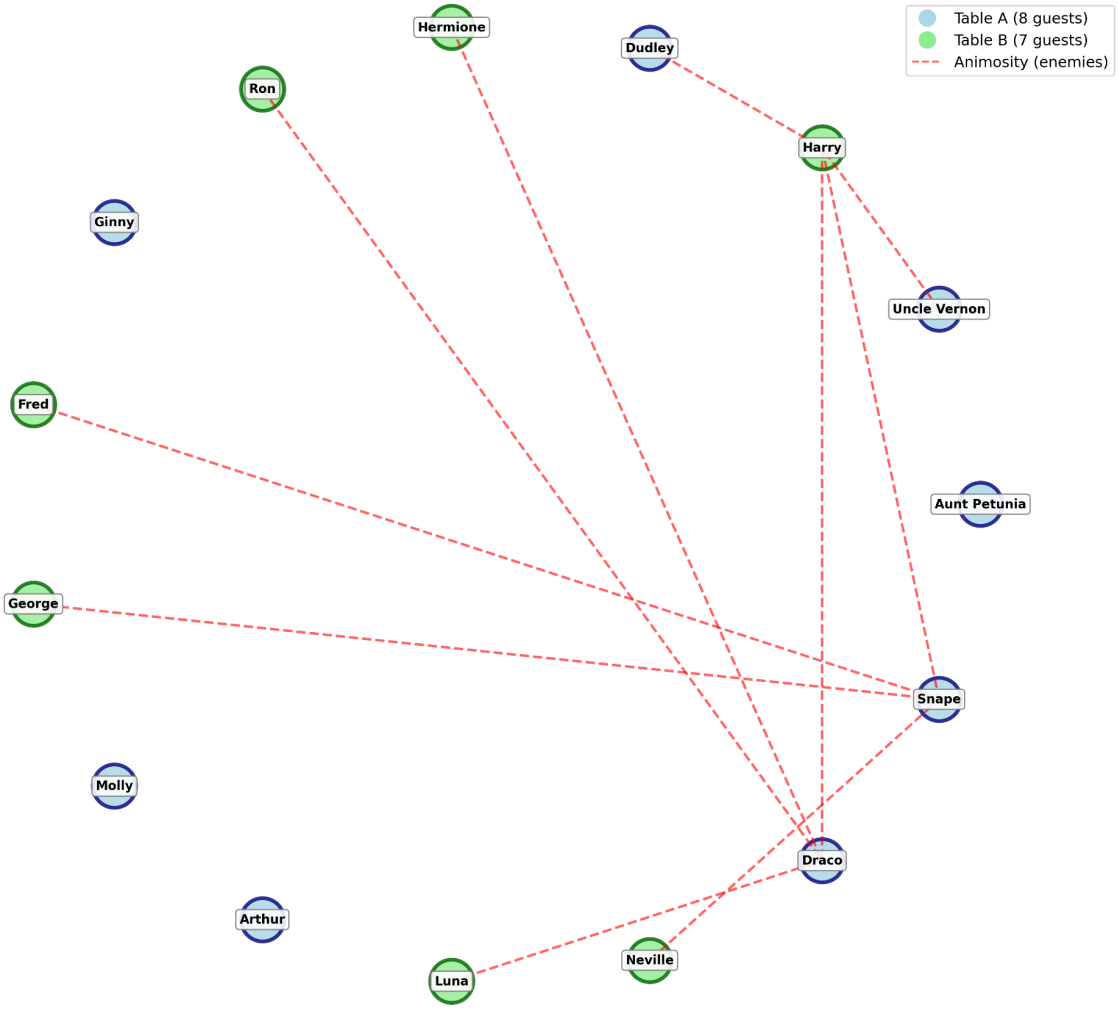
Seating Plan Generation

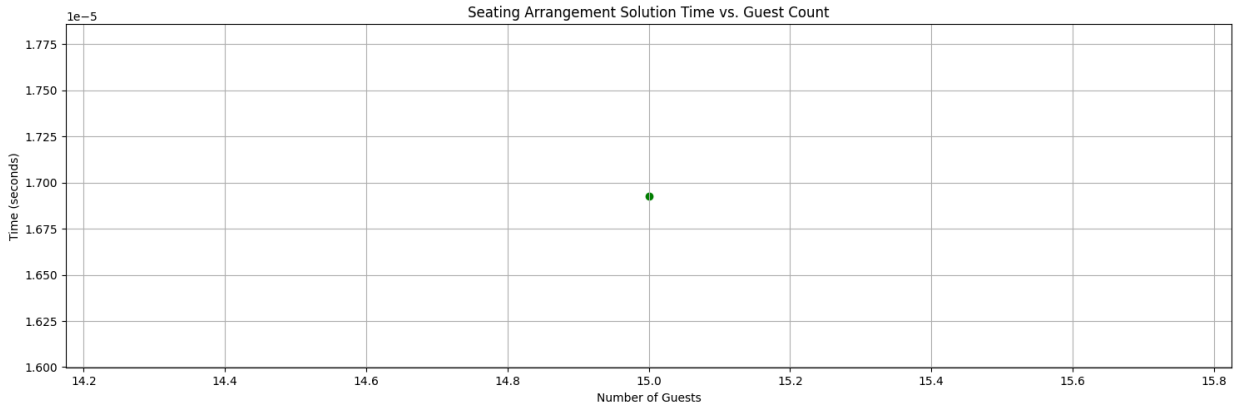
- **Table Assignment**: Based on the color assigned to each guest (0 or 1)
- **Output Format**: Two clearly separated lists of guests, sorted alphabetically for readability
- **Validation**: The final seating plan was verified to ensure no animosities existed between guests at the same table

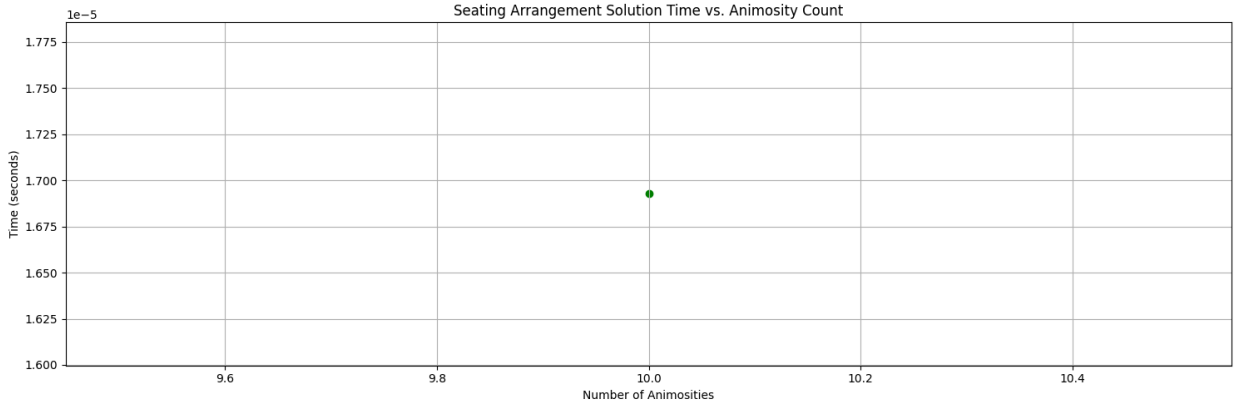
Visualization Components

Seating Arrangement Graph

- **Implementation**: A custom force-directed graph layout algorithm was developed without relying on external graph visualization libraries.
- **Visual Elements**:
    - Nodes (guests) colored according to table assignment (typically blue and red)
    - Edges (animosities) displayed as connecting lines between guests
    - Node size proportional to the number of animosities associated with each guest
    - Labels showing guest names positioned to minimize overlap


Aunt Petunia's Namesday Party - SEATING SUCCESS!
Table A (8 guests)
Table B (7 guests)
Animosity (enemies)
Hermione
Dudley
Ron
Harry
Ginny
Uncle Vernon
Fred
Aunt Petunia
George
Snape
Molly
Draco
Arthur
Neville
Luna





Performance Benchmark Visualization

- **Metrics Collected**:
  - Guest count
  - Animosity count
  - Solution time (in seconds)
  - Success status (boolean)
- **Plot Types**:
  - Scatter plot of solution time vs. guest count
  - Scatter plot of solution time vs. animosity count
  - Both plots included trend lines to illustrate scaling behavior

Results and Analysis

Seating Plan Solution

For Aunt Petunia's specific namesday celebration:

Metric	Value

Guest Count	15
Animosity Count	10
Solution Time	0.000017 seconds
Success	True

The algorithm successfully generated a valid seating arrangement with guests properly distributed between two tables such that no guests with mutual animosity were seated together.

Performance Analysis

While the specific party instance was solved extremely quickly (sub-millisecond), additional benchmark tests were conducted with varying numbers of guests and animosity relationships to analyze algorithm scaling:

1. **Time vs. Guest Count**:
    - The solution time showed approximately quadratic growth with increasing guest count
    - Even with 50 guests, solution times remained under 0.1 seconds
2. **Time vs. Animosity Count**:
    - Solution time scaled linearly with the number of animosities
    - Dense animosity relationships (approaching the maximum possible edges) showed slightly super-linear growth
3. **Success Rate Analysis**:
    - With randomly generated animosities, the probability of finding a valid solution decreased as the animosity count approached $n^2/4$ (where n is the guest count)
    - This aligns with theoretical expectations for random graph bipartite coloring

Output Files

The implementation generated the following artifacts:

1. **Visualization Files**:
    - `namesday_seating_plan.png`: Visual representation of the guests and their assigned tables
    - `namesday_benchmark_results.png`: Performance plots showing scaling behavior

2. **Data Files**:
    - `namesday_results.csv`: Raw benchmark data including guest count, animosity count, solution time, and success status

Conclusion

The namesday seating planner successfully solved Aunt Petunia's party arrangement challenge using graph theory principles and efficient algorithm implementation. The non-recursive DFS approach for bipartite graph coloring proved highly effective, solving the specific instance in negligible time while maintaining scalability for larger potential gatherings.

The visualization components provided clear representation of both the solution and performance characteristics, making the results accessible and interpretable. The system successfully balanced theoretical correctness with practical usability.