Examining the Stability Versus Efficiency of Algorithms Related to the Stable Marriage Problem

Design of Algorithms 001 Radian

By Maddison Barnwell, Cydney Miller, Esraa Marfing, David Oluwadara Owotomo

Overview

Stable Marriage Problem

The Stable Marriage Problem is concerned with pairing N men and N women, each having ranked all members of the opposite sex by preference. The goal is to form marriages such that no two individuals would prefer each other over their assigned partners.

Gale Shapley

The Gale Shapley algorithm is a deterministic algorithm that is used to match men and women into pairs based on their preferences. It guarantees stable matching and runs in $O(n^2)$ time in the worst case. Each man proposes to the women of his preference list in order, if the woman accepts his proposal, they become semi-engaged. If a man she prefers more proposes while she is already semi-engaged, she can break the engagement and switch to the more preferred partner. If she rejects the proposal, the man continues his list. This follows a greedy, iterative improvement method, ensuring that all participants are matched in a stable pair.

Irving's Algorithm

Irving's Algorithm is used to solve the Stable Roommates Problem. It is a deterministic iterative reduction algorithm which has a time complexity of $O(n^2)$. The Stable Roommate Problem does not always contain a stable pair, but Irving's Algorithm can determine when no matches exist. This algorithm proceeds in two phases. Phase 1: algorithm eliminates impossible pairings by removing candidates from each person's preference list and keeping the possible candidates. Phase 2: algorithm identifies and eliminates rotations (cyclic preference patterns). If, after all eliminations, each person has exactly one remaining partner, a stable matching exists; otherwise, the algorithm concludes that no stable matching is possible.

Randomized Serial Dictatorship (RSD) Algorithm

The RSD algorithm is a randomized greedy mechanism used for one-sided matching, where only one of the sets involved has a preference list, and is commonly demonstrated with an analogy of agents to houses, where the agents have a preference of which house they obtain, but the houses do not have a preference over which agent it receives. It becomes apparent, then, that this algorithm is a poor solution to the Stable Marriage Problem because it does not account for the women's preferences in constructing its

marriages. So, for this experiment, we are using this algorithm strictly to compare running time efficiency and applications of similar problems, and we acknowledge that this algorithm would not be used to solve the Stable Marriage Problem in practice.

Objective/Research Question

Determine how the stability guarantees and asymptotic time complexities differ between the Gale Shapley algorithm for the Stable Marriage Problem, Irving's Algorithm for the Stable Roommates Problem, and the Randomized Serial Dictatorship algorithm, and what trade-offs emerge between deterministic stability and computational efficiency? Comparing the methods highlights when, how, and why each one would be implemented for different circumstances or how superior one method is to another one. It allows a baseline to be established and may empower optimization of code by combining different aspects of each method to create a better hybridized version or specialized method.

Description of Experiment

Implementation Details

This experiment was designed and carried out using the free version of Google Colab which uses Python, version 3.12.12 as its current standard programming language, verified in later section titled *Hardware Characteristics*. The libraries used throughout the various functions are as follows:

1. random
2. typing (specifically Dict, List, Union)
3. heapq
4. time
5. pandas as pd
6. matplotlib.pyplot as plt
7. seaborn as sn
8. platform
9. subprocess

Code scaffolding and documentation suggestions were partially assisted by generative AI tools[1], however, countless revisions were made to adjust algorithms to experiment logic, verify correctness, and all final implementation decisions were performed manually.

Input Description

We had 6 distinct input types for this experiment:

1. **Hand-Created Preference Lists for Gale Shapley and RSD-based algorithms**

- The number of *n* men and women in this case was strictly 3.
- Used for manual checking and making sure that algorithms performed as expected.
- Artificial human-generated data.

2. **Hand-Created Preference Lists for Irving's Algorithm**
   - The number of *n* participants in this case was strictly 4.
   - Used for manual checking and making sure that the algorithm could successfully produce a stable matching when such a matching exists.
   - Artificial human-generated data.
3. **Randomly Generated Preference Lists for Gale Shapley and RSD-based algorithms**
   - Used to randomly populate preference lists for a given *n* women and women passed in by the user.
   - Artificial computer-generated random data.
4. **Randomly Generated Preference Lists for Irving's Algorithm**
   - Used to randomly populate preference lists for a given *n* participants passed in by the user.
   - Artificial computer-generated random data.
5. **Worst-Case Generated Preference Lists for Gale Shapley and RSD-based algorithms**
   - Used to populate preference lists that result in worst-case running time for a given *n* men and women.
   - Generated using a deterministic pattern known to induce maximal proposal chain length for GS and maximal scanning/heap operations for RSD.
   - Artificial computer-generated data.
6. **Worst-Case Generated Preference Lists for Irving's Algorithm**
   - Used to populate preference lists that result in worst-case running time for a given *n* participants.
   - Artificial computer-generated data.

Experimentation Setup

The Colab cell titled "Data Collection for Performance Chart" serves as the primary data collection module for performance analysis. It prompts the user for a maximum input size (max_n), subsequently generating a series of discrete input sizes (N) across a defined range (10 steps up to max_n). For example, with max_n = 100, we generated N = {10, 20,...,100}. For each N value, the cell performs the following:

1. **Preference Data Generation**:
    - **Random**: Using generate_random_preferences (for Gale Shapley and RSD) and generate_irving_preferences (for Irving's Algorithm), creating diverse and uniformly distributed preferences.
    - **Worst-Case**:
      Employing generate_worst_case and generate_irving_worst_case_preferences to construct scenarios designed to elicit maximal computational effort from the algorithms.
2. **Algorithm Execution and Timing**: Each of the four algorithms—Gale Shapley, Randomized Serial Dictatorship (RSD), RSD with Heaps, and Irving's Algorithm—is executed against both the random and worst-case preference sets. Execution times are precisely measured using time.perf_counter() and recorded in microseconds.
3. **Data Structuring**: All collected metrics (input size N, scenario type, algorithm name, and execution time) are compiled into a Pandas DataFrame (df_results). Df_results contains the following columns:
    - 'N': (int),
    - 'Scenario': (‘Worst-Case’ or ‘Random’),
    - 'Algorithm': ('Gale Shapley', ‘RSD’, ‘RSD-Heaps’, ‘Irving’s’),
    - 'Time (microseconds)': (float)

The df_results DataFrame is the resulting dataset, structured for subsequent quantitative analysis and visualization in performance comparison charts, thereby facilitating the empirical evaluation of algorithmic efficiency under varied conditions.

The input sizes for the hand-created samples were fixed, but for our other input types we chose four different maximum input sizes to be tested for each algorithm: 50, 100, 500, 1000. These input values were selected to test the running time of our selected algorithms as input grows large but seeing as our worst-case running time is $O(n^2 \log n)^2$ and we are running on a limited virtual machine (Google Colab), we did not want to increase too far beyond reasonable limits to prevent very long computation times. It is important to note that Irving’s Algorithm operates on N participants, where N must be even. If a generated N was odd, we incremented it by 1 (N → N+1) only for the Irving trial, ensuring valid input without affecting the other algorithms.

Since our data is randomly generated, we need to run each input size more than once to get an accurate average, so for each N and each scenario type (random/worst-case), we executed each algorithm twice and recorded both runtimes in microseconds.

Additionally, the data that was generated using the "Data Collection for Performance Chart" cell as described above was fed into a cell that plots the runtime for each algorithm for the randomly generated inputs as well as the worst-case generated inputs.

Hardware Characteristics

All parts of this experiment were executed in Google Colab, which runs code on a remote virtual machine. The environment provided the following hardware configuration when running the experiment on November 18th, 2025, at 7:30 P.M. CST:

- CPU: Intel(R) Xeon(R) @ 2.20GHz
- RAM: ~13.3 GB
- Operating System: Ubuntu 22.04.4 LTS (Jammy Jellyfish)

Because Google Colab dynamically allocates hardware, exact specifications may vary between sessions. If running notebook on your own device, use the cell titled "Get Hardware Characteristics" to get the specific configuration at time of running.

[1] For full details on the AI model that were used see items [1], [3], [6] in *Works Cited*.

[2] The worst-case running time calculated for Random Serialized Dictatorship with Heaps.

Results

Hand-Created Inputs

Trial One:

- Gale Shapley: 15.3 ms
- Random Serial Dictatorship: 26.1 ms
- Random Serial Dictatorship with Heaps: 17.5 ms
- Irving's Algorithm: 18.8 ms

Trial Two:

- Gale Shapley: 14.8 ms
- Random Serial Dictatorship: 22.4 ms
- Random Serial Dictatorship with Heaps: 15.5 ms
- Irving's Algorithm: 17.5 ms

Computer-Generated Inputs

Trial One:

- Input Size 50: See *Table 1, Figure 1*

- Input Size 100: See *Table 3, Figure 3*
- Input Size 500: See *Table 5, Figure 5*
- Input Size 1000: See *Table 7, Figure 7*

Trial Two:

- Input Size 50: See *Table 2, Figure 2*
- Input Size 100: See *Table 4, Figure 4*
- Input Size 500: See *Table 6, Figure 6*
- Input Size 1000: See *Table 8, Figure 8*

Interpretation

Our experiment yielded behavior aligned with our expectations, but there were some unexpected findings in the empirical analysis of the runtimes for our algorithms. In terms of our research question, which aims to discover how algorithm stability affects the computational efficiency of algorithms which solve the Stable Marriage Problem and its related problems, we found that algorithms that preserve stable matchings (Gale Shapley and Irving's Algorithm), tended to have the greatest runtimes, while unstable algorithms, particularly the randomized approaches, were able find a solution very quickly (See *Figure 9* and *Figure 10*). This behavior was consistent with our expectations before we began experimenting with our chosen algorithms. What was unexpected, however, was the disparity between the runtimes of our algorithms with the worst-case inputs relative to our calculated time complexities for each algorithm.

Consistently, the Gale Shapley algorithm had the greatest running time for worst-case inputs among all other algorithms (See *Figure 10*), despite having the same theoretical time complexity as Irving's Algorithm and the Randomized Serial Dictatorship algorithm. Additionally, the implementation of the Randomized Serial Dictatorship algorithm using heaps was expected to perform faster with the addition of a heap structure, but it ended up slowing the algorithm down in practice. This slowed behavior can be understood after calculating the theoretical time complexity but was still a surprising finding when investigating the algorithms in practice.

Our experiment did have some limitations that should be addressed, and there are some changes that we would make if we were to redesign and rerun this experiment. Firstly, and most importantly, our algorithms are not all designed to solve the same problem. The Gale Shapley algorithm is the standard and most widely used solution to the Stable Marriage Problem, but Irving's Algorithm and the Randomized Serial Dictatorship (RSD) algorithm each solve their own related, but distinctly different problem. Because of this, we cannot say that our research is only with regards to the Stable Marriage Problem, but in fact

involves the Stable Roommates Problem (Irving's Algorithm) and the problem of assigning agents to houses (RSD). In another round of exploration, we may be more exacting with which algorithms we decide to experiment with, instead of opening the door to many different problems. As a result of using algorithms that solve different problems, our data sets could not be applied universally for all algorithms we were testing. Irving's Algorithm raised this issue, and separate data sets had to be generated for it each time. This made comparison between methods a bit more challenging because their differences must be explicitly stated and considered when examining the results of the experiment. Another small experimental design choice that we would change for this experiment was the decision to measure the runtimes of our algorithms in microseconds. This was decided in the earlier stages of designing our experiment when we were testing small input sizes, but we quickly discovered that runtimes would have been more meaningful if they were measured in seconds instead. Finally, our experiment was executed entirely in Google Colab's virtual environment, which comes with its own computational limitations, which are listed above in *Hardware Characteristics*.

For further research on this topic, we could run many more trials for each input size (perhaps 5) and run the experiment several times (perhaps 10 times) to find the normal distribution and standard deviation that represents the results of our experiment. This would allow us to say with greater confidence whether or not the results of our experiment are reproduceable and have a greater understanding of what to expect from the experiment to identify unusual behavior. Additionally, we would have liked to explore how we could use either one of, or a combination of our algorithms for some practical implementation. This could range anywhere from collecting and using real data instead of artificial data, to creating a tangible project that uses these algorithms to perform some task. We could also explore more in-depth how the Gale Shapley algorithm can be adjusted to handle preference lists that contain preference ties or incomplete lists and how such changes would affect stability and efficiency of the algorithm.

Overall, our experiment has some shortcomings that could be refined for further testing, but it ultimately succeeds in answering our research question:

"Determine how the stability guarantees and asymptotic time complexities differ between the Gale–Shapley algorithm for the Stable Marriage Problem, Irving's algorithm for the Stable Roommates Problem, and the Randomized Serial Dictatorship algorithm, and what trade-offs emerge between deterministic stability and computational efficiency?"

Gale Shapely and Irving's Algorithm are both stable algorithms that run in $O(n^2)$ time, whereas RSD (and RSD with heaps) are unstable with $O(n^2)$ and $O(n^2 \log n)$ running times, respectively. From our empirical analysis, we see a clear time trade-off for algorithms that guarantee stable matchings, especially in worst-case input scenarios.

Obstacles Encountered

The foremost barrier in this project was finding appropriate times to meet due to conflicting schedules, which we overcame via partial meetings and meeting summaries being sent to absent members. Using this method, the absent members were able to accomplish the objectives given to them at the meetings and communicate any questions or issues that they had while completing their tasks. Besides scheduling, the project went relatively smoothly except for a few hiccups, such as not being able to export graphs from Google Colab for the data, causing us to manually screenshot the graphs and input the data into custom tables. Another problem was that the three algorithms could not accept the exact same input, so the inputs had to be modified for each run of the experiment and then compared with those differences in mind. Those minor issues increased the time spent on the project but did not cause significant trouble with its completion.

Works Cited

[1] Anthropic. Claude, Sonnet 4.5, Anthropic, 2025. Accessed 17 Nov. 2025; Used to write sample data generation functions.

[2] Austin, David. "The Stable Marriage Problem and School Choice." *American Mathematical Society*, March 2015, ams.org. Accessed 13 Nov. 2025.

[3] Google. Google Gemini, Gemini 2.5 Flash, Google, 2025. Accessed 17 Nov. 2025; Used for debugging and chart generation.

[4] Irving, Robert W. “An Efficient Algorithm for the ‘Stable Roommates’ Problem.” *Journal of Algorithms*, Volume 6, Issue 4, December 1985, Pages 577-595, sciencedirect.com. Accessed 12 Nov. 2025.

[5] Kun, Jeremy. "Serial Dictatorships and House Allocation." *Math ∩ Programming*, October 2015, jeremykun.com. Accessed 13 Nov. 2025.

[6] OpenAI. ChatGPT, version 5.1, OpenAI, 2025. Accessed 12 Nov. 2025; Used to generate algorithms.

Annex

An important note for all tables: Tables only reflect the runtime for the maximum input size.

Table 1
Trial One Input Size 50

N	Scenario	Algorithm	Time (Microseconds)
50	Random	Gale Shapley	1166.437
50	Random	RSD	89.504
50	Random	RSD Heaps	1751.769
50	Random	Irving's	1375.502
50	Worst-Case	Gale Shapley	7510.798
50	Worst-Case	RSD	179.319
50	Worst-Case	RSD Heaps	1982.512
50	Worst-Case	Irving's	904.758

Table 2
Trial Two Input Size 50

N	Scenario	Algorithm	Time (Microseconds)
50	Random	Gale Shapley	2790.376
50	Random	RSD	88.110
50	Random	RSD Heaps	1465.616
50	Random	Irving's	1295.849
50	Worst-Case	Gale Shapley	7986.669
50	Worst-Case	RSD	257.494
50	Worst-Case	RSD Heaps	2328.705
50	Worst-Case	Irving's	1048.446

Table 3
Trial One Input Size 100

N	Scenario	Algorithm	Time (Microseconds)
100	Random	Gale Shapley	6638.945
100	Random	RSD	117.973
100	Random	RSD Heaps	3639.181
100	Random	Irving's	4411.511
100	Worst-Case	Gale Shapley	29073.434
100	Worst-Case	RSD	492.954
100	Worst-Case	RSD Heaps	7584.171
100	Worst-Case	Irving's	4027.348

Table 4
Trial Two Input Size 100

N	Scenario	Algorithm	Time (Microseconds)
100	Random	Gale Shapley	5314.195
100	Random	RSD	141.300
100	Random	RSD Heaps	3916.666
100	Random	Irving's	6335.825
100	Worst-Case	Gale Shapley	32303.438
100	Worst-Case	RSD	495.572
100	Worst-Case	RSD Heaps	5802.072
100	Worst-Case	Irving's	3969.069

Table 5
Trial One Input Size 500

N	Scenario	Algorithm	Time (Microseconds)
500	Random	Gale Shapley	180399.332
500	Random	RSD	725.460
500	Random	RSD Heaps	76398.328
500	Random	Irving's	446762.259
500	Worst-Case	Gale Shapley	3554985.508
500	Worst-Case	RSD	8939.546
500	Worst-Case	RSD Heaps	243081.147
500	Worst-Case	Irving's	751977.077

Table 6
Trial Two Input Size 500

N	Scenario	Algorithm	Time (Microseconds)
500	Random	Gale Shapley	158769.301
500	Random	RSD	1028.323
500	Random	RSD Heaps	267376.437
500	Random	Irving's	745282.950
500	Worst-Case	Gale Shapley	3597140.727
500	Worst-Case	RSD	5391.648
500	Worst-Case	RSD Heaps	136736.064
500	Worst-Case	Irving's	395954.194

Table 7
Trial One Input Size 1000

N	Scenario	Algorithm	Time (Microseconds)
1000	Random	Gale Shapley	6.510133e+05
1000	Random	RSD	1.869495e+03
1000	Random	RSD Heaps	3.049748e+05
1000	Random	Irving's	4.487172e+06
1000	Worst-Case	Gale Shapley	2.939497e+07
1000	Worst-Case	RSD	2.491481e+04
1000	Worst-Case	RSD Heaps	5.650260e+05
1000	Worst-Case	Irving's	4.162644e+06

Table 8
Trial Two Input Size 1000

N	Scenario	Algorithm	Time (Microseconds)
1000	Random	Gale Shapley	1.052032e+06
1000	Random	RSD	2.222065e+03
1000	Random	RSD Heaps	4.849887e+05
1000	Random	Irving's	3.915931e+06
1000	Worst-Case	Gale Shapley	2.932051e+07
1000	Worst-Case	RSD	2.428830e+04
1000	Worst-Case	RSD Heaps	5.691563e+05
1000	Worst-Case	Irving's	4.763814e+06

Figure 1

Trial One Input Size 50

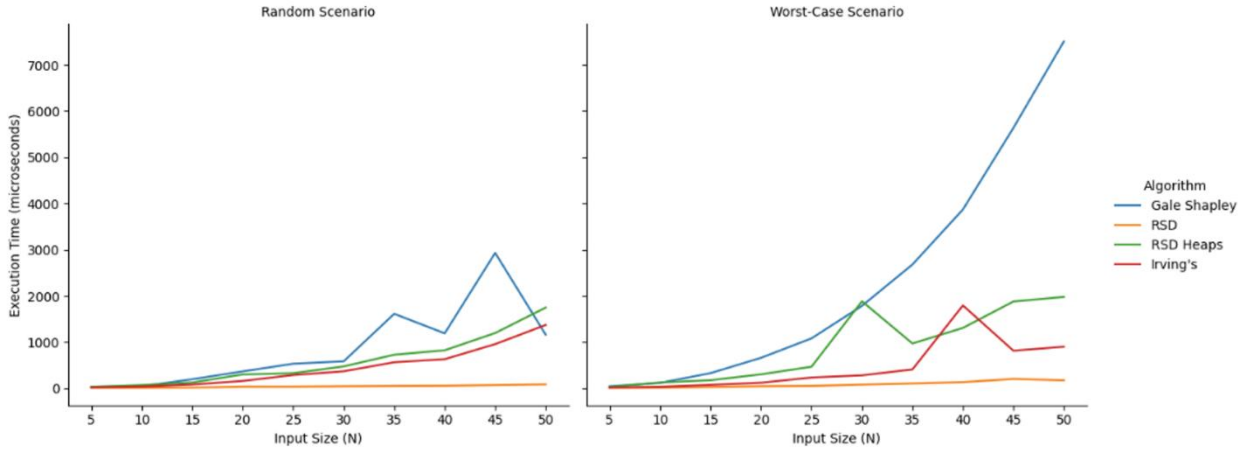


Figure 2

Trial Two Input Size 50

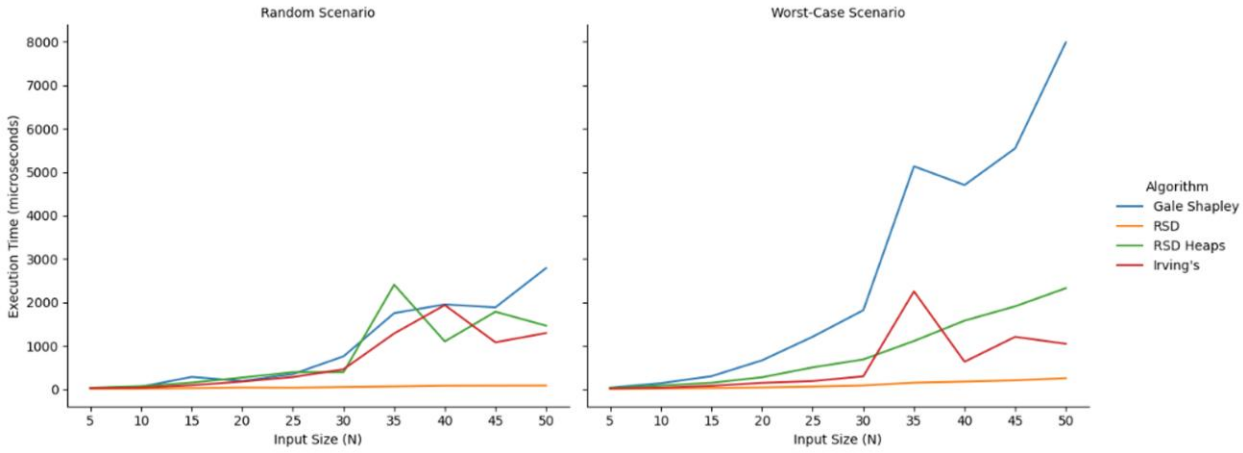


Figure 3

Trial One Input Size 100

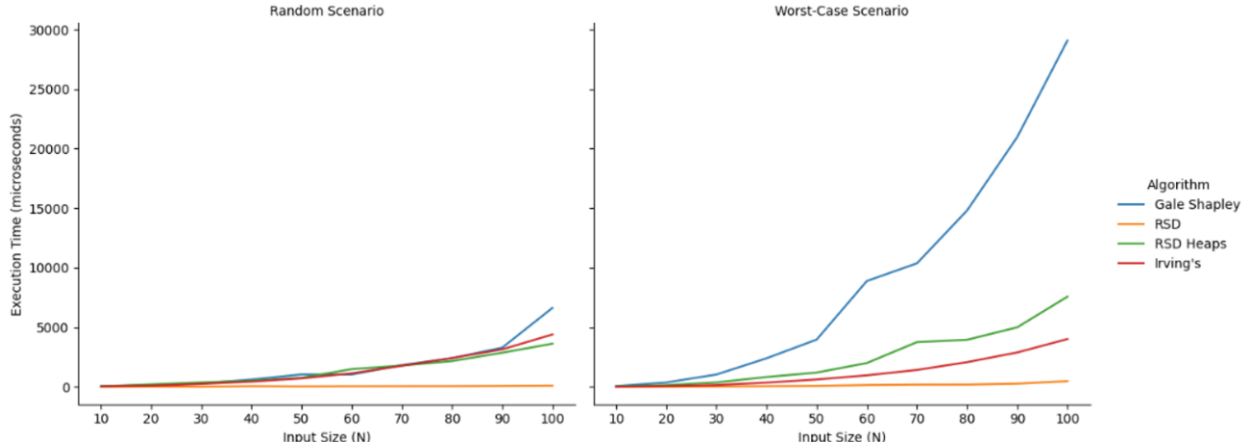


Figure 4

Trial Two Input Size 100

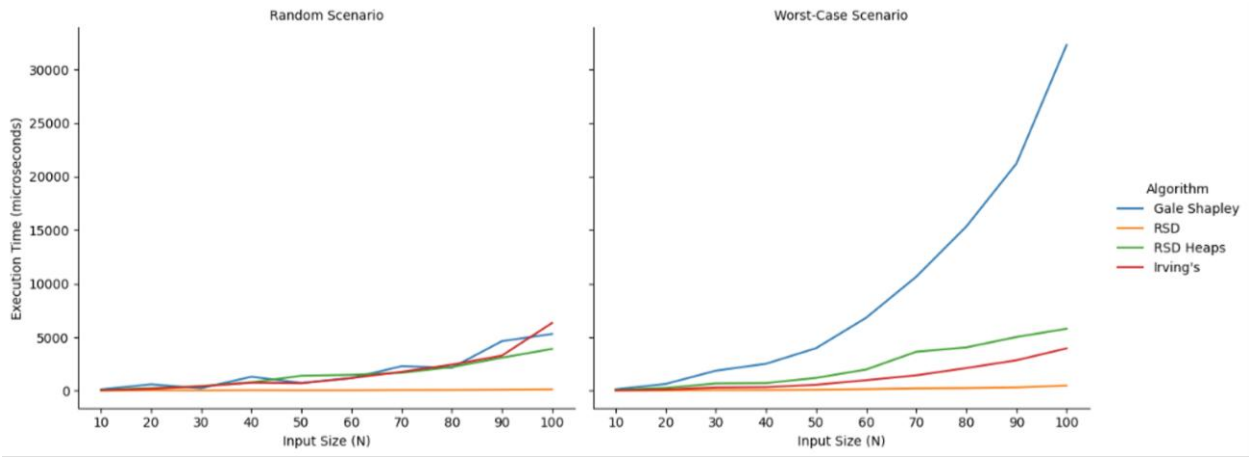


Figure 5

Trial One Input Size 500

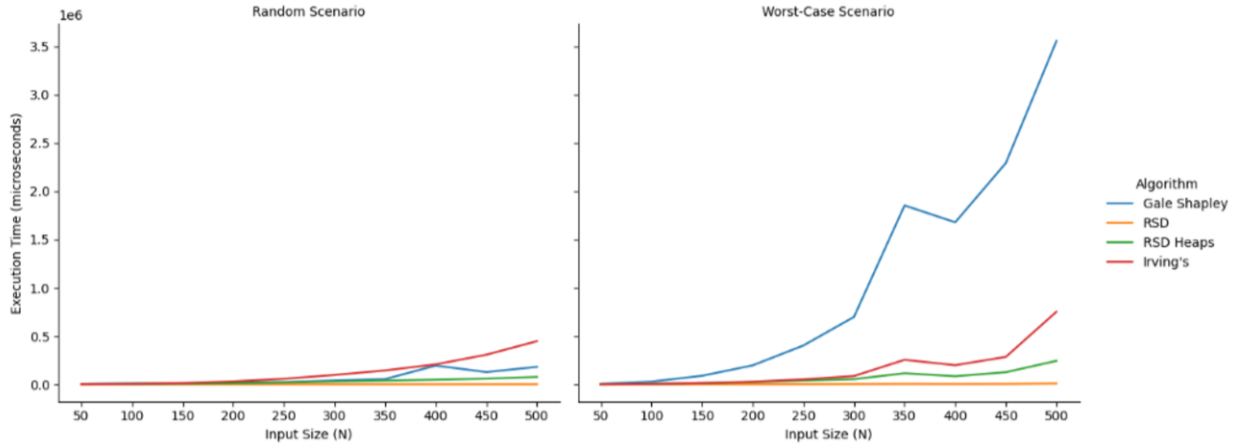


Figure 6

Trial Two Input Size 500

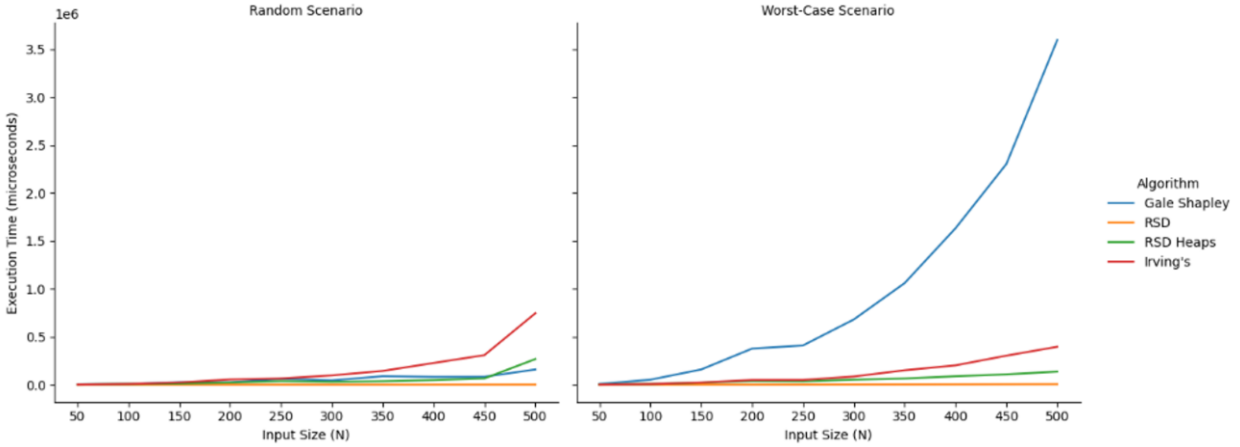


Figure 7

Trial One Input Size 1000

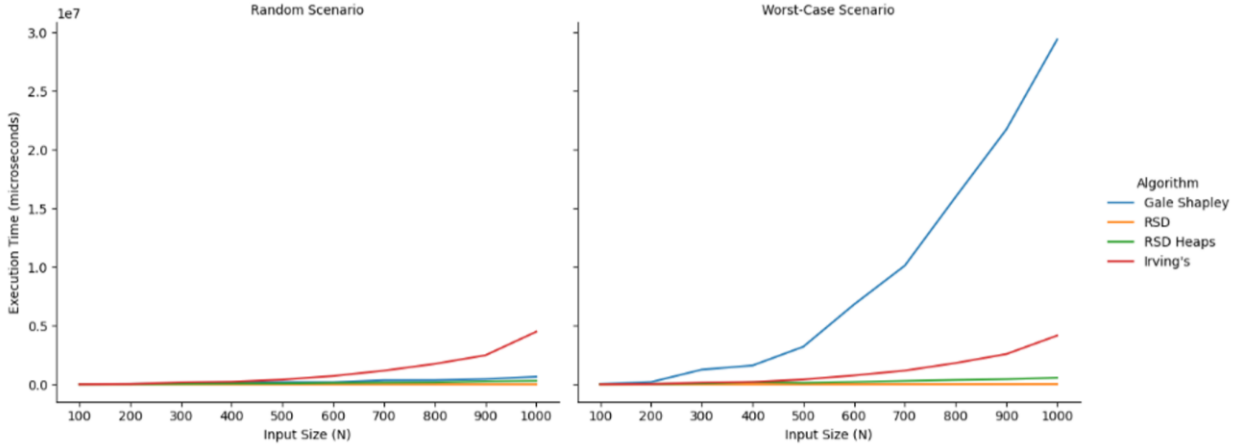


Figure 8

Trial Two Input Size 1000

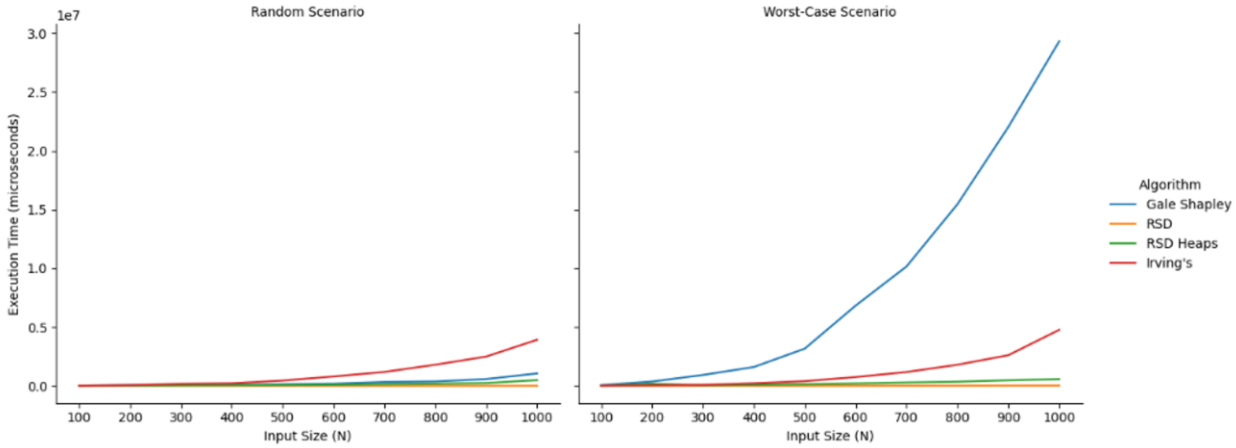


Figure 9

Average of All Trials for Random Data

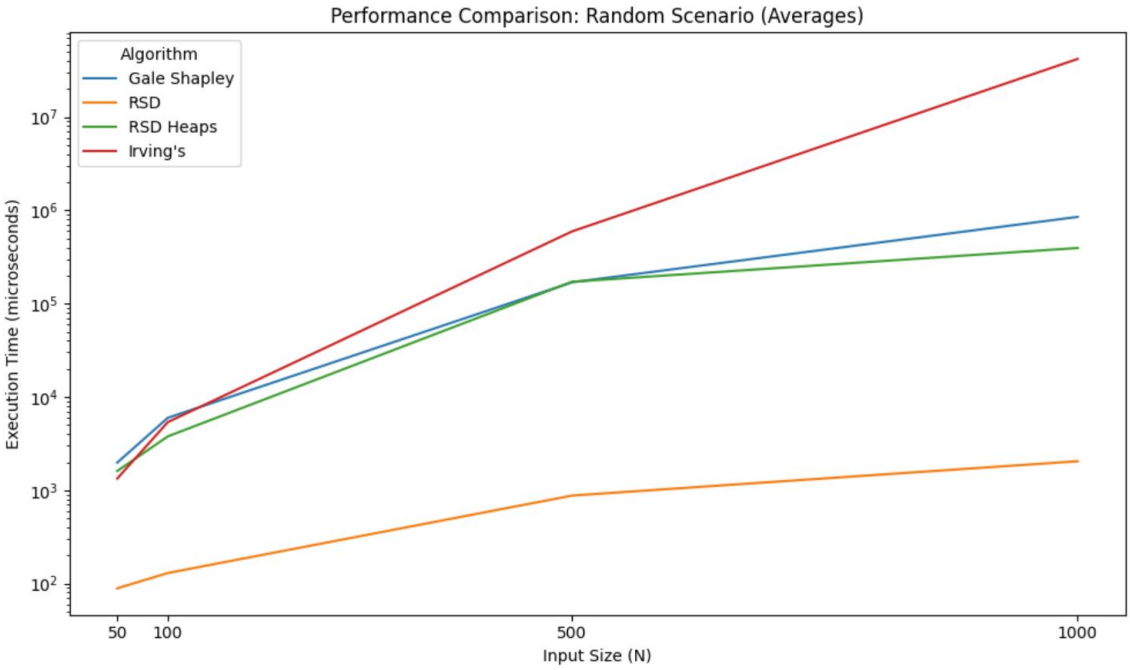
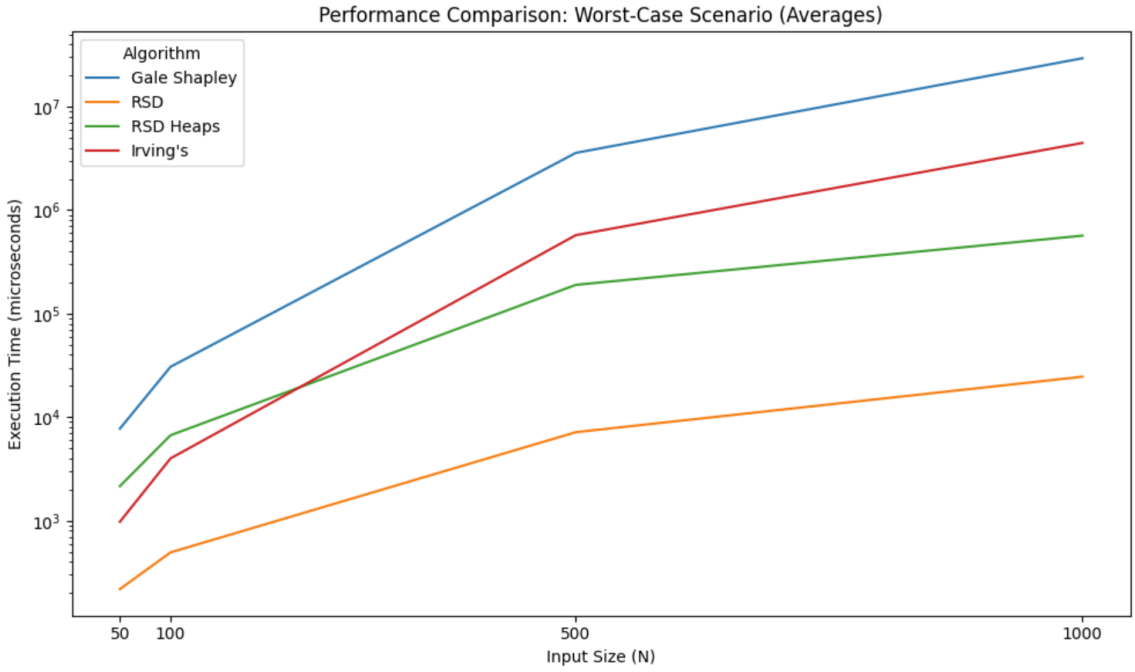


Figure 10

Average of All Trials for Worst-Case Data



Work Breakdown Table

Brainstorming		
Cyd	Organized important documents and notes	25%
Maddie	Generate data and confirming it	25%
Esraa	Research questions and rough draft of report	25%
David	Additional research and visual of the algorithms or related concepts	25%
Experiment Design/Coding		
Cyd	Coded literally everything with the help of generative AI	90%
Maddie	Focused on other items	0%
Esraa	Focused on other items	0%
David	Generated the test data	10%
Report Writing		
Cyd	Wrote sections description of experiment, results, annex, and interpretation	30%
Maddie	Wrote the objective, obstacles encountered, and the work breakdown table	20%
Esraa	Wrote the introduction in addition to the summary of the three algorithms and stable marriage problem	30%
David	Helped Cyd on her sections	20%
Visualization/Results		
Cyd	Generated the graphs and interpreted and generated the results	40%
Maddie	Found visuals online	15%
Esraa	Found visuals online	15%
David	Found visuals online and helped Cyd interpret and generate the results	30%
Communication		
Cyd	Set up meetings and sent meeting summaries	40%
Maddie	Helped divide up the work and ensured communication of labor	20%
Esraa	Attended meetings and responded to questions/issues that arose	20%

David	Attended meetings and responded to questions/issues that arose	20%
Submission		
Cyd	Submitted the report and presentation	25%
Maddie	Submitted the report and presentation	25%
Esraa	Submitted the report and presentation	25%
David	Submitted the report and presentation	25%
Presentation		
Cyd	Edited the presentation and will speak (Q&A and help David with experiment design/coding)	25%
Maddie	Organized the presentation and will speak (interpretation)	25%
Esraa	Edited the presentation and will speak (overview)	25%
David	Edited the presentation and will speak (experiment design/coding)	25%