Analyzing the Efficiency and Accuracy of Randomized Algorithms

A Project by Divide & ConquAAD

Introduction & Motivation

- Deterministic algorithms always follow one path
- Randomized algorithms explore many possible paths and can show:
  - Faster runtime
  - Simpler implementation
  - Avoids worst cases
  - Foundation of practical systems like cryptography
- This project explores how randomness can help reduce runtime, improve average performance, and trade off between speed and accuracy.
- By implementing and testing several well-known randomized algorithms, we aim to identify runtime patterns, measure accuracy, and analyze their relation with input size and number of runs.

Algorithms Studied

- Miller-Rabin Primality Test
- Randomized Quick Sort
- Kruskal’s Min Cut
- Monte Carlo’s Algorithm for finding π

Randomized Quick Sort

Variants analyzed:

- Standard Quick Sort
- Randomized Quick Sort
- Dual Pivot Quick Sort

Standard Quick Sort

- Select last element as pivot
- All elements $<$ pivot are moved to the left side and all elements $\geq$ pivot are moved to the right side. (Lomuto Partition)
- Recursively apply quick sort to the left and right sub array.
- Complexity:
  - Best/Average: $O(n \log n)$
  - Worst: $O(n^2)$ (sorted input)
  - Space: $O(\log n)$

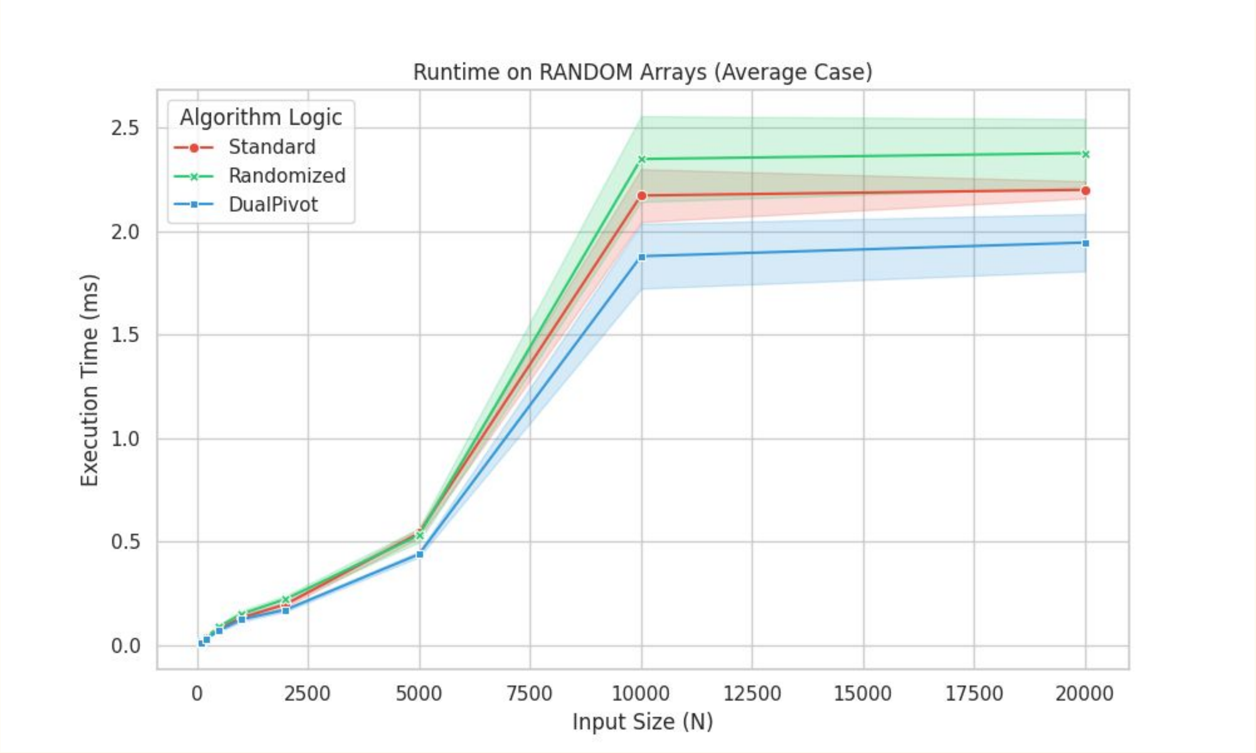
Randomized Quick Sort

- Randomly select pivot as: random = low + rand() % (high - low + 1);
- Swap with last element
- Same partitioning afterwards
- Complexity:
    - Best/Average: $O(n \log n)$
    - Worst: $O(n^2)$ (but very unlikely)
    - Space: $O(\log n)$

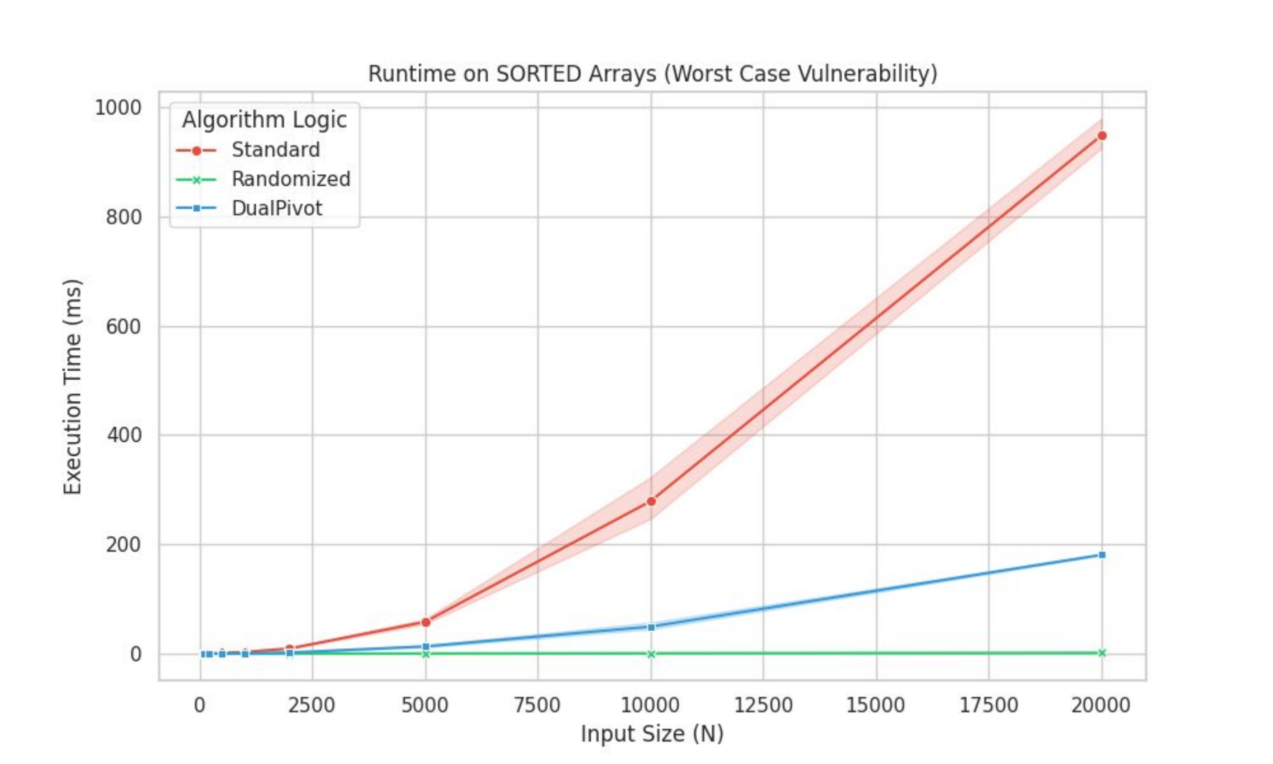
Dual Pivot Quick Sort

- Uses two pivots: left and right, dividing into three sub arrays
- Faster in practice, used in Java too
- Complexity:
  - Best/Average: O(n log n)
  - Space: O(log n)
  - Usually faster than Standard Quick Sort

Quick Sort: Results on random input



Quick Sort: Results on sorted input



Primality Tests

Fermat Test:

- Based on Fermat's Little Theorem: $a^{p-1} \equiv 1 \pmod{p}$
- Very fast, but can be broken by Carmichael numbers
- Carmichael number: A composite number that satisfies the condition of Fermat's Little Theorem for all bases relatively prime to it.
- Examples: 561 (3 x 11 x 17), 1105 (5 x 13 x 17), 1729 (7 x 13 x 19)

Miller-Rabin Algorithm

- Take n and express as $n - 1 = (d \times 2^r)$
- Compute sequence: $a^d, a^{2d}, a^{4d}, \ldots, a^{2^{\wedge}(r-1)*d} \pmod{n}$
- If n is prime, the sequence must end in 1, and the element immediately preceding the first 1 must be —1. If we see a 1 preceded by something else, n is composite.
- This is based on the lemma that if n is prime, the only solutions to $x^2 \equiv 1 \pmod{n}$ are $x \equiv 1$ and $x \equiv -1$.

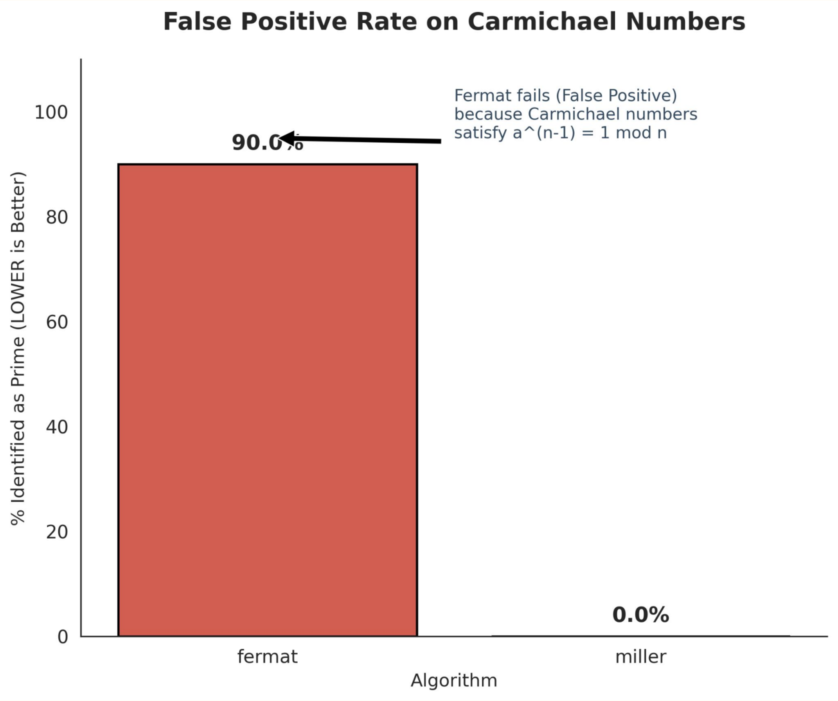
Complexity of Miller-Rabin

Dominated by modular exponentiation

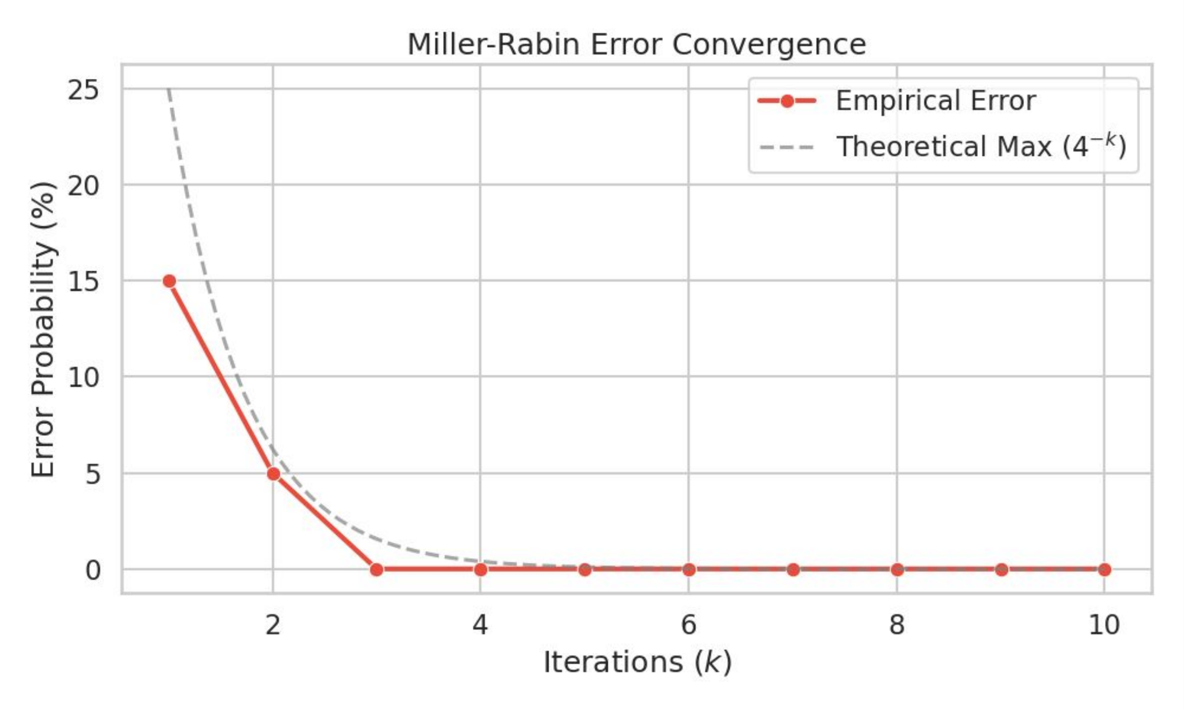
Time Complexity: $O(k \log^3 n)$

Space Complexity: $O(\log n)$

Results: Comparison of algorithms over Carmichael numbers

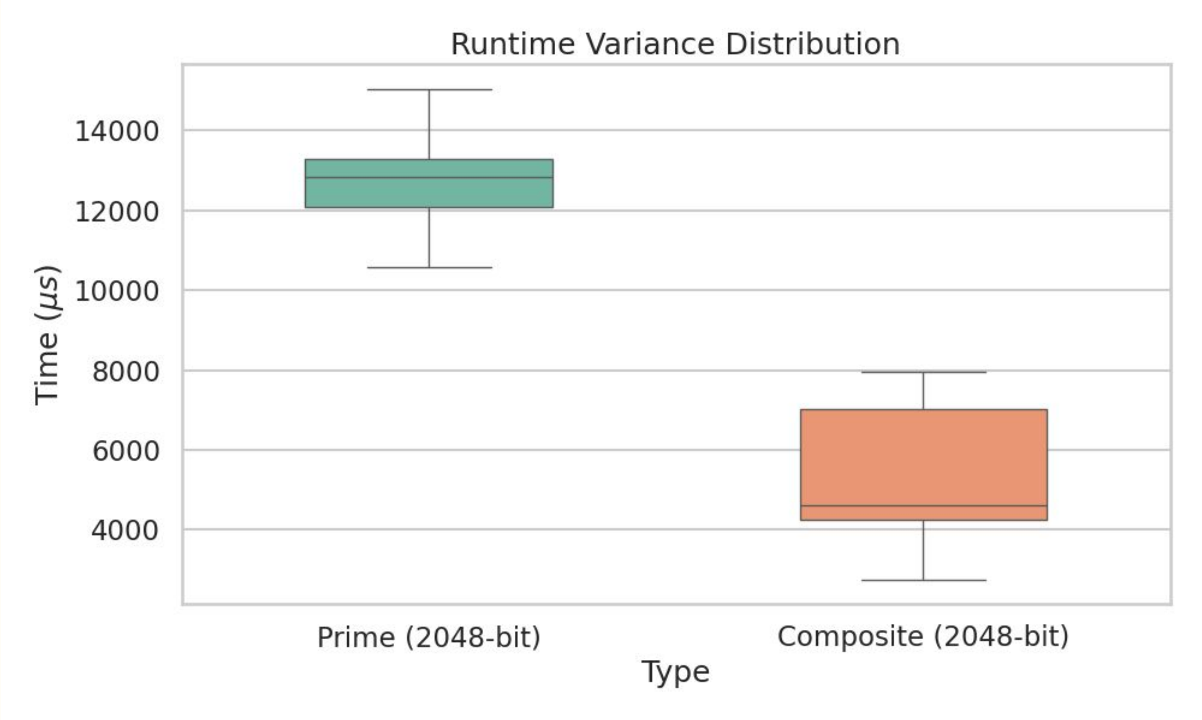


Error Convergence of Miller Rabin



Variance Analysis

Prime numbers require longer time for checking as a full check is needed, therefore they are tightly knitted to form the group of numbers that takes up the most time.



Randomized Min-Cut

- Min Cut Problem: Find the minimum number of edges required to be removed to make the graph disconnected.
- Finds practical uses in networking and circuitry
- Two algorithms have been used:
    - Karger’s Algorithm
    - Karger-Stein Modification to the algorithm

Min-Cut & Karger's Random Algorithm

- Repeatedly contract random edges
- Only two nodes remain → cut value
- Fast but large probability of error
- Success probability $\approx 2/(n(n—1))$
- Accuracy gets worse as the number of edges increase.
- Time Complexity: $O(n^2)$

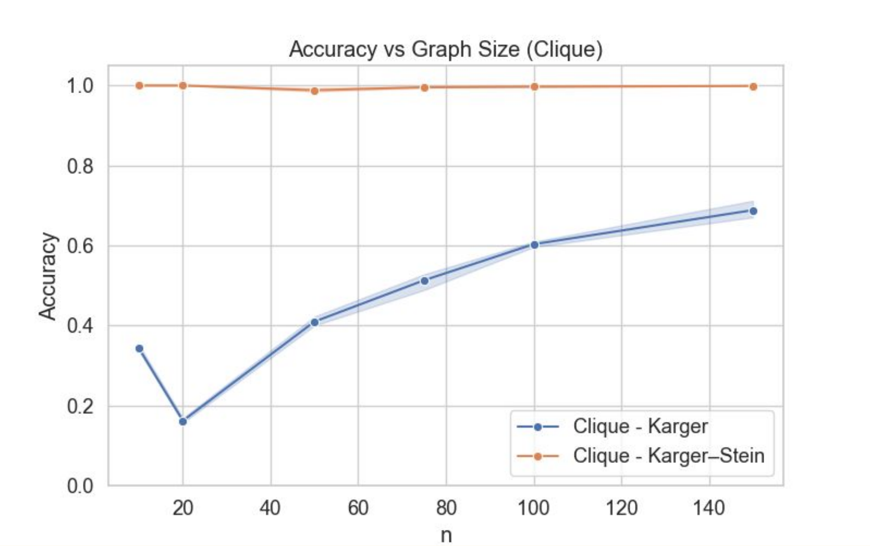
Karger-Stein Algorithm

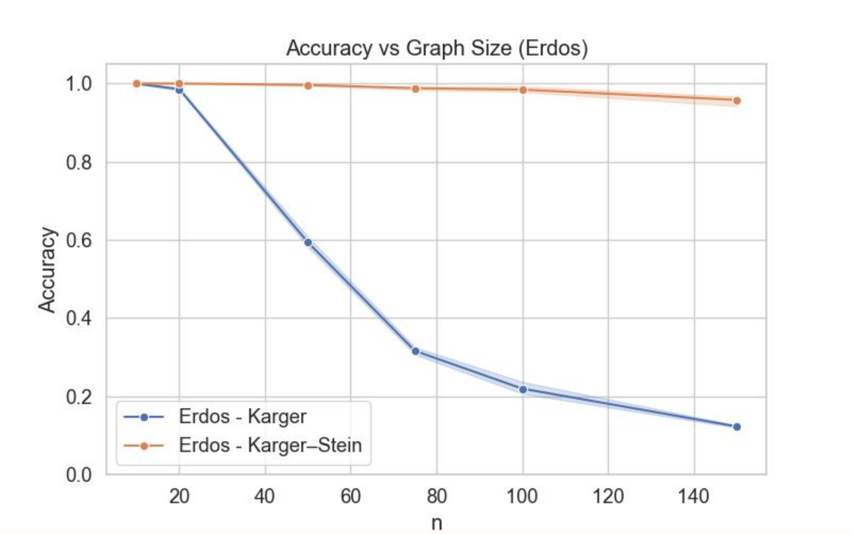
- Avoids over-contracting the graph, contracts it to only $n/\sqrt{2}$ vertices
- Recurses through the reduced graph twice independently, then takes the minimum of the two results
- Success probability $\approx 1/\log n$, which is significantly better than the regular Karger algorithm
- Time Complexity: $O(n^2 \log n)$ which is higher

Implementation

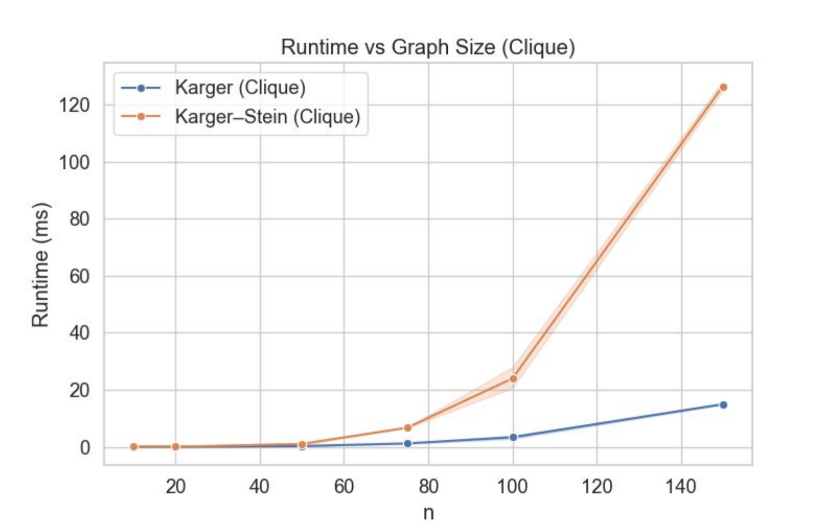
- Stoer-Wagner algorithm was used to secure ground correctness.
- Two sets of graphs were generated:
  - Erdős–Rényi: Generating a set of vertices, with any pair (u,v) having a probability p of having an edge between them.
    - Here the min-cut is large and random contraction behaves fairly well
  - 2-clique: Building 2 cliques of size n/2 and adding k random edges between them
    - Here the min-cut is small (k), making it a worst-case scenario test case for Karger’s

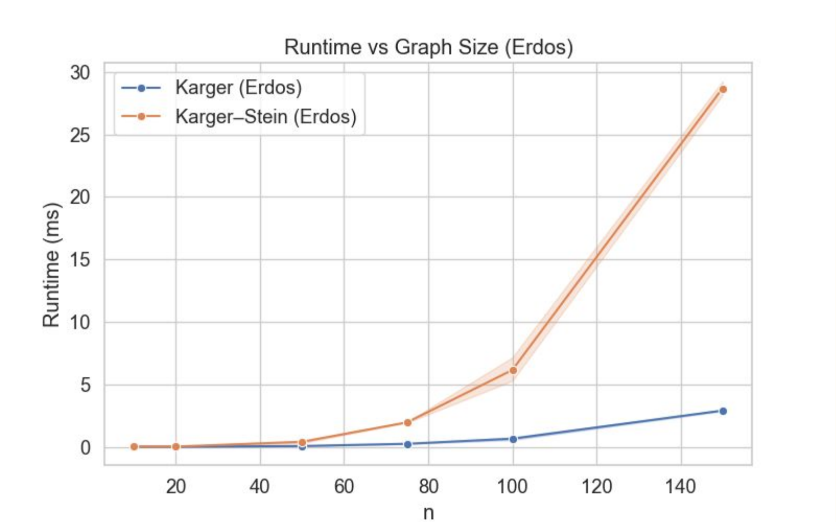
Results: Accuracy Comparison





Results: Runtime Comparison

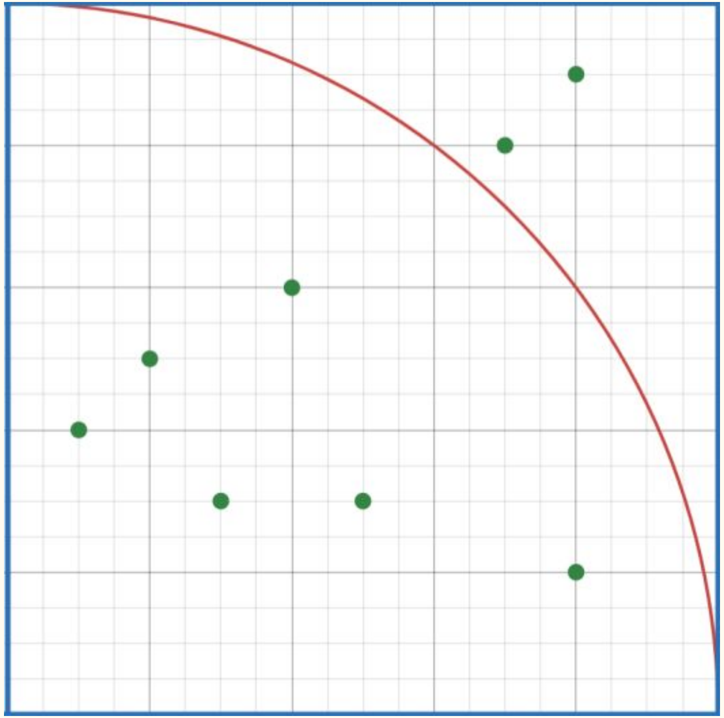




Monte Carlo's Algorithm For Finding π

- We generate random points in the unit square.
- We check which of them lie in the quarter circle by checking whether the sum of squares of their coordinates is less than or equal to 1.
- As the area in the quarter circle is $\pi/4$, we calculate π as follows:

$$\pi = (4 \text{ x pts. inside})/\text{total pts.}$$



Other Types of Sampling Points

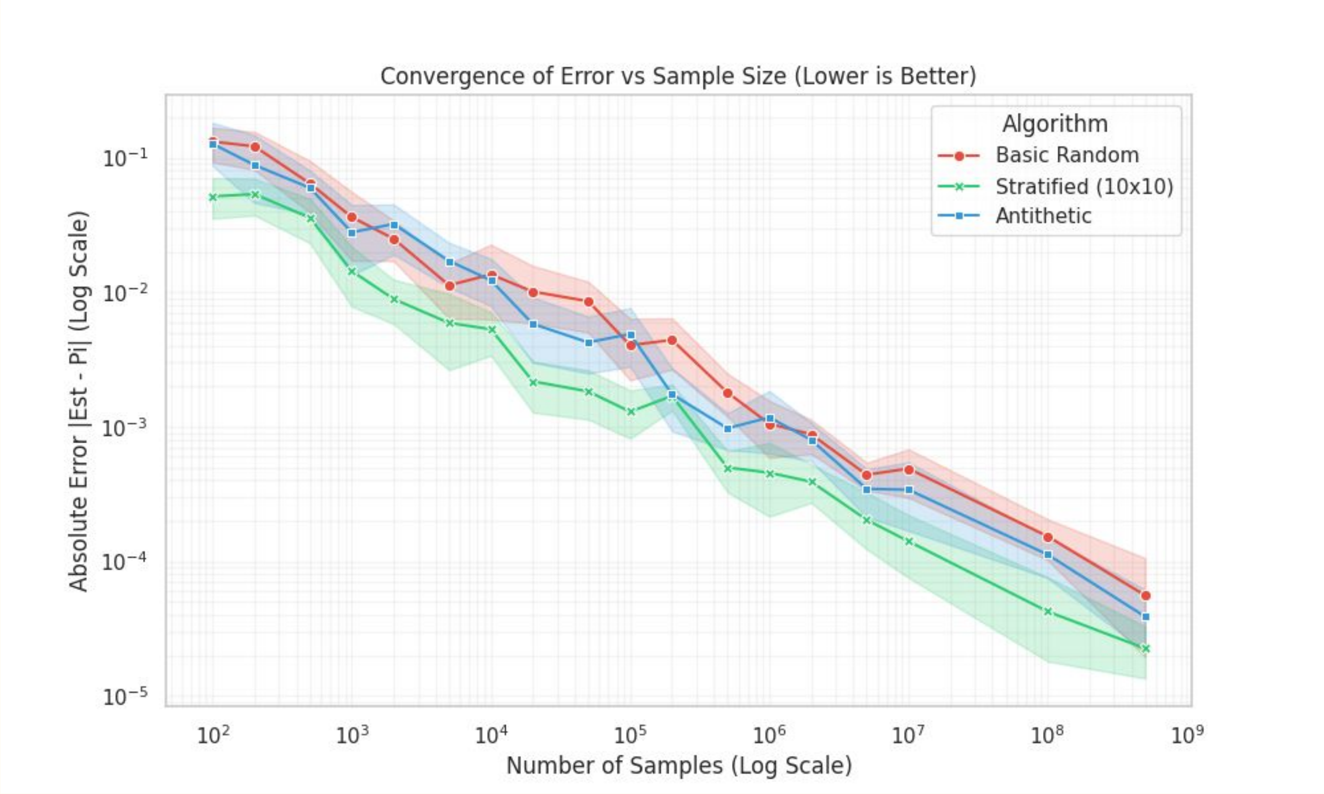
- Stratified: Split the unit square into a k x k grid and sample exactly one point from each grid cell
  - This ensures uniform coverage of the domain and avoids random clustering
- Antithetic: For every point (x, y) selected, also select (1 - x, 1 - y)
  - This ensures that if one point overshoots outside the region, the other one balances it in.

Both these sampling strategies help in reducing variance and noise

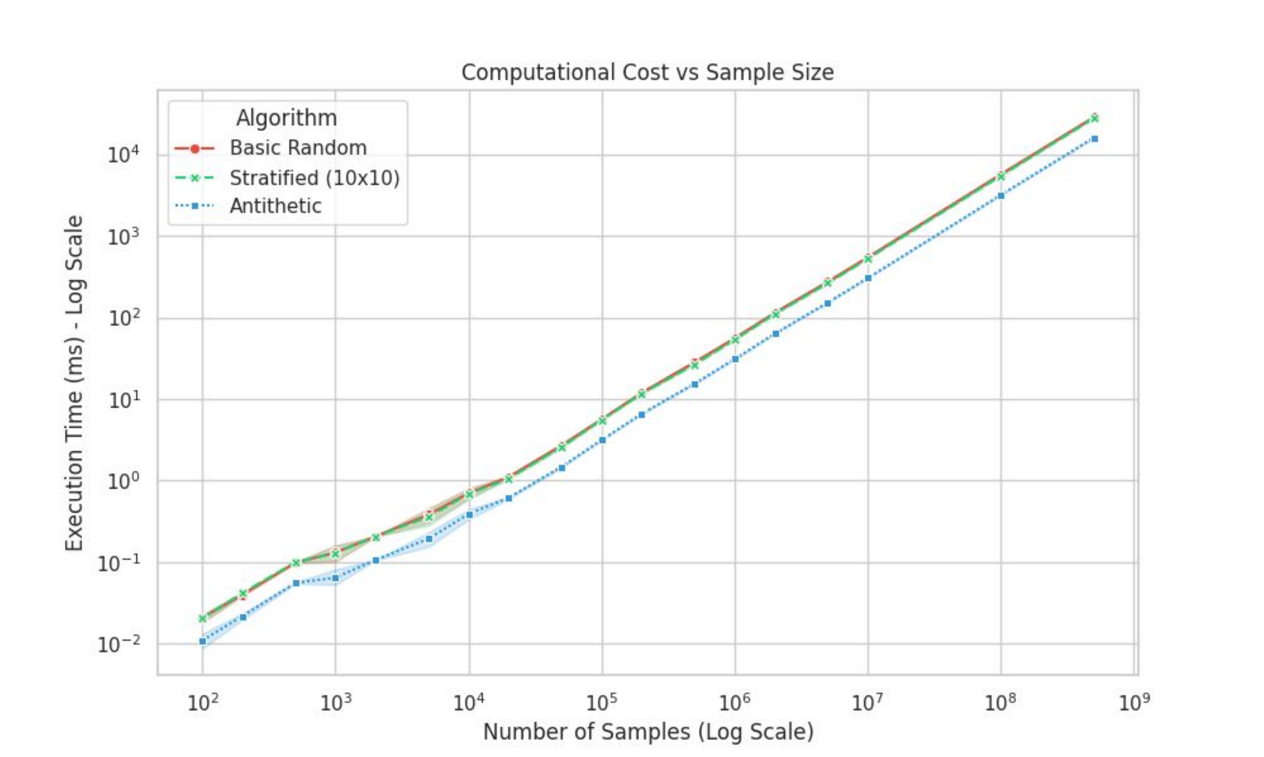
Theoretical Analysis

- Error converges to $O(1/\sqrt{n})$
- Time Complexity: $O(n)$
- Space Complexity: $O(1)$

Results: Convergence of Error



Results: Runtime



Practical Applications

Randomized QuickSort

- High-performance sorting in standard libraries (C++, Java, Python).
- Used in systems where **consistent average speed** matters (DBMS, search engines).

Miller–Rabin Primality Test

- **Cryptography** → RSA key generation, blockchain protocols
- Efficient primality testing for large (1024–4096 bit) integers
- Used in OpenSSL, GMP, Java BigInteger

Karger / Karger–Stein Min-Cut

- Network reliability analysis (find weak links)
- Image segmentation and clustering
- Cut-based graph partitioning in large-scale data processing

Monte Carlo π and Variance-Reduction Methods

- Numerical integration in physics and finance
- Risk estimation in stock markets
- Simulating stochastic processes (weather models, AI/ML sampling)
- High-dimensional problems where deterministic methods fail

Conclusion

- Randomized algorithms provide a powerful balance between speed, simplicity, and probabilistic accuracy.
- Through this project, we observed how randomness can dramatically improve performance (Randomized QuickSort), guarantee efficiency through repeated sampling (Monte Carlo π), and enhance correctness through probabilistic reasoning (Miller–Rabin).
- Even in complex tasks like finding graph min-cuts, randomness helps escape worst-case structures and achieve strong expected results.
- Across all experiments, the empirical outcomes closely matched theoretical predictions, reinforcing randomness as a practical and reliable design principle in modern computing.

THANK YOU