Final Submission

Bill Qin

CSC600: Research and Development

For my final submission, you will see two things.

- **Paper**. The bulk of the project is found here. Images/screenshots/graphs from code written can also be found here. My bibliography is also here.
- **GitHub repository**. https://github.com/bqin01/randomized-algorithms
    - All code done for the project should be there. Also, you'll find the JFrame project I worked on.

The paper is divided into four sections.

- The first section talks briefly about why we care about algorithms in the first place.
- The second section talks about the meaning of randomness and how we can even talk about generating pseudo-random numbers (as well as some ways to generate pseudo-random numbers)
- The third section talks about randomization in certain algorithms while also comparing these to their deterministic counterparts.
- The fourth section is a short blurb on the implications of the project and how randomization should continue to be integrated in algorithmic CS.

In the GitHub Repository, there are multiple different algorithms (all sorted under "algorithms"). Each type of algorithm is written in the C language and has its own class framework (which took me a scary amount of time to complete).

- primality_tests/fermat
    - The Fermat Primality test employs Fermat's Little Theorem, which states that $a^{p-1} \equiv 1 \ (mod \ p)$ for all primes p and all integers a not divisible by p. It turns

out that the converse is *almost always true*. If $a^{n-1} \not\equiv 1 \ (mod\ n)$ almost always implies that n is composite. Using this, we can use fast power algorithms on random values of a such that $1 < a < n$ to check whether the above statement is true (which it almost always is) in order to check for n's compositeness. Because the only operation of substance is the fast power algorithm of $O(\log_2 n)$, so too does this algorithm run in $O(\log_2 n)$.

- primality_tests/naïve
    - The naïve method, of course, checks whether every integer from 2 to $\sqrt{n}$ divides n. This algorithm is an $O(\sqrt{n})$ solution, making it slower than its Fermat counterpart in large cases. (but it's deterministic!)
- matrix_multiplication/freivald
    - Freivald's algorithm for the verification of matrix multiplication generates a random 0-1 vector r (a vector whose contents are only 0 and 1) and checks if $ABr - Cr = 0$, which will always be false if $A \times B \neq C$. And will more than half the time be true if $A \times B = C$. By repeating this at least k times, we reduce our chance of a false negative from $\frac{1}{2}$ to $\frac{1}{2^k}$. This algorithm runs in $O(kn^2)$.
- matrix_multiplication/naïve
    - The naïve method involves recomputing $A \times B$ and checking term-by-term whether or not that matrix is C. Since each one of the n^2 terms requires n operations, the algorithm overall is $O(n^3)$.

Also in the GitHub repository is the "gui" folder. This folder houses four Java files (and their respective class files). Together, they form an interactive window that allows users to simulate throwing as many darts at dartboards as they want in order to empirically determine the value for Pi.

Overall, I had a very fun time with this project. I was happy I got to touch on everything: research, coding algorithms, and software development. While a lot of what I want is there, there are still so many algorithms and ideas that I never get to touch upon simply because I (a) was not able to understand them myself in the time constraint, or (b) could not find a feasible way to code/explain it and thus didn't make it to the paper nor the repository.

I think I finally got my organization to some standard through working on this project! This was initially a concern for me while jumping into the project, but I feel like I've been able to manage my work and mark my progress as I went.

One thing that could've gone better was definitely the time management side of things. With so much other things going on in and out of school, it was sometimes hard to set time aside to work on something that felt more personal (even though I'm still getting graded on this).

Thanks so much for helping me through my project!