**Probabilistic Analysis**

## **Complexity Analysis**

**Time Complexity Analysis:**

The time complexity of the algorithm is primarily determined by the while loop, where we generate random samples, perform calculations, and check conditions. Break down the main time-consuming steps:

1. **Generating Random Samples:** Generating a random sample from the distribution typically takes constant time, assuming that the random number generator has a constant time complexity.
2. **Calculating Estimated Success Rate:** Calculating the estimated success rate involves simple arithmetic operations (addition and division), which have a constant time complexity.
3. **Calculating Upper Bound Using Chernoff Bound:** The calculations involving exponentiation, multiplication, and subtraction have constant time complexity.

The overall time complexity of the while loop is constant, considering the arithmetic operations and random sample generation.

Since the while loop continues until the upper bound is less than or equal to δ, the number of iterations is determined by the properties of Chernoff bound. The number of iterations will be determined by the properties of the Chernoff bound, but in practical terms, it typically converges relatively quickly.

**Space Complexity Analysis:**

The space complexity is determined by the variables used in the algorithm. Break down the space complexity:

1. **count\_success and total\_trials:** These are integer variables used to keep track of the number of successes and total trials. They require a constant amount of space, so their combined space complexity is O(1).
2. **estimated\_success\_rate and upper\_bound:** These are floating-point variables used to store calculated values. They also require a constant amount of space, so their combined space complexity is O(1).

Overall, the algorithm's space complexity is O(1), indicating that the amount of memory used does not depend on the input size.

In summary, the algorithm has a time complexity of O(1) and a space complexity of O(1), making it very efficient in terms of computational resources. The main factor affecting the number of iterations is the behavior of the Chernoff bound, which can vary based on the input δ and the underlying distribution.

## **Proof of Correctness**

To prove the correctness of the algorithm we show that:

1. **The algorithm terminates:** The ‘**while True’** loop will terminate and return a value within a finite number of iterations.
2. **The output is a valid estimate:** The estimated success rate returned by the algorithm is a valid estimate that satisfies the condition of success at least 2/3 of the time with a probability of at least 1 - δ.

**Proof of Termination:**

The while True loop will terminate because at each iteration, the number of trials (**total\_trials**) increases, and the upper bound (**upper\_bound**) calculated using Chernoff bound decreases. Eventually, the upper bound will be less than or equal to δ, causing the loop to exit and return an estimate.

**Proof of Valid Estimate:**

Let **n** be the total number of trials (**total\_trials**) at the point where the loop terminates.

* **Success Rate Estimate (estimated\_success\_rate):** The estimated success rate is calculated as count\_success / n, where count\_success is the number of successes observed during the trials.
* **Upper Bound using Chernoff Bound:** The upper bound is calculated as

Since the loop terminates when the upper bound is less than or equal to δ, we have:

Rearranging the inequality and taking the natural logarithm on both sides, we get:

Solving for estimated\_success\_rate, we get:

By choosing δ such that , we ensure that estimated\_success\_rate is at least 1/3, meaning the algorithm succeeds at least 2/3 of the time.

Therefore, the algorithm provides a valid estimate that satisfies the desired condition.