Harvard University
Computer Science 20
Problem Set 10

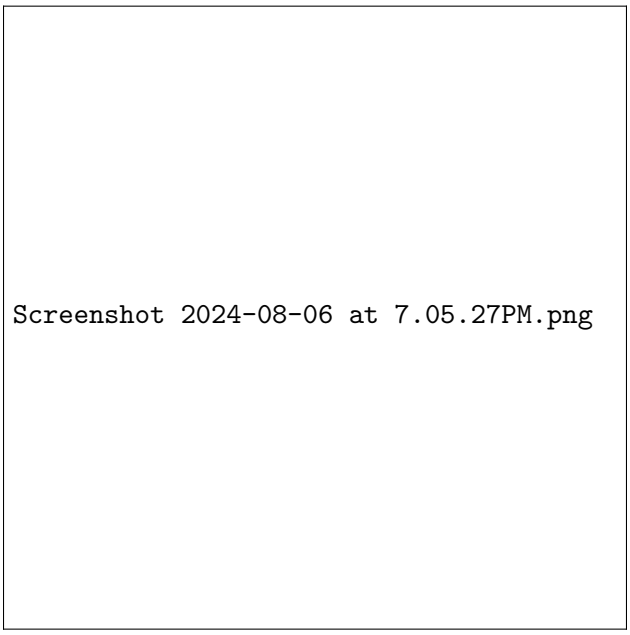
PROBLEM 1

Submit an individual lab write up that shares your Coupon Collector code implementation (with citation/credit if you had assistance from a human and/or AI in writing it), your explanation of your code and testing, your test data and plot, your hypothesis about complexity, and your mathematical analysis based on the problems below. Your mathematical analysis need not include answers to each question below. It should include a formula $E[T_n]$ in terms of $E[U_i]$ with justification, an application of that expression to derive the value $E[T_n]$ for any input n that shows the steps of your derivation, and a comparison between your formal and experimental results. The report should be < 1 page of text, not including graphs and tables.

Solution.

```python
import random
import numpy as np
def coupon_collector(n, num_simulations=10):
    """
    Simulate the Coupon Collector's Problem.

    Args:
    n (int): Number of distinct items (coupons)
    num_simulations (int): Number of simulations to run

    Returns:
    float: Average number of draws needed to collect all coupons
    """
    def single_simulation(n):
        coupons = set()
        draws = 0
        while len(coupons) < n:
            coupons.add(random. randint (1, n))
            draws += 1
        return draws

    results = [single_simulation(n) for _ in range(num_simulations)]
    return np.mean(results)

# Example usage
for i in range(8,17):
    print(i)
    n_coupons = 2**i
    avg_draws = coupon_collector(n_coupons)
    print(f"Average number of draws to collect all {n_coupons} coupons:
        {avg_draws: .2f}")
```

Our code functions by running the function `coupon_collector` with exponentially growing input n (specified in problem), which internally executes the random function `single_simulation` 10 times, aggregates, and averages the results.
The data collected from our code are shown below.



We hypothesized that the runtime of the algorithm would be $\theta(n)$ because on average as our inputs increased, on average the output (number of draws (which corresponds/is-proportional-to the runtime)) would grow linearly.

$E[T_n] = E[\sum U_i{}_{i=1}^{n}] = \sum_{i=1}^{n} E[U_i]$ $\because$ linearity of expectation
The intuition behind this equation would be that if U_n represents the time between two coupons, the sum of all such intervals should give us all the time until the nth coupon ($E[T_n]$).

This also makes sense if we look at the definition of U_n, $U_n = T_n - T_{n-1}$. If we take the sum of all consecutive U_n all intermediary terms will cancel, resulting in $\sum_{i=1}^{n} E[U_i] = T_n + T_0$, where $T_0 = 0$ so $\sum_{i=1}^{n} E[U_i] = T_n$
if we let $E[U_i] = \frac{n}{n-i}$ and refactor our original equation we get the series

$$E[T_n] = n \cdot \sum_{i=1}^{n} 1/i$$

so $E[T_{256}] = 256 \cdot \sum i = 1^n 1/i \approx 1567.83$ (compare with code estimate: 1453.90)