**2.5.31;Problem Statement 3.**

Write a client that takes integers M, N, and T as command-line arguments,

then uses the code given in the text to perform T trials of the following experiment: Generate N random int values between 0 and M – 1 and count the number of duplicates. Run your program for T = 10 and N = 10^3, 10^4, 10^5, and 10^6, with M = N/2, and N, and 2N. Probability theory says that the number of duplicates should be about (1 – e^(–alpha) ) where alpha = N/M—print a table to help you confirm that your experiments validate that formula.

**APPROACH:**

I have used the formula (1 – e^(–alpha) ) in the given problem statement to find the number of distinct values, I have compared them by taking mean of the all T trails, and the both values for N = 10^3, 10^4, 10^5, and 10^6, with M = N/2, and N, and 2N. Are approximately equal. These are the results.

**Results:**

**Values Generated | Max Value | Distinct Values | Expected Distinct Values**

**1000000 500000 432384.60 432332.36**

**1000000 1000000 632184.60 632120.56**

**1000000 2000000 786948.20 786938.68**

**Time complexities:**

TIME COMPLEXITY: **O(T\*(M+N))**

**3.5.30:Problem Statement 2:**

Redo Exercise 2.5.31 using the Dedup filter given on page 490. Compare the running times of the two approaches. Then use Dedup to run the experiments for N = 10^7, 10^8, and 10^9, repeat the experiments for random long values and discuss the results.

**APPROACH:**

I have used dedup method to store the randomly generated keys, the dedup method is using hashset data structure. The maximum long number I have considered as 10^7. These are the results.

**Results of hashset:**

Method | Values type | Values Generated | Max Value | Time spent

DeDup Long 10000000 5000000 25.69

DeDup Long 10000000 10000000 29.37

DeDup Long 10000000 20000000 33.38

**Time complexities:**

TIME COMPLEXITY for dedup method : **O(N)**

Time complexity for add method in hashset : **O(1)**

Time complexity for contains method in hashset: **O(1)**

**5.2.23: Problem Statement 1:**

Redo Exercise 3.5.30 using StringSET (see Exercise 5.2.6) instead of HashSET. Compare the running times of the two approaches. Then use Dedup to run the experiments for N = 10^7, 10^8, and 10^9, repeat the experiments for random long values and discuss the results.

**APPROACH:**

I have used dedup method to store the randomly generated keys, the dedup method is using StringSet data structure. The maximum long number I have considered as 10^7. These are the results.

**Results of stringset:**

Method | Data structure | Values type | Values Generated | Max Value | Time spent

DeDup StringSet Long 10000000 5000000 74.32

DeDup StringSet Long 10000000 10000000 83.83

DeDup StringSet Long 10000000 20000000 92.99

**Time complexities :**

TIME COMPLEXITY for dedup method : **O(Nlog(n))**

Time complexity for add method in hashset : **O(log(n))**

Time complexity for contains method in hashset: **O(log(n))**

**Conclusion:**

* **By the above results we can clearly see that the time taken by the HashSet is less compared to StringSet.**
* **This is because we are using Trie data structure in the StringSet and the time complexity for add and contains methods in the StringSet is logn.**
* **But in the HashSet we generate the hashcode to add, get and contains methods so, the time complexity for them is constant.**
* **HashSet does not provide the ordered keys but the StringSet will provide in sorted order.**