UMC201L Assignment 4

Gavish Bansal, Pratham Gupta, Krishna Agarwal

7 November, 2024

1 Testing methodology:

In order to verify the claim experimentally, we used the Splay Tree implementation and wrote functions to pass queries to the data structure and obtain the outputs, at the same time recording the total time taken to answer all c.m queries. In order to make the testing fulfilling and comprehensive, we passed inputs of the initial n numbers to be stored in the Splay Tree randomly i.e. all numbers of the list of n distinct numbers were chosen randomly.

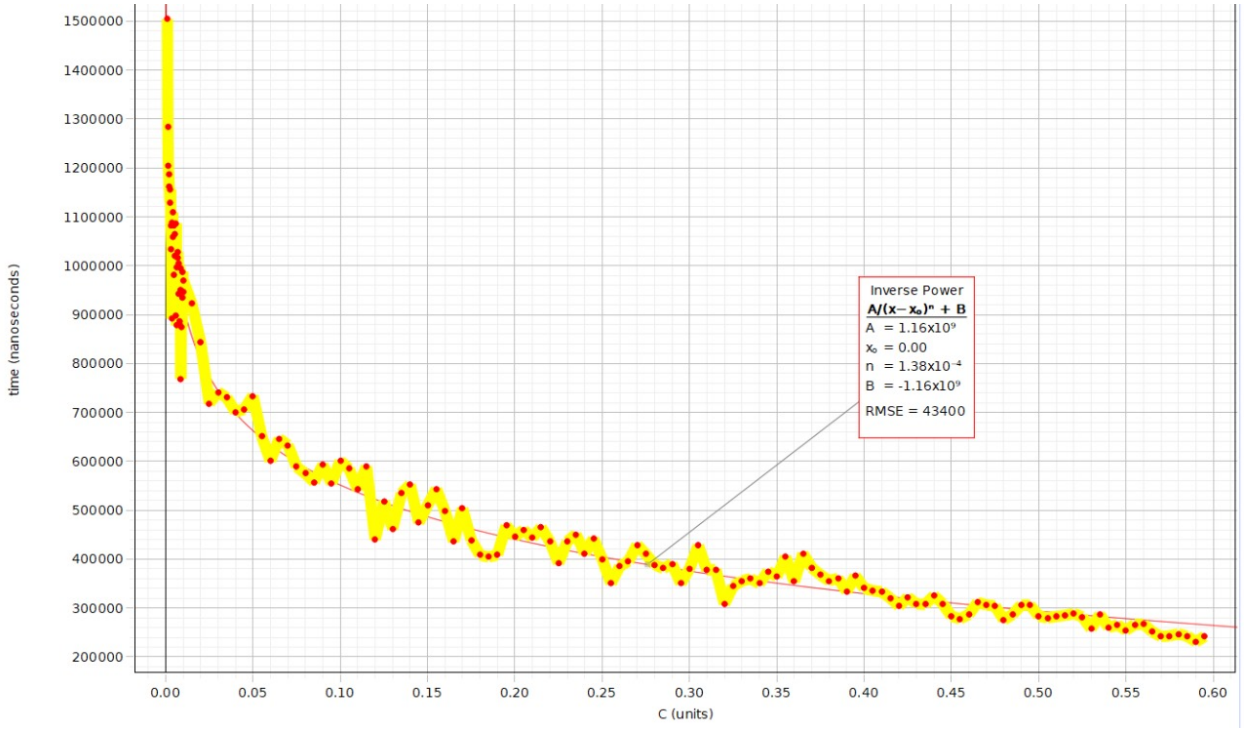
At the same time, we have defined a new quantity called the hit rate which determines the percentage of queries among the m queries that will reap Truth i.e. the requested number will be present in the Splay Tree. Consequently, we have also ensured that c remains less than or equal to the hit rate in each test case. Moreover, the queries were again chosen randomly for given values of m, n, c, and the hit rate.

Note that the major reason behind choosing the input and queries randomly was to maintain uniformity in the operations and not introduce any irrelevant biases, Hitting the worst cases was also more probable with randomly chosen values instead of biasing ourselves to any particular worst case where the conclusions can be inapt.

2 Test Cases:

2.1 Changing c while keeping m, n, and c.m(=20000) fixed:

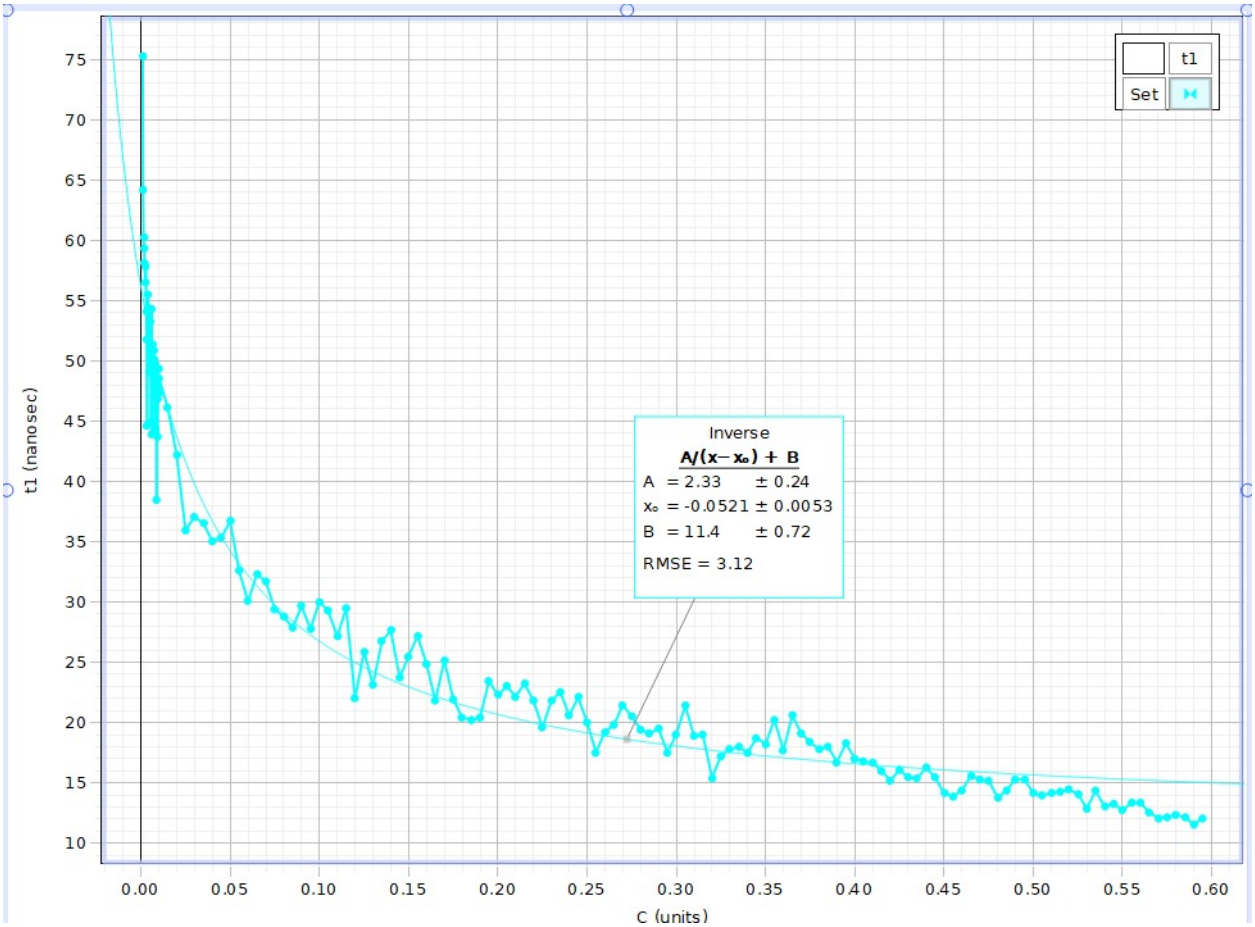
By our assumption, if the amortised cost of searching for the element x in the Splay Tree when asked for c.m times is O(1), the constant in this cost depends on c. Hence, to identify the function of c that makes up this constant, we run the program for test cases given the constraints mentioned in the heading and plot the c vs time taken graphs to fit the appropriate function as the constant of the O(c.m) total time taken for the queries for x if it is amortised O(1). The graph is shown below:



The appropriate curve fitting for the constant in the total time taken for c.m queries for the particular element x in the list of n distinct numbers.

$$constant = \frac{A}{x^n} + B$$

where A $= 1.16 \times 10^9$, $n = 1.38 \times 10^{-4}$, and $B = -1.16 \times 10^9$

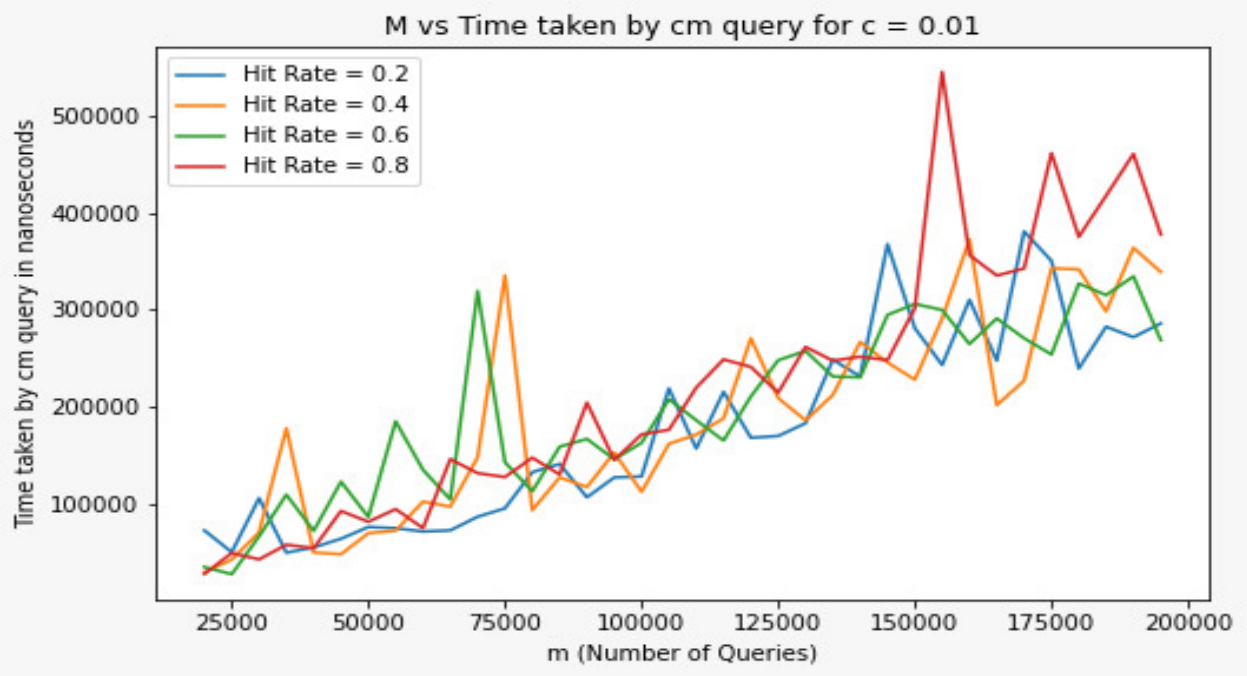


We divided time taken by c.m = 20000 to find the actual constant in O(1) search of element x.

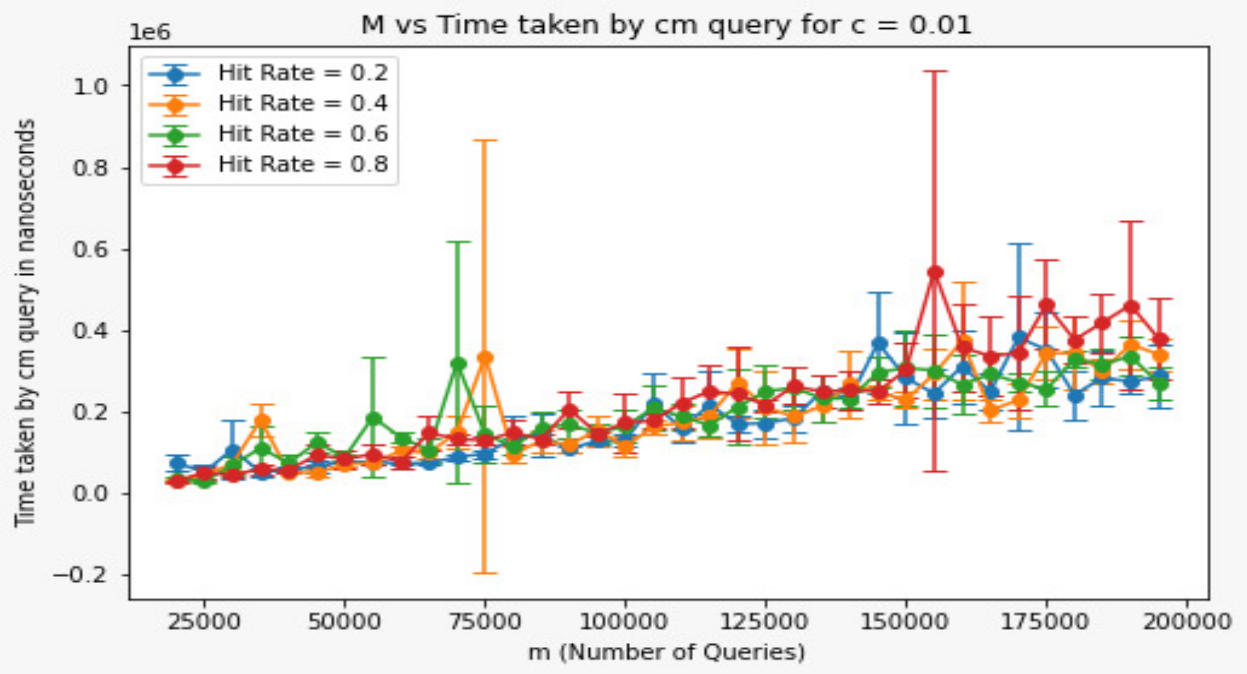
2.2 Keeping c and n fixed, we vary m:

In order to obtain the explicit dependence of the total time taken to execute the c.m queries for x among the m queries on m, we plot the graph for m vs time taken while keeping the remaining parameters(c and n) fixed. We have attached the graphs for different values of c and can observe that the relation between the time and m can be very well approximated to be a linear dependence.

2.2.1 c=0.01 and n=5000

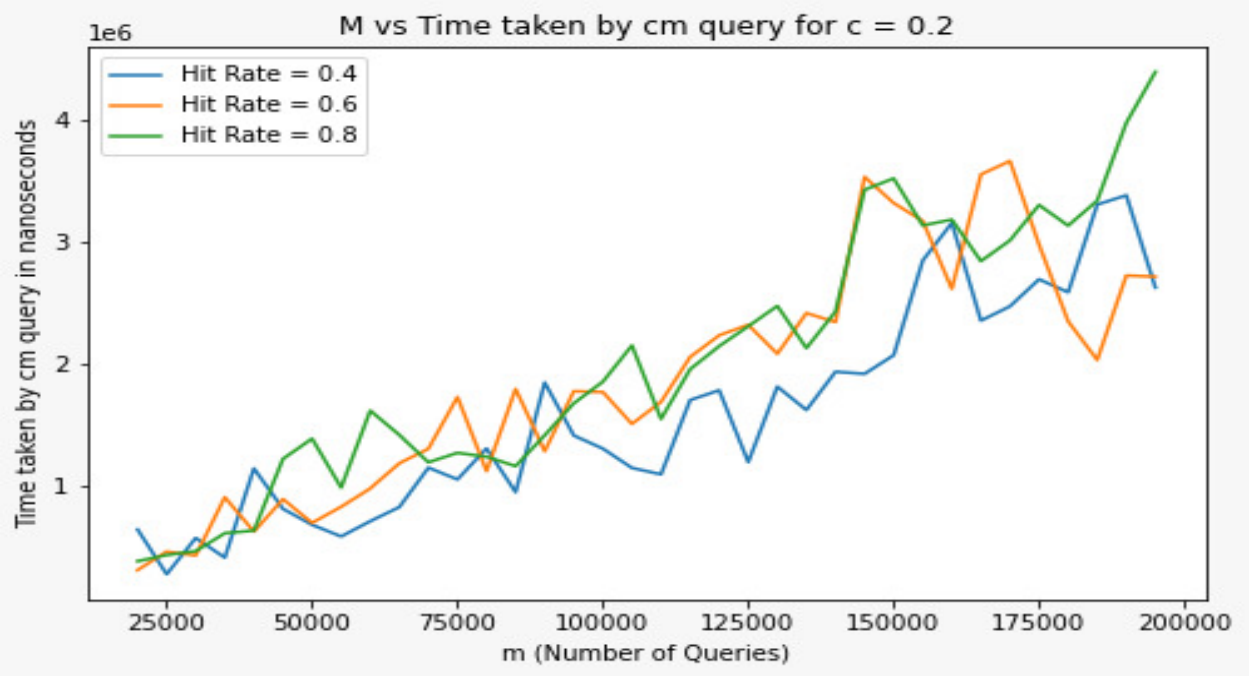


Without error bars

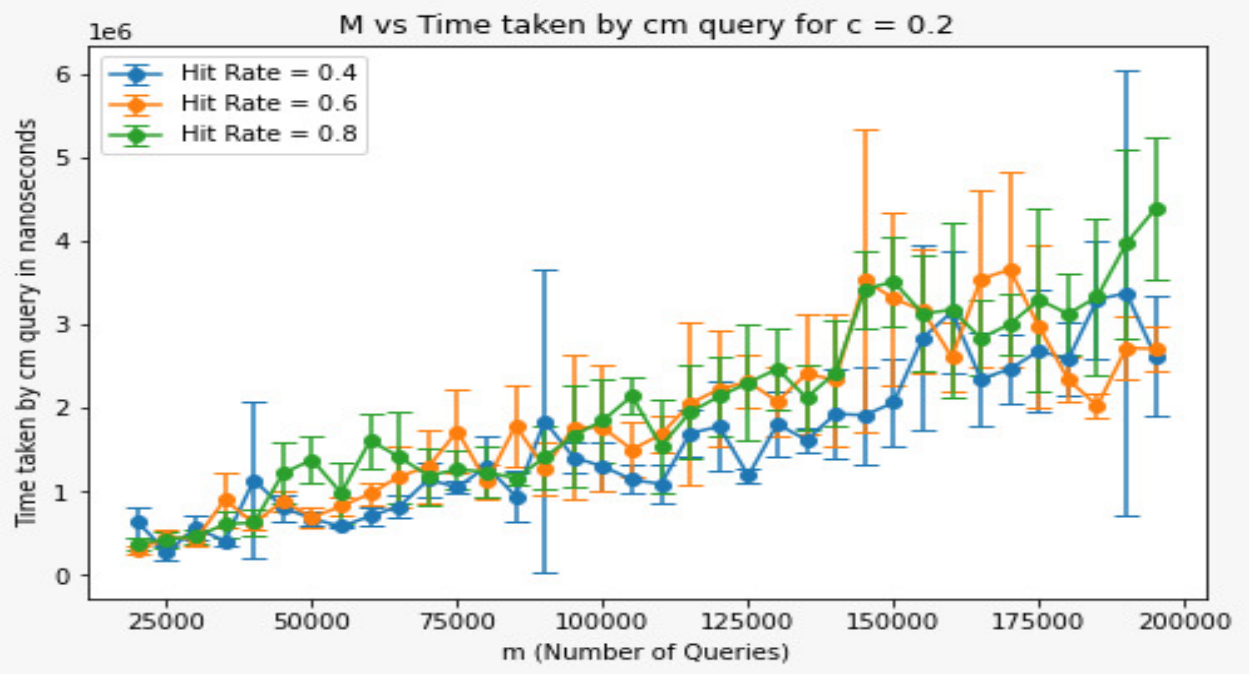


With error bars

2.2.2 c=0.2 and n=5000

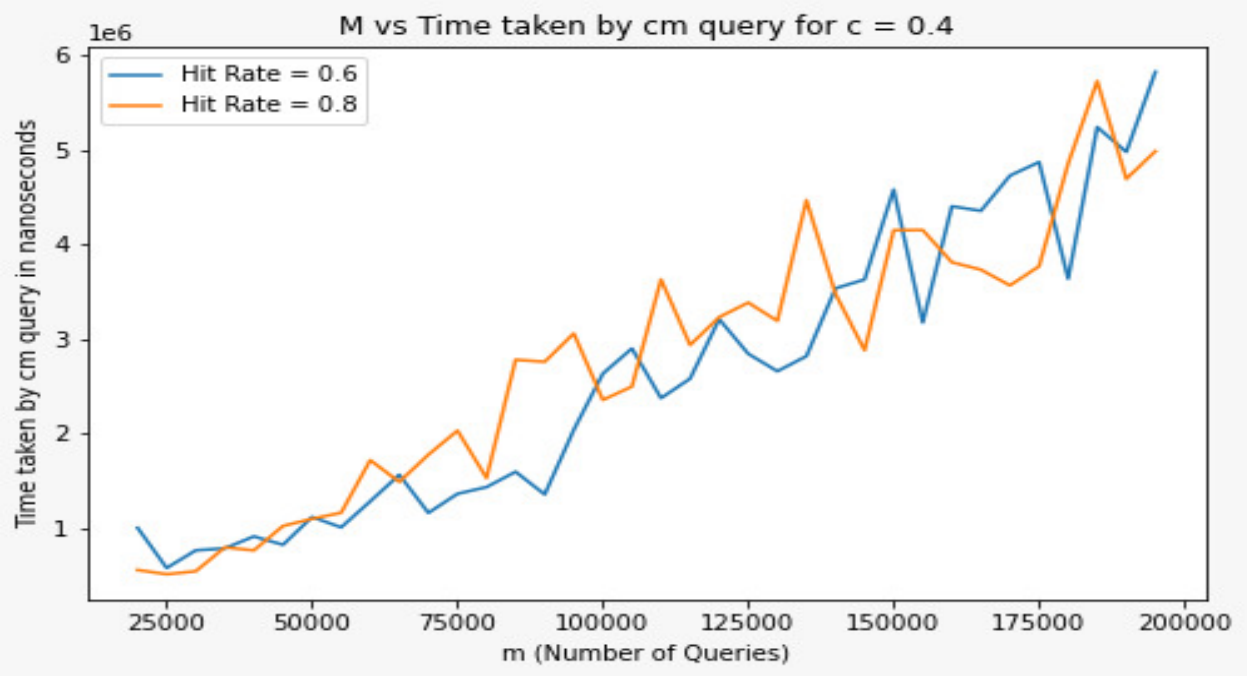


Without error bars

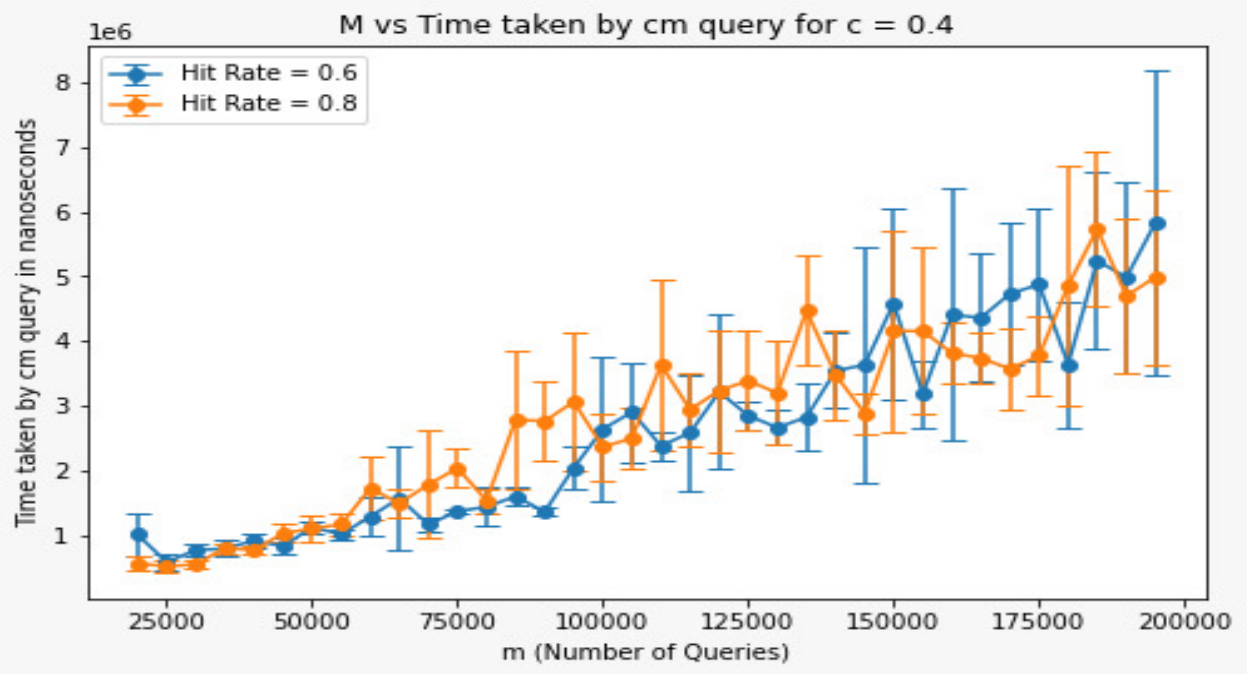


With error bars

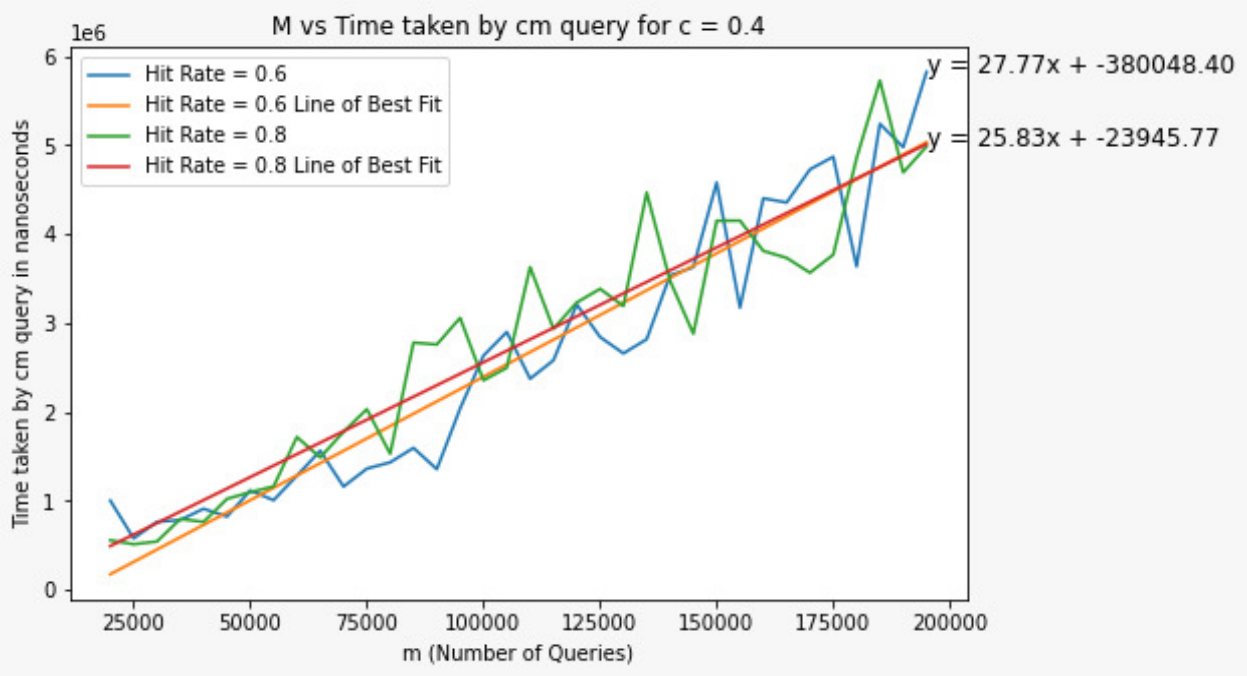
2.2.3 c=0.4 and n=5000



Without error bars



With error bars



Showing lines of best fit

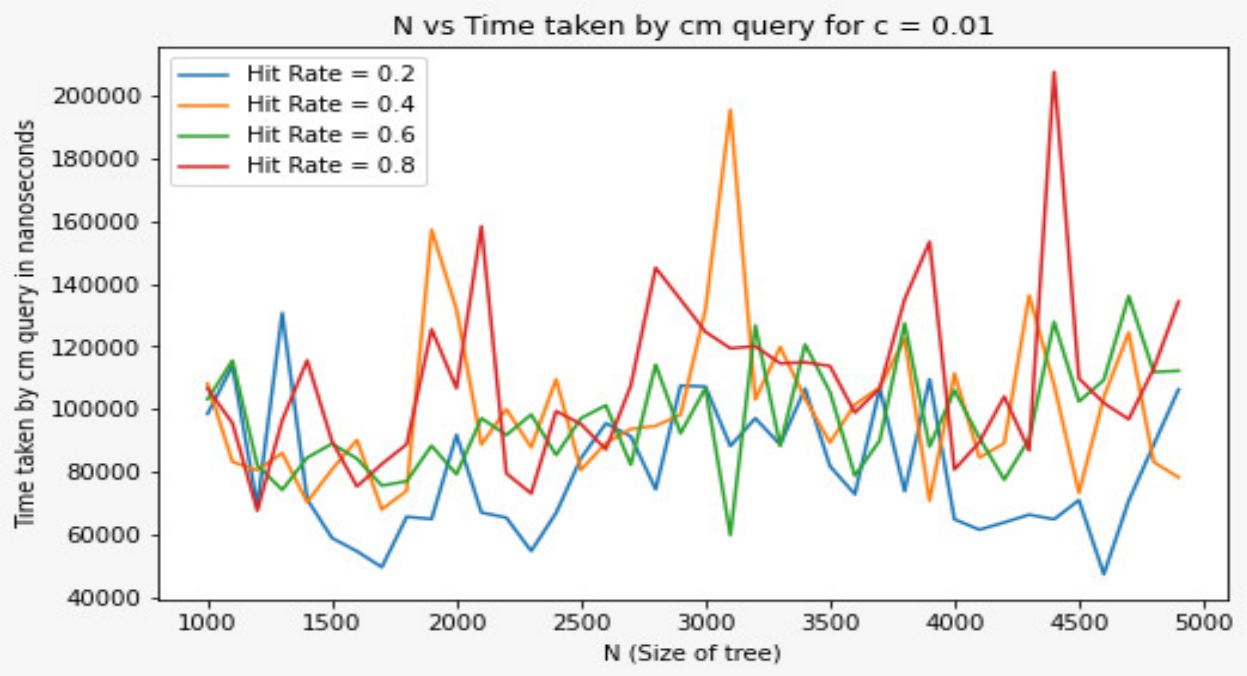
We have thus, successfully shown that the time taken for c.m queries for the element x among the m queries is linearly dependent on the value of m as is also evident on the appropriate linear fits shown for the c = 0.4 case.

Another very interesting observation here is that, when we divide the slopes of the above plots by c, so we should ideally obtain the constant k(c) of O(1) search consistent with our originally derived function. Having done the calculations, it was observed that the expected and calculated values were approximately equal. The reason for slight anomalies would include data specificity, background processes cause inconsistency in processing time, etc. Hence, the reason for very large error bars in our data. Keeping the error bars in mind, the theoretical results are in agreement with the experimental results.

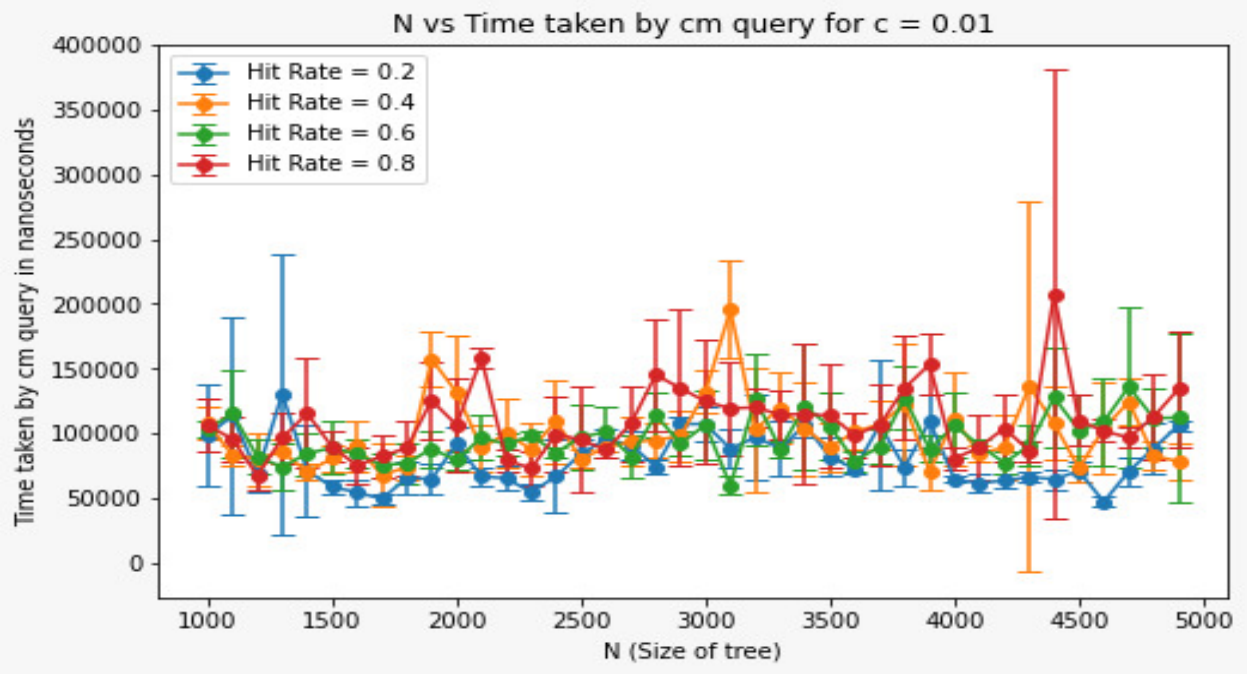
2.3 Keeping c and m fixed, varying n:

In order to obtain the explicit dependence of the total time taken to execute the c.m queries for x among the m queries on n, we plot the graph for n vs time taken while keeping the remaining parameters(c and m) fixed. We have attached the graphs for different values of c and can observe that the relation between the time and n can be very well approximated to be a constant relation i.e time = constant given the above mentioned constraints.

2.3.1 c = 0.01 and m = 50000

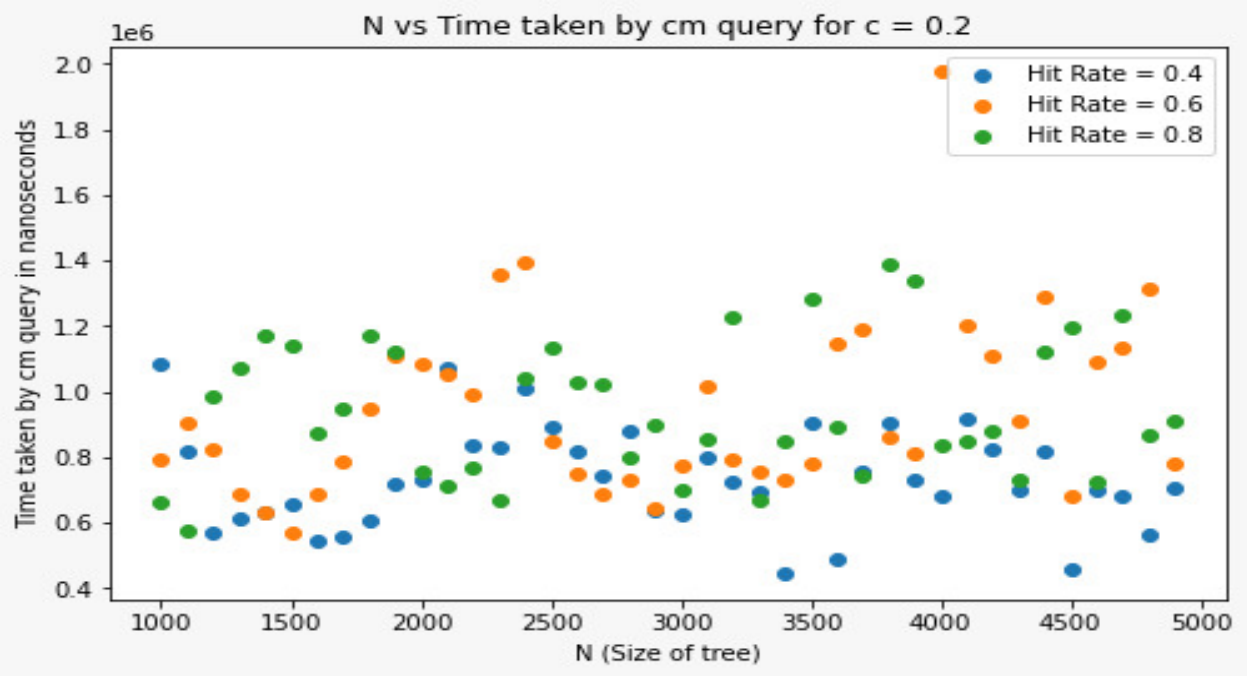


Without error bars

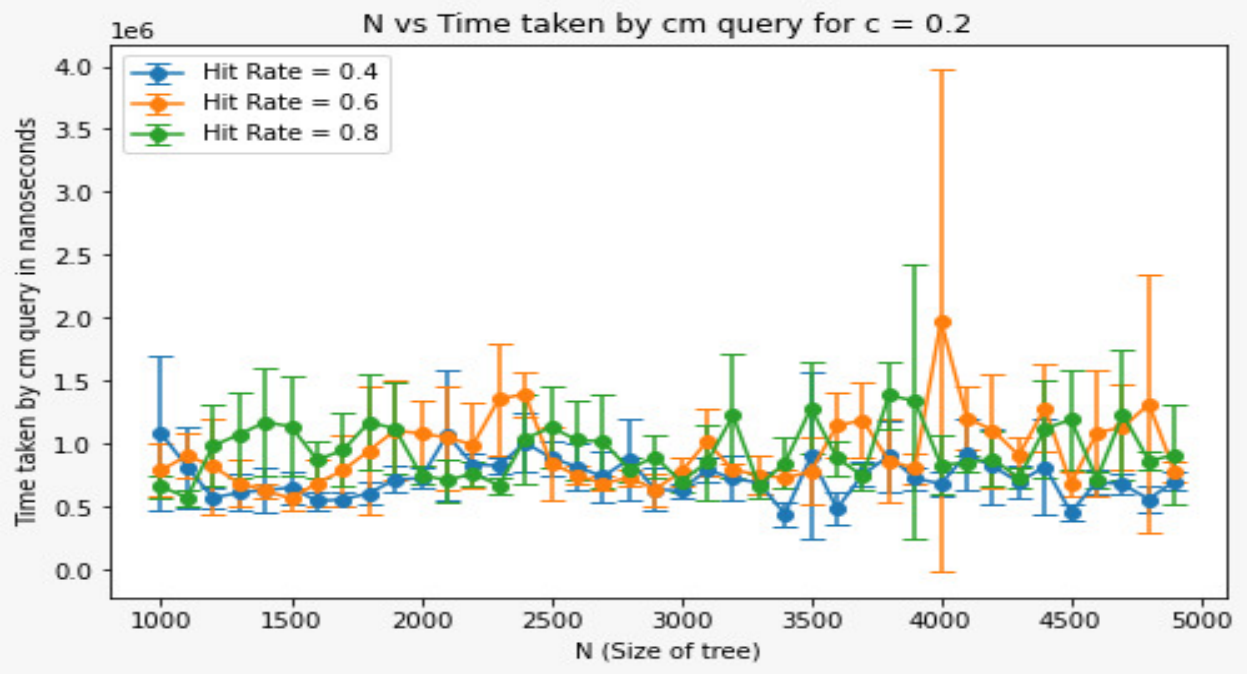


With error bars

2.3.2 c = 0.2 and m = 50000

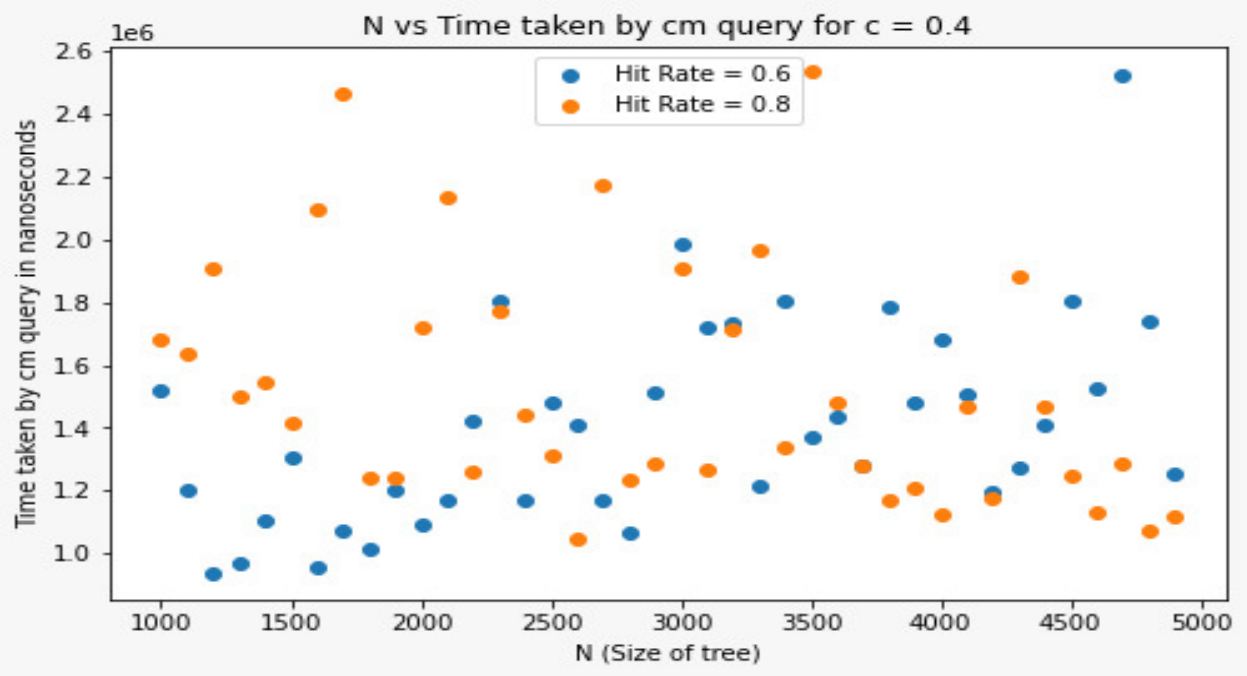


Without error bars

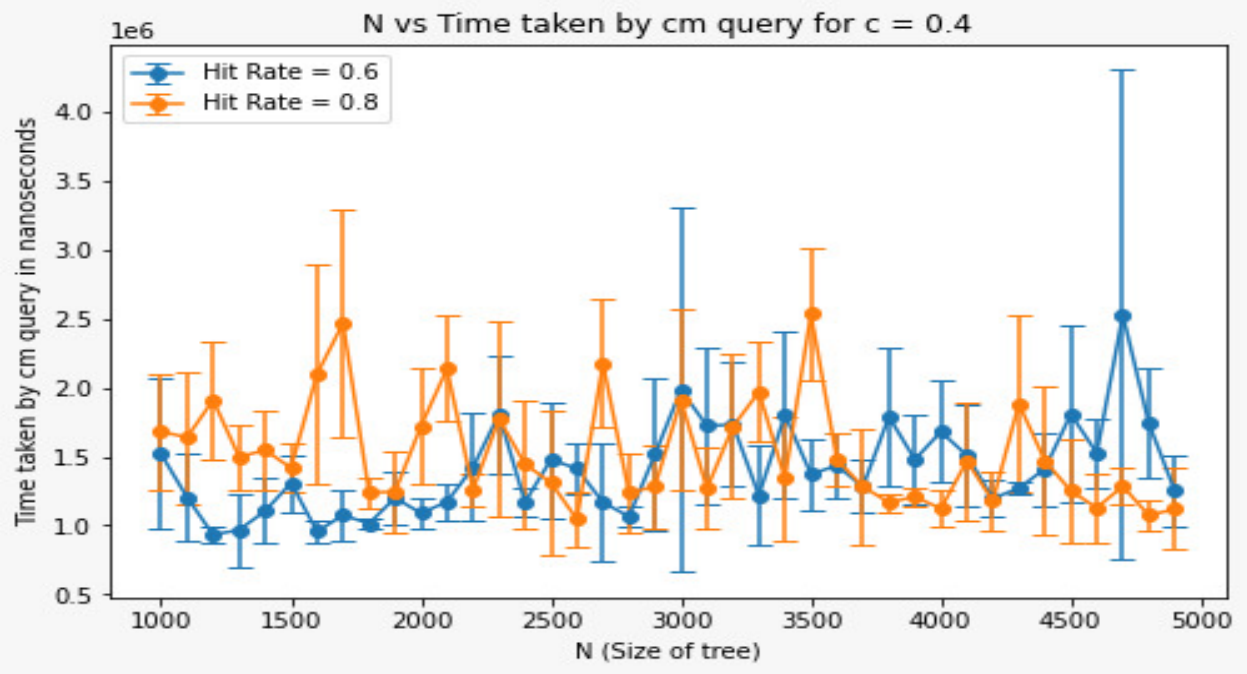


With error bars

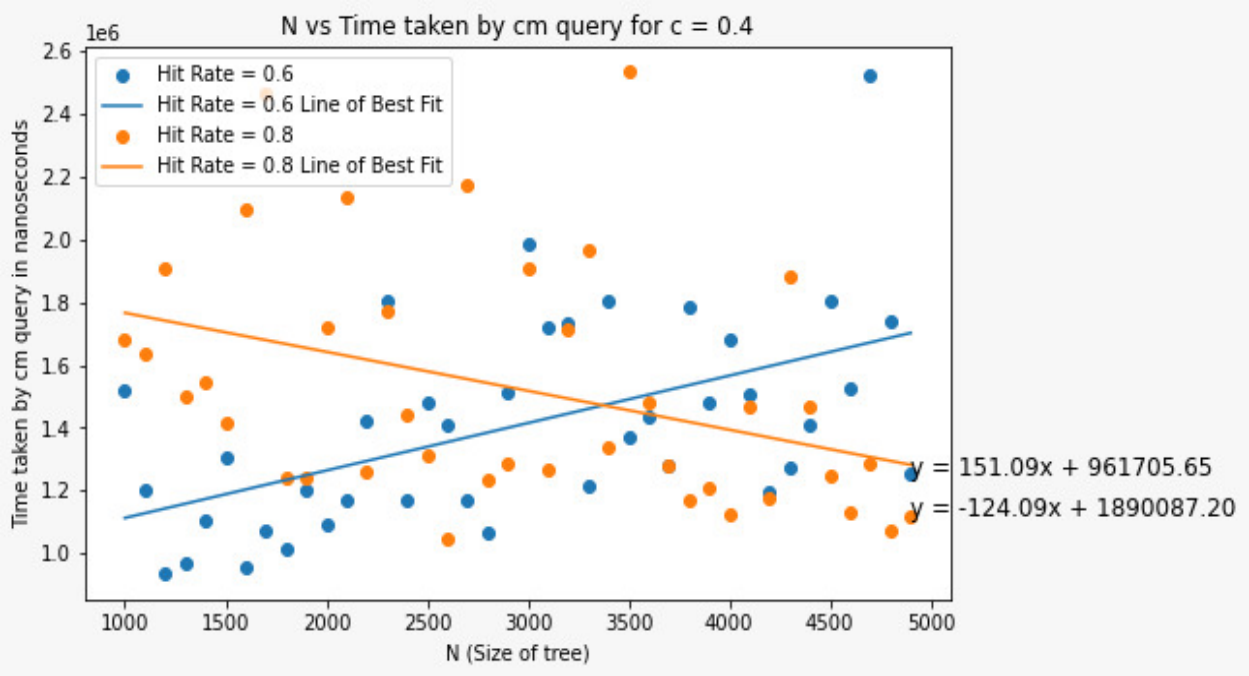
2.3.3 c = 0.4 and m = 50000



Without error bars



With error bars

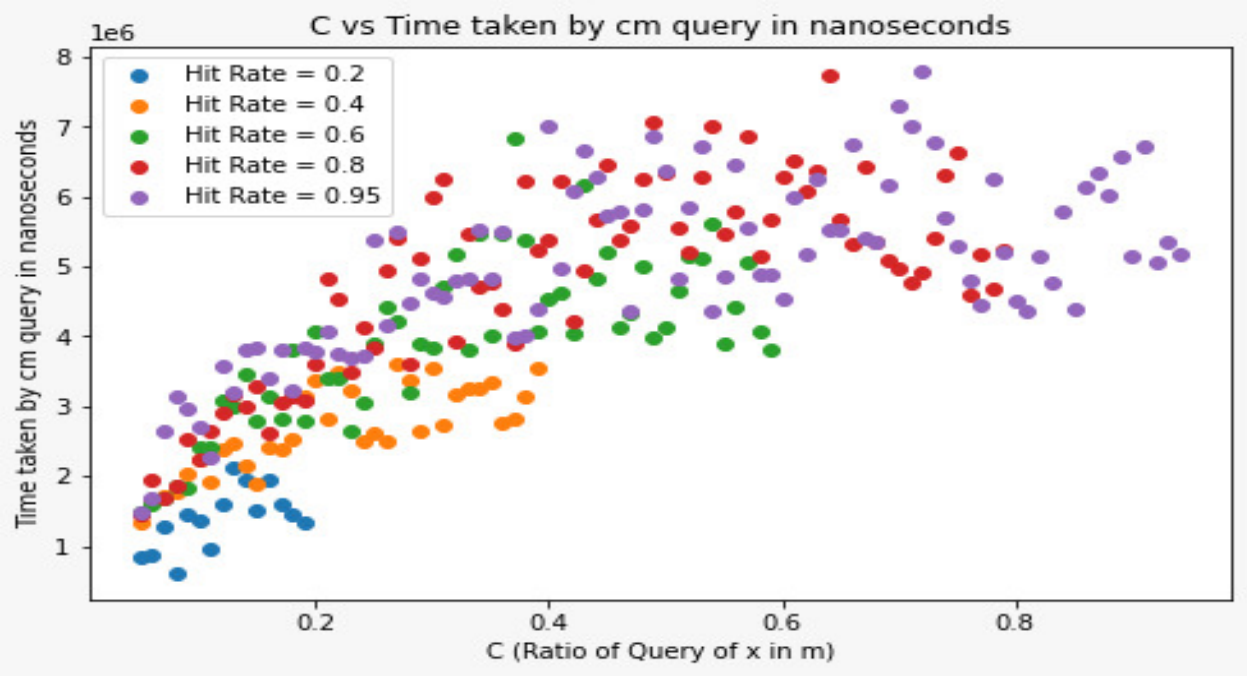


Showing lines of best fit

We have thus, successfully shown that the time taken for c.m queries for the element x among the m queries is independent of the value of n as is also evident on the appropriate curve fits shown for the c = 0.4 case.

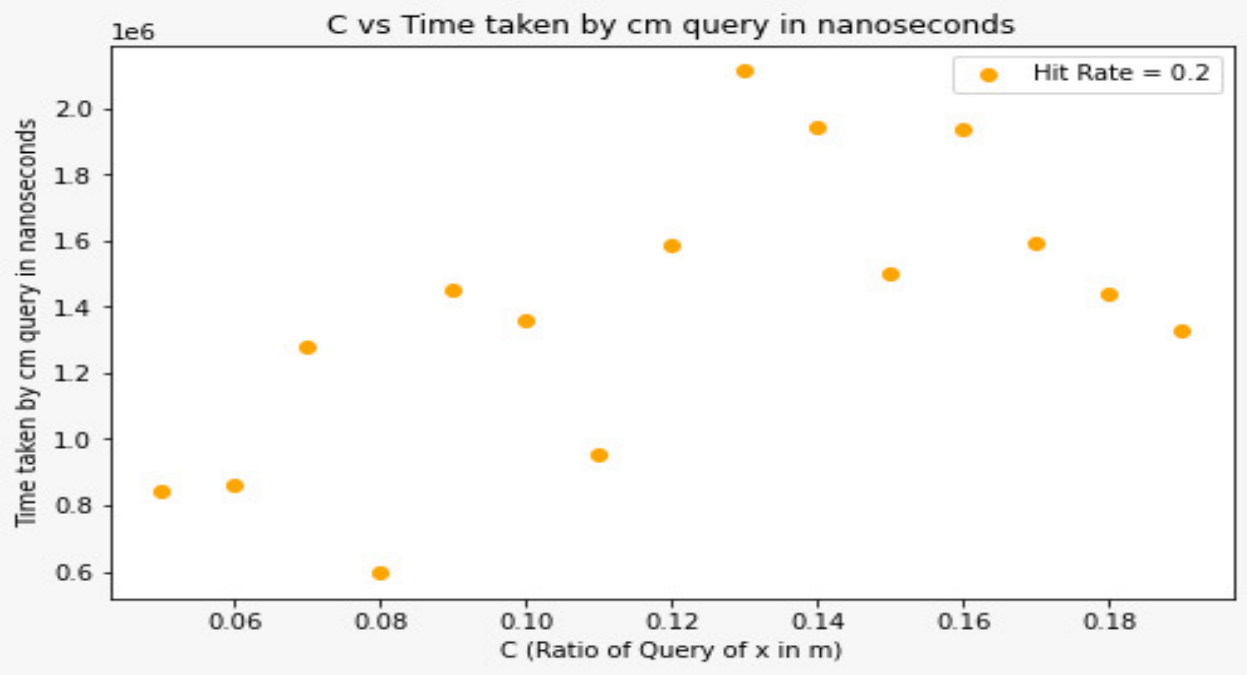
2.4 Keeping m and n fixed, we vary c:

In order to obtain the explicit dependence of the total time taken to execute the c.m queries for x among the m queries on c, we plot the graph for c vs time taken while keeping the remaining parameters(m and n) fixed. We have attached the graphs for different hit rates and can observe that the relation between the time and n can be very well approximated to be a linear dependence.
n = 5000 m = 200000

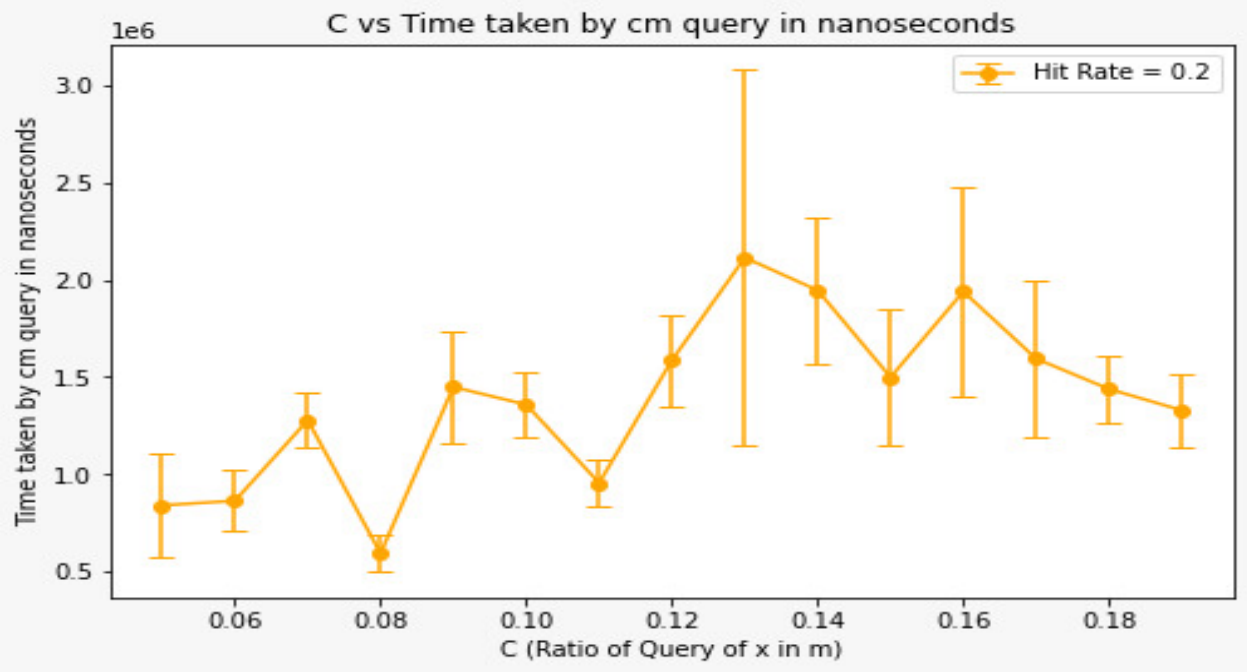


Cumulative data for all hit rates

2.4.1 Hit rate = 0.2

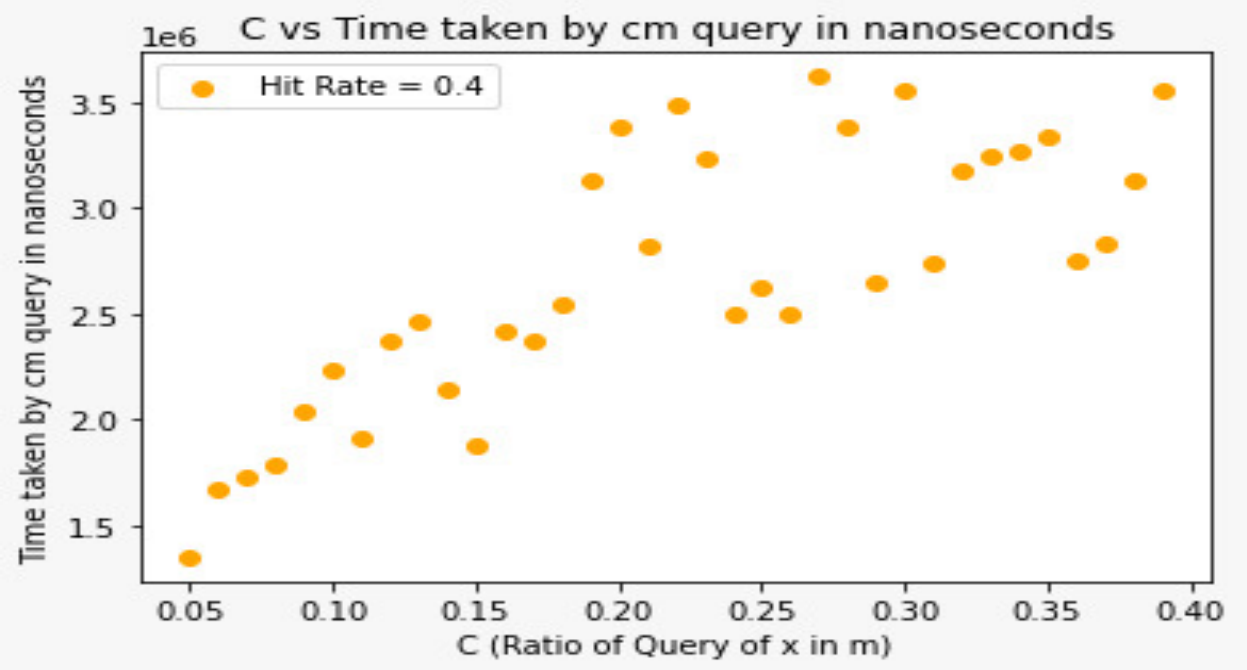


Without error bars

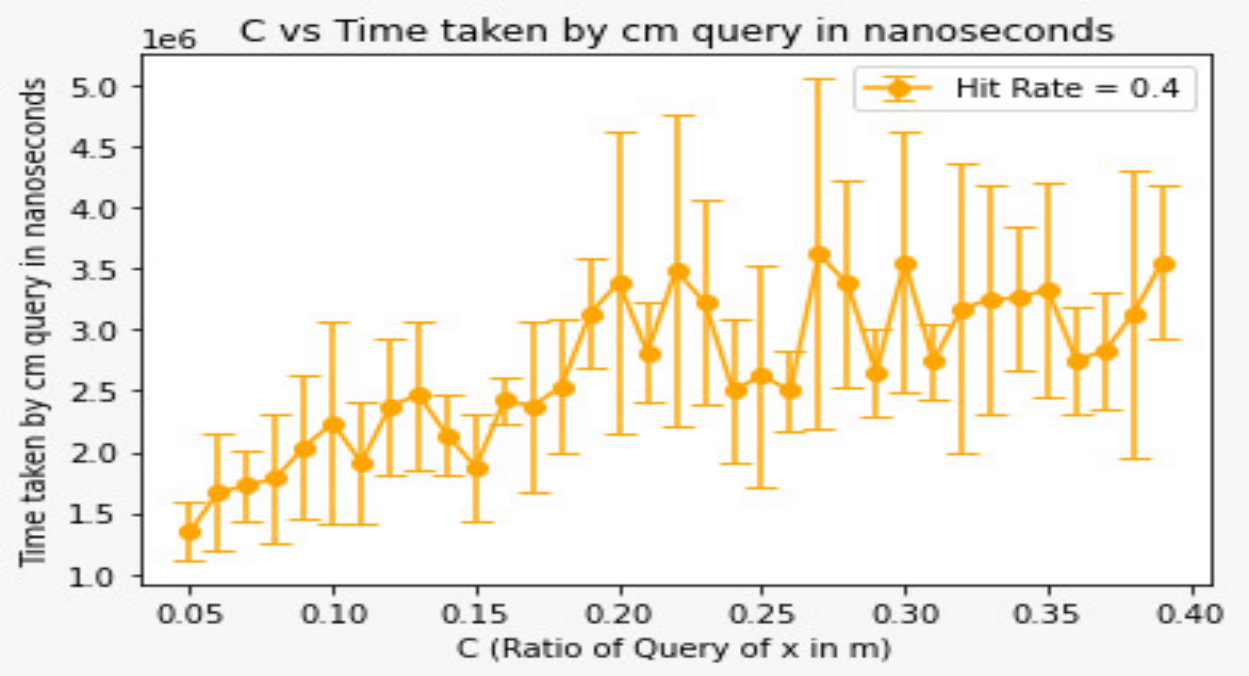


With error bars

2.4.2 Hit rate = 0.4

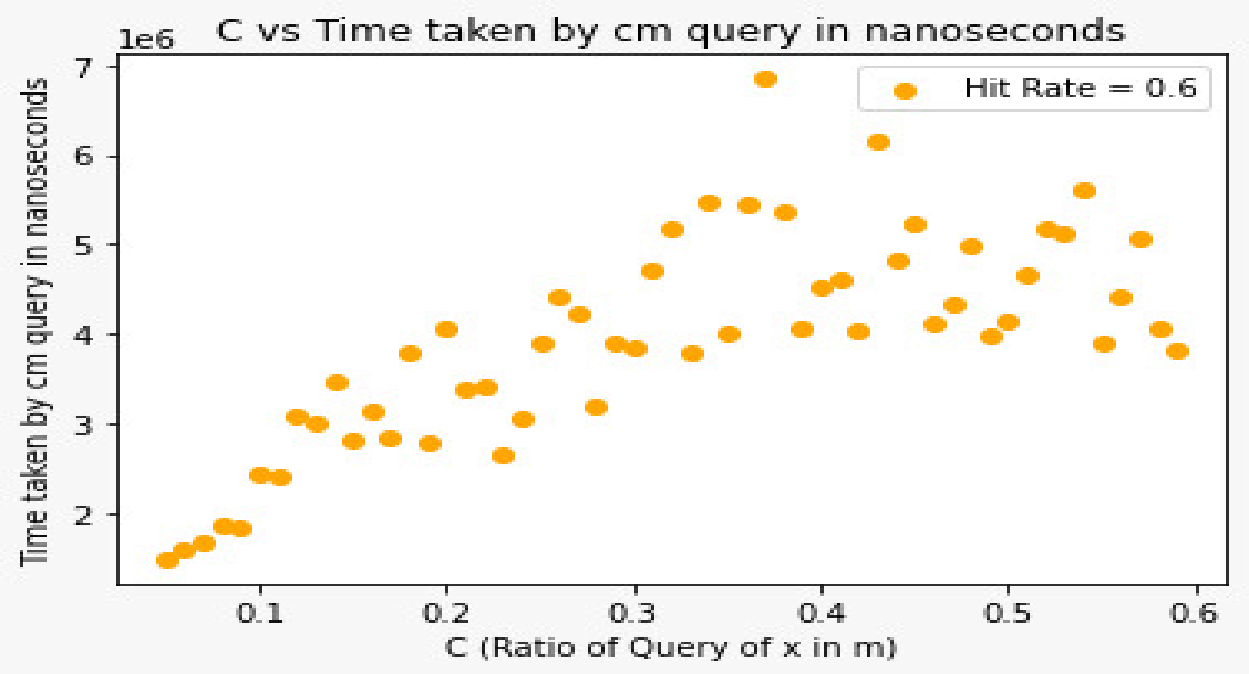


Without error bars

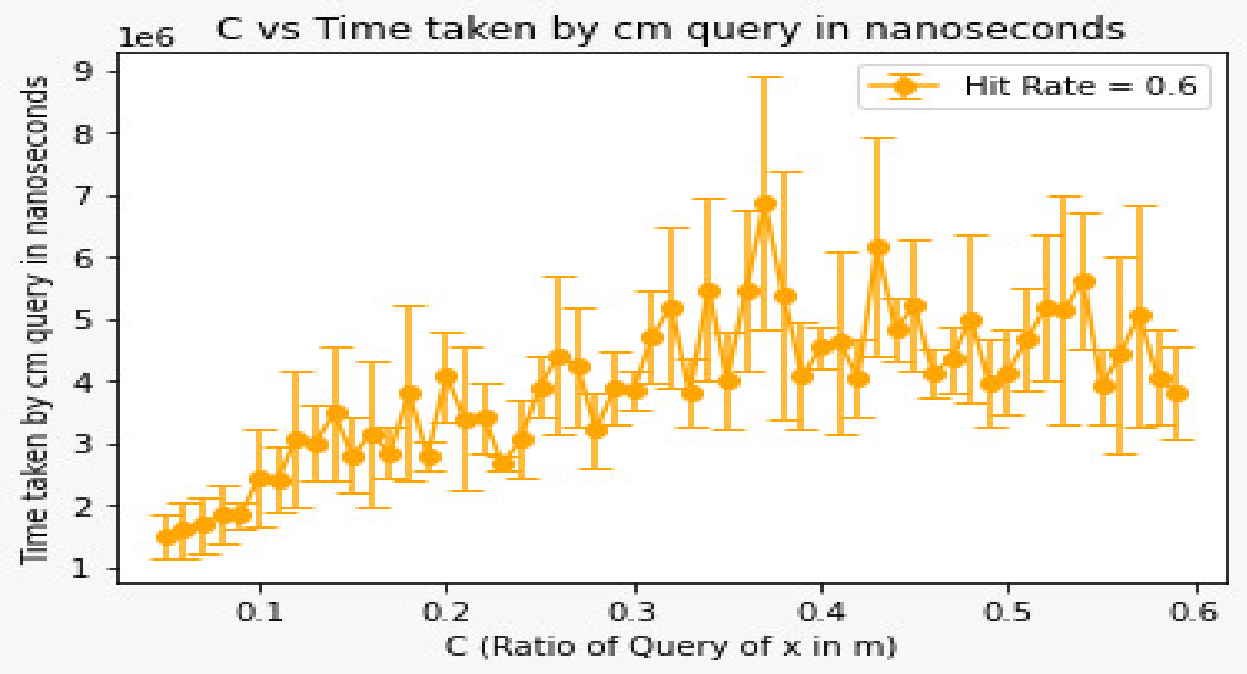


With error bars

2.4.3 Hit rate = 0.6

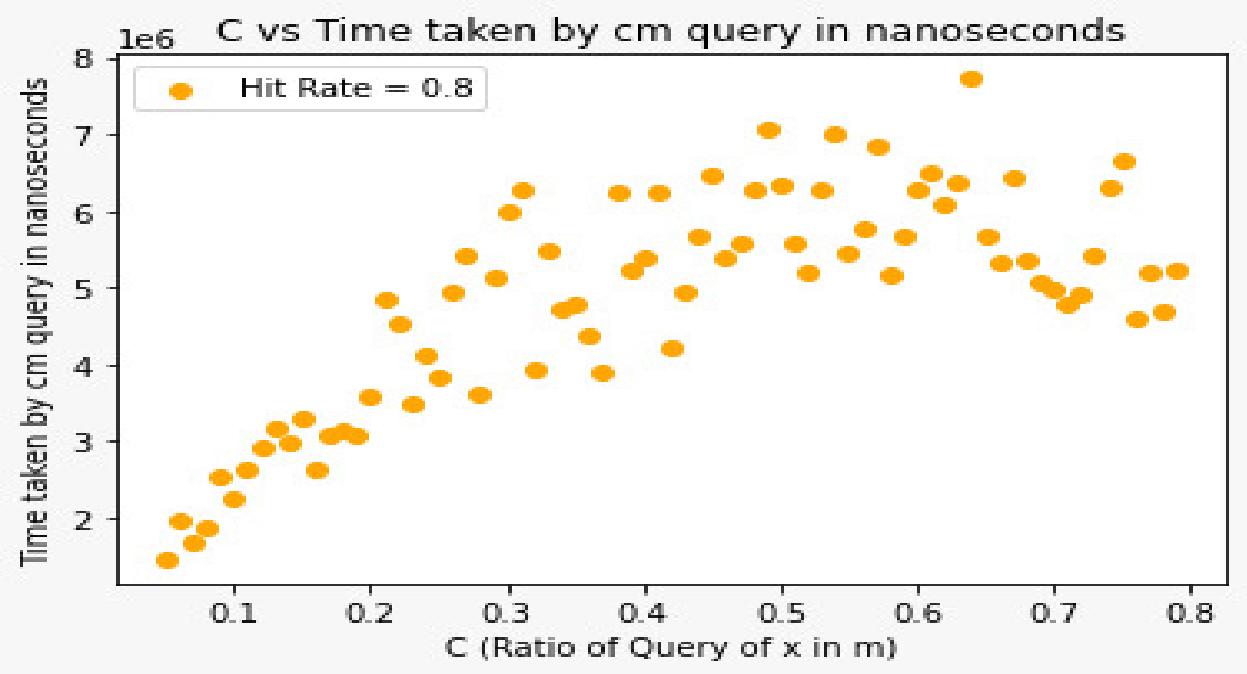


Without error bars

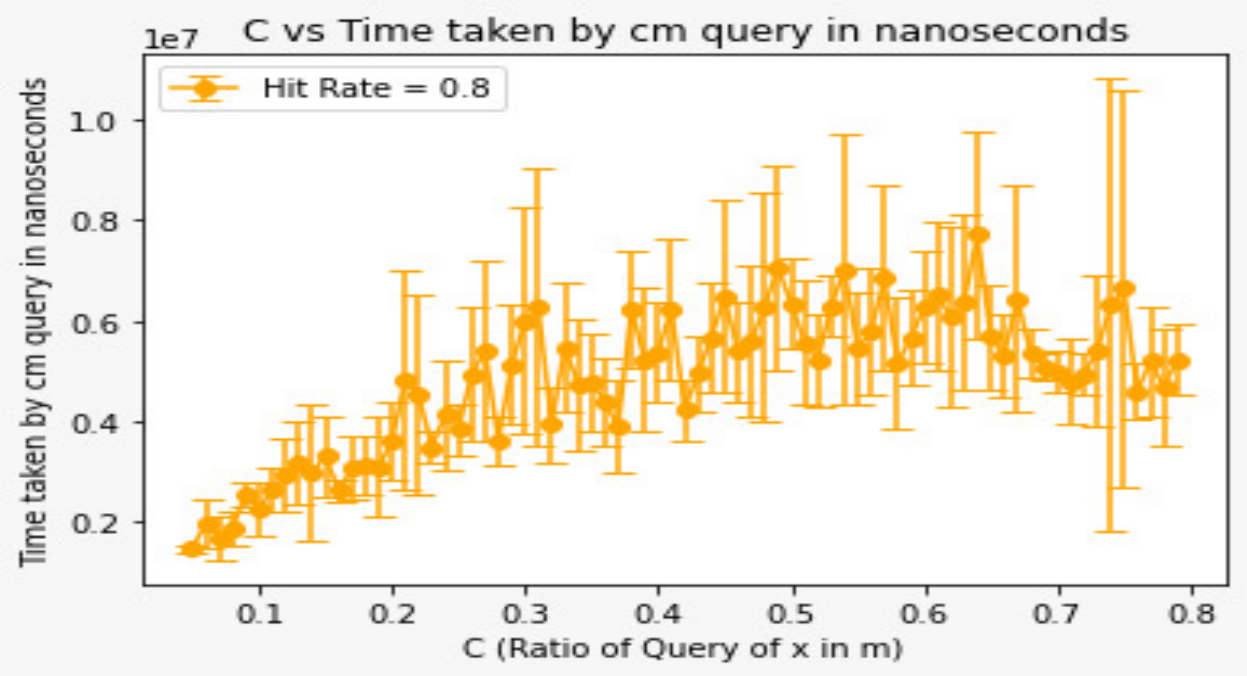


With error bars

2.4.4 Hit rate = 0.8

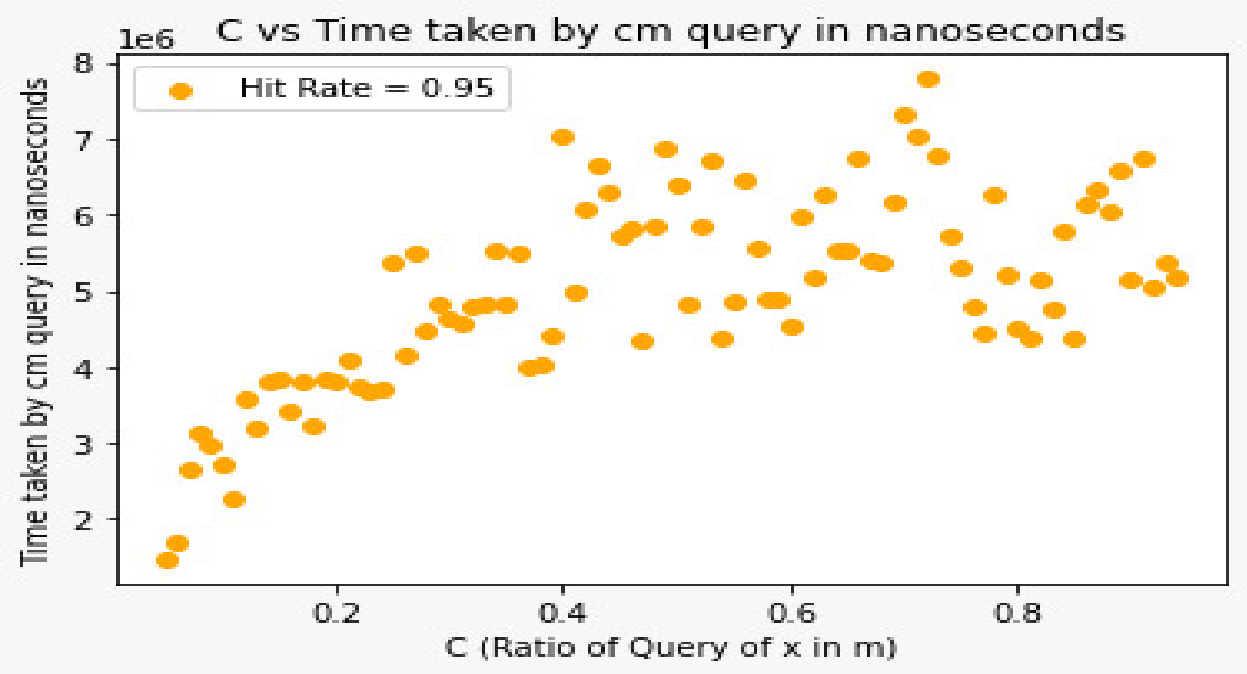


Without error bars

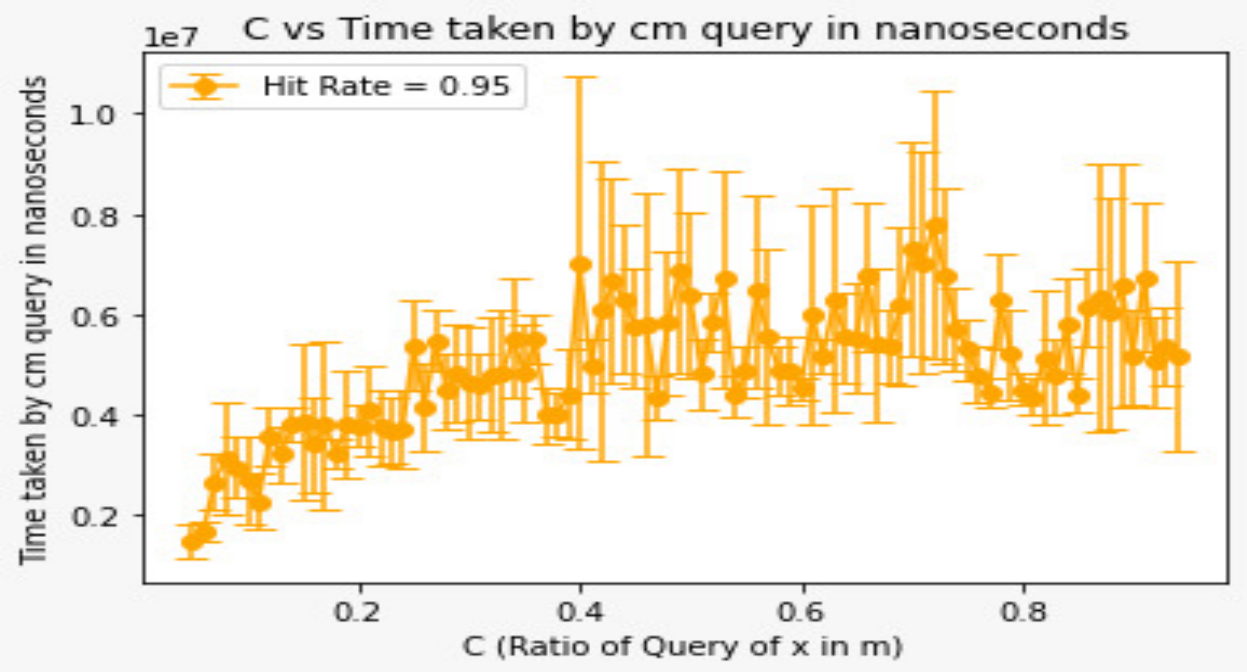


With error bars

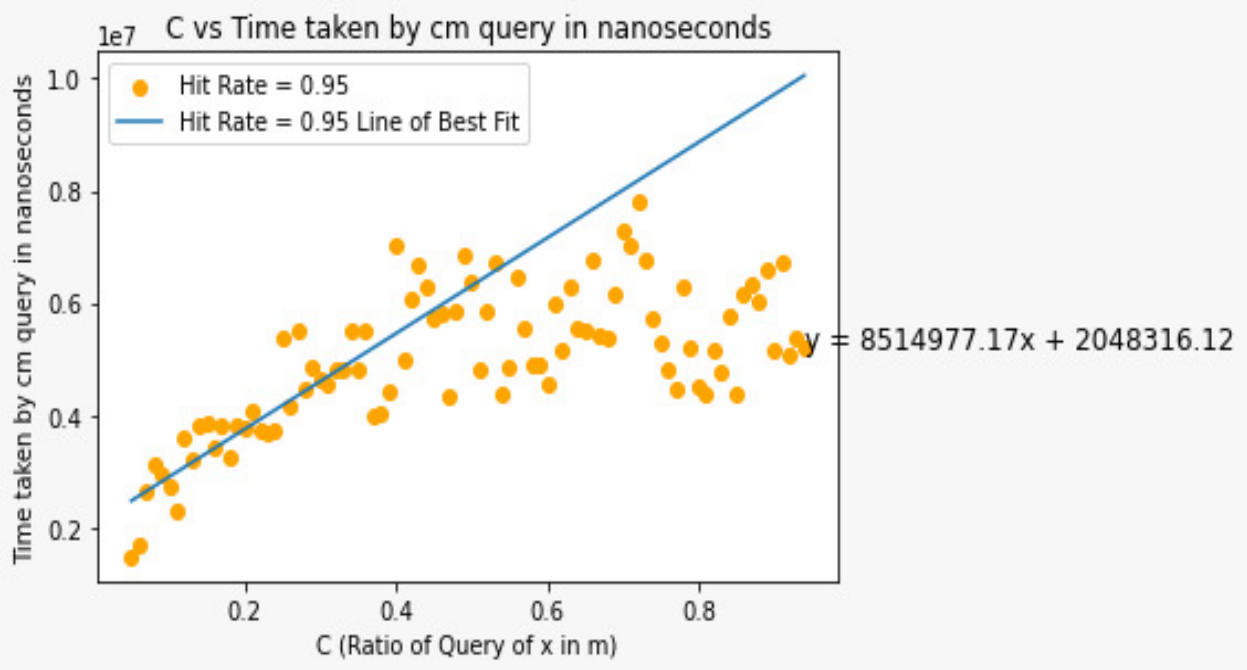
2.4.5 Hit rate = 0.95



Without error bars



With error bars



Showing lines of best fit

We have thus, successfully shown that the time taken for c.m queries for the element x among

the m queries is linearly dependent on the value of c as is also evident on the appropriate linear fits shown for the hit rate = 0.95 case.

Note that: We see a drop off from linearity for large c. Reason for that can be attributed to our chosen number x taking up all of available successful queries (ie hit rate c) hence the value of x won't move much from top of the tree (nothing else is able to splay).
More over we know the constant in O(1) is not fully constant with c (as we regression on.

3 Code Snippet Links:

3.1 Load testing of the Splay Tree:

https://onlinegdb.com/ndOXpFJE4

3.2 Generating test cases:

https://onlinegdb.com/LizY8QncW

4 Conclusion:

We have thus successfully shown that using the Splay Tree data structure containing n distinct valued nodes, executing m queries among which c.m are for a particular element x in the tree is a sequence of operations in which the amortised cost of finding x turns out to be O(1) and also that the constant involved in O(1) is dependent on c given all other parameters are fixed with the relation mentioned in **2.1**.