ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 9| Issue: 12| December 2023|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2023: 8.224 || ISI Value: 1.188

IMPLIMENTATION OF randomize_traverse() MATHEMATICAL MODEL TO CHECK THE OCCURANCE OF GIVEN NUMBER IN A VECTOR. FURTHER COMPARISON BETWEEN randomize_traverse() AND STANDARD linear_search() FUNCTION. - A CASE STUDY

6371 – Cadet A Vinoth Kumar[1]
Class- XII 2023-24, Sainik School Amaravathinagar
Post: Amaravathinagar,Udumalpet Taluka,Tirupur Dt,Tamilnadu State

ABSTRACT

In computer science efficiency of a particular software is determined by the two factors and they are, the amount of memory that is being used to store the data and the amount of time used for processing .These two parameters are used in calculating the space complexity and time complexity of the program.

This manuscript specifically examines the execution of checking the given number exists in a vector or not, if it exists how many times the number occurred by using randomize_traverse() and further comparing with the standard searching methodology linear_search() function. The purpose is to provide a alternative methodology for counting the occurrence of a given number in a vector.

KEYWORDS: ***linear search(ls), randomize traverse(rt), Runtime Complexity (rc), Big OO(n), Big ThetaΘ(n), Big OmegaΩ(n), Generalised approach (ga)***

1. INTRODUCTION

Searching means locating a particular element in a collection of elements. Searching allows you to find if a particular element is present in the list or not. You can use searching to count the number of occurrences of a specific element in a list. This is beneficial when you need to determine how many times a particular value appears in the list.

The uses of searching the occurance of a element in vector is: Counting the occurrences can be helpful in removing duplicate elements from a list and when working with data, counting occurrences can be helpful for summarizing and understanding the distribution of values. It allows you to see how often each value appears and identify outliers.

2. RELATED WORK

Linear search and binary search are two common searching algorithms used in computer science to find the position of a target value within a list or array. Linear search is a simple algorithm that sequentially checks each element of a list until it finds a match with the target value. It starts from the first element and compares it with the target value. If the values are equal, the search is complete. If there is no match, it moves to the next element until the end of the list is reached. Linear search has a time complexity of O(n), where n is the number of elements in the list.

Binary search is a more efficient algorithm for searching elements in a sorted list or array. It compares the target value with the middle element of the list. If they match, the search is complete. If the target value is greater than the middle element, the search continues in the right half of the list. Otherwise, it continues in the left half. This process is repeated by dividing the list in half until the target value is found or the list is exhausted. Binary search has a time complexity of O(log n), where n is the number of elements in the list.

3. METHODOLOGY

The randomize traverse(rt) and linear search(ls) method is used for checking the occurance of a number in a vector ,the randomize traverse(rt) will generate numbers and store it in a list and with the help of this list it will check the occurrence of a number in a vector.

Further to check the efficiency of randomize traverse approach with the linear search methodology of checking the occurance of a number in a vector and examining the length of the input.

ALGORITHM FOR randomize_traverse()
MATHEMATICAL MODEL TO CHECK THE OCCURANCE OF GIVEN NUMBER IN A VECTOR

STEP 01: START
STEP 02: IMPORT RANDOM
STEP 03: ENTER A LIST
STEP 04: ENTER THE SEARCHING KEY
STEP 05: COUNT=0
STEP 06: GENERATE RANDOM NUMBERS IN A LIST a
STEP 07:FOR I IN RANGE(0,LEN(l1))
STEP 08: IF SK==l1[a[i]]:
STEP 09: COUNT+=COUNT
STEP 10: PRINT COUNT
STEP 11: STOP

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 9| Issue: 12| December 2023|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2023: 8.224 || ISI Value: 1.188

PYTHON PROGRAM TO CHECK THE OCCURRANCE OF A NUMBER IN A VECTOR USING randomize traverse(rt).

```
import random
L1=eval(input("enter a list"))
sk=int(input("enter the search number:"))
count=0
a=random.sample(range(0,len(L1)),len(L1))
for i in range(0,len(L1)):
        if sk==L1[a[i]]:
                count=count+1
print(count)
```

ALGORITHM FOR linear search(ls) MATHEMATICAL MODEL TO CHECK THE OCCURANCE OF GIVEN NUMBER IN A VECTOR

STEP 01: START
STEP 02: ENTER A LIST
STEP 03: ENTER THE SEARCHING KEY
STEP 04: COUNT=0
STEP 05: FOR I IN L1:
STEP 06: IF I==SK:
STEP 07: COUNT=COUNT+1
STEP 08: PRINT (COUNT)
STEP 09: STOP

PYTHON PROGRAM TO CHECK THE OCCURRANCE OF A NUMBER IN A VECTOR USING linear search(ls).

```
L1=eval(input("enter a list"))
sk=int(input("enter the search number:"))
count=0
for i in L1:
        if i==sk:
                count=count+1
print("no of numbers:",count)
```

4. COMPLEXITY OF ALGORITHM

In computer science, analysis of algorithms is a very crucial part. It is important to find the most efficient algorithm for solving a problem. It is possible to have many algorithms to solve a problem, but the challenge here is to choose the most efficient one.[1]

There are multiple ways to design an algorithm, or considering which one to implement in an application. When thinking through this, it's crucial to consider the algorithm's **time complexity** and **space complexity**.[2]

5. SPACE COMPLEXITY

The space complexity of an algorithm is the amount of space (or memory) taken by the algorithm to run as a function of its input length, n. Space complexity includes both auxiliary space and space used by the input.[2]
Auxiliary space is the temporary or extra space used by the algorithm while it is being executed. Space complexity of an algorithm is commonly expressed using **Big (O(n))** notation.[2]

The Space complexity is ignored in this research paper, since the space complexity of particular problem is not considered so important.

6. TIME COMPLEXITY

The time complexity of an algorithm is the amount of time taken by the algorithm to complete its process as a function of its input length, n. The time complexity of an algorithm is commonly expressed using asymptotic notations:[2]

Big O - O(n)
Big Theta - Θ(n)
Big Omega - Ω(n)

It's valuable for a programmer to learn how to compare performances of different algorithms and choose the best time-space complexity to solve a particular problem in the most efficient way possible.[2]

Big O notation is used in Computer Science to portrait the performance or complexity of an algorithm.

Big O specifically defines the worst-case scenario of an algorithm, and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm. here O stands for order of growth.

Big Theta(Θ) is used to represent the average case scenario of an algorithm and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

Big Omega (Ω)is used to represent the best case scenario of an algorithm and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

These three methods are the most common and very popular methods of design and analysis of an algorithm which are used for finding the efficiency of the program.

7. RUNTIME COMPLEXITY OF CHECKING A randomize_traverse

Input	Linear search	Randomize traverse
100	0.0	0.0
1000	0.0	0.0
10000	0.0	0.015637874
50000	0.009718418	0.010993957
100000	0.008007764	0.020989698
200000	0.021995544	0.052467088
300000	0.026982545	0.078321424
400000	0.041989564	0.095634537
500000	0.060964345	0.125586376

Graphical Representation of Runtime complexity of both the methods

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 9| Issue: 12| December 2023|| Journal DOI: 10.36713/epra2013 || SJIF Impact Factor 2023: 8.224 || ISI Value: 1.188

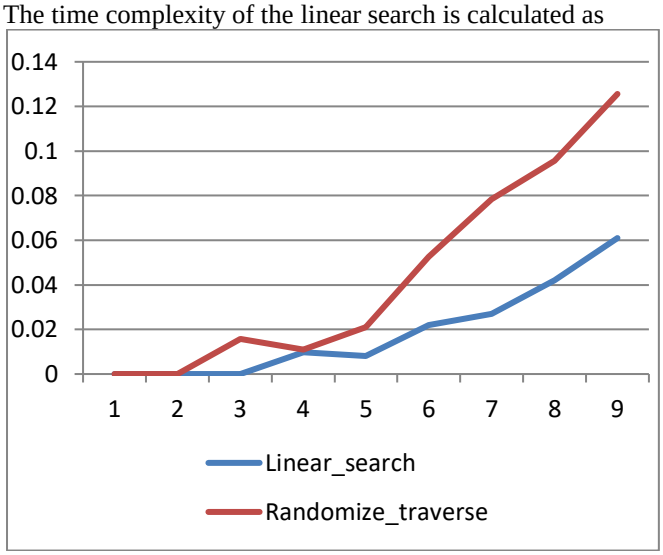
8.RANDOMIZE TRAVERSE-rc

In the randomize traverse approach the program checks for the occurrence of a number in a vector. The time complexity of the algorithm for any value of n is denoted as;

Big (O(n))

9.LINEAR SEARCH-rc

The time complexity of the linear search is calculated as



Big (O(14))

10. CONCLUSION

The Linear search has the greater efficiency for checking occurrence of a number in a vector when comparing with randomize traverse.The linear search proves to be very efficient as it uses the iterative process to check each and every element of the list in order.This proves to be very efficient while dealing with very huge numbers like in zillions.

11.ACKNOWLEDGEMENT

Apart from the efforts of me, the success of any work or project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this research paper.

I express deep sense of gratitude to almighty God for giving me strength for the successful completion of the research paper.

I express my heartfelt gratitude to my parents for constant encouragement while carrying out this research paper.

I express my deep sense of gratitude to the luminary **The Principal Capt. (IN) K Manikandan, Sainik School Amaravathinagar** who has been continuously motivating and extending their helping hand to us.

I express my sincere thanks to the academician The **Vice Principal Wg Commander Deepti Upadhyaya, Sainik School Amaravathinagar**, for constant encouragement and the guidance provided during this research.

My sincere thanks to **Mr.Praveen Kumar MurigeppaJigajinni**, Master In-charge, A guide, Mentor and great motivator,who critically reviewed my paper and helpedin solving each and every problem, occurred during implementation of this research paper.

12. REFERENCES

1. *https://www.freecodecamp.org/news/time-complexity-of-algorithms/*
2. *https://www.educative.io/edpresso/time-complexity-vs-space-complexity*