REVERSE A GIVEN VECTOR USING LINEAR APPROACH *rev_vect() function* AND REVERSE A VECTOR USING *vect_slice() method*. EXAMINING AND COMPARING THE EFFICIENCY OF ALGORITHM OF REVERSING VECTOR USING SLICING *vect_slice() function* WITH REVERSING VECTOR USING LINEAR APPROACH *rev_vect() function* - A CASE STUDY

6840 – Cadet D Deepak[1]
Class- XII 2023-24, Sainik School Amaravathinagar
Post: Amaravathinagar,Udumalpet Taluka,Tirupur Dt,Tamilnadu State

ABSTRACT
In computer science, design analysis of algorithms is a very significant part. It is crucial to find the most efficient algorithm for solving a problem. There can be many algorithms to solve a problem, but the challenge here is to choose the most efficient one. There are multiple ways to design an algorithm, or taking into account which one to implement while solving critical applications in day to day life. When thinking about this, it is crucial to consider the algorithm's time complexity and space complexity.

This manuscript specifically examines the execution of reversing a vector using rev_vect() function with vect_slice(). Further comparing with the linear or sequential approach of reversing a vector with slicing methodology by calculating the time complexity of both the algorithms. In addition space complexity is also examined. The purpose is to provide efficient algorithm to reverse a vector.

KEYWORDS: ***Linear Approach(la), Vector Slice (vs), Reverse Vector(rv), Runtime Complexity (rc), Big OO(n), Big ThetaΘ(n), Big OmegaΩ(n), Generalised approach (ga).***

1. INTRODUCTION

A vector, in programming, is a type of array that is one dimensional. A vector is often represented as a 1-dimensional array of numbers, referred to as components and is displayed either in column form or row form. Vector is a logical element in programming languages that are used for storing data. Vector plays a crucial role in storing the data.

In modern programming libraries this name "vector" has come to generally mean a variable sized sequence of values.Vectors are the same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted.

2. RELATED WORK

Many mathematics research scholars have done extensive research and have changed the way we reverse a number. There are many methods discovered by many programmers but this is one of the efficient methods where time consumption is very less .

3. METHODOLOGY

Slicing is the extraction of a part of a string, list, or tuple. It enables users to access the specific range of elements by mentioning their indices. Syntax: Object [start:stop:step] "Start" specifies the starting index of a slice. "Stop" specifies the ending element of a slice. It is simple to execute. First import the time module and use the function. Then get the input from the user in vector i.e, in list form. Assign values m=0 and n=len(l1)-1 then use assignment operator and assign k=l1[m] , l1[m]=l1[n] and l1[n]=k.Increase the value of m by 1 and decrease the value of 1.Then using slicing method that is l1[::-1] finish the code.

ALGORITHM

```
STEP 01: START
STEP 02: IMPORT TIME
STEP 03: GET INPUT OF LIST l1
STEP 04: ASSIGN M=0 AND N=LEN(L1)-1
STEP 05: START TIME FOR rev_vect()
STEP 06: WHILE M<N REPEAT STEPS 6 TO 8
STEP 07: ASSIGN K=L1[M] AND L1[M]=L1[N]
STEP 08: SET VALUE OF L1[N]=K
STEP 09:INCREASE THE VALUE OF M AND
REDUCE THE VALUE OF N BY 1
STEP 10: END TIME FOR vect_slice()
STEP 11: START TIME FOR vect_slice()
STEP 12: USE l1[::-1]
STEP 13: END TIME FOR vect_slice()
```

PYTHON PROGRAM TO REVERSE A VECTOR AND FINDING THE EFFICIENCY OF THE CODE.

```
import time
def rev_vect(l1):
   m=0
   n=len(l1)-1
   start=time.time()
   while m<n:
      k=l1[m]
      l1[m]=l1[n]
      l1[n]=k
      m+=1
      n-=1
   end=time.time()
   print("Time taken for completion of execution by method 1:",(end-start)*10**3)
   return l1
def vect_slice(l1):
   start=time.time()
   l1[::-1]
   end=time.time()
   print("Time taken for completion of execution by method 2:",(end-start)*10**3)
   return l1
def main():
   l1=eval(input("Enter a list:"))
   a=rev_vect(l1)
   print("The reversed list method 1:",a)
   l=vect_slice(l1)
   print("The reversed list method 2:",l)
main()
```

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.[2]

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**.[3]

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.[3]

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.[3]

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:[3]

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.[3]

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.

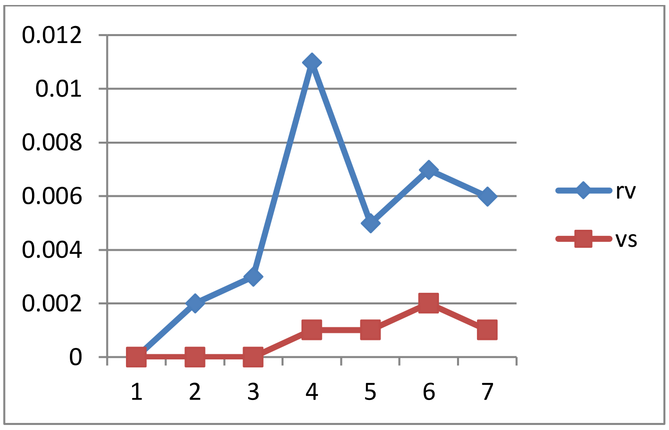
Big Theta(Θ) is used to represent the average case scenario of an algorithm and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

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 REVERSING A VECTOR

Input	rv	vs
1000	0.0	0.0
6000	0.001994609	0.0
50001	0.002992391	0.0
76789	0.010969877	0.001000642
83567	0.004987955	0.000997066
98765	0.006981372	0.001996994
100000	0.005983829	0.000998020



Graphical Representation of Runtime complexity of both the methods

8.GENERALISED APPROACH - *rc*

In the normal approach the program reverses the vector. The time complexity of the algorithm for worst case is denoted as:

Big (O(n))

9.LUCAS METHOD (LMM) - *rc*

The time complexity of the reversing a vectoris calculated as

Big (O(14))

10. CONCLUSION

The Linear method and slice method have the greater efficiency for reversing a vector when comparing with general approach. Further it is also observed that reversing a vectors and storing in a file is one time process and it is time consuming but once the file is prepared the performance of the code is much higher than the normal approach. In addition to this it is also observed that the execution of expression also depends on the hardware configuration.

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 Wing Commander DEEPTI UPPADHAYAY and Administrative officer Lieutenant colonel DEEPU** 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 helped in 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*