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

CONCATENATION OF TWO OR MORE VECTORS USING *concat_vect() function* A LINEAR METHOD AND CONCATENATION OF TWO OR MORE VECTORS USING *concat_assign_vect()* FUNCTION USING SIMPLIFIED ASSIGNMENT OPERATION. FURTHER ASSESSING THE TIME COMPLEXITY OF BOTH ALGORITHMIC FUNCTIONS - A CASE STUDY

6440– Cadet Prasant Kumar[1]
Class- XII 2023-24, Sainik School Amaravathinagar
Post: Amaravathinagar,Udumalpet Taluka,Tirupur Dt,Tamilnadu State

ABSTRACT

Design and analysis of algorithm is one of the core component of implementation of any software. Its a very pivotal part. It is important to plan and design the most efficient algorithm for solving a problem. There are various means of algorithms to solve a problem, but the challenge here is to choose the most effectual and well organised algorithm.

This manuscript indeed describes the concatenation of list of elements using a linear method called concat_vetc() function and concatenates two or more vectors, further the same functionality is being executed by using concat_assign_vect() function simplified assignment operation, in addition to it the time complexity of the function is being calculated to examine the performance of both the algorithms. The purpose is to provide efficient algorithm for concatenation of more than one vector.

KEYWORDS: ***Simplified Assignment operations(sa), concatenation of vector (cv), linear method(lm), Runtime Complexity (rc), Big OO(n), Big ThetaΘ(n), Big OmegaΩ(n), Generalised approach (ga)***

1. INTRODUCTION

A List is an ordered data structure with elements separated by comma and enclosed within square brackets. The operation of joining string end to end. A list is one of the most common data structures used, not just in Python but in programming in general. It is an ordered and mutable Python container. To create a list, the elements are placed inside square brackets ([]) and each element is separated by a comma

2. RELATED WORK

The most conventional method to perform the list concatenation, the use of “+” operator can easily add the whole of one list behind the other list and hence perform the concatenation.
List comprehension can also accomplish this task of list concatenation. In this case, a new list is created, but this method is a one-liner alternative to the loop method discussed above.

3. METHODOLOGY

concat_assign_vect() function utilizes the input_vector_elements() function twice to get two lists, L1 and L2, from user input. After obtaining both lists, it concatenates them into a new list L3 using the + operator and then prints the concatenated list.
concat_vect() function collects two lists, L1 and L2,fromtheuserusingtheinput_vector_elements() function. However, instead of using the + operator for concatenation, it uses a for loop to iterate through the elements in L2 and appends each element to L1. It then prints L1 after each element is appended. This function essentially prints the intermediate state of the L1 list as elements from L2 are added to it.

ALGORITHM FOR concat_assign_vect() and concat_vect()
STEP 01: START
STEP 02: ASK THE USER TO INPUT THE NUMBER OF ELEMENTS, N.
STEP 03: INITIALIZE AN EMPTY LIST, L1.
STEP 04: Use A For Loop To Iterate From 0 To N-1.
STEP 05: INSIDE THE LOOP, ASK THE USER TO INPUT AN ELEMENT, ELE.
STEP 06: APPEND THE ELEMENT ELE TO THE LIST L1.
STEP 07: RETURN THE LIST L1 CONTAINING THE INPUT ELEMENTS.
STEP08: DEFINE THE CONCAT_ASSIGN_VECT()FUNCTION
STEP09: CALL THE INPUT_VECTOR _ELEMENTS FUNCTIONS TWICE TO GET TWO LISTS,L1 AND L2, FROM USERS.

STEP 10: CONCATENATE THE TWO LISTS USING THE + OPERATOR AND STORE THE RESULT IN A NEW LIST, L3.
STEP 11: PRINT THE CONCATENATED LIST, L3.
STEP 12: DEFINE THE CONCAT_VECT() FUNCTION
STEP13: CALL THE INPUT VECTOR_ELEMENT()FUNCTION TWICE TO GET TWO LISTS, L1 AND L2, FROM THE USER.
STEP 14: INITIALIZE A LOOP TO ITERATE THROUGH ELEMENTS IN L2.
STEP 15: INSIDE THE LOOP,TAKE AN ELEMENT, X, FROM L2.
STEP 16: APPEND THIS ELEMENT TO THE LIST L1.
STEP 17: APPEND THIS ELEMENT TO THE LIST L1.
STEP 18: PRINT THE UPDATED LIST L1 AFTER ADDING EACH ELEMENT FROM L2.
STEP 19:STOP

PYTHON PROGRAM FOR concat_vect() AND concat_assign_vect():

```
import time
import random
def concat_assign_vect():
    L1=[]
    L2=[]
    for i in range(0,100):
        L1.append(random.randint(0, i))
    for i in range(0,100):
        L2.append(random.randint(0, i))

    start = time.time()
    L3=L1+L2
    end = time.time()
    print("Total Time For Enumeration Using Assign Vector is : ",end - start)
    print(L3)

def concat_vect():
    L1=[]
    L2=[]
    for i in range(0,100):
        L1.append(random.randint(0, i))
    for i in range(0,100):
        L2.append(random.randint(0, i))
    start = time.time()
    for x in L2:
        L1.append(x)
        print(L1)
    end = time.time()
    print("Total Time For Enumeration Using CONCAT VECTOR is : ",end - start)
concat_assign_vect()
concat_vect()
```

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.

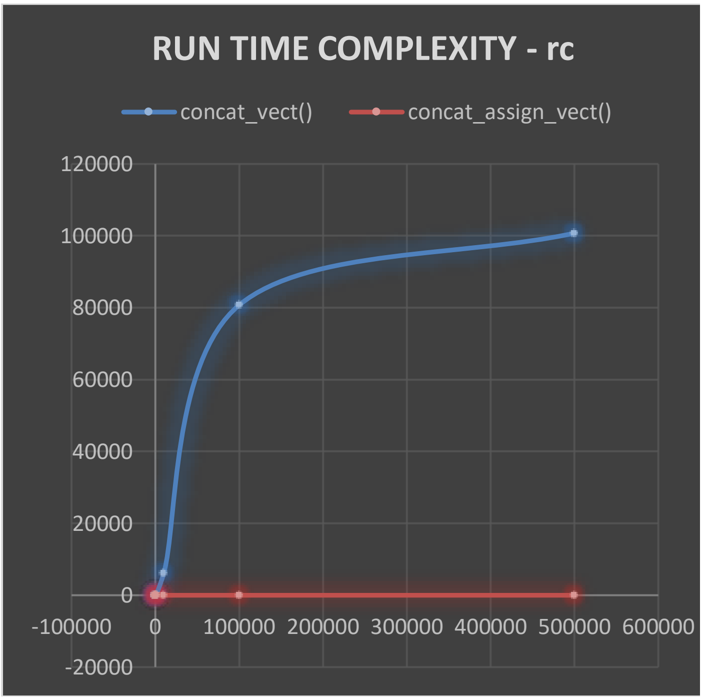
Big Omega (Ω)is used to represent the best case scenario of an algorithm and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

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

Input	concat_vect()	concat_assign_vect()
5	0.015622138	0.0
10	0.031253814	0.0
100	1.890174627	0.0
500	40.86535811	0.0
1000	87.53732275	0.0
10000	6073.405734	0.0
100000	80735.405734	0.0
500000	100735.40573	0.0

Graphical Representation of Runtime complexity of both the methods



8.GENERALISED APPROACH - *rc*

In the normal approach the program checks for the given number prime or not. The time complexity of the algorithm for worst case is denoted as:

Big (O(n))

9.concat_vect() METHOD (LMM) - *rc*

The time complexity of the **concat_vect()** Method is calculated as

Big (O(n))

10. CONCLUSION

The concat_vect() mathematical methodology has the greater efficiency for checking prime when comparing with general approach. Further it is also observed that generating prime series 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 Wg Cmd Deepti Upadhyay, 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 helped in solving each and every problem, occurred during implementation of this research paper.

12. REFERENCES

1. *https://en.wikipedia.org/wiki/Concat_vect()_number*
2. *https://www.freecodecamp.org/news/time-complexity-of-algorithms/*
3. *https://www.educative.io/edpresso/time-complexity-vs-space-complexity*