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 replica_vector() TO CHECK THE EFFICIENCY OF REPLICATION OF THE ELEMENTS OF GIVEN VECTOR USING append() METHOD OF LIST, TO CREATE AND REPLICATE VECTOR WITH THE copy() function of the copy module WITH RESPECT TO THE AMOUNT OF ELEMENTS OF A VECTOR. - A CASE STUDY

6394 – Cadet M P Indresh[1]
Class- XII 2023-24, Sainik School Amaravathinagar
Post: Amaravathinagar,Udumalpet Taluka,Tirupur Dt,Tamilnadu State

ABSTRACT

All the programming languages support copying functions which is used to copy or create replica of objects these objects represent the set of possible interpretations of any possibly-infinite sequence of symbols.

This manuscript reveals replica of vector using replica_vector () function and examines the execution of replication of elements of the vector. Further comparing of replica_vector () approach with the replicating of vector elements using copy() function in copy module in python. In addition to this time complexity and space complexity of an algorithm is examined. The purpose of this manuscript is to examine two different approaches efficiency when it comes to handling large data sets.

KEYWORDS: ***replica vector (Rv), copy() function (Cf), Runtime Complexity (rc), Big O𝑂(n), Big Theta𝛩(n), Big Omega𝛺(n), Generalised approach (ga)***

1. INTRODUCTION

The copying modules are basically functions which are generally used to create a new object of same data type of the data which is to be copied and copies all the objects present in it to the old object into the newly created object. This function is usually found in nearly all programming languages available across the world under different names usually providing the same feautures

The Copying modules mostly consists of two types of copying namely deep copy and shallow copy. In deep copy a collection of objects is constructed at first and the newly created object is recursively populated with copies of object found in the original.Wheras in shallow copy we create a duplicate of the original element but do not make copies of any elements referenced by the original element. In *copy()* function (Cf) we use the methodology of shallow copy .Therefore we tend to duplicate the elements in a new object rather than copying every elements like deep copy as it tends to be more effevtive.

In *replica_vector(Rv)* we use the append() fuction of list data type in Python to copy every object individually.

2. RELATED WORK

Many programmers around the world have researched thoroughly about the copying function around the world and have increased the number of ways through which the elements in a list can be copied very efficiently even while dealing with number of elements numbering upto zillions .

3. METHODOLOGY

The *copy() function(Cf)* is used to copy the elements of a list using the method of shallow copying.The time module will be imported using the import function.Two list data types will be created using the list function.

Inside a iteration upto certain number till which the number of elements in the first list or the list to be copied is needed.

Then using the *copy() function(Cf)* of the copy module of the Python you can assign the second list to the *copy() function(Cf)* and in the function where the argument of the list to be copied is required put the name of the first list.

In the *replica_vector(Rv)* we first create two list data types using the list function. The first list is then iterated to a certain number of times and the number of times the iteration is completed a new number will be added to the first list data type.

After creating a list data type of certain number of elements ,once again a new iteration is created to transverse through each and every elements of the first data type and each element transverse individually is then individually appended into the second list data type using the append() function in Python.

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

ALGORITHM FOR COPYING ELEMENTS OF A LIST DATA TYPE USING copy() function(Cf)
STEP 01: START
STEP 02: L1=LIST
STEP 03: L2=LIST
STEP 04: FOR I IN RANGE (1,N):
STEP 05:APPEND IN L1
STEP 06:L2=copy.copy(L1)
STEP 07: STOP
PYTHON PROGRAM TO COPY THE ELEMENTS OF A LIST DATA TYPE USING THE copy() function(Cf)

```
import copy
def copy_():
    l1=list()
    l2=list()
    for i in range(1,N):
        l1.append(i)
    l2=copy.copy(l1)
copy_()
```

ALGORITHM TO COPY THE ELEMENTS IN A LIST DATA TYPE USING replica_vector()(Rv)
STEP 01: START
STEP 02: L1=LIST
STEP 03:L2=LIST
STEP 04:FOR I IN RANGE (1,N):
STEP 05: APPEND IN THE L1
STEP 06: FOR I IN L1:
STEP 07: APPEND IN THE L2
STEP 08: STOP

PYTHON PROGRAM TO COPY THE ELEMENTS OF A LIST DATA TYPE USING THE replica_vector()(Rv)

```
def list_method():
    l1=list()
    l2=list()
    for i in range(1,N):
        l1.append(i)
    begin=time.time()
    for i in l1:
        l2.append(i)
list_method()
```

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.

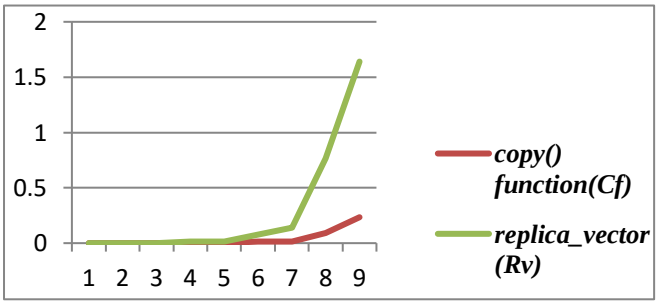
Big Omega (Ω)is used to represent the best case scenario of an algorithm and can be used to describe the execution time required or the space used (e.g. in memory or on disk) by an algorithm.

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.

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

7. RUNTIME COMPLEXITY OF CHECKING A PRIME NUMBER

Input	copy() function(Cf)	replica_vector (Rv)
100	0.0	0.0
10000	0.0	0.0
20000	0.0	0.0
100000	0.0	0.015615940
200000	0.0	0.016032696
1000000	0.015654087	0.078124761
2000000	0.0156323909	0.140627384
10000000	0.0937488079	0.765634536
20000000	0.2343809604	1.6406536102



Graphical Representation of Runtime complexity of both the methods

8.Time Complexity of copy() function (Cf)

In the copy() function of the program the time copmplexity of the code for any value of N in the given code is given as

Big (O(n))

9.Time Complexity of replica_vector(Rv)

The time complexity of the replica_vector()(Rv) for any value of N of the given code is

Big (O(n))

10. CONCLUSION

The copy() function(Cf) of the copy module of the Python has greater efficiency compared to that of the replica_vector(Rv) which employs the append() function of the list data type in Python.Further the copy function also uses a method of shallow copying which allows it to work very efficiently when dealing with bigger number of elements in the list data type of the Python.

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 Cdr Deepti Upadyay,Sainik School Amaravathinagar**, for constant encouragement and the guidance provided during this research.

My sincere thanks to **Mr.Praveen Kumar Murigeppa Jigajinni**, 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*