ASYMPTOTIC PERFORMANCES OF POPULAR PROGRAMMING LANGUAGES FOR POPULAR SORTING ALGORITHMS

Mr. Omar Khan Durrani
Dept.t of Computer Science & Engineering, Ghousia College of Engneering, Ramanagram India

Dr.Sayed Abdulhayan
Dept.t of Computer Science & Engineering, P A college of Engineering, Mangalore, India

Abstract—In modern technology where disciplines like data sciences, data Analytics, and machine learning are emerging and infrastructure being set for Internet of things, important operations like sorting and searching plays a vital role. Hence efficient sorting along with searching is still important for the efficiency of other algorithms. These emerging areas are using Python and Java languages than the general purpose languages C and C++ for its implementation. Also, it is seen fever asymptotic analysis is made on popular sorting algorithms using Java and Python though there exist sufficient asymptotic analysis and results using like C/C++. In this paper we have made a thorough asymptotic performance measures by implementing popular sorting algorithm using C++, Python and Java languages. We conducted our experiments on Pseudo random data for average case analysis and ordered(Presorted) data for worst case analysis. The research shows that Merge sort doing well for its Python implementation instead of Quick sort which very much lacks in its Performance. Quick sort remains excellent in performance for Java implementation. C++ implementation performing as per the Verified behaviors of Popular sorting Algorithms. Finally we have indicated the miss behaviors made by Java and Python implementation results.

Keywords—Popular sorting algorithms, asymptotic analysis, test bed, python, java , C++, implementation, observations, modern compiler design;

I. INTRODUCTION

In computer science, a sorting algorithm is an algorithm that puts elements of a list in a certain order. The most-used orders are numerical order and lexicographical order. Efficient sorting is important for optimizing the use of other algorithms (such as search, merge, Graph etc.) that require sorted lists to work correctly.

Since the dawn of computing, the sorting problem has attracted a great deal of research, perhaps due to the complexity of solving it efficiently despite its simple, familiar statement. For example, bubble sort was analyzed as early as 1956. later other algorithms like selection sort, insertion sort merge sort, quick sort and many others methods were introduced which we term as popular sorting algorithms. Generally, the sorting algorithms considered are classified into n2 class and nlog n class with respect to their order of growth (related to time complexities)

which are are discussed in [2][5]. These algorithms have been Asymptotically experimented using C and C++ in [1][8] and verified with respect to their behaviors. By Asymptotic we mean we have considered a large size input. In Section III we have briefly explained the Asymptotic Analysis.

Later when Java and Python were introduced in 1990s they gained a good popularity and acceptance in their own arena of software development. As part our research study to we wanted to realize the behaviors of popular sorting algorithms by implementing using C++, Java and Python languages. These three languages, we have named as Popular Programming Languages with reference to a latest report in [14].

Section II briefs about the Popular Programming Languages. The Subsections A, B, C illustrates important features of Popular Programming Languages in[35][36]. Section III briefly explains Asymptotic Analysis and Notations. In Section IV we have briefly explained the theoretical aspects of the popular sorting algorithm namely merge sort, quick sort ,Insertion sort, Selection and Bubble sort. Section V speaks about the related work with respect to our experimental studies. In section VI we exhibit the experiment conduction and measurements and discussion over the results. The experiments were conducted on two different data sets for analyzing the algorithms average case and worst case. The experiments have shown interesting findings which made us to give a satisfactory Conclusion with respect to the performances and compiler design especially of Python and Java. Further we have listed some future works to be made and suggestions to the theoreticians and Practitioners.

II. POPULAR PROGRAMMING LANGUAGES

A Computer is a machine which works on set of Instructions (a Program) given to it, which initially are made to store in its Memory and later are executed by the computer to act upon. Initially programming started with binary codes which we called it as Machine language (the language of computer) which needed highly skilled professionals. Later to make it simpler mnemonic symbolic codes which felicitated in remembering and identifying the instructions, hence it was named Assembly language. When computer programming started to benefit because of its performance in the field of Science and Technology and for some famous Commercial companies, the computer scientist designed interpreters and compilers which translated the computer programs written in very much human readable form making programming more Programmer friendly and hence High level language gradually High level programming languages like BASIC, FORTRAN, PASCAL, COBOL etc. came into the market. With this, Problem Solving through Computer was like a innovation for all its applications.

As the applications of programming increased, and as part of a natural growth, the Programming community felt difficulties, such as the lines of codes were increasing in the application programs, difficulty in debugging, etc. Later the Software Crisis in 1968 made the software sector to conduct a Conference [22], which highlighted the problems which the software development was facing , like: Project running over-budget, Software was very inefficient, Software was low quality, Software often did not meet requirements, Projects were unmanageable and code difficult to maintain, Software was never delivered in time.

Soon after this Brian.W.Kernighan and Dennis.M.Ritchie, introduced C (in 1970s) in the Third Generation period, The introduction of Cas a programming language which brought a tremendous change as it featured support for instruction level programming being a high level

language. It topped over the other programming languages of its generation like Fortran, Pascal, COBOL due to its rich data types structures and libraries and many other features [28]. C was so strong and rich it was used to develop various operating system.

Later as time progressed and software development for bigger projects were on demand list yet the issues of software crisis were not fully solved, also the software community were realizing the weaknesses of Procedure Oriented concepts. On the other side Object Oriented Technology was into discussion which made to introduce Smalltalk a object Oriented language in 1972. In 1985 C++ was introduced by Bjarne Stroustrup as an extension to C Language. Though C++ exhibited many of the Object Oriented Concepts but could not give much of the performance rather it achieved the opportunity to introduce most of the Object Oriented concepts as a programming language [29]. Hence the invent of object oriented technology solved the issues of Software crisis.

At the same time we find many researchers like C.A.R Hoare in [31], A.L.Tharp in [24] and many others promoted the Programming language design in their own perspective which has influenced the compiler designers not only restricted to their time rather a foresee.

In 1991, Java was proposed by James Gosling and his team from Sun Microsystems and was released in 1995 its special features mentioned in [16], which helped to fulfill the need of time. Soon, along with Java, Python was introduced in 1989, by Guido van Rossum at Central Wiskunde & Informatica (CWI) in the Netherlands which had not matured much in its early days but later took up its phase of growth as soon as its 3.x Version was released. Past a decade Java and Python along with C and C++ have became the popular programming languages as per a report in [13]. Java and Python are on top(since 2018) in the development of modern applications and technologies than the C and C++ as shown in [14].

Both Java and Python were born during the Fourth Generation Period which demanded to make programming as simple as possible by mean of programming in natural human language[23][24][31]. 4GLs aim to make programming easier, more efficient and more effective for the users with less programming skills. The objectives of 4GL was at-most fulfilled by Python , where as Java lacked in it. Java due to its special features of Multi-threading, Internet programming took up its own arena. With reference to the above information we term C++, Java and Python as Popular Programming Languages.

A. Features of C++ Language

1. Apart from the main features of object orientation, C++ programming language has all the features and characteristics of the C programming language[29].
2. It supports modularity: Commonly usable modular programs.
3. It is efficient and close to the machine programming.
4. C++ language can be defined as a hybrid language which includes the same functionality of C.
5. Large amount of the existing C source code can be used in C++ programs[29].
6. It is one of the most popular and suitable languages for developing applications on PCs and UNIX systems.
7. When developing system software like Drivers and Operating Systems, C still holds strong as compared to C++.
8. The lack of some features in C++ language have increased the complexity of developing concise and robust distributed applications.

9. It is not a purely Object-Oriented Programming language.
10. Though being Object Oriented it is also considered a General Purpose Programming Language, which the Java and Python lack.

B. Features of Java Language

1. Java's most prominent feature is that it is platform independent.
2. Java is easy to use, write, compile, debug, and learn compared to C, C++.
3. Being Platform-independent, distributed language, supporting multi-threading, providing Automatic Garbage Collection, etc., are some of the specialties of Java[16].
4. Java particularly focuses on storage and not on the backup of data.
5. Memory management is costly as large memory space is required.
6. Java is slower and memory consuming when compared with C or C++[11].
7. The programs written in Java runs faster than Python but is slower when compared with C++.
8. Java is a compiled language that is statically typed i.e., their variables are to be declared before assigning values.
9. Java libraries are built using Multi-threading which make their functioning faster[16].
10. Java programs that run on a Java Virtual Machine tend to perform slower than equivalent programs written in C++. The system neutrality of byte-code acts as a disadvantage where performance is concerned. This is because code optimization relies heavily on system-specific features. Since Java byte-code is system-neutral, it cannot be optimized for a specific hardware set[27].

C. Features of Python Language

1. Python is easy to read, learn, and write making it beginner-friendly. It is a productive language[38].
2. Lesser code (utilizes less memory) is required by Python compared to other languages like C++, Java for the same task.
3. Python is interpreted and is dynamically typed, meaning that the programmer does not need to define the data-type of the variables and no need for compilation and with the use of the interactive command-line, they get prompt assessment without having to wait for the whole program to be finished[38].
4. Due to its vast libraries the programmer can execute complex functions easily.
5. Python has become the fastest-growing language. The popularity of Python in data science is one of the main reasons for the hike of Python[13].
6. As Python is an interpreted language, it is slower in execution when compared with other languages.
7. Python is not favorable for mobile development.
8. The Global Interpreter Lock (GIL) of Python allows the execution of only one thread at a time.
9. As Python is dynamically typed raises run time error leading to restriction in design[38].
10. Due to 2-tier hierarchy Python can suffer Performance loss[38].

III. ASYMPTOTIC ANALYSIS

Asymptotic is the study of function say f(n), of a parameter n(in our study it is the input size), as n become larger and larger without any bounds. Here we are concerned with how the runtime

of an algorithm increases with the size of the input. We can do analysis of algorithms which will accurately reflect the way in which the computation time will increase with the size, but ignores the other details with little effect on total. It permits us to provide upper and lower bounds on the value of a function f for suitably large values. The efficiency analysis frame work in [2] concentrates on order of growth of an algorithms' basic operation count as the principal indicator of the algorithms' efficiency. To compare and rank such orders of growth, computer scientist use three notations:

- O(big-oh)
- Ω(big-omega)
- Θ(big-theta)

O(big-oh) bounds from above, Ω (big-omega) from bottom and Θ(big-theta) does from both above and below, details study is given in [1][2]. The performance of algorithms are only possible by considering asymptotic input size or data sets while conducting the experiments.

IV. POPULAR SORTING ALGORITHMS

A. Merge Sort

The merge sort splits the list to be sorted into two equal halves, and places them in separate arrays. This sorting method is an example of the DIVIDE-AND-CONQUER paradigm i.e. it breaks the data into two halves and then sorts the two half data sets recursively, and finally merges them to obtain the complete sorted list. The merge sort is a comparison sort and has an algorithmic complexity of O (n log n). Elementary implementations of the merge sort make use of two arrays, one for each half of the data set. The following image shown in figure 1 depicts the complete procedure of merge sort.

Pros:Marginally faster than the heap sort for larger sets as tested in [8]. Merge Sort always does lesser number of comparisons than Quick Sort. Worst case for merge sort does about 39% less comparisons against quick sort's average case.

Cons:At least twice the memory requirements of the other sorts because it is recursive. This is the BIGGEST cause for concern as its space complexity is very high. It requires about a Θ (n) auxiliary space for its working at every level. Function overhead calls (2n-1) are much more than those for quick sort (n). This causes it to take more time marginally to sort the input data.

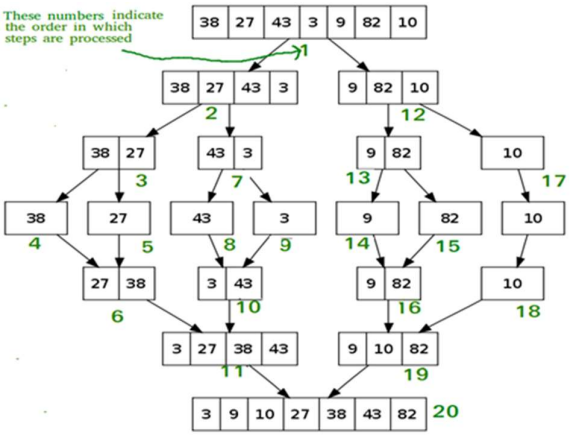


Figure 1: shows recursive dividing and Merge

B. Quick Sort

Quick Sort is an algorithm based on the DIVIDE-AND-CONQUER paradigm that selects a pivot element (in our example it is the right most element of list) and reorders the given list in

such a way that all elements smaller to it are on one side and those bigger than it is on the other side. Then the sub lists are recursively sorted until the list gets completely sorted as shown in Figure 2 also you observe the Pivot element occupying its position as part of list being sorted. The time complexity of this algorithm is O (n log n) due partitioning into two along with two scans in opposite directions.

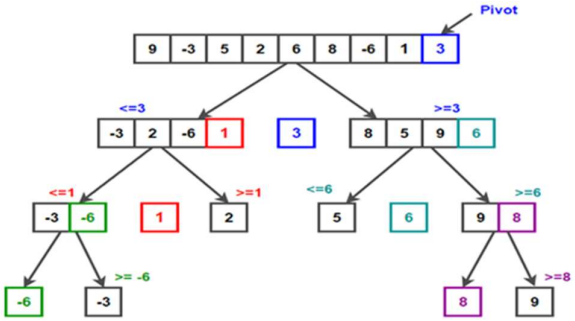
Figure 2: shows partitioning considering Pivot element

Pros:All comparisons are being done with a single pivot value, which can be stored in a register. The list is being traversed sequentially, which produces very good locality of reference and cache behavior for arrays and hence to speedup. Quick sort is the fastest algorithm on randomly distributed data set. Our paper [15] has shown a way to make the quick sort worst case performance O(n) instead of O(nlogn).

Cons:Auxiliary space used in the average case for implementing recursive function calls is O (log n) and hence proves to be a bit space costly, especially when it comes to large data sets. It is also seen that recursion stack overflow being occurred when large data set are considered in programming languages like Java and very much in Python, hence we need to switch to its iterative versions.

C. Insertion Sort, Selection Sort and Bubble Sort

The bubble sort is the oldest and simplest sort in use. The bubble sort works by comparing each item in the list with the item next to it, and swapping them if required. The algorithm repeats this process until it makes a pass all the way through the list without swapping any items. This causes larger values to "bubble" to the end of the list while smaller value "sink" towards the beginning of the list.



Pros:Simplicity and ease of implementation. Auxiliary Space used is O (1).

Cons:Very inefficient. General complexity is O (n2). Best case complexity is O (n).

Selection Sort's philosophy most closely matches human intuition: It finds the largest element and puts it in its place. Then it finds the next largest and places it and so on until the array is sorted. To put an element in its place, it trades positions with the element in that location (this is called a swap). As a result, the array will have a section that is sorted growing from the end of the array and the rest of the array will remain unsorted

Pros:Specifically an in-place comparison sort. Selection sort is noted for its simplicity, and also has performance advantages over more complicated algorithms in certain situations. It yields a 40% performance improvement over the bubble sort[8][17].

Cons: It has O (n2) complexity, making it inefficient on large lists. Generally performs worse than the similar insertion sort.

The insertion sort works just like its name suggests- it inserts each item into its proper place in the final list. The simplest implementation of this requires two list structures - the source list and the list into which sorted items are inserted. To save memory, most implementations use an in-place sort that works by moving the current item past the already sorted items and repeatedly swapping it with the preceding item until it is in place.

Pros:Auxiliary space used is O (1). The insertion sort is twice as efficient as the bubble sort. Performs well in sorting small sample sizes along with nlogn class algorithms which go slow on small data sets.

Cons:General Complexity is O (n2). Best Case is O (n) when the list is already sorted

These basic algorithms, the Bubble sort or Selection sort are inefficient in real world applications. They have importance in academic studies but are not in concern at practices. As our intention of this paper is to make analysis and performance measurement of sorting algorithms implemented using the Popular Programming Languages C++, Java and Python.

V. RELATED WORK

Sartaj Sahni and Lavetin .A in their books [1][2] both have very clearly described the sorting algorithms by representing them in terms of equations belonging to orders of growth(considering time), he further makes Asymptotic analysis of each sorting algorithm's equation and has represented them with respect to 3 different possible cases(best, average and worst) in terms of time and space complexities using Asymptotic notations. Further in[1] Sahni verifies the Asymptotic complexities of each sorting algorithm by conducting respective code execution for large input sizes in different ranges. The author has used C and C++ languages. Similar works are exhibitted by the pioneers like Weiss M A in [9] and Thomas H Coremen in [3].

In [5], 2011, C Canaan et al has perfectly described a set sorting algorithms and their computational complexity using Asymptotic notations. Further they have named the set as popular sorting algorithms. This set includes Bubble sort, selection sort, insertion sort, Merge sort, Heap sort and quick sort. A similar kind of work is done in [20] by You, Ping & Yan for five sorting algorithm. As a whole these works have verified the behaviors of Sorting algorithms.

In papers [8][17] have made asymptotic analysis and performance measurement of popular sorting algorithms by implementing them in C and C++ language and has exhibited the behaviors of the popular sorting algorithms. The experiments and results in [8] show the Asymptotic Point (AP) or the time taken for the data sample from which onwards the sorting algorithm enter into the Asymptotic Behavior Range(ABR), here Range refers to data samples or set greater than that sample size for which we achieved the AP.

As our study focuses to analyse and measure the behaviors of popular sorting algorithm by considering Java and Python languages implementations. Hence we found from [13][14] that Java Python C++ and C are the most popular programming languages. To know and confirm the characteristics and features of the popular programming languages apart from the books [16][28][29][33] we found in [38], Shadman Salih(2014) has described in his research study detail about Object Oriented programming languages. In[35],Selina Khoiron et al(2020) has performed a study on Java & Python and has very precisely described their features. In [37], Saquib Ali & Ammar Quayyum has performed a Pragmatic Comparison of Four Different Programming Languages and illustrated considering both theoretical and practical aspects.

After conducting experiments they concluded that, if the goal is speed efficiency and reliability then C++ is preferred even over C. Java is the middle ground of all languages, if you can't agree on the complexity of C but also want the speed that Python lacks then Java is best. Python is best used if you want to write software where speed is not of major concern.

At the end Third Generation of Programming language (3GL) professionals like C. A .R Hoare in [31] motivated towards good compiler design and have illustrated their views and predictions which has its influence in the field of programming language design. Also A.L Tharp in [24] during 1984 and later in 1990s Darious S Baer in[23] presented regarding the objectives and efficiency needed about the Fourth Generation Languages(4GLs). In[30] Bertrand Meyer(2000), suggested the professional community to take up experimental study and analyse the efficiency of popular programming languages like C++,Python and Java.

In[12],on february22,2021 Mike McMillan a practitioner has published in level up coding by comparing programming language efficiency in 4 programming languages with respect to selection sort (on 10000 random Numbers)and has ranked java to run fast followed by C++ and Java.

By considering all the above made us to experiment C++ and Java, and later Java and Python which is published in [17][18] respectively. The present work is performed using three Popular Programming languages for Popular Sorting Algorithms. In our experiments we try analyze by considering a deep survey over the features of Programming Languages.

In [32], we found a similar experiments on large data sets (of about n=600000) which verifies the Mergesort performance, Quick sort being a real case to be studies is not considered also there is no worst case results present.

VI. PERFORMANCE MEASUREMENT

A. Experiment Set-up

Performance measurement is concerned with obtaining the actual time measure of a program. To obtain the execution time of a program, we need proper clocking mechanism. In C++, we have used the clock() which returns the number of clock ticks elapsed since the program got executed. We need to include the Header File "time.h" and declare clock_t clock(void); which returns a value : On success, the value returned is the CPU time used so far as a clock_t; To get the number of seconds used, divide by CLOCKS_PER_SEC on error -1 is returned. For, Python,we have used the Python method time() by importing time class from its library, which measures time in seconds. Similarly, In Java, the system class in library has methods like nanoTime() or currentTimeMillis() which clocks in nano seconds and milli seconds respectively later the same was converted to seconds.

Since we expect the asymptotic behavior to begin for larger values of input size n as in [8]. We have our sample size n= 1000, 2000,..10000, 20000, ...100000…500000. As we have considered two types of data samples for conducting our experiment in order to testify the average and the worst cases of algorithms.

Random samples (for average case analysis), for which we use the random number generator methods available in both python and java languages, by importing their respective classes. which is given below:

#For Python:

```
Import random #random is the class
random.randint()  # the method of class random
```

```
//for Java:
Import java.util.Random;  //Utility class from library
Generator.nextInt()     //the method of Random class
//through its object generator.
//for C++
#include<cstdlib>.   // includes the library file
Rand() /*function is an inbuilt function used to generate a series of random
numbers. The srand() function sets he starting point for producing a series of
        pseudo random  integers*/
```

The Sorted samples for worst case analysis, here the array index is assigned as the array element, a[i]=i for ascending order or a[i]=n-i-1for descending order. We did not consider the best-case analysis as there is not much consideration in practice for sorting algorithms. The driver codes, which initiates the sorting code executions for C++, python and Java are given below:

```
# Driver code of C++
int main ()
{
int max=500000;
int data[max];
float x,y;
clock_t time_req;
int step=100000;
for(int n=100000;n<=max;n=n+step){
        if (n==100000)step=100000;
        srand((unsigned) time(0));
        for (int i=0;i<n; i++)
          data[i]= n-i-1; //for descending order data set        //data[i]=rand();  //  for
random data set
        time_req = clock();
        bubbleSort(data,n);
        time_req = clock() - time_req;
        cout << (float)time_req/CLOCKS_PER_SEC <<'\n';
        }
return 0;
}
//time_req is the time taken by the algorithm to sort the        //sample
# Driver code of Python
a=[]
for n in range (10000,100001,10000):
#generating random numbers in range 1..50
#Append method adds t generated number into the        #list
#For i in range(n):
                #a. append(random.randint(1, 50))
    """ By enabling the above two statements we generate the sample data for worst
```

```
case analysis “””
for i in range(n):
                    a.append(i)
a.reverse() # to generate a reverse ordered   sample
start=time.time()
quickSortIterative(a, 0, n-1)
#similar call for other sorting methods is placed here
elapsed=time.time()-start
print(elapsed)
    “””elapsed is the time taken by the algorithm to sort                the  Sample “””
// Driver code of Java program
public static void main(String[] args){
int[] a;
int i,step=1000;
for(int n=1000;n<=max;n=n+step){
        if (n >= 10000) step=10000;
        a = new int[max];
        Random generator = new Random();
        for(i=0;i<n;i++)
                a[i]=generator.nextInt(100));
        long startTime = System.nanoTime();
        SelectionSort ss = new SelectionSort();
        ss.selectionSort(a);
        // similar call for other sorting methods is placed here
        long stopTime = System.nanoTime();
        long elapseTime = (stopTime - startTime);
System.out.println(elapseTime);
         } //sample loop
} // main body end
```

The sample sizes are specified as per the range needed in case of python with the help of range() function and using for loop with suitable step size in case of Java to form the range of data. The random method is used in both python and java to generate pseudo random numbers. We have used the system configurations specified in test bed below for conducting our experiments.

Test bed: Memory of 3.7 GiB, Intel® Core™ i3-6006U CPU @ 2.00GHz × 4 ,64-bit, 970.9 GB, Ubuntu 16.04 LTS

B. Experiment results and Observations

This Sub-Section is further divided into two (1) Experimenting with Random data section and (2)Experimenting with Sorted data section, the former shows the results of experiments and observations made on different samples of Pseudo random number(average case analysis) and the later shows the results and observations made on different samples of Ordered data(worst case analysis)). Both these Sections have Tables and their respective graph plots corresponding to Python, Java and C++ languages .The first column in the Table has different sizes of data set ranging from 10000, 20000 …..100000..500000 and the remaining columns of tables show

execution times for different sorting methods.

1) Experimenting with Random data

The results(execution times for random data) of sorting algorithms are showed in Table 1 for Python implementation, Table 2 for Java implementation and Table 3 for C++ implementation. The tables are supported with their respective graphical representation. Further for simplicity and discussion purpose we have considered the execution times of sorting algorithms for n=100000 from Table 1, Table 2 and Table 3 and created Table 4.

TABLE 1.
EXECUTION TIME TAKEN ON TEST BED USING PYTHON CODE FOR RANDOM DATA SAMPLES

Sample size	Quick Sort time(s)	Merge sort time(s)	Insertion Sort time(s)	Selection sort time(s)	Bubble sort time(s)
10000	0.193	0.354	0.615	3.389	4.615
20000	0.700	0.160	10.191	13.166	25.718
30000	1.543	0.328	16.443	32.573	55.480
40000	2.725	0.556	23.398	61.848	90.736
50000	4.510	0.848	29.950	101.888	130.150
60000	6.142	0.639	36.043	156.848	183.345
70000	8.777	1.209	42.907	219.166	248.706
80000	10.786	1.635	49.679	301.854	322.293
90000	13.948	2.130	57.422	325.083	397.109
100000	**17.463**	**2.689**	**63.824**	**348.351**	**748.495**

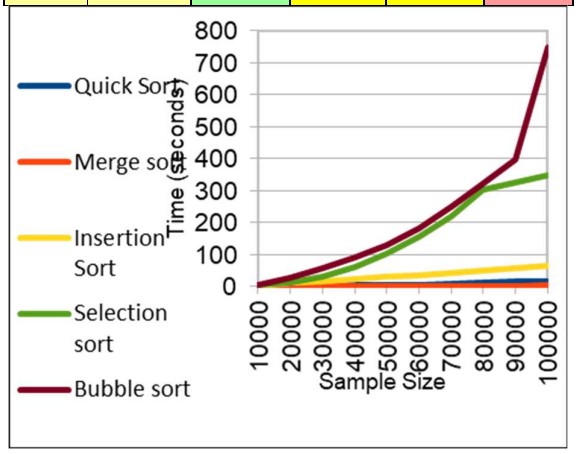


Figure 3: Graph plot for table 1

TABLE 2.
EXECUTION TIME TAKEN ON TEST BED USING JAVA CODE FOR RANDOM DATA SAMPLES

Sample size	Quick Sort time(s)	Merge sort time(s)	Insertion Sort time(s)	Selection sort time(s)	Bubble sort time(s)
10000	0.008	1.968	0.033	0.120	.0.273
20000	0.007	3.411	0.089	0.428	1.199
30000	0.013	5.029	0.209	0.647	2.364
40000	0.019	6.789	0.355	1.124	5.216
50000	0.029	8.425	0.556	1.859	6.572
60000	0.045	10.032	0.831	2.548	9.482
70000	0.061	11.817	1.133	3.511	12.800
80000	0.079	13.584	1.806	4.621	16.842
90000	0.100	15.080	1.829	5.985	21.250
100000	**0.1122**	**17.651**	**2.297**	**7.299**	**26.347**

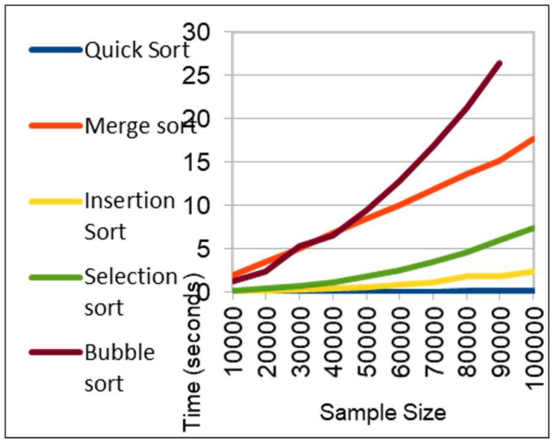


Figure 4: Graph plot for table 2

TABLE 3.

EXECUTION TIME TAKEN ON TEST BED USING C++ CODE FOR RANDOM DATA SAMPLES

Sample Size	Quick Sort time(s)	Merge sort time(s)	Insertion Sort time(s)	Selection sort time(s)	Bubble sort time(s)
10000	**0.010**	0.003	0.110	0.180	**0.403**
20000	0.019	0.006	0.435	0.720	1.779
30000	0.013	0.009	0.972	1.682	4.102
40000	0.012	0.012	1.719	2.896	7.438
50000	0.013	0.016	2.700	4.494	11.844
60000	0.015	0.018	4.011	6.465	17.129
70000	0.020	0.022	5.362	8.860	23.586
80000	0.021	0.025	6.867	12.421	31.208
90000	0.024	0.029	8.860	15.280	39.373
100000	**0.027**	**0.032**	**10.880**	**18.248**	**48.755**

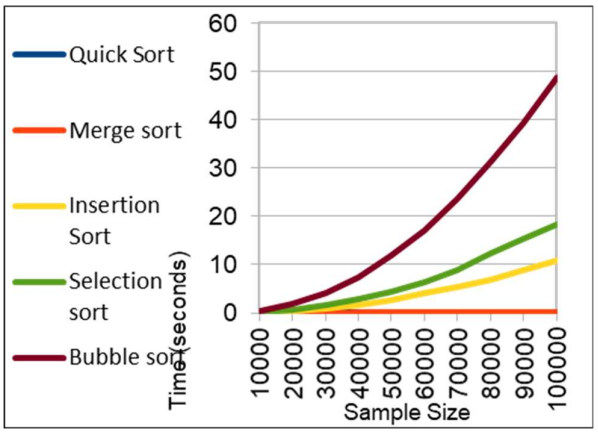


Figure 5: Graph plot for table 3

TABLE 4.

EXECUTION TIME OF SORTING ALGORITHMS WHEN SAMPLE SIZE IS 100000 FOR C++, JAVA AND PYTHON(average case analysis)

Programming language	Quick sort time(s)	Merge sort time(s)	Insertion sort time(s)	Selection sort time(s)	Bubble sort time(s)
C++	0.027	0.032	10.248	18.248	48.75
Java	0.1122	17.651	2.97	7.299	26.347
Python	17.463	2.689	63.824	348.35	748.49

Hence we make the following observation:

1. From Table 1, Table 2, and Table 3 and their respective Graphs it shows clearly that sorting algorithms i e., Quick sort, Merge sort, Insertion sort, Bubble sort and Selection sort have performed as per their asymptotic behavior and order of growths given in the theories and performance measurements in [1][2][8]only with respect C++ , and the behaviors are not normal with Java and Python. As Java performance slower for Merge sort and for Python Merge sort has performed good but all others are slow.
2. We observe from Table 4 that the n2 class algorithms ie., Bubble, selection and insertion sort have performed good with Java code than C++ code.
3. Quick sort as per the theories and mathematical analysis done by [2][3][9] and performance measured by [1][8] and also realized from Table 2 and Table 3 it is the fastest among all, but we see in Table 1 and its Graph plots that with Python is very slow than Merge sort. Its execution time for sample size 100000 is 17.463 seconds and merge sort is 2.689 seconds. Hence Python has not performed for Quick sort.
4. We observe from Table 4 that though Java has out performed for n2 class algorithms it has gone slow for Merge sort with execution time 17.651seconds, where as Python and C++ has executed in 2.689 seconds and 0.032 seconds respectively. Also, we observe that Insertion sort performing better than Merge sort in case of Java implementation,This shows Java has not performed well.
5. We notice from Table 4 that when it comes to sorting with java we suggest Quick sort , because it out performs all with execution time 0.1122 seconds. Python should prefer Merge sort because it out perform all other sorting methods as its execution time is 2.689 seconds.
6. We came across a runtime error while executing the quick sort implementation using python,which was due to the recursion stack overflow. The exception was removed by

increasing the recursion stack limit by importing the sys(import sys) class and using its method sys.setrecursionlimit(size). The details of recursion stack are given in [6]. The runtime stack overflow was not encountered with C++ and java implementation for random data samples. This observation indicates that C++ and Java could be preferred over python for sorting larger samples.

7. As we notice inconsistency in performance of sorting algorithms with respect to Python language and Java Language, we prefer C and C++ for efficiency and stability.

As part of discussion, we make the following statements:

1. C and C++ have exhibited consistently the behaviors of Popular sorting algorithms as in [1[2][3], the reason for this is stated in Sub-Section II-A point No. 3.
2. The lack of robustness in C++ languages is stated in Sub-Section II(A) point No. 8 & 9.
3. Python implementation for sorting gets slower due the point no. 3,6,8, & 10 in the Sub-section II-B.
4. Java which has performend well and good for all but Merge sort is due to point No. 5,6,7& 10 specified in II-C Sub-Section.

2) Experimenting with Sorted data set

The results(execution times for ordered data) of sorting algorithms are showed in Table 5 for Python implementation, Table 6 for Java implementation and Table 7 for C++ implementation. The tables are supported with their respective graphical representation. Also it is made clear that this is a unrealistic case which have considered for our study to know the strengths and weaknesses of Programming languages use for implementing the algorithms. we have created Table 8 where we have the worst case execution times of sorting algorithms for n=100000 from Table 5 Table 6 and Table 7 along with the average case execution time from Table 4.

TABLE 5
EXECUTION TIME TAKEN ON TEST BED FOR PYTHON CODE ON ORDERED DATA SAMPLES

Sample size	Quick Sort time(s)	Merge Sort time(s)	Insertion sort time(s)	Selection sort time(s)	Bubble sort time(s)
10000	5.47	0.04	6.83	3.38	4.10
20000	21.97	0.15	27.00	13.78	24.36
30000	49.89	0.33	63.06	30.75	39.98
40000	89.03	0.53	129.82	54.94	99.15
50000	141.23	0.82	182.84	87.71	114.12
60000	209.51	1.16	259.94	130.88	220.95
70000	293.68	1.57	378.23	178.36	286.96
80000	429.04	2.06	461.70	238.01	337.61
90000	536.65	2.61	582.57	300.06	405.04
100000	**598.79**	**1.51**	**684.93**	**375.28**	**460.40**
200000	2590.36	1.67	2109.76	1447.73	1751.17
300000	5858.01	3.53	6167.30	3777.81	5862.24
400000	10255.04	6.08	10991.70	6090.13	8104.42

500000	**15713.08**	**9.410**	**20052.8**	**9125.70**	**11748.8**

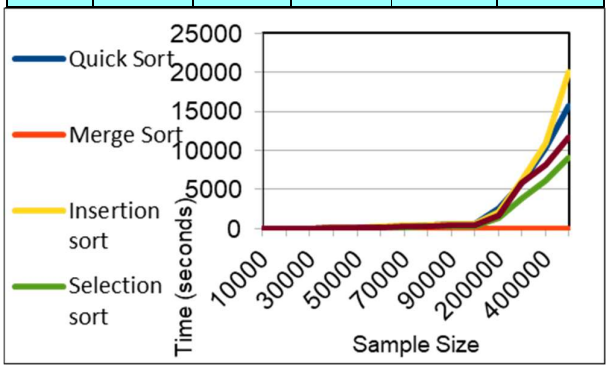


Figure 6: Graph plot for table 5

TABLE 6.

EXECUTION TIME TAKEN ON TEST BED FOR JAVA CODE ON ORDERED DATA SAMPLES

Sample Size	Quick sort Time in sec	Merge sort Time in sec	Insertion sort Time in sec	Selection sort Time in sec	Bubble sort Time in sec
10000	0.87	25.25	0.54	0.32	0.47
20000	3.09	33.48	1.84	0.92	1.56
30000	7.07	51.26	3.98	2.37	2.93
40000	12.54	67.51	7.14	3.67	6.19
50000	16.26	84.46	11.24	5.70	9.49
60000	25.56	102.03	16.02	8.25	13.85
70000	34.56	119.61	21.99	10.96	18.87
80000	45.05	137.71	28.88	14.88	24.54
90000	57.04	153.69	36.37	18.40	31.16
100000	70.44	169.90	44.90	25.86	32.58
200000	283.43	337.16	181.17	104.96	131.53
300000	650.18	530.29	415.99	236.41	295.66
400000	1142.17	696.76	749.28	429.95	529.66
500000	**1812.31**	**875.73**	**1167.27**	670.42	**848.90**

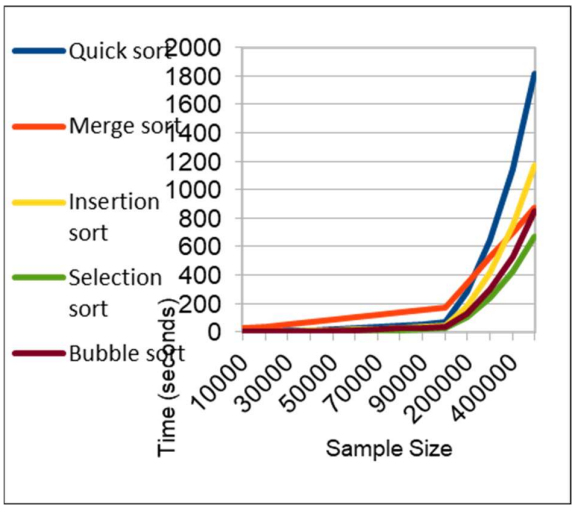


Figure 7: Graph plot for table 6

TABLE 7

EXECUTION TIME TAKEN ON TEST BED FOR C++ CODE ON ORDERED DATA SAMPLES

Sample size	Quick Sort Time in sec	Merge sort Time in sec	Insertion Sort Time in sec	Selection sort Time in sec	Bubble sort Time in sec
10000	0.327025	**0.001947**	0.217146	0.188348	**0.348765**
20000	1.30349	0.003275	0.861631	0.75181	1.31706
30000	2.92996	0.005278	1.94164	1.69115	2.98455
40000	5.20315	0.006751	3.44723	3.0071	5.29415
50000	8.13089	0.010155	5.42916	4.76164	8.52272
60000	11.7192	0.010365	7.74651	6.94479	11.9832
70000	15.9394	0.012186	10.539	9.37635	16.2143
80000	20.8201	0.015152	13.7893	12.4428	23.6834
90000	26.4484	0.015903	17.3839	15.5903	28.5272
100000	32.6818	**0.01901**	21.479	19.1	**33.6542**
200000	134.072	0.039253	86.1923	77.4486	133.496
300000	295.033	0.058425	193.365	174.771	299.489
400000	529.134	0.079388	344.659	301.901	530.479
500000	**809.665**	**0.099834**	537.935	483.659	**827.23**

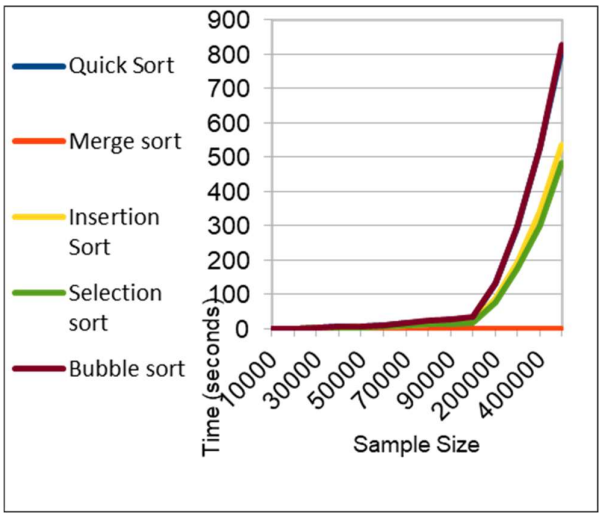


Figure 8 :Graph plot for table 7

TABLE 8.
EXECUTION TIME OF SORTING ALGORITHMS WITH SAMPLE SIZE 100000 (ORDERED DATA)

Programming language		Quick sort (seconds)	Merge sort (seconds)	Insertion sort (seconds)	Selection sort (seconds)	Bubble sort (seconds)
C++	Average case	**0.027**	0.032	10.248	18.248	48.75
	Worst case	32.68	**0.01901**	21.479	19.1	33.65
Java	Average case	**0.1122**	17.651	2.97	7.299	26.347
	Worst case	70.44	**169.90**	44.90	25.86	32.58
Python	Average case	17.463	2.689	63.824	348.35	748.49
	Worst case	**598.79**	**1.51**	**684.93**	**375.28**	**460.40**

The following observation were made from the results of experiments:

1. We observe from Table 8 that both average case execution times and worst case execution times of sorting algorithms implemented using C++ are consistence with respect to the theories established in [1][2][3]. ie., Quick sort being fastest among all followed by Merge sort ,insertion sort , selection sort and bubble sort. Also we find with respect to worst case, that quick sort complexity becomes O(n2) like bubble sort. T1=32.68 seconds and T2=33.65 seconds for quick sort and Bubble sort respectively.
2. We observe from Table 8 that Merge sort performs excellent in Python implementation with execution time 1.51 seconds in worst case (with ordered data) and execution time 2.689 seconds for average case(with random data). All others go too slow in both average and worst case executions.
3. We observe from Table 8 that all three n2 class algorithms ie., bubble sort ,selection sort and insertion sort perform faster with Java than the C++ implementation for average case executions. Also the worst case being almost the same speed. Further we observe that for quick sort java has executed with 0.1122 seconds in average cases and 70.44 seconds with respect to

worst case. We also see that merge sort ‘s implementation in Java has slowed down its performance with 17.651seconds and 169.99 seconds in case of average and worst case respectively.

4. Finally we observe that Python and Java implementations for popular sorting algorithms are not consistent with respect to the theories established in [1][2][3].

Also, the above results and observations opens the doors of research studies to give proper criteria of analysis and measurements when we deal to implement the sorting algorithms using modern programming languages like Python and Java.

CONCLUSION

Our research shows that when speed and efficiency is of concern C++ is to be preferred to implement the algorithms. When speed is not important and you still need sorting then Merge sort is preferable especially in the field of artificial intelligence, data analytics, machine learning etc where Python is popularly used as programming language. As in the results of our experiments we see in Python implementation of popular sorting algorithm, merge sort is the fastest one, where as all others go too slow. When the context of application development opts Java and speed is not of much concern, you can use Quick sort for sorting the data. Merge sort was slow when implemented in Java. As a peculiar case, we found Merge sort getting too slower when handling large ordered data sets (worst case). The reason behind this as per our study, is due to more memory consumption by Java compiler, also we know that merge sort consumes double the memory than other popular sorting algorithms. Finally, by seeing the behaviors of popular sorting algorithms described in theories and practices in [1][2][3], we find C and C++ languages exhibiting the asymptotic exact behaviors. Where as it is found that the 4GL programming languages like Python and Java failed in exhibiting the behaviors.

ACKNOWLEDGEMENT

Primarily we are thankful to almighty Allah who made this successful, then our college managements and administration and my Guide for motivation. Finally our department heads and our beloved Principals for their supports.

REFERENCE

[1] Sartaj Sahni, Data structures and Algorithms in C++, university press publication 2004, chapter 4: performance measurement, pages 123-136.

[2] Levitin, A. Introduction to the Design and Analysis of Algorithms. Addison-Wesley, Boston MA, 2007.

[3] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, “Introduction to Algorithms”,Second Edition,Prentice-Hall New Delhi,2004.

[4] Vandana Sharma, Parvinder S. Sandhu, Satwinder Singh, and Baljit Saini,” Analysis of Modified Heap Sort Algorithm on Different Environment", World Academy of Science, Engineering and Technology 42 2008.

[5] C.Canaan, M.S Garai, M Daya, Popular Sorting Algorithms,World Applied Programming, Vol 1,No. 1,April 2011,pages 42-50.

[6] James Gallagher, Python maximum recursion depth exceeded in comparison solution. Available: http://www.careerkarma.com

[7] Mike Mc Millan, Comparing Programming language efficiency in 4 programming languages, Available:http//www.gitconnected.com/

[8] Omar Khan Durrani, Shreelakshmi V, Sushma Shetty & Vinutha D C Analysis and Determination of Asymptotic Behavior Range For Popular Sorting Algorithms,Special Issue of International Journal of Computer Science & Informatics , ISSN (PRINT) :2231– 5292, Vol.- II, Issue-1, 2.

[9] Weiss, M. A., Data Structures and Algorithm Analysis in C,Addison-Wesley,Second Edition, 1997 ISBN: 0-201-49840-5.

[10] Gina Soileau, Muhammad Younus, Suresh Nandlall,Tamiko Jenkins, Thierry Ngouolali, Tom Rivers, Sorting Algorithm Analysis, (SMT-274304-01-08FA1), Professor James Iannibelli, December 21, 2008.

[11] In which language JVM is written?, Available at https://community.oracle.com/ tech/developers/discussion/1260753/

[12] Mike Mc Millan, "comparing Programming language efficiency in 4 programming languages",Available: http//www.gitconnected.com/levelup

[13] The Most Popular Programming Languages – 1965/2022, Available at https://statisticsanddata.org/data .

[14] TIOBE Index for August 2022: Python going through the roof. Available at, https://www.tiobe.com/tiobe-index/

[15] O.K. Durrani, S.A.K. Nazim, Modified quick sort: worst case made best case. Int. J. Emerg. Technol. Adv. Eng.5(8). Website: www.ijetae.com, ISSN 2250-2459, August 2015).

[16] Herbert Schildt ,Java™ : The Complete Reference, Seventh Edition,2007 by The McGraw-Hill Companies,ISBN: 978-0-07-163177-8

[17] Omar Khan Durrani, Chirag U, Ayub Moez Khan, Performance Measurement of popular Sorting Algorithms implemented using C++ and Java , in the Proceedings of National Conference on Innovation and ng technology(NCIET-2022) held at GCE,ramanagara Karnataka India on 13 August 2022.

[18] Omar Khan Durrani, Dr, Sayed Abdulhayan, Performance Measurement of Popular Sorting Algorithms implemented using Python and Java, in the Proc. of the International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)16-18 November 2022, Maldives, 978-1-6654-7095-7/22/$31.00 ©2022 IEEE.

[19] Bal, A.B., Chakraborty, S. (2020). An Experimental Study of a Modified Version of Quicksort. InAdvances in Intelligent Systems and Computing, vol 988. Springer, Singapore. https://doi.org/10.1007/978-981-13-8222-2_27

[20] You Yang, Ping Yu and Yan Gan, "Experimental study on the five sort algorithms," 2011 Second International Conference on Mechanic Automation and Control Engineering, 2011, pp. 1314-1317, doi: 10.1109/MACE.2011.5987184.

[21] M. Marcellino, D. W. Pratama, S. S. Suntiarko and K. Margi, "Comparative of Advanced Sorting Algorithms (Quick Sort, Heap Sort, Merge Sort, Intro Sort, Radix Sort) Based on Time and Memory Usage," 2021 1st International Conference on Computer Science and Artificial Intelligence (ICCSAI), 2021, pp. 154-160, doi: 10.1109/ICCSAI53272.2021.9609715.

[22] Ian Sommerville. 2010. Software Engineering (9th. ed.). Addison-Wesley Publishing Company, USA.

[23] Darius S. Baer. 1990s Expectations for a Fourth Generation Language, IBM Corporation available at https://support.sas.com/resources/papers/

[24] A. L. Tharp, “The impact of fourth generation programming languages,” ACM SIGCSE Bulletin, vol. 16, no. 2, pp. 37–44, Jun. 1984.
[25] Abdullah A H Alzahrani, “4GL Code Generation: A Systematic Review” International Journal of Advanced Computer Science and Applications(IJACSA), 11(6), 2020. http://dx.doi.org/10.14569/IJACSA.2020.0110623
[26] Details of Python and its features are made available at : https://www.python.org/doc/essays/comparisons/
[27] https://www.educba.com/comments-in-python
[28] Brian.W.Kernighan and Dennis.M.Ritchie, The C Programming Language. 2nd.Ed Prentice.Hall,1988.
[29] Bjarne Stroustrup ,The C++ Programming Language (4th Edition),Addison-Wesley ISBN 978-0321563842. May 2013.
[30] Bertrand Meyer, Principles of Language Design and Evolution, in Millenial Perspectives in Computer Science (Proceedings of the 1999 Oxford-Microsoft Symposium in Honour of Sir Tony Hoare), eds. Jim Davies, Bill Roscoe and Jim Woodcok, Cornerstones of Computing, Palgrave, 2000, pages 229-246
[31] Hoare, C. A. R., Hints on Programming Language Design, invited ad-dress at SIGACT/SIGPLAN Symposium on Principles of Programming Languages, Boston, Mass., October 1-3, 1973.
[32] Syed Muqeet Aqib et al., Analysis of Merge Sort and Bubble Sort in Python, PHP, JavaScript, and C language. International Journal of Advanced Trends in Computer Science and Engineering, 10(2), March - April 2021, 680 – 686. ISSN 2278-3091.
[33] Donaldson, Toby. "Python as a first programming language for everyone." Western Canadian Conference on Computing Education. Vol. 232. 2003
[34] Vineesh Cutting and Nehemiah Stephen. Comparative Review of JAVA and Python, International Journal of Research and Development in Applied Science and Engineering (IJRDASE) ISSN: 2454-6844,Volume 21, Issue 1, December 2021.
[35] Mrs. Selina Khoirom et al., Comparative Analysis of Python and Java for Beginners i.nternational Research Journal of Engineering and Technology (IRJET) ,Volume: 07 Issue: 08 | Aug 2020 .
[36] Shadman Salih, selection of computer programming languages for developing distributed systems, software technology research paper, de montfort university faculty of technology the gateway, leicester le1 9bh, uk ,may 15, 2014.
[37] Saquib Ali , Ammar quayyum, A Pragmatic comparison of four different programming languages, University of Management And
Technology, Daska Campus,Orcid ids: 0000-0001-8533-4103[1], 0000-0001-5231-2936[2], 21 June 2021.
DOI: 10.14293/S2199-1006.1.SOR-.PP5RV1O.v1.
[38] Muhammad Ateeq et al, “C++ or Python? Which One to Begin With: A Learners Perspective”, https://www.researchgate.net/publication /271425337
[39] Nallusamy, S., et al. "Implementation of total productive maintenance to enhance the overall equipment effectiveness in medium scale industries." International Journal of Mechanical and Production Engineering Research and Development 8.1 (2018): 1027-1038.
[40] Sharma, Shiv Kumar, et al. "Performance evaluation of diesel engine using biodiesel

fuel derived from waste cooking refined soyabean oil." Int. J. Mech. Prod. Eng. Res. Dev 7 (2017): 103-110.
[41] Neelakanteswara, P., and P. SURYANARAYANA Babu. "Efficient trust management technique using neural network in cloud computing." J Comput Netw Wirel Mobile Commun 9.1 (2019): 29-40.
[42] Jayaram, B., et al. "A Survey On Social Media Data Analytics And Cloud Computing Tools." International Journal of Mechanical and Production Engineering Research and Development, 8 (3) 243 (2018): 254.
[43] Wadhwani, Priyanka, Akanksha Gaur, and Vipin Jain. "Cryptanalytic JH and Blake Hash Function for Authentication and Proposed Work Over Blake-512 on C Language." International Journal of Computer Science Engineering and Information Technology Research 4 (2014): 187-198.