A Novel Approach in Test Case Selection using Code Coverage Analysis

Nishant Gupta, Mayank Singh, Vibhash Yadav

***Abstract*: A process of testing automation is to develop the automated tool which helps and reduce the testing efforts. It should focus on designing and implementation of a tool which automates the testing environment. The testing should also focus on designing and constructing a tool which automates test cases for regression testing. We have given a novel approach which selects the test cases based on code coverage analysis. This paper focuses on calculated operators and operands value on large programs in minimum amount of time and cost. The code coverage has been done using different parameters of Halstead metrics. The values of the different parameter may be used for the test case selection.**

***Index Terms*: Halstead Matrix, Regression Testing, Software Testing, Test Case Selection**

I. INTRODUCTION

Testing requires the execution of a program to identify the errors. Testing must efficiently reduce the dissimilar classes of errors in a less amount of time and resulting the effort reduction. It should verify and validate the software so that the end user requirements may be achieved. It is an important area of research and also help in development in software [1]. After each development phase, it is necessary to execute the result of that phase before executing the next phase. Manually this can be a difficult process, and will also increases the cost of overall software. The only way to make an effective and quality software both systematic and economically feasible is to automate it.

The sorting Software Engineering Environment is a system which integrate the tools with a common development database in which all of the software products are represented. Quality of the software depends on various factors like time, cost and the effort which has been put during development process. Time and costs are also tradeoffs while selecting the project or any software.

Regression testing makes sure that the old version of program runs in parallel with the new changes in program. Regression testing is to test the existing software applications to make sure that a change or modifications doesn't affect any existing functionality. It has to be implemented to find out bugs that has been introduced into a new build and

to ensure that it has not introduced a new bug in previous builds.

Every activity performs some tasks which has some output which in effect cascaded to another phase. At the end, a complete report of bugs has to be produced and documented. These bugs are then used by the testers and development team to find out the actual reason of errors and faults so that they may be corrected. As the report is elaborated in detail, the test plan analysis may be performed and exact objectives may be find out. The method of design procedures is defined which helps in deciding the test procedures.

By following the methods and test procedures, it may be decided that whether the particular test build has to be tested manually or automatically. The importance of web application has been growing in recent years. Sites have been highly used for businesses, scientific medical purposes such as diagnostic based expert systems and for government organization.

New properties are getting added to them to satisfy user's changing demands. Also the task is asked to be done in less time with fewer people. This complexity decreases test coverage and highly affects the software's quality. Their factors involved over time are the entire price of the product and the time to deliver the software. An automation effort provides more coverage area than the testing done manually and enhances the overall quality of the software. They reduce the testing duration and delivery price.

Automation testing is performed at the time of tester creates scripts and utilizes software to verify the product. They are also used to examine the application from work, result, observation and stress point of view. The automation software testing consists of various activities and methods which helps in designing a tool that helps in executing the test build and keeping the record of the test suites for further execution. The following activities include in testing process:

- Test planning
- Test design & Implementation
- Test execution
- Test report
- Test evaluation

The main thing for developers and testers should focus in brain is that a little amount of change in code can result in partitioning the operations into various sections that can cause unknown transformations in source code. At the time of performing regression testing tester ensures that modifications done in the programs will not affects it's working.

In regression testing, where huge effort is required to test the test suite, the efficient test case selection technique may reduce the cost and time.

In our proposed technique, we are using the Halstead matric to find out the code coverage. The code coverage has been analyzed for different set of programs using various parameters of Halstead matrices [19], [20].

Revised Manuscript Received on December 22, 2018.

Nishant Gupta, Mahatma Gandhi Mission's College of Engineering and Technology, Noida, India
Mayank Singh, University of Kwazulu-Natal, Durban, South Africa
Vibhash Yadav, Rajkiya Engineering College, Banda, India

The values received from code coverage using Halstead matrices may then be used for the test case selection process.

II. LITERATURE SURVEY

Pranali & D.M.Thakore [2] presented his work on automatic test data generation tools and techniques focused on object oriented code. The aim is to give an overview of test data generation tools and techniques. It focuses on the problem of how to choose the most appropriate tool that will fulfil developer requirement. Ritu & Sukaldip Singh [3] has given a review on test cases selection in regression testing. The focus is to select and prioritize test case for performing regression testing. Hussain & Touseef Ikram [4] presented the review on formalizing use cases and scenario based testing. The paper is focusing to explore in the field of software testing using artefacts in the design and requirement specification of software application. Maheshwari & Prasanna [5] conducted the experiment by generation of test cases using automation in software system. The main focus of the work is to explore about automatic test case generation. The author presented the existing testing approaches which were used to verify the critical based testing.
Monier & EI-mahdy [6] proposed an approach which evaluates automated web testing tools. The author has given the feasibility study which targeted the commercial and open source web testing tools for helping the testers and software developers. Its future work is based on different features to build a user based requirements. Ali Shah [7] proposed a review of class based test case generation techniques. The author discussed about the software development in recent years. The focus was also on development of an automated tool which may generate test cases through class diagram in order to focus on the problem.

Shahid & Ibrahim [8] presented a code-based test case prioritization technique. In this paper, they introduce a new algorithm for test cases prioritization that is based on the code coverage of the test cases. The proposed technique shows that the algorithm can be applied on large systems to verify its correctness. Pahwa & Solanki [9] presented the review by UML based test cases generation methods. It focuses on different techniques used for generating test cases that can decreases minimize the number of test cases.

Beena & Sarala [10] showed code coverage-based test case selection and prioritization. Author presented a novel approach where selective prioritization of test cases produces maximum code coverage. Kaur et al. [11] presented a survey of software test case generation testing. This paper is all about the study of different techniques used in test cases e.g. Test case generation using GA, RBT, and MBT etc. Martin Glinz [12] discussed about scenario-based approach to validate and test the test suite using state charts. In this paper, author has given a procedure to create scenario in the analysis phase and test cases has been determined by using the scenarios. Tahbildar & Kalita [13] presented that the automated software test data generation in the direction of research. The focus was on automatic test data generation and its approaches. In his study, the extension may be in the area of an automatic testing future in-improvement of code coverage, loops handling in path-oriented testing.

Shay Artzi et al. [14] has given an automatic generation of unit regression tests and also introduces a technique to automatically create class specific regression test from a program run. Shin & Harman [15] presented the regression testing minimization, selection and prioritization method. Author has given a complete survey for minimization, selection and prioritization technique and discusses open problems. Ming song & Xuedong [16] presented an automated test case generation for UML activity diagram. In this paper, use of UML activity diagram as a design specification was used and proposed an automatic test case generation technique.

III. METHODOLOGY

This procedure includes distinct operators and operands in large programs. This procedure is easy to understand, implement, calculate and can be used in many different programming languages with minimum errors and maintenance efforts.

- Parser: It is used in all high level programming languages. It is also called as syntax and syntactic analysis. It is the process of analyzing a string of symbols either in computer language and natural languages [17].
- Listener: It defines the methods used by components to dispatch events. It will have at least one corresponding dispatch method [18].

A. *Problem Formulation*

The most important objectives of this study is to implement the algorithm that is having many operands and operators.

- It has some real and imaginary formulations which are not easily proven.
- Code coverage will be done as per the Halsted algorithm.

The Software unpredictability recognize a basic part to decrease the drive to manufacture and update the basic structure of testing and encoding quality. The reason that not all estimation is giving similar results is that each plan covers a segment and ponder a couple of limitation while deserting some others. In this way, a merge of Estimations is allowed to be used to calculate the various qualities.

- To sketch out a crashing and improved model for symbols scope as well as remarks width examination, factors and operands utilize.
- To draw and actualize the powerful replica for code examination utilizing Monte Carlo Simulation technique [21]
- The proposed work will express the improved standard and activities so the relative parts of operands, constants and remarks can be utilized as a part of the source code.
- Comparison might be done on many parameters in future Approach
- In the usual or base work, there is no the obstacle of the usage of the observations thickness and examination.
- In our proposed work, the relation assessment of the remarks thickness capacity is actualized.

This examination work revolves around the quality of source code using code extension and examination strategy. In the proposed work, an effective model has to be generated and improve the execution of code similarly as general code, scope, time, code comprehensive nature and related estimation.
We can prioritize our test case based on the different parameter values as given below [19]:

$\eta 1$ = the number of distinct operators
$\eta 2$ = the number of distinct operands
N 1= the total number of operators
N 2= the total number of operands

The different parameters as given by Halstead Metrics [20] are

- Program Vocabulary(h)=h1+h2
- Program Length(N)=N1+N2
- Volume (V)=N log2h
- Potential Volume(V*)=(2+h2)log2(2+h2
- Program Level (L)=V*/V
- Difficulty (D)=V/V*
- Effort (E) =V/L
- Faults (B) = V/S*

The above parameters have been implemented for different source code to find out the code coverage and the time taken to cover a particular source code.
As an example, we have implemented this algorithm with the various parameters through the program as given below:

```
main ()
{
int a, b, c, avg;
scanf("%d %d %d", &a, &b, &c); avg = (a + b
+ c) / 3; printf("avg = %d", avg); }
```

The unique operators are:
main, (), {}, int, scanf, &, =, +, /, printf,'',

The unique operands are:
a, b, c, avg, "%d %d %d", 3, "avg = %d"

Operator	Operands
main	a
( )	b
{}	c
int	avg
scanf	%d%d%d
&	3
=	"avg=%d"
+	
;	
,	
/	
printf	
n1= 12	n2 = 7

Number of distinct operators & operands are:
η 1= 12,
η 2 = 7,
η = 19,

The total number of operands & operators are:
N1 = 27,
N2 = 15, N= 42

Now, as per the above operands & operators, the values of different parameters are calculated as shown in Table 1.

Table 1: Values of calculated Parameters

S.No	Parameters	Values
1	Estimated program length	N=12*log2 (12) +7*log2 (7) =62.67
2	Volume	V= 42*log2 (19) =178.4
3	Difficulty	D= 12/2 * 15/7 = 12.85
4	Effort	E = 12.85 * 178.4 =2292.5
5	Time required	T= 2292.5/ 18 = 127.36 millisec
6	No. of delivered bugs	0.05

As above in the table 1, the values of the different parameters have been calculated using Halstead metrics which may be further used to find out the best suited test cases based on code coverage and the time required.

IV. PROPOSED WORK

The flow of the proposed work has been shown in Figure 1. The entire study work is done with several prospect of research works as follows.

- The code coverage should be design in a systematic way or in a precise way.
- It should be based on the source code. In the classical or base work, there is no limitation of the implementation.
- The proposed work will deliver the optimized rules and solutions. So, that the proportional aspects of operands, constant and comments can be used in the source code.
- Halstead metrics analysis: Halstead diverse quality measures are customizing estimations introduced by Maurice Howard Halstead [20] as a noteworthy part of his treatise on working upon observational investigation of programming progression. Halstead said that the target truth that estimations of the item should reflect the use or verbalization of figuring in different lingos, anyway be self-ruling of their execution on a specific stage. The estimations are thus analyzed statically from the code given.

A. *Execution Scenario*

We have given a simulator that deals with different Halsted metrics parameters as shown in Fig 2. As per the simulator there are different parameters on different source code is given in Table 2

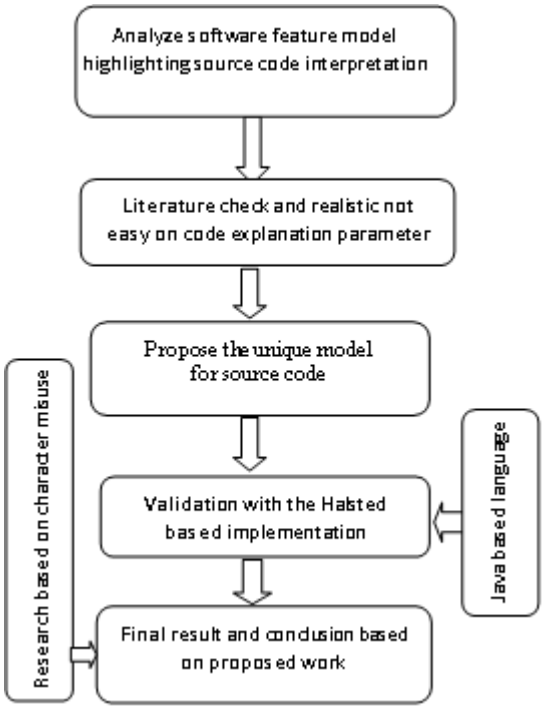

Fig1. Flow of the proposed work

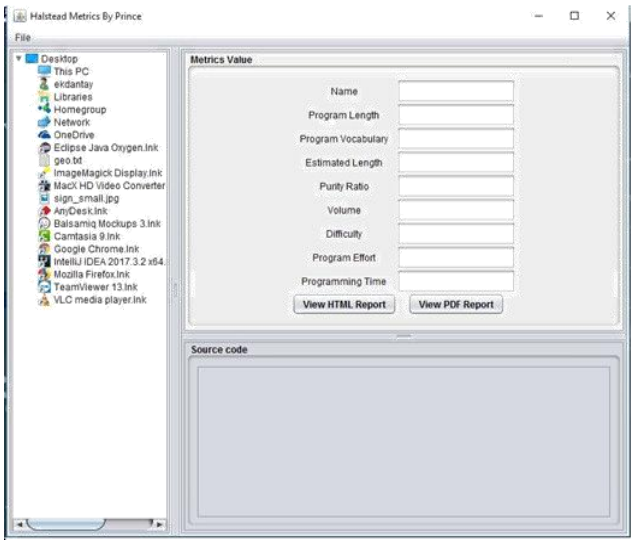

Fig.2 Execution Scenario of code coverage

Table 2: Code covering time (in milisec) for different source file

File Name	Length	Purity ratio	Vocabulary	Estimated Length	Volume	Difficulty	Efforts	Time
quicksort.java	37	36.75	203	1359.68	192.75	6.18	1191.97	66.17
insertionsort.java	39	44.46	244	1734.04	206.13	4.18	860.97	47.83
Fibonacci series.java	56	44.12	327	2470.64	325.21	6.37	2071.91	115.11
bubblesort.java	67	45.09	385	3020.9	406.43	6.93	2815.58	156.42
linearsearch.java	73	42.27	428	3450.36	451.80	6.58	2473.76	165.21
selectionsort.java	99	45.86	542	4540.2	656.31	10.33	6778.83	376.6
countingsort.java	44	16	116	704.72	240.21	5.61	1348.2	74.9
bucketsort.java	44	46.2	283	2032.8	240.21	9.4	2257.19	125.4
radixsort.java	69	37.88	335	2614	421.49	4.73	1994.75	110.82
mergesort.java	41	50.4	284	2065.73	219.66	5.25	1153.2	64.07
heapsort.java	53	26.1	202	1383.88	303.58	6.09	1847.4	102.6

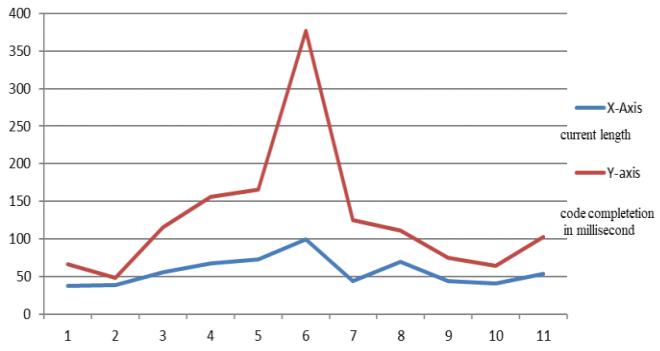

Fig 3: Graph of current length with completion time

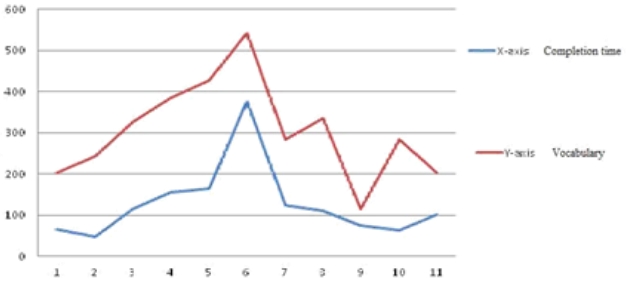

Fig 4: Graph of vocabulary of code with Respect to time.

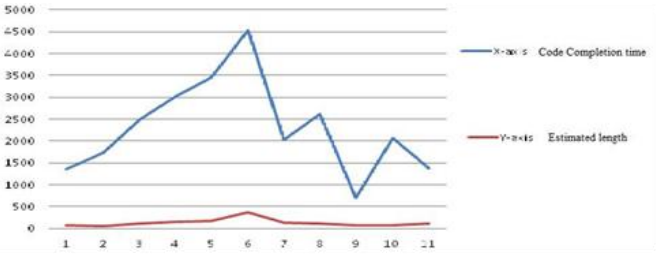

Fig 5: Graph of completion time with estimated length of code

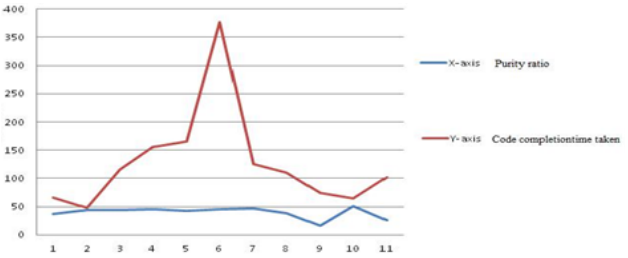

Fig 6: Graph of code purity ratio with code completion time taken

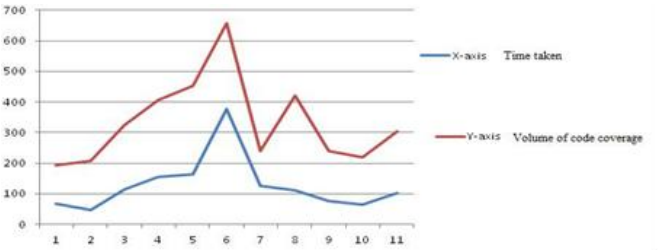

Fig7: Graph for volume of a code with time taken

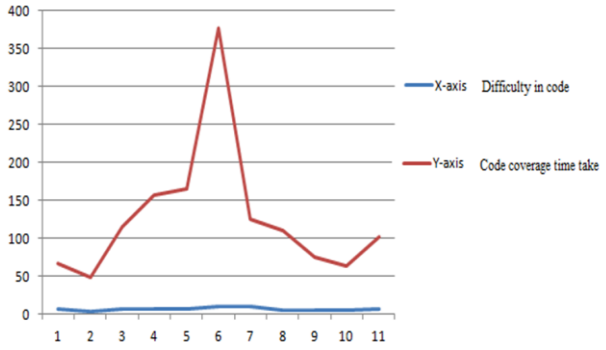

Fig 8: Graph for difficulty of code with time taken

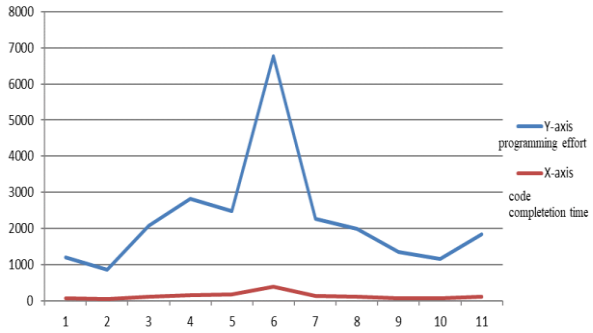

Fig 9: Graph of programming effort with time

V. CONCLUSION AND FUTURE SCOPE

We have executed different types of source code of Java to test and measured the complexity parameters. The finishing time of the planned and usual approach is calculated so that the experimental comparison can be done. Code coverage study is used to compute the value of software testing, usually using vibrant execution flow analysis. The parameters values may be used to select the test cases as per the requirement of the testing team.

There are lots of different types of code exposure analysis, some very basic and others that are very rigorous and complicated to carry out without higher tool support. The planned work can be included with inherent algorithm or ant colony optimization for further improvements and enhancements in the optimization charge. As the area of software testing be a great deal expand, there is lots of range of investigate used for the scholar and practitioners. In Code Base Software test, the following study area can be worked out by the research scholars –

- Factor Based Code search
- Study of explanation concreteness
- Investigation of Operands plus virtual presentation on overall secret code
- Halstead Metrics testing

To improve the base work done in the existing algorithm having the classical approach with random manner of operands and comments, we will calculate the execution time and complexity of the existing algorithm in the base paper. At the end, the whole research work will be concluded with some future research work. To design an effective and improved model for code coverage including comments density analysis, variables and operands used. Evaluation shall be done on several parameters in active and future move towards.

REFERENCES

1. Amit P. Raut, Garima A. Chudiwale, Priyanka A. Kawale. A Survey And Study Of Software Testing Tools, International Journal of Research In Science & Engineering .e-ISSN: 2394-8299 Special Issue: Techno-Xtreme 16 p-ISSN: 2394-8280
2. Pranali Prakash Mahadik,.D. M. Thakore; International Journal of Innovative Research in Computer and Communication Engineering; Vol. 4, Issue 1, January 2016
3. Ritu, Sukhdip Singh , "A Review Paper on Test Case Selection in Regression Testing" .International Journal of Advanced Res earch in Computer Science and Software Engineering. Volume 6, Issue 5, May 2016.
4. Altaf Hussain, Aamer Nadeem, Muhammad Touseef Ikram, "Review on Formalizing Use Cases and Scenarios: Scenario Based Testing" IEEE 11th International Conference on Emerging Technologies - ICET 2015. DOI: 10.1109/ICET.2015.7389203
5. V.Maheshwari and M. Prasanna, "Generation of Test Case using Automation in Software Systems – A Review". Indian Journal of Science and Technology,Vol8(35),DOI:10.17485 /ijst/2015/v8i35/72881, December 2015
6. Mohamed Monier, Mahmoud Mohamed El-mahdy, "Evaluation of automated web testing tools". International Journal of Computer Applications Technology and Research. Volume 4– Issue 5, 405 - 408, 2015.
7. Syed Asad Ali Shah, Raja Khaim Shahzad, Syed Shafique Ali Bukhari, Nasir Mehmood Minhas, Mamoona Humayun. "A Review of Class Based Test Case Generation Techniques".Manuscript submitted January 13, 2015; accepted March 13, 2015. doi:10.17706/jsw.11.5.464-480.
8. Muhammad Shahid and Suhaimi Ibrahim, "A New Code Based Test Case Prioritization Technique" .International Journal of Software Engineering and Its Applications Vol.8, No.6 (2014), pp.31-38.
9. Neha Pahwa, Kamna Solanki, " UML based Test Case GenerationMethods: A Review" International Journal of Computer Applications (0975 – 8887) Volume 95– No.20, June 2014
10. R.Beena , .S.Sarala , "Code Coverage Based Test Case Selection And Prioritization". International Journal of Software Engineering & Applications (IJSEA), Vol.4, No.6, November 2013.
11. Karambir and Kuldeep Kaur, "Survey of Software Test Case Generation Techniques"International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 6, June 2013.
12. Johannes Ryser and Martin Glinz ,"A Scenario-Based Approach to Validating and Testing Software Systems Using Statecharts" Presented at the 12th International Conference on Software and Systems Engineering and their Applications ICSSEA'99. Proceedings: CNAM, Paris, France 2011.
13. Hitesh Tahbildar and BichitraKalita."Automated Software Test Data Generation: Direction Of Research" International Journal of Computer Science & Engineering. Survey (IJCSES) Vol.2, No.1, Feb 2011.
14. Shay Artzi, Adam Kie˙zun Carlos Pacheco Jeff Perkins, 2010. "Automatic Generation of Unit Regression Tests". MIT CSAI
15. Shin Yoo, Mark Harman. Regression Testing Minimisation, Selection and Prioritisation: A Survey King's College London, Centre for Research on Evolution, Search & Testing, Strand, London, WC2R 2LS, UK ,2009.
16. Chen Mingsong, Qiu Xiaokang, and Li Xuandong, "Automatic Test Case Generation for UML Activity Diagrams" AST'06, May 23, 2006, Shanghai,China.
17. http://en.wikipedia.org/wiki/parsing.
18. .http://www./world.com/javaworld/javaqa/2000-08/01-qa-0804-even.html.
19. Muhammad Hassnain, and Anjum Abbas. 2014. A Literature Survey on Evaluation and Comparison of Regression Testing Techniques for Web Applications.International. Journal of Information Technology and Electrical Engineering ITEE Journal Information Technology & Electrical Engineering .*Volume 3, Issue 3*

20. Ali Athar Khan, Amjad Mahmood, Sajeda M. Amralla and Tahera H. Mirza. 2016. "Comparison of Software Complexity Metrics" International Journal of Computing and Network Technology. Vol 4, No.1
21. Varun Jasuja1, Rajesh Kumar Singh. 2017. Effective Approach for Code Coverage Using Monte Carlo Techniques in Test Case Selection. International Journal of Discrete Mathematics. Vol. 2, No. 3, 2017, pp. 100-106. doi: 10.11648/j.dmath.20170203.17