P- ISSN 1991-8941 E-ISSN 2706-6703 *Journal of University of Anbar for Pure Science (JUAPS)* Open Access
2012,(6), (2) :01-05

Performance Testing Technique for Applied Programs

Mortadha M. Hamad* **Falath M. Mohammad****

* University of Anbar - College of Computer
** University of Anbar - College of arts.

ARTICLE INFO

Received: 00 / 00 /00
Accepted: 00 / 00 /00
Available online: 9/12/2012
DOI: 10.37652/juaps.2012.63156

Keywords:
performance,
time complexity,
space complexity,
reliability,
efficient.

ABSTRACT

In this paper, software was preparedto use for measuring the programs performance because of the importance of measuring the programs performance. The performance of any program basically depends on spent time and storing area needed to implement any program. This work implemented manually is based on trusted rules to guess the executive time. In this research we used the same rules of complexity through a program to give the same manual results automatically and speedily. In addition to the time and storing area, the prepared software uses other standards to analyze the performance of a program like reliability, documentation and others as shown later. All these standards help in taking the appropriate decision about performance.This research was accomplished the performance test of program samples written with Pascal language as easy to understand with simple structures which is provide clear and easy start to test the performance of programs in other languages like c + +.

Introduction

Software has found an enormous dissemination in the past years. There are few machines or facilities left today that are not controlled by software or at least include software. In automobiles, for example, from the engine to the transmission and up to the brakes, more and more functions are controlled by microprocessors and their software. The smooth operation of an enterprise or organization depends largely on the reliability of the software systems used for supporting the business processes or particular task. One way to achieve this goal is systematic evaluation and testing of the developed software. In this paper we depended on some standers to achieve this testing like:

- The time complexity of a program is the amount of computer time it needs to run to completion.
- The space complexity of a program is the amount of memory it needs to run *to completion.*
- *Software Reliability is the probability of failure-free software operation* for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability.We were able in this research to create the program for measuring reliability, where the program measurement and reliable arithmetic expressions that contain the division to avoid division by zero.
- Statistical testing,It is useful to analyse the program or algorithm to compute the number of used loops (for statements) and number of condition statements (if statements) and others. When we compare the qualification of two programs written to solve the same problem, the program uses minimum number of loops to be considered qualified. Also statistics helps the user in improving the program especially when he treats big programs in size.
- Documentation, it is important to know if the programmer enters enough comments to shows the functions of the program's instructions or writes some information about the program like, the time in which the written program starts, ends, and the test

* Corresponding author at: University of Anbar - College of Computer.E-mail address: mortadha61@yahoo.com

development. These comments help user to understand the program and make it simpler in using. The result gives through accounting the rate of the executive lines in the program, to the rate of comments in it.

The proposed method

The system we prepared contains several axes. In addition to the axis of executive time and storing area, we can find another several axes in this system like reliability, documentation and knowledge of the number of subprograms in the program. All these are shown in the following diagram:

Fig (1) the general flow chart of proposed system

The proposed algorithms

Executive Time Algorithm:

This algorithm represents an accurate description for the guessing or spent time to implement the program. We account the time as n where n is the size of income.

Step1: assume that
c = the time complexity of program
n = size of input.
c=0
Step2: Read the program lines
Step3: if program line content iteration statement then c=c + (n+1) .
Step4: if program line content iteration statement previous by another iteration statement then c=c+n^2.
Step5: if program line content If - Then-Else Statement then c=c+2
Step6: if program line content If - Then-Else statement previous by iteration statement then c = c + 2(n^2)
Step7: if program line content Assignment Statements then c=c+1
Step8: if the assignment statement was previous by iteration statement then c=c+n^2.

Example 1:

The following program ready to reading by the purposed program to compute its time complexity as follows,

Program A1

```
var
b, z , I , a : integer;
Begin
b:=0;
z:=0;
For i: = 1to n do
Begin
readln (a);
If a mod 2=0 then
b:=b+1;
Else
z:=z+1;
End;
writeln ('b=',b, 'z=', z);
End.
```

The lines 1, 2, 3 and 4 Do not take any time complexity, the lines 5 and 6 will be carried out once there for it will take timecomplexity equal to 1 for each one, the line 7 take n+1 timescomplexity, the line 10 will be carried out(n) times, the line 11 or 13 will be carried out (n) time, then the total of execution times of line 11 and 13 equals to (n) times. There for the total execution times for all program's lines will be(3+3n) times.

Space Algorithm

Input: Pascal program
Output: the value of space complexity
The process:

```
I=0
Top=0
If a (i) like "function" or "procedure" then
Do until a (i) = end
J=i
T=1
Stack (top) =a (j)
a (j) = ''
If a (j) like "integer" then
Count the number of variables
Space =space + (t*2)
 Else if a (j) like "real" then
Space = space + (t*4)
Top =top +1
Loop
I=1
```

```
Do until i=N
If a (i) like "Type" then
I=i+1
```

Read the variable that pleased before the assignment signal and store it in variable (k3).

If the variable that pleased after the assignment signal like "record" then

```
 J=i
Do until a (j) = "end"
If a (j) like 'integer" then p=p+2
If a (j) like "real" then     p=p+4
If a (j) like "string" then p=p+255
J=j+1
Loop
```

Else if the variable pleased after the assignment signal like "array" then

Count the number of its elements (t) and the kind of it, and then store this value in variable (P).

```
If a (i) like "var" then
J=i+1
Do until a (j) like "Begin"
If a (j) like "array" then
```

Count the number (t) and kind of its elements and added that to the total space

```
If a (j) like "integer" then
Space =space + (t*2)
Else if a (j) like "real" then
Space =space + (t*4)
Else if a (j) like "string "then
Space = space + (t*255)
J=j+1
Loop
Print space
```

Statistical Analysis Algorithm

The input: Pascal program

The output = number of control statements

```
        The process:
        For I =1 to EOF
        Read program's lines from begin statement
        3. Ifa (I) like "for*" then
        4. f=f+1
        5 else if a (I) like"if*" then
        6. n=n+1
        7. Returnf, n
```

Reliability Algorithm

It is one of the important standards which we add to our system as it discovers the cases of failure in programs because the mathematical statements like division on zero, algorithm warns the programmer to these cases.

The process:

Input: Mathematic expression written in Pascal language with Infix state.

Output = Massage box which states whether the program is reliable or not.

The Process:

Step 1: Read the expiration, and see if the expiration contains div symbol.

Step 2: Convert the state of expression to postfix state.

Step 3Set all variables in the expression by value number and set first variable by zero.

Step 4:: Find the total value of expression to avoid division by zero.

Example 2:

```
    Program xz;
    var
    k,a,b,c,d,e,f,g,h : integer;
    Begin
    Read(c,d,a,b,d,f,g,h,e);
    If g<=0 then
    Write ('the value of 'g' must be > 0');
    k:= a – b * (c+d) / (e-f) + g * h;
writeln ('k=',k);
End;
```

In the previous program, the line 8 in which the dividing process is carried out. The purposed program can test if the divisor Expression *(e-f)* + *g* * *h* equal to zero or not and what the programmer doing to avoid dividing by zero.

Results and Discussion

Through the implementation of our program about testing a number of programs written in Pascal, we got the results in table (1).

- We analyze five different programs written to solve different problems. These programs are various in the style of writing and programming tools as our program (system) succeeds in analyzing all these programs one by another. The table contains several standards in addition to time, storing area which used in analyzing the performance of any program, so we added some other standards like documentation, structured and statistical analyses. All these help in analyzing and evaluating the program performance and qualifications. As shown in the table below our program succeeds in guessing time and storing area to implement all these programs. Every program has different executive time from the other program.Seetable (2).

- When applied the purposed program on three programs made to solve the same problem in different manners. The problem is how to find the value of main diagonal in square matrix. When we test and analyze these programs by our system, we get different results for every program from the executive time and this is good when we compare the program performance. see table (3).

Three programs contain arithmetic expressions in mathematical equations are used, if these equations contain division operations, then our program changes automatically. These arithmetic expressions change shape infix into postfix to find the value of this expression. After that we enter the value zero for every variables one after another and notice the total Value of expression if it becomes zero or not according to algorithm mentioned earlier. If any variable makes the total value of the expression zero, then our system will determine this variable and investigate if the programmer who wrote this program warned the user that the value of this variable would not be zero because this will lead to stop the program.

From the three programs R1, R2 and R3 we notice that the programmer in R1 did not warn the user about the value of variable (f) as if it took zero so that the value of denominator will be zero. Then according to our test, the program is not trusted.

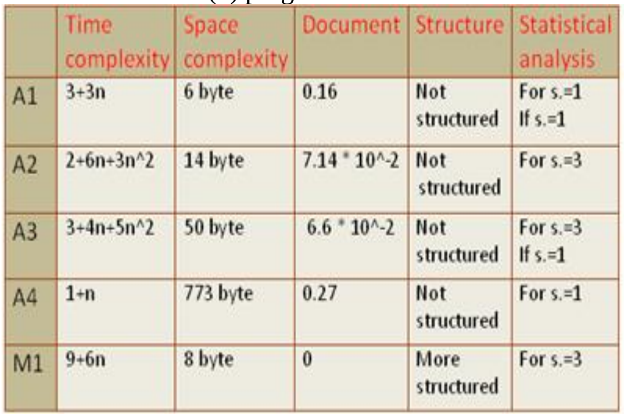
In the program R2, the programmer warns that the user must not enter a value of variable (f) equals zero, so this program is trusted and it does not stop.

In the program R3, it does not need warning or decision because it does not contained division or root operation, so it will not stop because it is trusted from mathematical operation side.

References

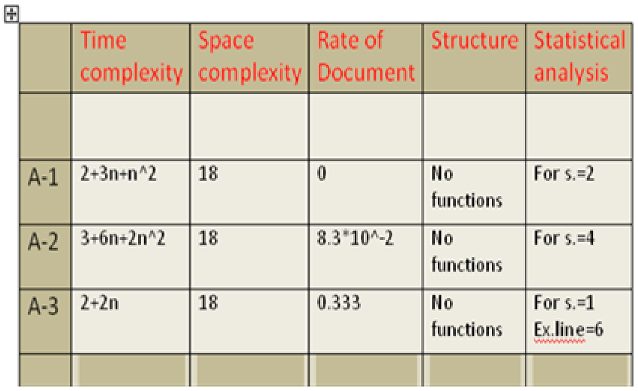
1. Ammann Paul and Offutt Jeff, (2008). Introduction to Software Testing.*Cambridge University Press*. New York
2. Pan Jiantao.(1999). Software Reliability. Carnegie Mellon University.Spring
3. Nyhoff, (2005) Algorithm Efficiency
4. Spillner Andreas, Linz Tilo(2007). Software Testing Foundations, Rocky Nook
5. BhAkshaya.(2009). Software Quality Metrics.
6. Sabetta1 Antonino , Measuring Performance Metrics: Techniques and Tools , Università di Roma, "Tor Vergata", Italy
7. Mette Anne and Hass Jonassen, . (2008). "Guide to Advanced Software Testing, *Artech House inc.*
8. Ajax apis. (2010). Introduction to Parallel Programming and Map Reduce, Goggle Code University.

Table (1) programs characteristic



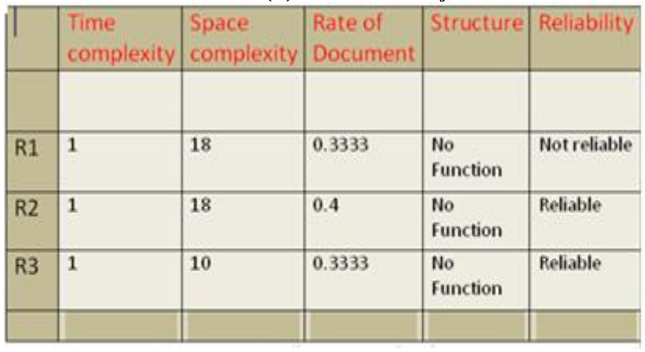
	Time complexity	Space complexity	Document	Structure	Statistical analysis
A1	3+3n	6 byte	0.16	Not structured	For s.=1 If s.=1
A2	2+6n+3n^2	14 byte	7.14 * 10^-2	Not structured	For s.=3
A3	3+4n+5n^2	50 byte	6.6 * 10^-2	Not structured	For s.=3 If s.=1
A4	1+n	773 byte	0.27	Not structured	For s.=1
M1	9+6n	8 byte	0	More structured	For s.=3

Table (2) the programs characteristics



	Time complexity	Space complexity	Rate of Document	Structure	Statistical analysis
A-1	2+3n+n^2	18	0	No functions	For s.=2
A-2	3+6n+2n^2	18	8.3*10^-2	No functions	For s.=4
A-3	2+2n	18	0.333	No functions	For s.=1 Ex.line=6

Table (3) the Reliability



	Time complexity	Space complexity	Rate of Document	Structure	Reliability
R1	1	18	0.3333	No Function	Not reliable
R2	1	18	0.4	No Function	Reliable
R3	1	10	0.3333	No Function	Reliable

تقنية لاختبار اداء البرامج التطبيقية

مرتضى محمد حمد **فلذ منصور محمد**

E.mail: mortadha61@yahoo.com

الخلاصة:

نظرا لأهمية مستوى اداء البرامج قمنا في هذا البحث بتحضير برنامج يقوم بقياس اداء وكفاءة البرامج التطبيقية , حيث يعتمد مستوى الاداء على الوقت المصروف لتنفيذ ذلك البرنامج والمساحة الخزنية. اعتمدنا في هذا العمل على قوانين التعقيد المعتمدة لتخمين الوقت والمساحة الخزنية وبشكل الي وسريع. اعتمد البرنامج معايير اخرى لأجل تحليل الاداء مثل الموثوقية والتوثيق واحصاءات اخرى تساعد في اتخاذ قرار بشأن كفاءة الاداء. تم في هذا البحث اختبار اداء عينات برامج بسيطة مكتوبة بلغة باسكال كونها لغة سهلة الفهم وبسيطة التركيب مما يوفر بداية واضحة وسهلة لاختبار اداء برامج بلغات اخرى مثل ++c.