

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Качество и метрология программного обеспечения»

Тема: «Построение операционной графовой модели программы (ОГМП)
и расчет характеристик эффективности ее выполнения методом
эквивалентных преобразований»

Студент гр. 6304

Зыль С.Е.

Преподаватель

Кирияничков В.А.

Санкт-Петербург

2020

Задание.

1. Построение ОГМП.

Для рассматривавшегося в лабораторных работах 1-3 индивидуального задания разработать операционную модель управляющего графа программы на основе схемы алгоритма. При выполнении работы рекомендуется для упрощения обработки графа исключить диалог при выполнении операций ввода-вывода данных, а также привести программу к структурированному виду.

Выбрать вариант графа с нагруженными дугами, каждая из которых должна представлять фрагмент программы, соответствующий линейному участку или ветвлению. При расчете вероятностей ветвлений, зависящих от распределения данных, принять равномерное распределение обрабатываемых данных в ограниченном диапазоне (например, $[0,100]$ - для положительных чисел или $[-100,100]$ - для произвольных чисел). В случае ветвлений, вызванных проверкой выхода из цикла, вероятности рассчитываются исходя априорных сведений о числе повторений цикла. Сложные случаи оценки вероятностей ветвлений согласовать с преподавателем.

В качестве параметров, характеризующих потребление ресурсов, использовать времена выполнения команд соответствующих участков программы. С помощью монитора Sampler выполнить оценку времен выполнения каждого линейного участка в графе программы.

2. Расчет характеристик эффективности выполнения программы методом эквивалентных преобразований.

Полученную в части 1 данной работы ОГМП, представить в виде графа с нагруженными дугами, у которого в качестве параметров, характеризующих потребление ресурсов на дуге ij , использовать тройку $\{P_{ij}, M_{ij}, D_{ij}\}$, где:

P_{ij} - вероятность выполнения процесса для дуги ij ,

M_{ij} - мат.ожидание потребления ресурса процессом для дуги ij ,

D_{ij} - дисперсия потребления ресурса процессом для дуги ij .

В качестве потребляемого ресурса в данной работе рассматривается время процессора, а оценками мат. ожиданий времен для дуг исходного графа следует

принять времена выполнения операторов (команд), соответствующих этим дугам участков программы. Дисперсиям исходных дуг следует присвоить нулевые значения.

Получить описание полученной ОГМП на входном языке пакета CSA III в виде поглощающей марковской цепи (ПМЦ) – (англ.) AMC (absorbingMarkovchain) и/или эргодической марковской цепи (ЭМЦ) - EMC (ergodicMarkovchain).

С помощью предоставляемого пакетом CSA III меню действий выполнить расчет среднего времени и дисперсии времени выполнения как для всей программы, так и для ее фрагментов, согласованных с преподавателем.

Построение операционной графовой модели.

Исходный текст программы

```
program bubble_sort;
uses sampler;

const    max      = 100;
szUnit: String = 'KMPO.pas';

type     ary      = array[1..max] of real;

var      x        : ary;
i,n      : integer;

procedure sort1(var a: ary; n: integer);
var      i,j      : integer;
        hold : real;

begin
  for i:=1 to n-1 do //1
    for j:=i+1 to n do //2
      begin
        if a[i]>a[j] then //3
          begin
            hold:=a[i];
            a[i]:=a[j];
            a[j]:=hold; //4
          end
        end
      end
    end
  end; //5

procedure sort2(var a: ary; n: integer);

var      no_change : boolean;
        j          : integer;
```

```

procedure swap(var a: ary; p,q: integer);
var hold      : real;
begin
  hold:=a[p];
  a[p]:=a[q];
  a[q]:=hold;
end;
begin
  repeat
no_change:=true; //1
    for j:=1 to n-1 do //2
      begin
        if a[j]>a[j+1] then //3
          begin
            swap(a, j, j+1);
no_change:=false; //4
          end
        end
      until no_change //5
    end; //6

var  forFirstSort, forSecondSort      : ary;
    randomNumber                      : real;
begin
  randomize;
  for i:=1 to max do
    begin
      randomNumber:=random*1000;
      forFirstSort[i]:=randomNum;
      forSecondSort[i]:=randomNum;
    end;

    sort1(forFirstSort, max);
    sort2(forSecondSort, max);
  end.

```

Граф управления программы.

Две функции рассматривал отдельно.

Графы управления строились только для функции, а не для главного тела, так как в главном теле программы только инициализация всех элементов и вызов функции.

Графы управления представлены на рис. 1.

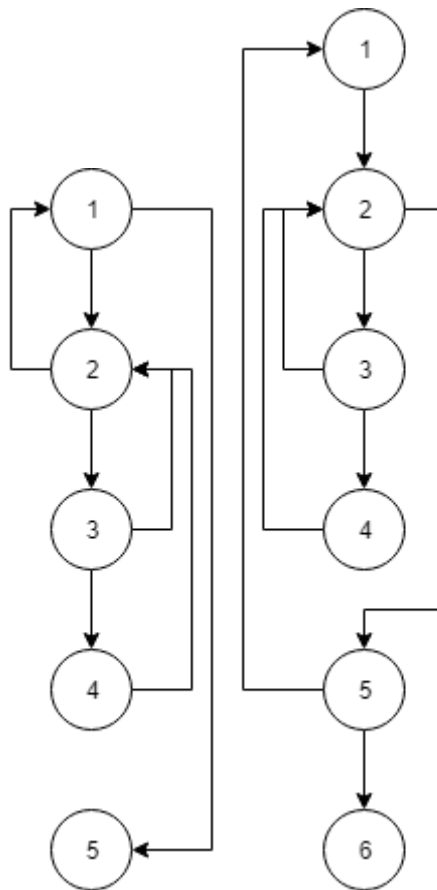


Рисунок 1 – Графы управления программы

Профилирование.

Текст программы (подготовленный для профилирования).

```

program bubble_sort;
uses sampler;

const    max      = 100;
szUnit: String = 'KMPO.pas';

type     ary       = array[1..max] of real;

var      x         : ary;
i,n      : integer;

procedure sort1(var a: ary; n: integer);
var      i,j       : integer;
        hold : real;

begin
    Sample(szUnit, 1);
    for i:=1 to n-1 do
        begin
            Sample(szUnit, 2);
            for j:=i+1 to n do

```

```

begin
  Sample(szUnit, 3);
  if a[i]>a[j] then
    begin
      Sample(szUnit, 4);
      hold:=a[i];
      a[i]:=a[j];
      a[j]:=hold;
      Sample(szUnit, 5);
    end;
  Sample(szUnit, 6);
end;
Sample(szUnit, 7);
end;
Sample(szUnit, 8);
end;

procedure sort2(var a: ary; n: integer);

var      no_change : boolean;
         j          : integer;

procedure swap(var a: ary; p,q: integer);
var hold      : real;
begin
  hold:=a[p];
  a[p]:=a[q];
  a[q]:=hold;
end;
begin
  Sample(szUnit, 9);
  repeat
    Sample(szUnit, 10);
    no_change:=true;
    Sample(szUnit, 11);
    for j:=1 to n-1 do
      begin
        Sample(szUnit, 12);
        if a[j]>a[j+1] then
          begin
            Sample(szUnit, 13);
            swap(a, j, j+1);
no_change:=false;
            Sample(szUnit, 14);
          end;
        Sample(szUnit, 15);
      end;
    Sample(szUnit, 16);
  until no_change;
  Sample(szUnit, 17);
end;

var  forFirstSort, forSecondSort      : ary;

```

```

randomNum                                     : real;
begin
  randomize;
  for i:=1 to max do
  begin
    randomNum:=random*1000;
    forFirstSort[i]:=randomNum;
    forSecondSort[i]:=randomNum;
  end;

  sort1(forFirstSort, max);
  sort2(forSecondSort, max);
end.

```

Результаты профилирования

Отчет о результатах измерений для программы kmpo.

Создан программой Sampler(версия от Feb 15 1999)
1995-98 (с) СПбГЭТУ, Мойсейчук Леонид.

Список обработанных файлов.

NN Имя обработанного файла

1. КМРО.pas

Таблица с результатами измерений (используется 17 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 1	1 : 2	0.00	1	0.00
1 : 2	1 : 3	0.00	99	0.00
1 : 3	1 : 6	13.47	2912	0.00
1 : 3	1 : 4	18.26	2038	0.01
1 : 4	1 : 5	289.44	2038	0.14

1 : 5 1 : 6	0.00	2038	0.00
1 : 6 1 : 3	0.00	4851	0.00
1 : 6 1 : 7	0.00	99	0.00
1 : 7 1 : 2	10.54	98	0.11
1 : 7 1 : 8	0.00	1	0.00
1 : 8 1 : 9	0.84	1	0.84
1 : 9 1 : 10	0.00	1	0.00
1 : 10 1 : 11	1.68	99	0.02
1 : 11 1 : 12	0.00	99	0.00
1 : 12 1 : 15	24.83	7307	0.00
1 : 12 1 : 13	12.94	2494	0.00
1 : 13 1 : 14	456.32	2494	0.18
1 : 14 1 : 15	0.00	2494	0.00
1 : 15 1 : 12	52.64	9702	0.01
1 : 15 1 : 16	0.00	99	0.00
1 : 16 1 : 10	13.24	98	0.13
1 : 16 1 : 17	0.84	1	0.84

Расчет вероятностей и затрат ресурсов для дуг управляющего графа.

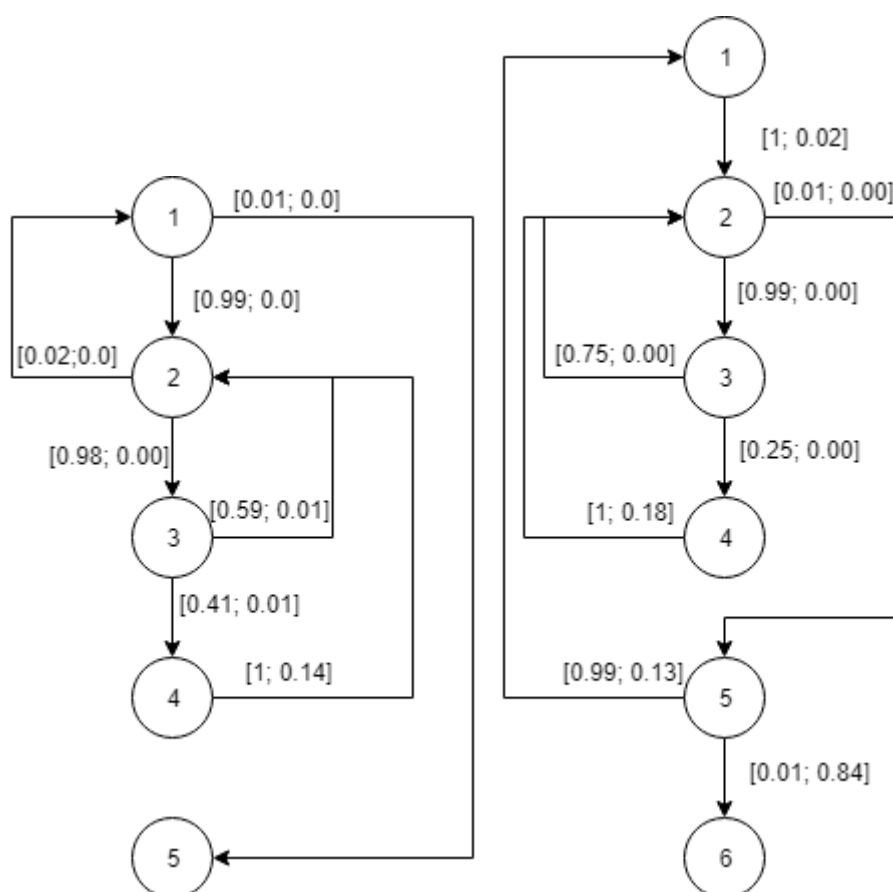
Для первого графа

	Номера точек	Количество проходов
L1= 0.11	1-2; 7-2	1; 98
L2= 0.00	2-3; 6-3	99; 4851
L3= 0.01	3-4; 3-6	2038; 2912
L4= 0.14	4-5	2038

Для второго графа

	Номера точек	Количество проходов
L1= 0.00	9-10; 16-10	1; 98
L2= 0.01	11-12; 15-12	99; 9702
L3= 0.00	12-13; 12-15;	2494; 7307
L4= 0.18	13-14	2494
L5= 0.00	15-16	99

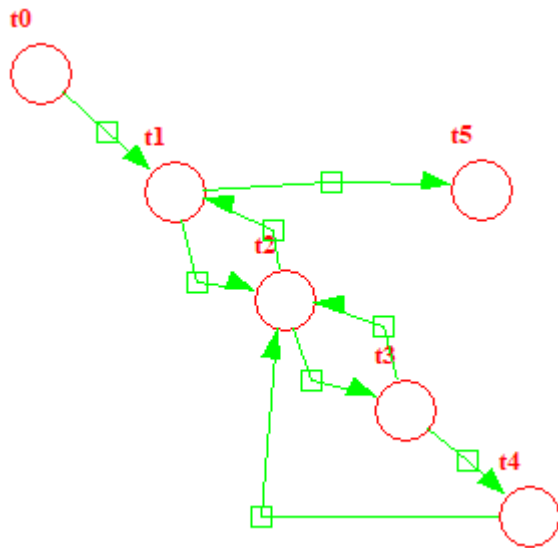
Операционная графовая модель программы.



Расчет характеристик эффективности выполнения программы с помощью пакета CSAIII методом эквивалентных преобразований.

Первая функция

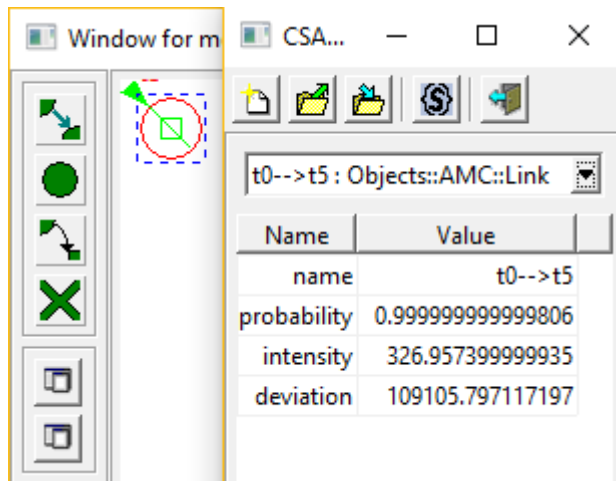
ГНД



Описание модели

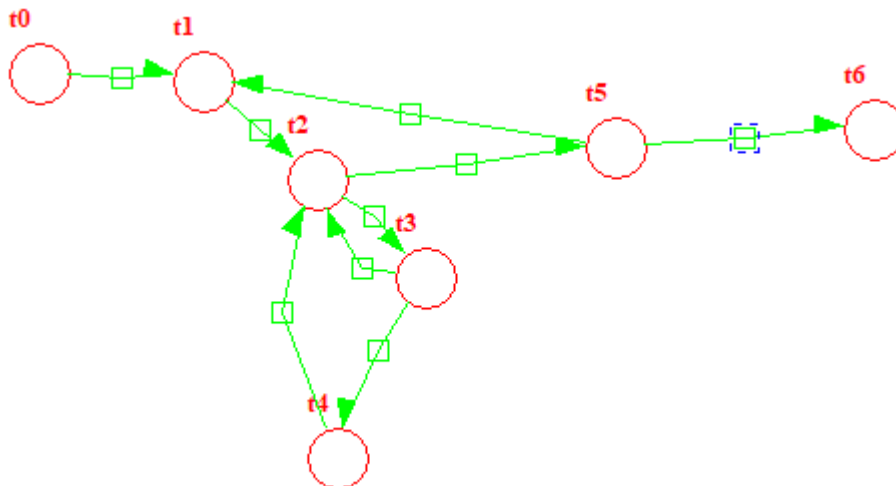
```
<model type = "Objects::AMC::Model" name = "LR1">
  <node type = "Objects::AMC::Top" name = "t0"></node>
  <node type = "Objects::AMC::Top" name = "t1"></node>
  <node type = "Objects::AMC::Top" name = "t2"></node>
  <node type = "Objects::AMC::Top" name = "t3"></node>
  <node type = "Objects::AMC::Top" name = "t4"></node>
  <node type = "Objects::AMC::Top" name = "t5"></node>
  <link type = "Objects::AMC::Link" name = "t0-->t1" probability =
"1.0" intensity = "0.0" deviation = "0.0" source = "t0" dest =
"t1"></link>
  <link type = "Objects::AMC::Link" name = "t1-->t2" probability =
"0.99" intensity = "0.0" deviation = "0.0" source = "t1" dest =
"t2"></link>
  <link type = "Objects::AMC::Link" name = "t2-->t3" probability =
"0.98" intensity = "0.0" deviation = "0.0" source = "t2" dest =
"t3"></link>
  <link type = "Objects::AMC::Link" name = "t2-->t1" probability =
"0.02" intensity = "0.0" deviation = "0.0" source = "t2" dest =
"t1"></link>
  <link type = "Objects::AMC::Link" name = "t3-->t2" probability =
"0.59" intensity = "0.01" deviation = "0.0" source = "t3" dest =
"t2"></link>
  <link type = "Objects::AMC::Link" name = "t3-->t4" probability =
"0.41" intensity = "0.01" deviation = "0.0" source = "t3" dest =
"t4"></link>
  <link type = "Objects::AMC::Link" name = "t4-->t2" probability =
"1.0" intensity = "0.14" deviation = "0.0" source = "t4" dest =
"t2"></link>
  <link type = "Objects::AMC::Link" name = "t1-->t5" probability =
"0.01" intensity = "0.0" deviation = "0.0" source = "t1" dest =
"t5"></link>
</model>
```

Результат



Вторая функция

ГНД



Описание модели

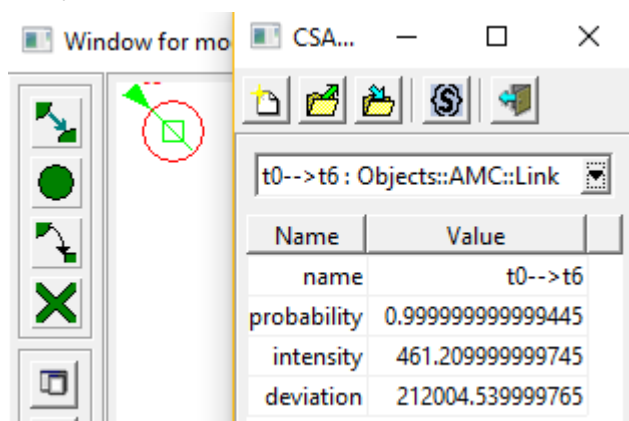
```
<model type = "Objects::AMC::Model" name = "LR2">
  <node type = "Objects::AMC::Top" name = "t0"></node>
  <node type = "Objects::AMC::Top" name = "t1"></node>
  <node type = "Objects::AMC::Top" name = "t2"></node>
  <node type = "Objects::AMC::Top" name = "t3"></node>
  <node type = "Objects::AMC::Top" name = "t4"></node>
  <node type = "Objects::AMC::Top" name = "t5"></node>
  <node type = "Objects::AMC::Top" name = "t6"></node>
  <link type = "Objects::AMC::Link" name = "t0-->t1" probability =
"1.0" intensity = "0.0" deviation = "0.0" source = "t0" dest =
"t1"></link>
  <link type = "Objects::AMC::Link" name = "t1-->t2" probability =
"1.0" intensity = "0.02" deviation = "0.0" source = "t1" dest =
"t2"></link>
```

```

<link type = "Objects::AMC::Link" name = "t2-->t3" probability =
"0.99" intensity = "0.0" deviation = "0.0" source = "t2" dest =
"t3"></link>
<link type = "Objects::AMC::Link" name = "t2-->t5" probability =
"0.01" intensity = "0.0" deviation = "0.0" source = "t2" dest =
"t5"></link>
<link type = "Objects::AMC::Link" name = "t3-->t4" probability =
"0.25" intensity = "0.0" deviation = "0.0" source = "t3" dest =
"t4"></link>
<link type = "Objects::AMC::Link" name = "t3-->t2" probability =
"0.75" intensity = "0.0" deviation = "0.0" source = "t3" dest =
"t2"></link>
<link type = "Objects::AMC::Link" name = "t4-->t2" probability =
"1.0" intensity = "0.18" deviation = "0.0" source = "t4" dest =
"t2"></link>
<link type = "Objects::AMC::Link" name = "t5-->t1" probability =
"0.99" intensity = "0.13" deviation = "0.0" source = "t5" dest =
"t1"></link>
<link type = "Objects::AMC::Link" name = "t5-->t6" probability =
"0.01" intensity = "0.84" deviation = "0.0" source = "t5" dest =
"t6"></link>
</model>

```

Результат



Name	Value
t0-->t6	Objects::AMC::Link
probability	0.999999999999445
intensity	461.2099999999745
deviation	212004.5399999765

Вывод.

При выполнении лабораторной работы была построена операционная графовая модель заданной программы, нагрузочные параметры которой были оценены с помощью профилировщика Samplern методом эквивалентных преобразований с помощью пакета CSAIII были вычислены математическое ожидание и дисперсия времени выполнения для всей программы. Результаты полученных характеристик с помощью пакета CSAIII совпали с результатами, полученными в лабораторной работе 3.