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. 7304 _____ Абдульманов Э.М

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы:

Изучение возможности построения операционной графовой модели программы (ОГМП) и расчета характеристик эффективности ее выполнения методом эквивалентных преобразований.

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, [0,100.00] - для положительных чисел или [-100.00, 100.00] - для произвольных чисел). В случае ветвлений, вызванных проверкой выхода из цикла, вероятности рассчитываются исходя из 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, полученные с помощью монитора Sampler в процессе выполнения работы №3. Если требуется, оценить с помощью монитора Sampler времена выполнения неучтенных ранее участков 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{Pij,Mij,Dij}, где:

Pij - вероятность выполнения процесса для дуги ij,

Mij - мат.ожидание потребления ресурса процессом для дуги ij,

Dij - 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.

Выполнить описание построенной ОГМП на входном языке пакета CSA III в виде поглощающей марковской цепи (ПМЦ) – (англ.) AMC (absorbing Markov chain) или эргодической марковской цепи (ЭМЦ) - EMC (ergodic Markov chain).

Ход работы:

1. Построение операционной графовой модели

Исходный код программы представлен в приложении А.

2. Граф управления программы

Граф управления представлен на Рисунке 1.

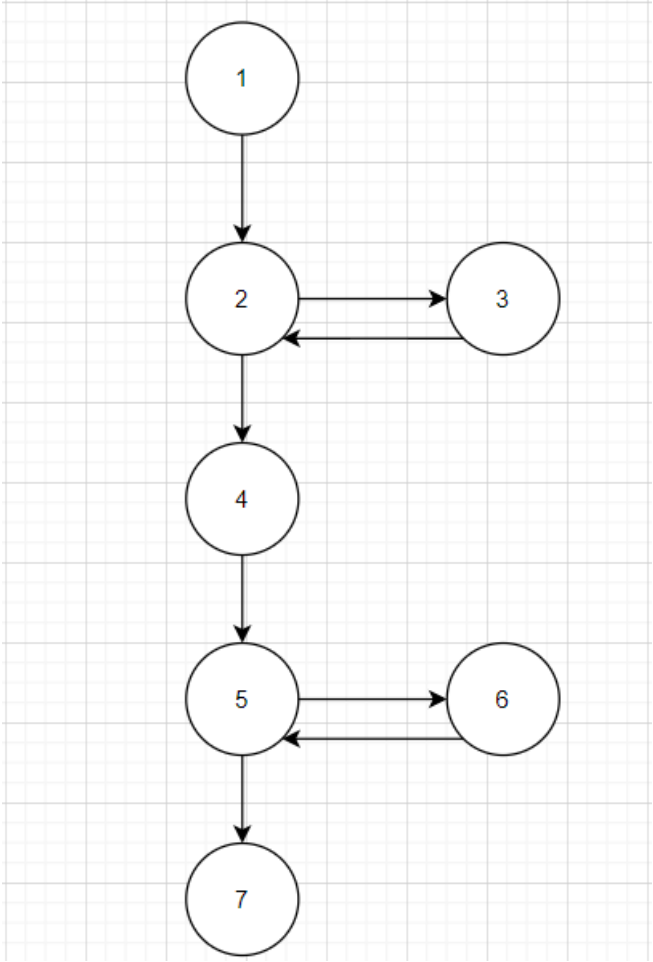

Рисунок 1 – Граф управления

3. Профилирование

Исходный текст программы для профилирования представлен в приложении Б. Результаты профилирования представлены на Рисунке 2.

Список обработанных файлов.

NN	Имя обработанного файла
1.	LAB4.CPP

Таблица с результатами измерений (используется 14 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 8	1 : 19	4.19	1	4.19
1 : 19	1 : 23	1.68	1	1.68
1 : 23	1 : 35	18115.46	1000	18.12
1 : 35	1 : 23	1420.77	999	1.42
1 : 35	1 : 38	0.84	1	0.84
1 : 38	1 : 46	469.33	1	469.33
1 : 46	1 : 50	1.68	1	1.68
1 : 50	1 : 54	8533.50	1000	8.53
1 : 54	1 : 50	1428.12	999	1.43
1 : 54	1 : 57	1.68	1	1.68
1 : 57	1 : 83	2.51	1	2.51
1 : 67	1 : 71	1.68	1	1.68
1 : 71	1 : 76	30876.31	1000	30.88
1 : 76	1 : 71	1769.22	999	1.77
1 : 76	1 : 79	1.68	1	1.68
1 : 79	1 : 8	7.54	1	7.54

Рисунок 2 - Результаты профилирования

4. Расчет вероятностей и затрат ресурсов для дуг управляющего графа

Расчет вероятностей и затрат ресурсов для дуг управляющего графа представлен в Таблице 1.

	Ресурсы, мкс	Номера строк	Количество проход
L1	4.19	8:19	1
L2	1.68, 1.42, 0.84	19:23, 35:23, 35:38	1, 999, 1
L3	18.12	23:35	1000
L4	469	38:46	1
L5	1.68, 1.43, 1.68	46:50, 54:50, 54:57	1, 999, 1
L6	8.53	50:54	1000
L7	-	-	-

Таблица 1 – Вероятности и затраты ресурсов для дуг управляющего графа

5. Операционная графовая модель программы

Операционная графовая модель представлена на Рисунке 3.

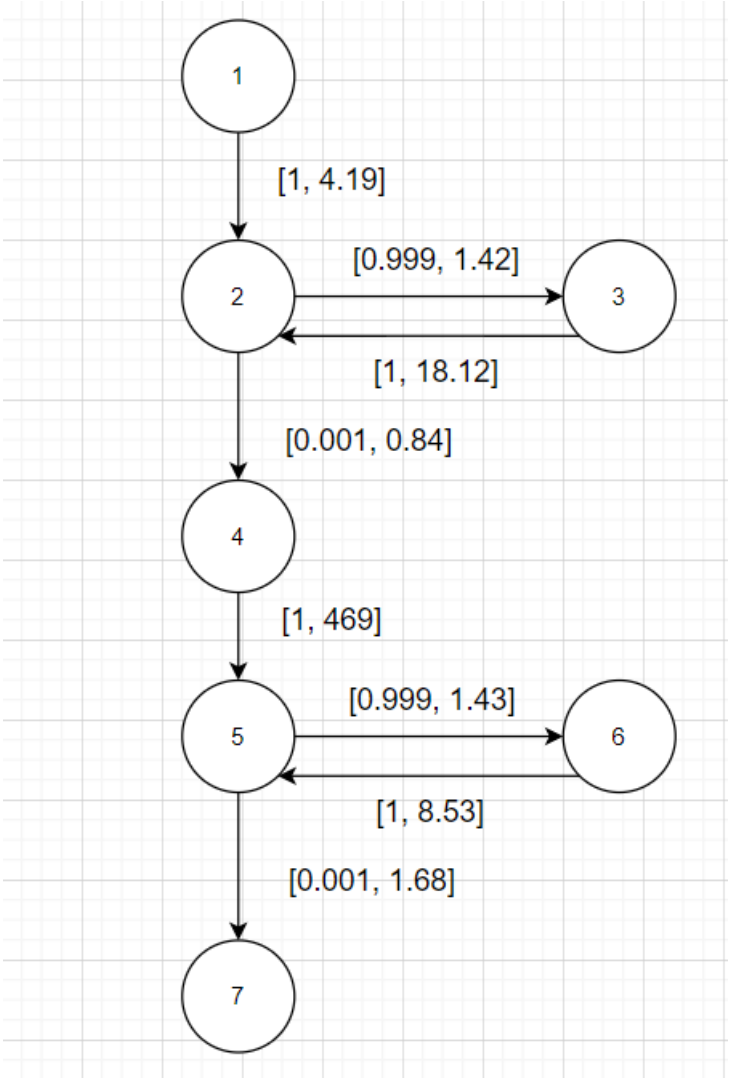

Рисунок 3 – Операционная графовая модель

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

Граф с нагруженными дугами представлен на Рисунке 4.

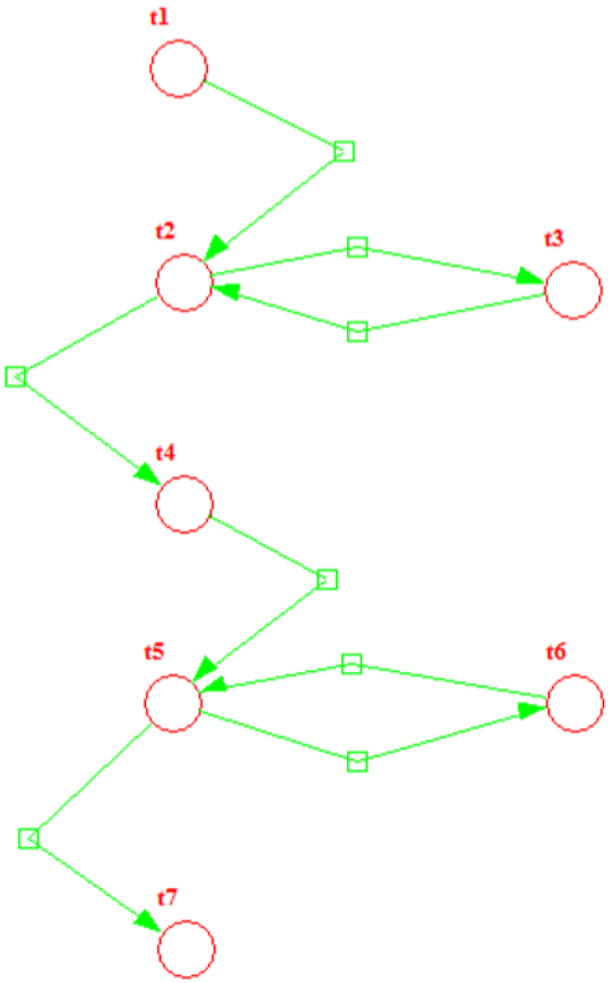

Рисунок 4 – Граф с нагруженными дугами

Описание модели:

```
<model type="Objects::AMC::Model" name="lab4">
      <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>
      <node type="Objects::AMC::Top" name="t7"></node>
      <link type="Objects::AMC::Link" name="t1-t2" probability="1"
intensity="4.19" deviation="0.0" source="t1" dest="t2"></link>
```

```
      <link type="Objects::AMC::Link" name="t2-t3" probability="0.999"
intensity="1.42" deviation="0.0" source="t2" dest="t3"></link>
      <link type="Objects::AMC::Link" name="t3-t2" probability="1"
intensity="18.12" deviation="0.0" source="t3" dest="t2"></link>
      <link type="Objects::AMC::Link" name="t2-t4" probability="0.001"
intensity="0.84" deviation="0.0" source="t2" dest="t4"></link>
      <link type="Objects::AMC::Link" name="t4-t5" probability="1"
intensity="469" deviation="0.0" source="t4" dest="t5"></link>
      <link type="Objects::AMC::Link" name="t5-t6" probability="0.999"
intensity="1.43" deviation="0.0" source="t5" dest="t6"></link>
      <link type="Objects::AMC::Link" name="t6-t5" probability="1"
intensity="8.53" deviation="0.0" source="t6" dest="t5"></link>
      <link type="Objects::AMC::Link" name="t5-t7" probability="0.001"
intensity="1.68" deviation="0.0" source="t5" dest="t7"></link>
</model>
```

Результат представлен на Рисунке 5.

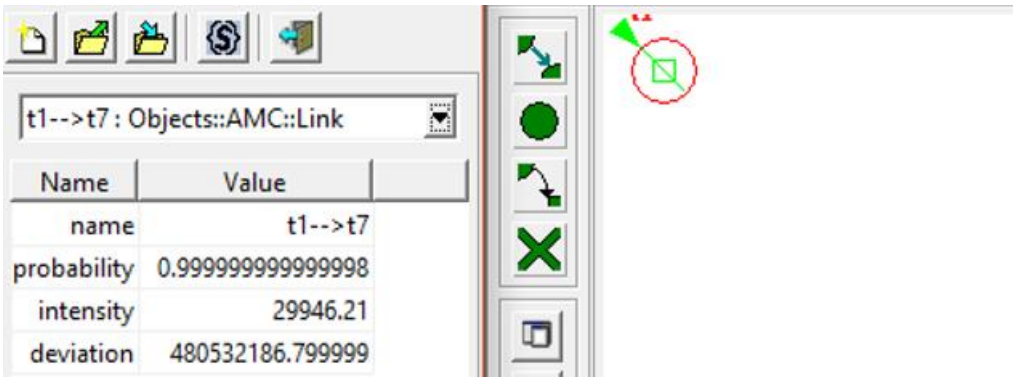

Рисунок 5 – Результат

Выводы:

При выполнении лабораторной работы была построена операционная графовая модель заданной программы и выполнено профилирование ее функциональных участков с помощью SAMPLER, а также 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 отличаются от полученного с помощью SAMPLER на 0.1%, что говорит о адекватности построенной модели.

ПРИЛОЖЕНИЕ А.
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
1 #include <stdlib.h>
2
3 #define SIZE 1000
4
5 void linfit(const float* x, const float* y, float* y_calc, float* a, float* b,
int n) {
6     float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
7
8     sum_x = 0.0;
9     sum_y = 0.0;
10    sum_xy = 0.0;
11    sum_x2 = 0.0;
12
13    int i = 0;
14
15    for (i = 0; i < n; i++) {
16          xi = x[i];
17          yi = y[i];
18
19          sum_x += xi;
20          sum_y += yi;
21
22          sum_xy += xi * yi;
23
24          sum_x2 += xi * xi;
25    }
26
27    sxx = sum_x2 - sum_x * sum_x / n;
28    sxy = sum_xy - sum_x * sum_y / n;
29
30    *a = sxy / sxx;
31    *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
32
33    for (i = 0; i < n; i++) {
34          y_calc[i] = *a + *b * x[i];
35    }
36 }
37
38 int main(int argc, char* argv[]) {
```

```
39    float x[SIZE];
40    float y[SIZE];
41    float y_calc[SIZE];
42
43    float a, b;
44
45    for (int i = 0; i < SIZE; i++) {
46          x[i] = rand() % 200 - 100; // [-100, 100)
47          y[i] = rand() % 200 - 100; // [-100, 100)
48    }
49
50    linfit(x, y, y_calc, &a, &b, SIZE);
51
52    return 0;
53 }
```

ПРИЛОЖЕНИЕ Б.

ТЕКСТ ПРОГРАММЫ ДЛЯ ПРОФИЛИРОВАНИЯ

```
1 #include <stdlib.h>
2
3 #include <SAMPLER.H>
4
5 #define SIZE 1000
6
7 void linfit(const float* x, const float* y, float* y_calc, float* a, float* b,
int n) {
8     SAMPLE;
9
10    float sum_x, sum_y, sum_xy, sum_x2, xi, yi, sxx, sxy;
11
12    sum_x = 0.0;
13    sum_y = 0.0;
14    sum_xy = 0.0;
15    sum_x2 = 0.0;
16
17    int i = 0;
18
19    SAMPLE;
20
21    for (i = 0; i < n; i++) {
22
23          SAMPLE;
24
25          xi = x[i];
26          yi = y[i];
27
28          sum_x += xi;
29          sum_y += yi;
30
31          sum_xy += xi * yi;
32
33          sum_x2 += xi * xi;
34
35          SAMPLE;
36    }
37
38    SAMPLE;
```

```
39
40    sxx = sum_x2 - sum_x * sum_x / n;
41    sxy = sum_xy - sum_x * sum_y / n;
42
43    *a = sxy / sxx;
44    *b = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
45
46    SAMPLE;
47
48    for (i = 0; i < n; i++) {
49
50          SAMPLE;
51
52          y_calc[i] = *a + *b * x[i];
53
54          SAMPLE;
55    }
56
57    SAMPLE;
58 }
59
60 int main(int argc, char* argv[]) {
61    float x[SIZE];
62    float y[SIZE];
63    float y_calc[SIZE];
64
65    float a, b;
66
67    SAMPLE;
68
69    for (int i = 0; i < SIZE; i++) {
70
71          SAMPLE;
72
73          x[i] = rand() % 200 - 100;
74          y[i] = rand() % 200 - 100;
75
76          SAMPLE;
77    }
78
79    SAMPLE;
80
```

```
81    linfit(x, y, y_calc, &a, &b, SIZE);
82
83    SAMPLE;
84
85    return 0;
86 }
```