

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

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

Ковынев М.В.

Преподаватель

Кирияничков В.А.

Санкт-Петербург

2020

Формулировка задания

1.1. 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.

1.2. Расчет характеристик эффективности выполнения программы методом эквивалентных преобразований.

Полученную в части 1.1 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.

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).

Это можно сделать двумя способами:

1) Создать с помощью набора стандартных инструментов, предусмотренных в графическом интерфейсе пакета программ CSA III (рисование); выходные операционные графовые модели отображаются на экране, и могут быть сохранены в виде PNG-изображения, а их описание – в виде XML-файла.

2) Описать непосредственно в виде XML-файла.

Подробные сведения по созданию моделей и их последующей обработке для расчета характеристик эффективности выполнения программы приведены в руководстве работы с пакетом CSA III – файл CSA3 Guide.doc.

Полученный XML-файл описания модели для контроля преподавателем следует **преобразовать в более компактную и удобную для понимания форму** с помощью преобразователя csa.xml, поместив его в каталог csa-model-html-exporter.

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.

3. Исходный код программы с расставленными метками для профилирования представлен в приложении Б.

4. Результаты профилирования представлены ниже.

NN	Имя обработанного файла
1.	..\SAMPLE~1\SAMP16\LAB_4.CPP

Таблица с результатами измерений (используется 27 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 16	1 : 19	0.00	1	0.00
1 : 19	1 : 21	68.72	5	13.74
1 : 21	1 : 24	7.54	5	2.51
1 : 24	1 : 26	166.78	10	16.68
1 : 26	1 : 24	5.87	5	1.17
1 : 26	1 : 28	0.00	5	0.00
1 : 28	1 : 30	75.43	5	15.09
1 : 30	1 : 19	5.87	4	1.47
1 : 30	1 : 32	1.68	1	1.68
1 : 32	1 : 85	2.51	1	2.51
1 : 37	1 : 40	0.00	1	0.00
1 : 40	1 : 43	5.03	3	1.68
1 : 43	1 : 45	78.78	6	13.13
1 : 45	1 : 48	10.90	6	2.82
1 : 48	1 : 50	538.90	30	17.96
1 : 50	1 : 57	6.70	15	0.45
1 : 50	1 : 53	9.22	15	0.61
1 : 53	1 : 55	167.62	15	13.7
1 : 55	1 : 57	0.00	15	0.00
1 : 57	1 : 48	36.04	24	1.50
1 : 57	1 : 59	0.00	6	0.00
1 : 59	1 : 61	4.19	3	1.40
1 : 59	1 : 43	3.35	3	1.12
1 : 61	1 : 63	61.18	3	20.39

1 : 63	1 : 66	0.00	3	0.00
1 : 66	1 : 68	249.75	15	16.65
1 : 68	1 : 66	14.25	12	1.19
1 : 68	1 : 70	4.19	3	1.40
1 : 70	1 : 40	3.35	2	1.68
1 : 70	1 : 72	0.00	1	0.00
1 : 72	1 : 87	1.68	1	1.68
1 : 83	1 : 16	4.19	1	4.19
1 : 85	1 : 37	5.03	1	5.03

5. Создадим операционную графовую модель программы. Исходный код xml представлен в Приложении В.

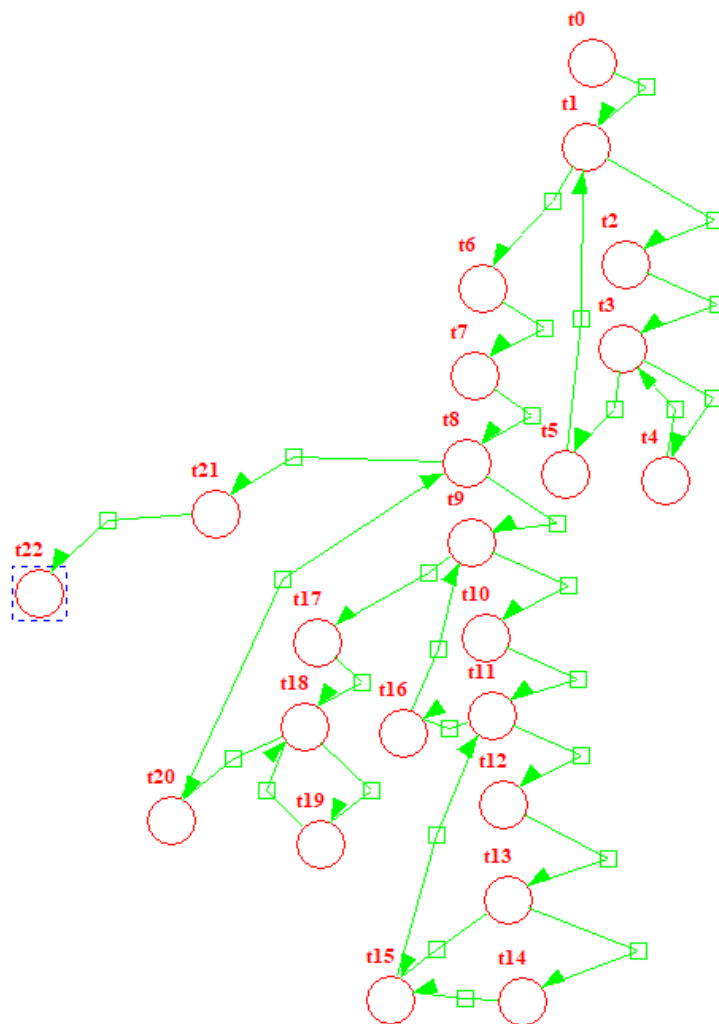
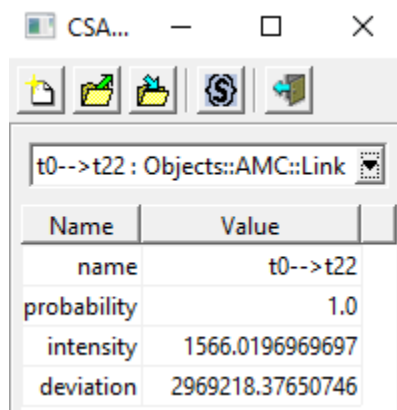


Рисунок 2 — Графовая модель

Таблица 1 — Расчет вероятностей и затрат ресурсов для дуг графа

Строки	Описание	i	j	P_{ij}	L_{ij}
83:16	Вызов get_data	1	2	1	4.19
16:32	Вход в цикл – 4 раз, выход – 1. $P=4/5=0.8, Q=1/5=0.2$	2	3	0.80	0
		2	7	0.20	0.84
21:24	Вход в цикл по j	3	4	1	0
21:28	Вход в цикл – 2 раза, выход – 1. $P=2/3=0.67, Q=1/3=0.33$	4	5	0.67	2.51
		4	6	0.33	0
24:26	Расчет $x[i][j]$	5	4	1	16.68
28:30	Расчет $y[i]$	6	2	1	15.09
30:85	Завершение get_data	7	8	1	2.51
85:37	Вызов square	8	9	1	0
41:43	Цикл. Вход – 3, Выход – 1. $P=3/4=0.75, Q=1/4=0.25$	9	10	0.75	0
		9	22	0.25	0
40:61	Цикл. Вход – 2, Выход – 1. $P=2/3=0.67, Q=1/3=0.33$	10	11	0.67	1.68
		10	18	0.33	0.61
43:45	Расчет $a[k][l]$	11	12	1	13.13
45:59	Цикл. Вход – 4, Выход – 1. $P=4/5=0.8, Q=1/5=0.2$	12	13	0.8	2.82
		12	17	0.2	0
48:50	Расчет $a[k][l]$	13	14	1	17.96
50:57	Ветвление. Вход – 15, Пропуск – 15. $P=27/54=0.5, Q=27/54=0.5$	14	15	0.5	0.61
		14	16	0.5	0.45
53:55	Расчет $a[l][k]$	15	16	1	13.7
57:48	Переход по циклу i	16	12	1	1.50
59:43	Переход по циклу l	17	10	1	1.12
61:63	Расчет $g[k]$	18	19	1	20.39
64:69	Цикл. Вход – 4, Выход – 1. $P=4/5=0.8, Q=1/5=0.2$	19	20	0.9	0
		19	21	0.1	0
66:68	Расчет $g[k]$	20	19	1	16.65
70:40	Переход по циклу k	21	9	1	1.68
72:87	Завершение square	22	23	1	5.03

6. Сравнение результатов



The screenshot shows a window titled 'CSA...' with a toolbar containing icons for file operations and settings. Below the toolbar is a table with the following data:

Name	Value
name	t0-->t22
probability	1.0
intensity	1566.0196969697
deviation	2969218.37650746

Рисунок 3 — Результат

Результат из предыдущей работы — 1673.95. Результаты совпадают на 93.5%.

Вывод

При выполнении лабораторной работы была построена операционная графовая модель заданной программы, нагрузочные параметры которой были оценены с помощью профилировщика Sampler и методом эквивалентных преобразований с помощью пакета CSA III были вычислены математическое ожидание и дисперсия времени выполнения всей программы.

ПРИЛОЖЕНИЕ А

Исходный код на С

```
1. #include <stdio.h>
2.
3.
4. #define rmax 9
5. #define cmax 3
6.
7. typedef double ary[rmax];
8. typedef double arys[cmax];
9. typedef double ary2[rmax][cmax];
10. typedef double ary2s[cmax][cmax];
11.
12.
13. void get_data(ary2 x, ary y, int nrow, int ncol)
14. {
15.     for (int i = 0; i < nrow; i++)
16.     {
17.         x[i][0] = 1;
18.         for (int j = 1; j < ncol; j++)
19.         {
20.             x[i][j] = (i + 1) * x[i][j - 1];
21.         }
22.         y[i] = 2 * (i + 1);
23.     }
24. }
25.
26. void square(ary2 x, double* y, ary2s a, double* g, int nrow, int ncol)
27. {
28.     for (int k = 0; k < ncol; k++)
29.     {
30.         for (int l = 0; l <= k; l++)
31.         {
32.             a[k][l] = 0;
33.             for (int i = 0; i < nrow; i++)
34.             {
35.                 a[k][l] = a[k][l] + x[i][l] * x[i][k];
36.                 if (k != l)
37.                 {
38.                     a[l][k] = a[k][l];
39.                 }
40.             }
41.         }
42.         g[k] = 0;
43.         for (int i = 0; i < nrow; i++)
44.         {
45.             g[k] = g[k] + y[i] * x[i][k];
46.         }
47.     }
48. }
49.
50. int main() {
51.     int nrow = 9;
52.     int ncol = 3;
53.     ary2 x;
54.     ary y;
55.     arys g;
56.     ary2s a;
57. }
```

```
58.     get_data(x, y, nrow, ncol);
59.     square(x, y, a, g, nrow, ncol);
60.
61.     return 0;
62. }
```

ПРИЛОЖЕНИЕ Б

Исходный код на С с контрольными точками

```
1. #include <stdio.h>
2. #include "sampler.h"
3.
4.
5. #define rmax 9
6. #define cmax 3
7.
8. typedef double ary[rmax];
9. typedef double arys[cmax];
10. typedef double ary2[rmax][cmax];
11. typedef double ary2s[cmax][cmax];
12.
13.
14. void get_data(ary2 x, ary y, int nrow, int ncol)
15. {
16.     SAMPLE;
17.     for (int i = 0; i < nrow; i++)
18.     {
19.         SAMPLE;
20.         x[i][0] = 1;
21.         SAMPLE;
22.         for (int j = 1; j < ncol; j++)
23.         {
24.             SAMPLE;
25.             x[i][j] = (i + 1) * x[i][j - 1];
26.             SAMPLE;
27.         }
28.         SAMPLE;
29.         y[i] = 2 * (i + 1);
30.         SAMPLE;
31.     }
32.     SAMPLE;
33. }
34.
35. void square(ary2 x, double* y, ary2s a, double* g, int nrow, int ncol)
36. {
37.     SAMPLE;
38.     for (int k = 0; k < ncol; k++)
39.     {
40.         SAMPLE;
41.         for (int l = 0; l <= k; l++)
42.         {
43.             SAMPLE;
44.             a[k][l] = 0;
45.             SAMPLE;
46.             for (int i = 0; i < nrow; i++)
47.             {
48.                 SAMPLE;
49.                 a[k][l] = a[k][l] + x[i][l] * x[i][k];
50.                 SAMPLE;
51.                 if (k != l)
52.                 {
53.                     SAMPLE;
54.                     a[l][k] = a[k][l];
55.                     SAMPLE;
56.                 }
57.                 SAMPLE;
58.             }
59.             SAMPLE;
60.         }
```

```

61.         SAMPLE;
62.         g[k] = 0;
63.         SAMPLE;
64.         for (int i = 0; i < nrow; i++)
65.         {
66.             SAMPLE;
67.             g[k] = g[k] + y[i] * x[i][k];
68.             SAMPLE;
69.         }
70.         SAMPLE;
71.     }
72.     SAMPLE;
73. }
74.
75. int main() {
76.     int nrow = 9;
77.     int ncol = 3;
78.     ary2 x;
79.     ary y;
80.     arys g;
81.     ary2s a;
82.
83.     SAMPLE;
84.     get_data(x, y, nrow, ncol);
85.     SAMPLE;
86.     square(x, y, a, g, nrow, ncol);
87.     SAMPLE;
88.
89.     return 0;
90. }

```

ПРИЛОЖЕНИЕ В

XML граф

```
<model type = "Objects::AMC::Model" name = "KOVINEV_LAB4">
<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>
<node type = "Objects::AMC::Top" name = "t7"></node>
<node type = "Objects::AMC::Top" name = "t8"></node>
<node type = "Objects::AMC::Top" name = "t9"></node>
<node type = "Objects::AMC::Top" name = "t10"></node>
<node type = "Objects::AMC::Top" name = "t11"></node>
<node type = "Objects::AMC::Top" name = "t12"></node>
<node type = "Objects::AMC::Top" name = "t13"></node>
<node type = "Objects::AMC::Top" name = "t14"></node>
<node type = "Objects::AMC::Top" name = "t15"></node>
<node type = "Objects::AMC::Top" name = "t16"></node>
<node type = "Objects::AMC::Top" name = "t17"></node>
<node type = "Objects::AMC::Top" name = "t18"></node>
<node type = "Objects::AMC::Top" name = "t19"></node>
<node type = "Objects::AMC::Top" name = "t20"></node>
<node type = "Objects::AMC::Top" name = "t21"></node>
<node type = "Objects::AMC::Top" name = "t22"></node>
<link type = "Objects::AMC::Link" name = "0-1" probability = "1" intensity = "4.19"
deviation = "0.0" source = "t0" dest = "t1"></link>
<link type = "Objects::AMC::Link" name = "1-2" probability = "0.80" intensity = "0"
deviation = "0.0" source = "t1" dest = "t2"></link>
<link type = "Objects::AMC::Link" name = "1-6" probability = "0.20" intensity = "0.84"
deviation = "0.0" source = "t1" dest = "t6"></link>
<link type = "Objects::AMC::Link" name = "2-3" probability = "1" intensity = "0"
deviation = "0.0" source = "t2" dest = "t3"></link>
<link type = "Objects::AMC::Link" name = "3-4" probability = "0.67" intensity = "2.51"
deviation = "0.0" source = "t3" dest = "t4"></link>
<link type = "Objects::AMC::Link" name = "3-5" probability = "0.33" intensity = "0"
deviation = "0.0" source = "t3" dest = "t5"></link>
<link type = "Objects::AMC::Link" name = "4-3" probability = "1" intensity = "16.68"
deviation = "0.0" source = "t4" dest = "t3"></link>
<link type = "Objects::AMC::Link" name = "5-1" probability = "1" intensity = "15.09"
deviation = "0.0" source = "t5" dest = "t1"></link>
<link type = "Objects::AMC::Link" name = "6-7" probability = "1" intensity = "2.51"
deviation = "0.0" source = "t6" dest = "t7"></link>
<link type = "Objects::AMC::Link" name = "7-8" probability = "1" intensity = "0"
deviation = "0.0" source = "t7" dest = "t8"></link>
<link type = "Objects::AMC::Link" name = "8-9" probability = "0.75" intensity = "0"
deviation = "0.0" source = "t8" dest = "t9"></link>
<link type = "Objects::AMC::Link" name = "8-21" probability = "0.25" intensity = "0"
deviation = "0.0" source = "t8" dest = "t21"></link>
<link type = "Objects::AMC::Link" name = "9-10" probability = "0.67" intensity = "1.68"
deviation = "0.0" source = "t9" dest = "t10"></link>
<link type = "Objects::AMC::Link" name = "9-17" probability = "0.33" intensity = "0.61"
deviation = "0.0" source = "t9" dest = "t17"></link>
<link type = "Objects::AMC::Link" name = "10-11" probability = "1" intensity = "13.13"
deviation = "0.0" source = "t10" dest = "t11"></link>
<link type = "Objects::AMC::Link" name = "11-12" probability = "0.8" intensity = "2.82"
deviation = "0.0" source = "t11" dest = "t12"></link>
<link type = "Objects::AMC::Link" name = "11-16" probability = "0.2" intensity = "0"
deviation = "0.0" source = "t11" dest = "t16"></link>
<link type = "Objects::AMC::Link" name = "12-13" probability = "1" intensity = "17.96"
deviation = "0.0" source = "t12" dest = "t13"></link>
```

```

<link type = "Objects::AMC::Link" name = "13-14" probability = "0.5" intensity = "0.61"
deviation = "0.0" source = "t13" dest = "t14"></link>
<link type = "Objects::AMC::Link" name = "13-15" probability = "0.5" intensity = "0.45"
deviation = "0.0" source = "t13" dest = "t15"></link>
<link type = "Objects::AMC::Link" name = "14-15" probability = "1" intensity = "13.7"
deviation = "0.0" source = "t14" dest = "t15"></link>
<link type = "Objects::AMC::Link" name = "15-11" probability = "1" intensity = "1.50"
deviation = "0.0" source = "t15" dest = "t11"></link>
<link type = "Objects::AMC::Link" name = "16-9" probability = "1" intensity = "1.12"
deviation = "0.0" source = "t16" dest = "t9"></link>
<link type = "Objects::AMC::Link" name = "17-18" probability = "1" intensity = "20.39"
deviation = "0.0" source = "t17" dest = "t18"></link>
<link type = "Objects::AMC::Link" name = "18-19" probability = "0.9" intensity = "0"
deviation = "0.0" source = "t18" dest = "t19"></link>
<link type = "Objects::AMC::Link" name = "18-20" probability = "0.1" intensity = "0"
deviation = "0.0" source = "t18" dest = "t20"></link>
<link type = "Objects::AMC::Link" name = "19-18" probability = "1" intensity = "16.65"
deviation = "0.0" source = "t19" dest = "t18"></link>
<link type = "Objects::AMC::Link" name = "20-8" probability = "1" intensity = "1.68"
deviation = "0.0" source = "t20" dest = "t8"></link>
<link type = "Objects::AMC::Link" name = "21-22" probability = "1" intensity = "5.03"
deviation = "0.0" source = "t21" dest = "t22"></link>
</model>

```