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»

Студентка гр. 6304 _____ Иванкова В.М.

Преподаватель _____ Кирьянчиков В.А.

Санкт-Петербург

2020

1. Цель работы

Построение операционной графовой модели программы (ОГМП) и расчет характеристик эффективности ее выполнения методом эквивалентных преобразований.

2. Задание

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] - 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.

1.2. Расчет характеристик эффективности выполнения программы методом эквивалентных преобразований.

Полученную в части 2.1 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.

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

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. Сравнить полученные результаты с результатами измерений, полученными в работе 3.

Построение операционной графовой модели программы

3. Текст программы (исходный)

```
1  #include <stdio.h>
2  #include <time.h>
3  #include <stdlib.h>
4  #include "Sampler.h"
5
6  void swap (float *x, float *y)
7  {
8         float temp;
9         temp = *x;
10        *x = *y;
11        *y = temp;
12 }
13
14 void shellsort(float arr[], int num)
15 {
16     int i, j, k;
17     for (i = num / 2; i > 0; i = i / 2)
```

```
18      {
19          for (j = i; j < num; j++)
20          {
21              for(k = j - i; k >= 0; k = k - i)
22              {
23                  if (arr[k+i] >= arr[k])
24                      break;
25                  else
26                  {
27                      swap(&arr[k], &arr[k+i]);
28                  }
29              }
30          }
31      }
32  }
33
34  void write_arr(float arr[], int num)
35  {
36        int i;
37        for (i = 0; i < num; i++)
38        {
39          printf("%f ", arr[i]);
40      }
41  }
42  int main()
43  {
44        int num = 80;
45        float my_max = 100.0;
46      float arr[80];
47      int k;
48
49      for (k = 0 ; k < num; k++)
50      {
51          arr[k] = (float)rand()/(float)(RAND_MAX/my_max);
52      }
53      shellsort(arr, num);
54      write_arr(arr, num);
55      return 0;
56  }
```

4. Граф управления программы

Граф управления строился для каждого из функциональных участков отдельно, т.к. подсчёт затрат времени в предыдущей л/р выполнялся на них отдельно. Граф представлен на рис.1.

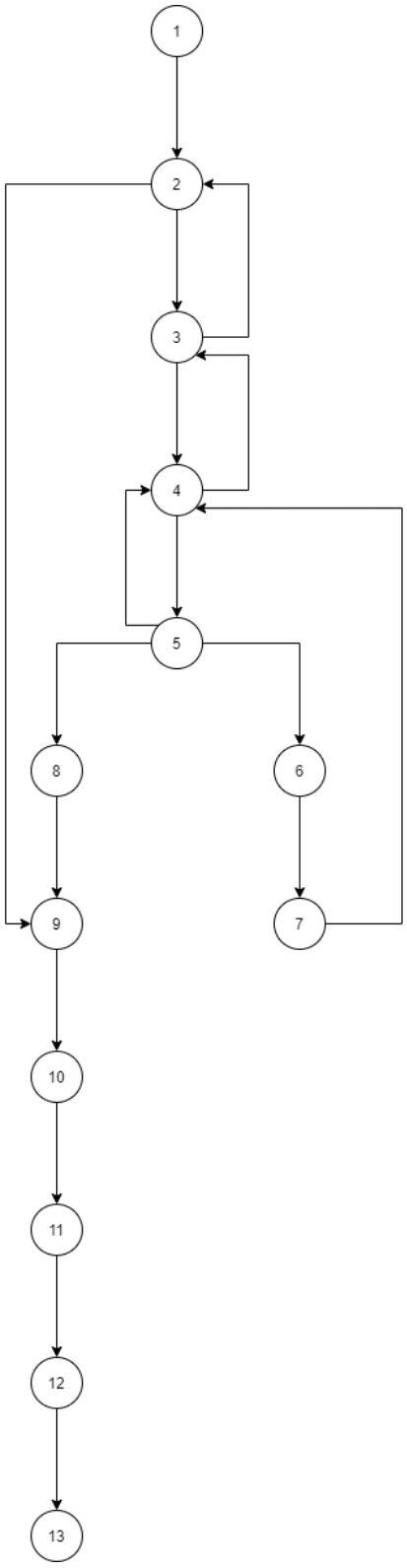


Рисунок 1 - Граф управления

5. Профилирование

Текст программы (подготовленный для профилирования)

```
1  #include <stdio.h>
2  #include <time.h>
3  #include <stdlib.h>
4  #include "Sampler.h"
5
6  void swap (float *x, float *y)
7  {
8         float temp;
9         temp = *x;
10        *x = *y;
11        *y = temp;
12 }
13
14 void shellsort(float arr[], int num)
15 {
16     int i, j, k;
17        SAMPLE;
18     for (i = num / 2; i > 0; i = i / 2)
19     {
20        SAMPLE;
21          for (j = i; j < num; j++)
22          {
23               SAMPLE;
24              for(k = j - i; k >= 0; k = k - i)
25              {
26                      SAMPLE;
27                  if (arr[k+i] >= arr[k])
28                            {
29                            SAMPLE;
30                      break;
31                            }
32                  else
33                  {
34                            SAMPLE;
35                      swap(&arr[k], &arr[k+i]);
36                            SAMPLE;
37                  }
38                            SAMPLE;
39              }
40                      SAMPLE;
41          }
42               SAMPLE;
43     }
44        SAMPLE;
45 }
46
47 void write_arr(float arr[], int num)
48 {
49        int i;
50        SAMPLE;
51        for (i = 0; i < num; i++)
52        {
53          printf("%f ", arr[i]);
54     }
55        SAMPLE;
56 }
57 int main()
```

```
58 {
59       int num = 80;
60       float my_max = 100.0;
61     float arr[80];
62     int k;
63
64     for (k = 0 ; k < num; k++)
65     {
66         arr[k] = (float)rand()/(float)(RAND_MAX/my_max);
67     }
68        SAMPLE;
69     shellsort(arr, num);
70        SAMPLE;
71     write_arr(arr, num);
72        SAMPLE;
73     return 0;
74 }
```

Результаты профилирован

NN	Имя обработанного файла
1.	..\..\LAB4\LAB4.CPP

Таблица с результатами измерений (используется 16 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 17	1 : 20	3.35	1	3.35
1 : 20	1 : 23	11.73	6	1.96
1 : 23	1 : 26	914.36	402	2.27
1 : 26	1 : 29	2663.47	356	7.48
1 : 26	1 : 34	1821.18	251	7.26
1 : 29	1 : 40	96.38	356	0.27
1 : 34	1 : 36	3579.51	251	14.26
1 : 36	1 : 38	0.00	251	0.00
1 : 38	1 : 40	72.08	46	1.57
1 : 38	1 : 26	403.96	205	1.97
1 : 40	1 : 23	635.28	396	1.60
1 : 40	1 : 42	9.22	6	1.54
1 : 42	1 : 20	14.25	5	2.85
1 : 42	1 : 44	2.51	1	2.51
1 : 44	1 : 70	2.51	1	2.51
1 : 50	1 : 55	29600.73	1	29600.73
1 : 55	1 : 72	0.84	1	0.84
1 : 68	1 : 17	2.51	1	2.51
1 : 70	1 : 50	2.51	1	2.51

6. Расчет вероятностей и затрат ресурсов для дуг управляющего графа

	Номера строк	Количество проходов
L1 = 2.51 мкс	68:17	1
L2 = 17.6 мкс	17:20, 42:20	1:5
L3 = 647.01 мкс	20:23, 40:23	6:396
L4 = 1318.32 мкс	23:26, 38:26	402:205
L5 = 8064.16 мкс	26:29, 26:34, 34:36	356:251:251
L6 = 0.00 мкс	36:38	251
L7 = 72.08 мкс	38:40	46
L8 = 11.73 мкс	40:42, 42:44	6:1
L9 = 2.51 мкс	44:70	1
L10 = 29600.73 мкс	70:50	1
L11 = 2.51 мкс	50:55	1
L12 = 0.84 мкс	55:72	1

7. Операционная графовая модель программ

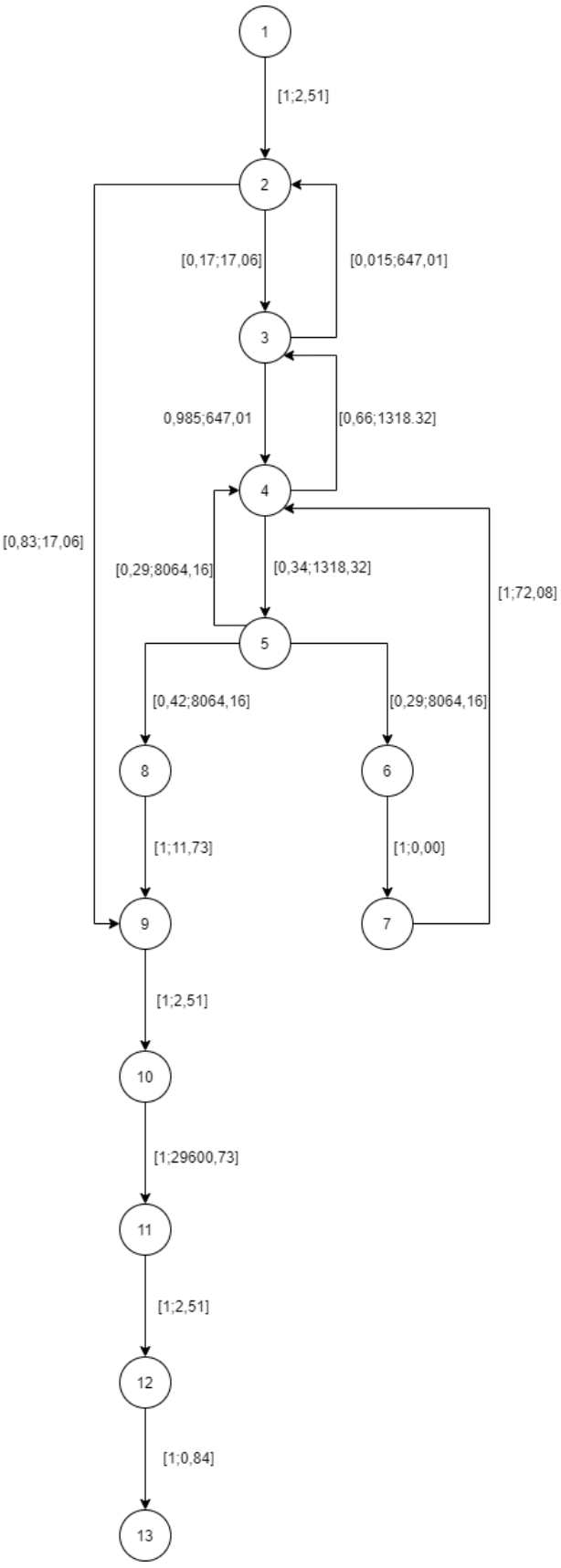
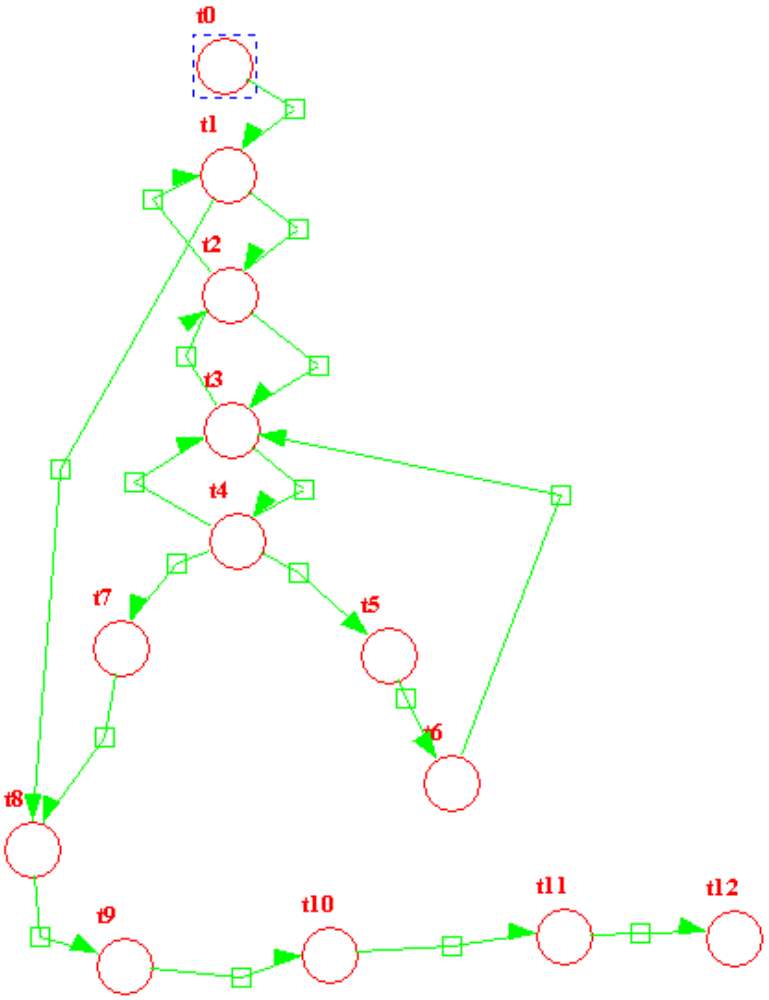


Рисунок 2 – Графовая модель

Расчет характеристик эффективности выполнения программы с помощью пакета CSA III методом эквивалентных преобразований

ГНД

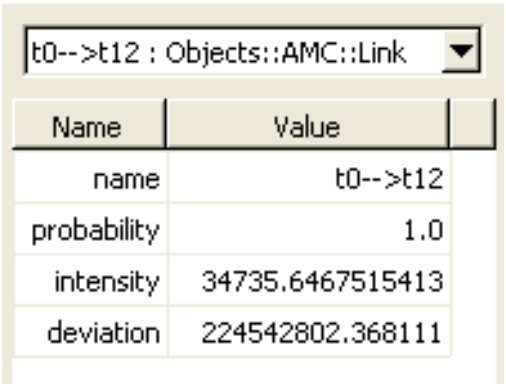


8. Описание модели model.xml

```
<model type = "Objects::AMC::Model" name = "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>
        <link type = "Objects::AMC::Link" name = "t0-->t1" probability = "1.0" intensity = "2.51"
deviation = "0.0" source = "t0" dest = "t1"></link>
```

<link type = "Objects::AMC::Link" name = "t1-->t2" probability = "0.17" intensity = "17.6" deviation = "0.0" source = "t1" dest = "t2"></link>
<link type = "Objects::AMC::Link" name = "t1-->t8" probability = "0.83" intensity = "17.6" deviation = "0.0" source = "t1" dest = "t8"></link>
<link type = "Objects::AMC::Link" name = "t2-->t1" probability = "0.015" intensity = "647.01" deviation = "0.0" source = "t2" dest = "t1"></link>
<link type = "Objects::AMC::Link" name = "t2-->t3" probability = "0.985" intensity = "647.01" deviation = "0.0" source = "t2" dest = "t3"></link>
<link type = "Objects::AMC::Link" name = "t3-->t2" probability = "0.66" intensity = "1318.32" deviation = "0.0" source = "t3" dest = "t2"></link>
<link type = "Objects::AMC::Link" name = "t3-->t4" probability = "0.34" intensity = "1318.32" deviation = "0.0" source = "t3" dest = "t4"></link>
<link type = "Objects::AMC::Link" name = "t4-->t7" probability = "0.42" intensity = "8064.16" deviation = "0.0" source = "t4" dest = "t7"></link>
<link type = "Objects::AMC::Link" name = "t4-->t5" probability = "0.29" intensity = "8064.16" deviation = "0.0" source = "t4" dest = "t5"></link>
<link type = "Objects::AMC::Link" name = "t4-->t3" probability = "0.29" intensity = "8064.16" deviation = "0.0" source = "t4" dest = "t3"></link>
<link type = "Objects::AMC::Link" name = "t5-->t6" probability = "1" intensity = "0.00" deviation = "0.0" source = "t5" dest = "t6"></link>
<link type = "Objects::AMC::Link" name = "t6-->t3" probability = "1" intensity = "72.08" deviation = "0.0" source = "t6" dest = "t3"></link>
<link type = "Objects::AMC::Link" name = "t7-->t8" probability = "1" intensity = "11.73" deviation = "0.0" source = "t7" dest = "t8"></link>
<link type = "Objects::AMC::Link" name = "t8-->t9" probability = "1" intensity = "2.51" deviation = "0.0" source = "t8" dest = "t9"></link>
<link type = "Objects::AMC::Link" name = "t9-->t10" probability = "1" intensity = "29600.73" deviation = "0.0" source = "t9" dest = "t10"></link>
<link type = "Objects::AMC::Link" name = "t10-->t11" probability = "1" intensity = "2.51" deviation = "0.0" source = "t10" dest = "t11"></link>
<link type = "Objects::AMC::Link" name = "t11-->t12" probability = "1" intensity = "0.84" deviation = "0.0" source = "t11" dest = "t12"></link>
</model>

9. Результаты



Согласно расчётам программы, среднее время выполнения процедуры из предыдущей лабораторной работы составляет 34735.65 мкс. В начале данного отчета приведен результат профилирования программы с использованием Sampler’a, где суммарное время выполнения составило 39836.38 мкс. В результате чего разница между результатами составляет менее 10%.

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 как для всей программы, так и для заданного фрагмента. Результаты сравнения этих характеристик с полученными в работе 3 согласуются.