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 (ОГПМ) и расчет характеристик эффективности ее выполнения методом эквивалентных преобразований

Студентка гр. 7304 _____ Каляева А.В.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург
2021

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 {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 (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.

Ход работы:

1. **Построение операционной графовой модели**

   Исходный код программы представлен в приложении А.

2. **Граф управления программы**

   Граф управления строился непосредственно для функции сортировки.

   Граф управления представлен на рисунке 1.

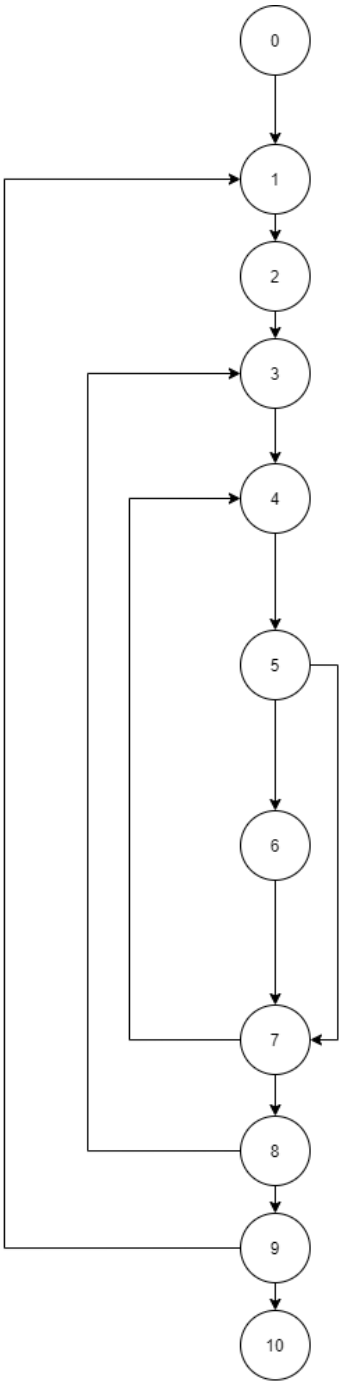

Рис. 1 – Граф управления

3. Профилирование

Исходный текст программы, подготовленный для профилирования представлен в приложении Б.

Результаты профилирования:

```
    Список обработанных файлов.

-------------------------------------------------------------------------
  NN            Имя обработанного файла
-------------------------------------------------------------------------
  1.   SHELL.CPP
-------------------------------------------------------------------------


  Таблица с результатами измерений ( используется 11 из 416 записей )

-------------------------------------------------------------------------
Исх.Поз. Прием.Поз.  Общее время(мкс)  Кол-во прох. Среднее время(мкс)
-------------------------------------------------------------------------
 1 :  11  1 :  13              1.68           1              1.68
-------------------------------------------------------------------------
 1 :  13  1 :  15              8.38           7              1.20
-------------------------------------------------------------------------
 1 :  15  1 :  18              2.51           7              0.36
-------------------------------------------------------------------------
 1 :  18  1 :  20             41.90          27              1.55
-------------------------------------------------------------------------
 1 :  20  1 :  30          12855.56        1849              6.95
 1 :  20  1 :  23           2370.98         311              7.62
-------------------------------------------------------------------------
 1 :  23  1 :  28           2862.10         311              9.20
-------------------------------------------------------------------------
 1 :  28  1 :  30              0.00         311              0.00
-------------------------------------------------------------------------
 1 :  30  1 :  20           2738.06        2133              1.28
 1 :  30  1 :  32             25.98          27              0.96
-------------------------------------------------------------------------
 1 :  32  1 :  18             27.66          20              1.38
 1 :  32  1 :  34              3.35           7              0.48
-------------------------------------------------------------------------
 1 :  34  1 :  13              5.87           6              0.98
 1 :  34  1 :  36              0.00           1              0.00
-------------------------------------------------------------------------
```

4. Расчет вероятностей и затрат ресурсов для дуг управляющего графа

	Номера точек	Количество проходов
L1= 2.66	11-13;34-13	1;6
L2= 1.20	13-15	7
L3= 1.74	15-18;32-18	7;20
L4= 2.83	18-20;30-20	27;2133
L5= 14.57	20-30;20-23	1849;311
L6= 9.20	23-28	311

L7=0.00	28-30	311
L8=0.48	32-34	7
L9=0.00	34-36	1

5. Операционная графовая модель программы

Операционная графовая модель представлена на рисунке 2.

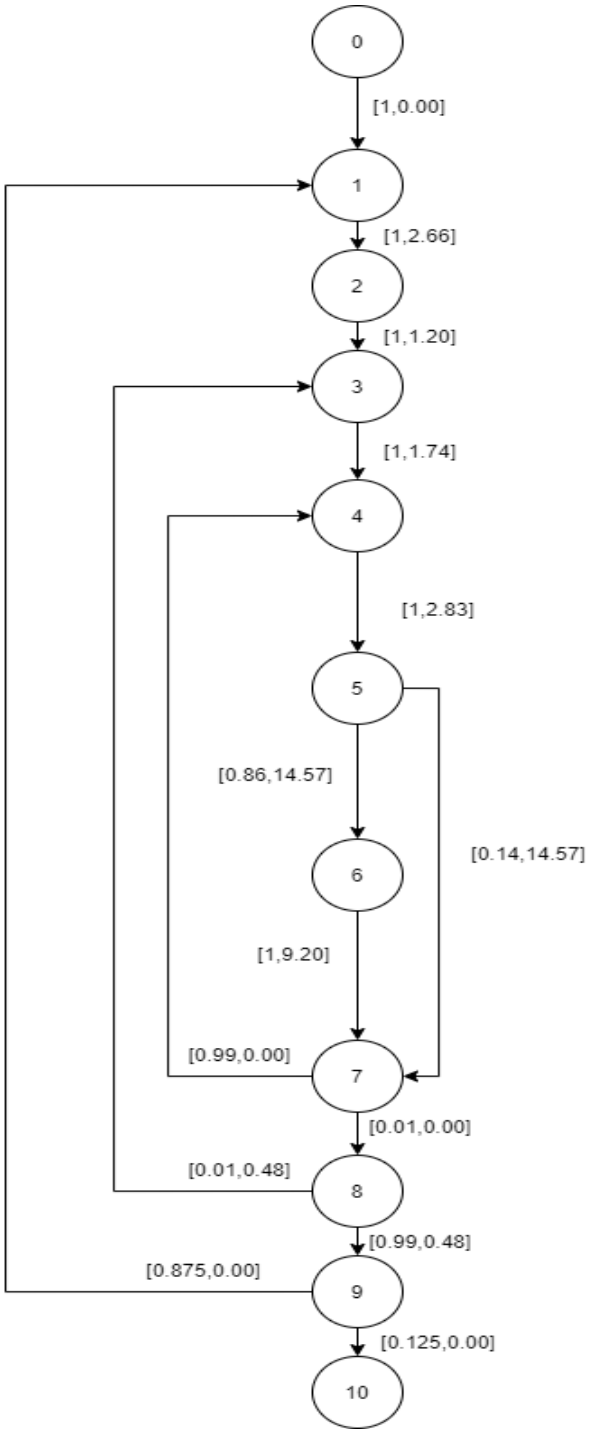

Рис. 2 – Операционная графовая модель

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

Граф с нагруженными дугами представлен на рисунке 3.

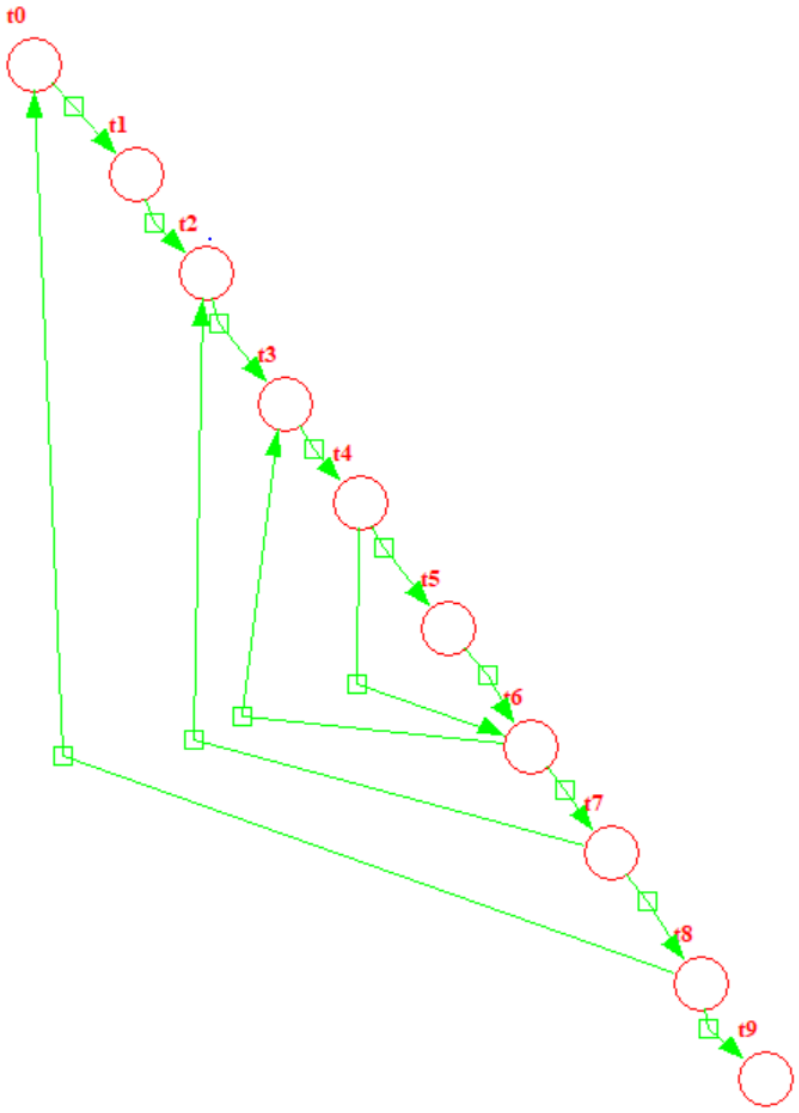

Рис.3 – Граф с нагруженными дугами

Описание модели:

```
<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>
    <link  type  =  "Objects::AMC::Link"  name  =  "t0-->t1"
probability = "1.0" intensity = "0.00" deviation = "0.0" source =
"t0" dest = "t1"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t1-->t2"
probability = "1.0" intensity = "2.66" deviation = "0.0" source =
"t1" dest = "t2"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t2-->t3"
probability = "1.0" intensity = "1.20" deviation = "0.0" source =
"t2" dest = "t3"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t3-->t4"
probability = "1.0" intensity = "1.74" deviation = "0.0" source =
"t3" dest = "t4"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t4-->t5"
probability = "1.0" intensity = "2.83" deviation = "0.0" source =
"t4" dest = "t5"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t5-->t6"
probability = "0.86" intensity = "14.57" deviation = "0.0" source
= "t5" dest = "t6"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t5-->t7"
probability = "0.14" intensity = "14.57" deviation = "0.0" source
= "t5" dest = "t7"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t6-->t7"
probability = "1.0" intensity = "9.20" deviation = "0.0" source =
"t6" dest = "t7"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t7-->t8"
probability = "0.01" intensity = "0.00" deviation = "0.0" source =
"t7" dest = "t8"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t7-->t4"
probability = "0.99" intensity = "0.00" deviation = "0.0" source =
"t7" dest = "t4"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t8-->t9"
probability = "1.0" intensity = "0.48" deviation = "0.0" source =
"t8" dest = "t9"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t8-->t3"
probability = "0.0" intensity = "0.48" deviation = "0.0" source =
"t8" dest = "t3"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t9-->t10"
probability = "0.125" intensity = "0.00" deviation = "0.0" source
= "t9" dest = "t10"></link>
    <link  type  =  "Objects::AMC::Link"  name  =  "t9-->t1"
probability = "0.875" intensity = "0.00" deviation = "0.0" source
= "t9" dest = "t1"></link>
</model>
```

Результат представлен на рисунке 4.

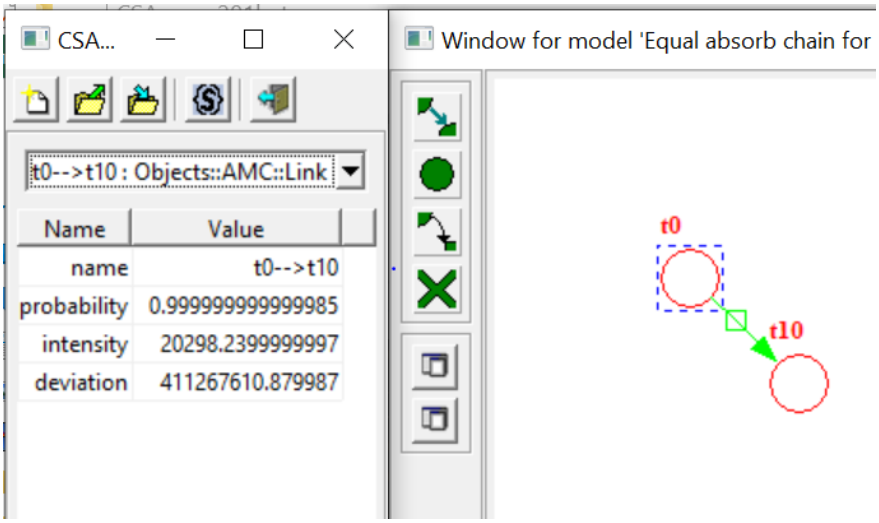

Рис.4 - Результат

Выводы:

При выполнении лабораторной работы была построена операционная графовая модель заданной программы, нагрузочные параметры которой были оценены с помощью профилировщика Samplerи методом эквивалентных преобразований с помощью пакета CSAIIIбыли вычислены математическое ожидание и дисперсия времени выполнения для всей программы. Результаты полученных характеристик с помощью пакета CSAIII оказались меньше на 1%, чем результаты из лабораторной работы 3.

ПРИЛОЖЕНИЕ А.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
1 # include <stdio.h>
2 # include <stdlib.h>
3 # include "Sampler.h"
4
5 void sort(float* a, int n) {
6    int i;
7    int j;
8    int flag;
9    int jump = n;
10    float hold=0;
11    while (jump > 0) {
12        jump = jump / 2;
13        do {
14            flag = 1;
15        for (j = 0; j < n; j++) {
16            i = j + jump;
17            if ((n>i) && (a[j] > a[i])) {
18                hold = a[i];
19                a[i] = a[j];
20                a[j] = hold;
21                flag = 0;
22            }
23        }
24        } while(!flag);
25    }
26 }
27
28 int main()
29 {
30    int n = 80;
31    float x[80];
32    int i;
33
34    for (i = 0; i < n; ++i) {
35        x[i] = (float) (rand() % 100);
36    }
37
38    sort(x, n);
39
40    return 0;
41}
```

ПРИЛОЖЕНИЕ Б.

ТЕКСТ ПРОГРРАММЫ ДЛЯ ПРОФИЛИРОВАНИЯ

```
1 # include <stdio.h>
2 # include <stdlib.h>
3 # include "Sampler.h"
4
5 void sort(float* a, int n) {
6    int i;
7    int j;
8    int flag;
9    int jump = n;
10    float hold=0;
11    SAMPLE;
12    while (jump > 0) {
13        SAMPLE;
14        jump = jump / 2;
15        SAMPLE;
16        do {
17            flag = 1;
18        SAMPLE;
19        for (j = 0; j < n; j++) {
20            SAMPLE;
21            i = j + jump;
22            if ((n>i) && (a[j] > a[i])) {
23            SAMPLE;
24                hold = a[i];
25                a[i] = a[j];
26                a[j] = hold;
27                flag = 0;
28            SAMPLE;
29            }
30            SAMPLE;
31        }
32        SAMPLE;
33        } while(!flag);
34        SAMPLE;
35    }
36    SAMPLE;
37 }
38
39 int main()
40 {
41    int n = 80;
42    float x[80];
43    int i;
44
45    for (i = 0; i < n; ++i) {
46        x[i] = (float) (rand() % 100);
47    }
48
49    sort(x, n);
50
51    return 0;
52}
```