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. 7304 _____ Нгуен Т.Т.З.

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

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

Постановка задачи.

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 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.

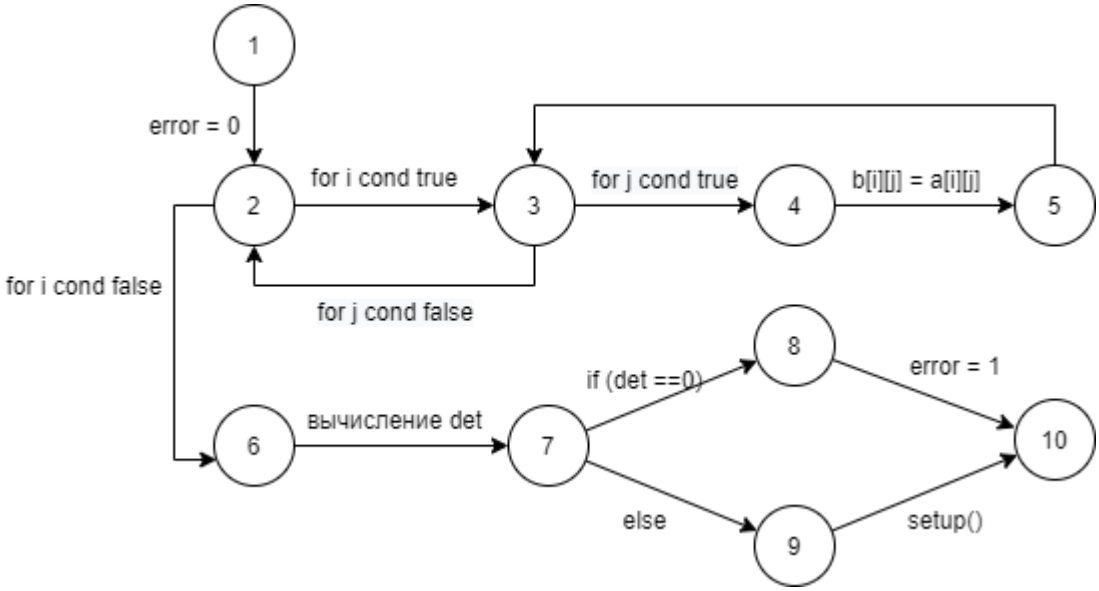
Получить описание полученной ОГМП на входном языке пакета CSA III в виде поглощающей марковской цепи (ПМЦ) – (англ.) AMC (absorbing Markov chain) и/или эргодической марковской цепи (ЭМЦ) - EMC (ergodic Markov chain).

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.

Ход выполнения.

1. **Граф управления программы**

Был построен операционный граф модели программы для программы из лабораторных работах №1-3 - граф управления для функции решение СЛАУ solve. Текст программы представлен в Приложении А.



Рисунка 1 – Граф управления

2. **Профилирование**

Для профилирования была подготовлена программа с использованием SAMPLER из лабораторной работы №3. Текст программы (подготовленный для профилирования) представлен в Приложении Б.

Результаты профилирования:

Таблица с результатами измерений (используется 11 из 416 записей)

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 34	1 : 39	0.84	1	0.84
1 : 39	1 : 41	2.51	1	2.51
1 : 41	1 : 43	5.03	3	1.68
1 : 43	1 : 45	135.77	9	15.09
1 : 45	1 : 43	11.73	6	1.96
1 : 45	1 : 47	6.70	3	2.23

```
 1 :  47  1 :  41              5.03            2              2.51
 1 :  47  1 :  50              2.51            1              2.51
-------------------------------------------------------------------
 1 :  50  1 :  54            382.17            1            382.17
-------------------------------------------------------------------
 1 :  54  1 :  57             88.84            1             88.84
-------------------------------------------------------------------
 1 :  57  1 :  59              0.84            1              0.84
-------------------------------------------------------------------
 1 :  59  1 :  71              1.68            1              1.68
-------------------------------------------------------------------
```

Суммарное время выполнения программы равно 643.65 мкс

3. Расчет вероятностей и затрат ресурсов для дуг управляющего графа

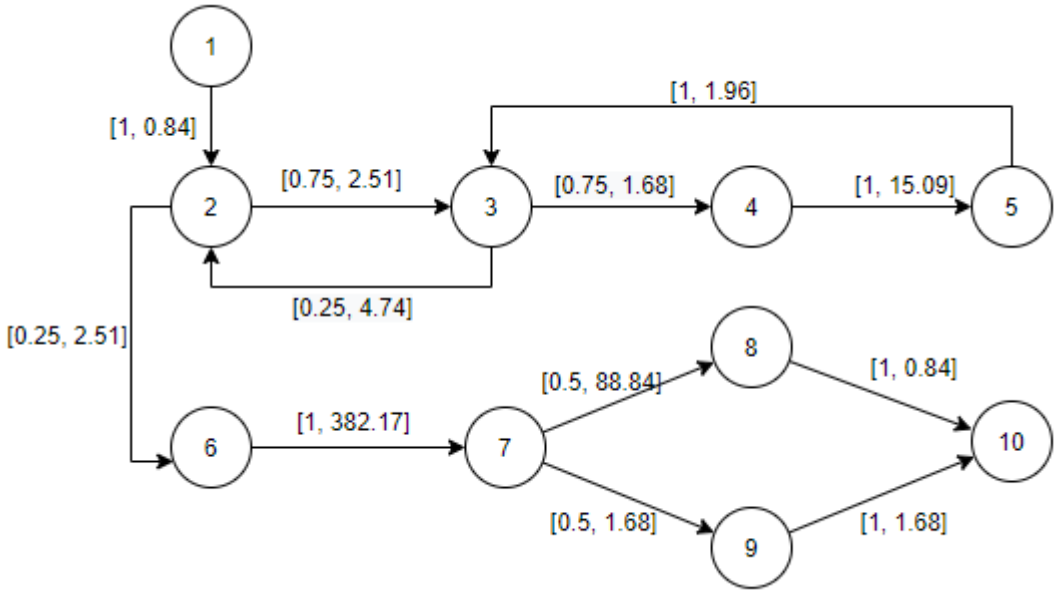
Был проведён расчёт вероятностей и затрат ресурсов для дуг управляющего графа.

Таблица 1: Расчёт вероятностей и затрат ресурсов

Дуга	Номера строк	Количество проходов	Расчет вероятности	Затраты ресурсов (Среднее время), мкс
L1 – L2	34 : 39	1	1	0.84
L2 – L3	39 : 41	1	3/(3+1) = 0.75	2.51
L3 – L4	41 : 43	3	9/(3+9)=0.755	1.68
L3 – L2	45 : 47, 47 : 41	3 2	1 – 0.75 = 0.25	4.74
L4 – L5	43 : 45	9	1	15.09
L5 – L3	45 : 43	6	1	1.96
L2 – L6	47 : 50	1	1 – 0.75 = 0.25	2.51
L6 – L7	50 : 54	1	1	382.17
L7 – L8	54 : 57	1	1/(1+1) = 0.5	88.84
L8 – L10	57 : 59	1	1	0.84
L7 – L9; L9 – L10	59 : 71	1	1 - 0.5 = 0.5	1.68

4. Операционная графовая модель программы

Операционная графовая модель представлена на рисунке 2.



Рисунка 2 – Операционная графовая модель

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

ГНД

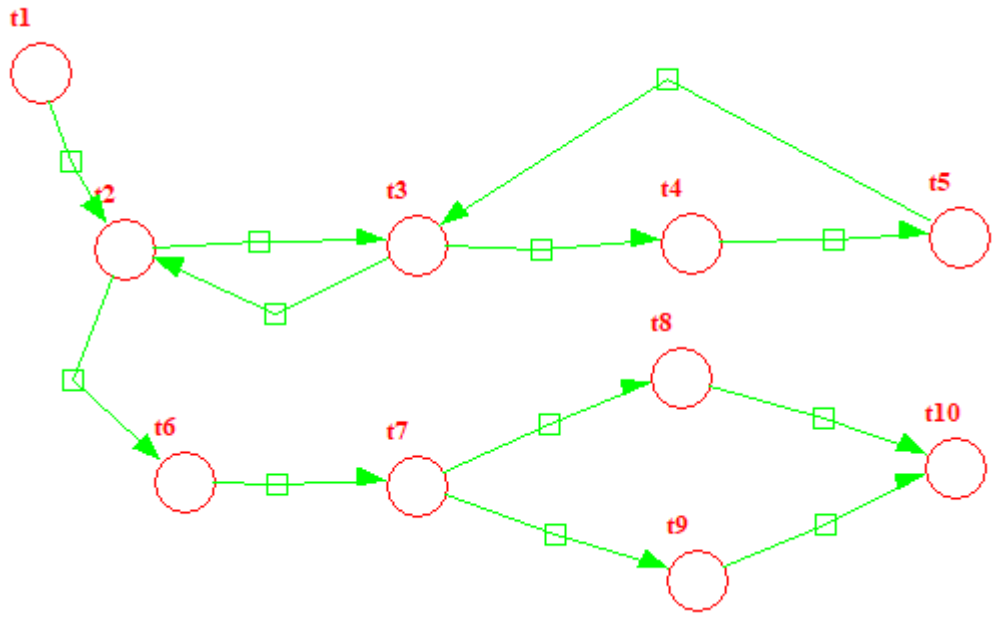


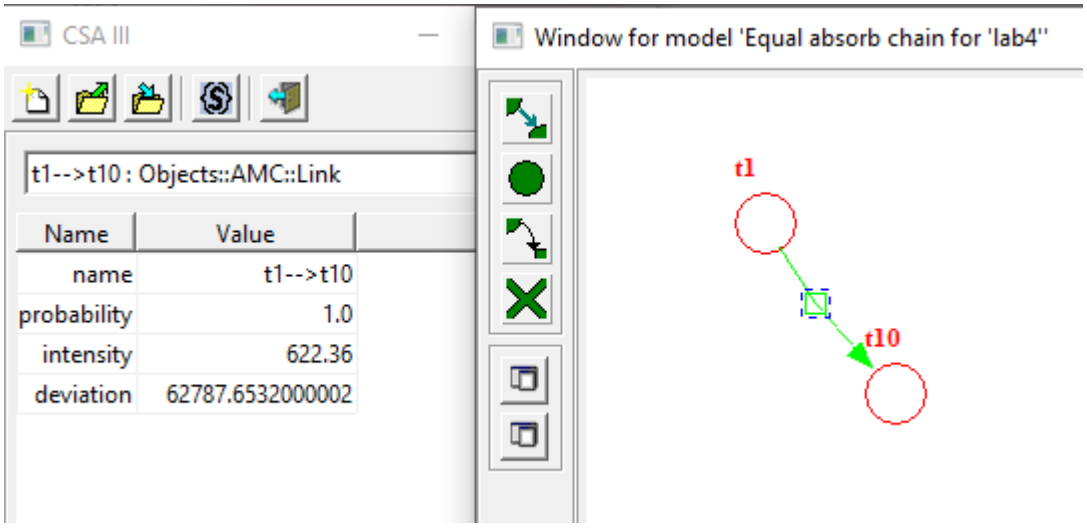
Рисунок 3: Модель программы в CSA III

6. Описаниие модели

Lab4.xml

```
<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>
     <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 = "t1-->t2" probability
= "1.0" intensity = "0.84" deviation = "0.0" source = "t1" dest =
"t2"></link>
     <link type = "Objects::AMC::Link" name = "t2-->t3" probability
= "0.75" intensity = "2.51" deviation = "0.0" source = "t2" dest =
"t3"></link>
     <link type = "Objects::AMC::Link" name = "t3-->t2" probability
= "0.25" intensity = "4.74" deviation = "0.0" source = "t3" dest =
"t2"></link>
     <link type = "Objects::AMC::Link" name = "t3-->t4" probability
= "0.75" intensity = "1.68" deviation = "0.0" source = "t3" dest =
"t4"></link>
     <link type = "Objects::AMC::Link" name = "t4-->t5" probability
= "1.0" intensity = "15.09" deviation = "0.0" source = "t4" dest =
"t5"></link>
     <link type = "Objects::AMC::Link" name = "t5-->t3" probability
= "1.0" intensity = "1.96" deviation = "0.0" source = "t5" dest =
"t3"></link>
     <link type = "Objects::AMC::Link" name = "t2-->t6" probability
= "0.25" intensity = "2.51" deviation = "0.0" source = "t2" dest =
"t6"></link>
     <link type = "Objects::AMC::Link" name = "t6-->t7" probability
= "1.0" intensity = "382.17" deviation = "0.0" source = "t6" dest =
"t7"></link>
     <link type = "Objects::AMC::Link" name = "t7-->t8" probability
= "0.5" intensity = "88.84" deviation = "0.0" source = "t7" dest =
"t8"></link>
     <link type = "Objects::AMC::Link" name = "t7-->t9" probability
= "0.5" intensity = "1.68" deviation = "0.0" source = "t7" dest =
"t9"></link>
     <link type = "Objects::AMC::Link" name = "t8-->t10" probability
= "1.0" intensity = "0.84" deviation = "0.0" source = "t8" dest =
"t10"></link>
     <link type = "Objects::AMC::Link" name = "t9-->t10" probability
= "1.0" intensity = "1.68" deviation = "0.0" source = "t9" dest =
"t10"></link>
</model>
```

Результат расчета среднего времени и дисперсии времени выполнения как для всей функции программы solve.



Вывод

В ходе выполнении лабораторной работы была построена операционная графовая модель заданной программы, нагрузочные параметры которой были оценены с помощью профилировщика Sampler и расчета характеристик эффективности выполнения программы методом эквивалентных преобразований с помощью пакета CSA III, были вычислены математического ожидании времени (662.36 мкс) и дисперсии времени выполнения (62787.6532 мкс) как для функции solve программы из лабораторной работы №3. Результаты сравнения этих характеристик с полученными в работе 3 примерно (отличается ~ 3%).

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include <STDIO.H>
#include <STDLIB.H>
#include "SAMPLER.H"

#define rmax 3
#define cmax 3
int n = rmax;

void get_data(float a[rmax][cmax], float y[cmax], int n)
{
      int i, j;
      for(i = 0; i < n; i++){
            for(j = 0; j < n; j++)
                  scanf("%f", &a[i][j]);
            scanf("%f", &y[i]);
      }
}

void setup(float b[rmax][cmax],float a[rmax][cmax], float y[cmax],
float coef[cmax], int j, float det)
{
      int i;
      for(i = 0; i < n; i++){
            b[i][j] = y[i];
            if(j > 0)
                  b[i][j - 1] = a[i][j - 1];
      }
      coef[j] = (b[0][0] * (b[1][1] * b[2][2] - b[2][1] * b[1][2])
            - b[0][1] * (b[1][0] * b[2][2] - b[2][0] * b[1][2])
            + b[0][2] * (b[1][0] * b[2][1] - b[2][0] * b[1][1]))/det;
}

void solve(float a[rmax][cmax], float y[cmax], float coef[cmax], int
n, int error)
{
      float b[rmax][cmax];
      int i,j;
      float det;
      error = 0;
      for(i = 0; i < n; i++)
            for(j = 0; j < n; j++)
                  b[i][j] = a[i][j];

      det = a[0][0] * (a[1][1] * a[2][2] - a[2][1] * a[1][2])
            - a[0][1] * (a[1][0] * a[2][2] - a[2][0] * a[1][2])
            + a[0][2] * (a[1][0] * a[2][1] - a[2][0] * a[1][1]);;

      if(det == 0)
            error = 1;
      else
            for(j = 0; j < n; j++)
                  setup(b, a, y, coef, j, det);
}
int main()
{
      float y[cmax], coef[cmax], a[rmax][cmax];
```

```
    int error;

    get_data(a, y, n);

    SAMPLE;
    solve(a, y, coef, n, error);
    SAMPLE;
   return 0;
}
```

ПРИЛОЖЕНИЕ Б. ТЕКСТ ПРОГРРАММЫ ДЛЯ ПРОФИЛИРОВАНИЯ

```
#include <STDIO.H>
#include <STDLIB.H>
#include "SAMPLER.H"

#define rmax 3
#define cmax 3
int n = rmax;

void get_data(float a[rmax][cmax], float y[cmax], int n)
{
      int i, j;
      for(i = 0; i < n; i++){
            for(j = 0; j < n; j++)
                  scanf("%f", &a[i][j]);
            scanf("%f", &y[i]);
      }
}

void setup(float b[rmax][cmax],float a[rmax][cmax], float y[cmax],
float coef[cmax], int j, float det)
{
      int i;
      for(i = 0; i < n; i++){
            b[i][j] = y[i];
            if(j > 0)
                  b[i][j - 1] = a[i][j - 1];
      }
      coef[j] = b[0][0] * (b[1][1] * b[2][2] - b[2][1] * b[1][2])
            - b[0][1] * (b[1][0] * b[2][2] - b[2][0] * b[1][2])
            + b[0][2] * (b[1][0] * b[2][1] - b[2][0] * b[1][1])/det;
}

void solve(float a[rmax][cmax], float y[cmax], float coef[cmax], int
n, int error)
{
      SAMPLE;
      float b[rmax][cmax];
      int i,j;
      float det;
      error = 0;
      SAMPLE;
      for(i = 0; i < n; i++){
      SAMPLE;
            for(j = 0; j < n; j++){
            SAMPLE;
                  b[i][j] = a[i][j];
            SAMPLE;
            }
      SAMPLE;
      }

      det = a[0][0] * (a[1][1] * a[2][2] - a[2][1] * a[1][2])
            - a[0][1] * (a[1][0] * a[2][2] - a[2][0] * a[1][2])
            + a[0][2] * (a[1][0] * a[2][1] - a[2][0] * a[1][1]);

      if(det == 0){
```

```
      SAMPLE;
            error = 1;
      SAMPLE;
      }

      else{
      SAMPLE;
            for(j = 0; j < n; j++){
            SAMPLE;
                  setup(b, a, y, coef, j, det);
            SAMPLE;
            }
      SAMPLE;
      }
      SAMPLE;
}
int main()
{
      float y[cmax], coef[cmax], a[rmax][cmax];
      int error;

      get_data(a, y, n);
      solve(a, y, coef, n, error);
     return 0;
}
```