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. 6304 _____ Тимофеев А.А.

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

2.1. 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.

2.2. 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.

3. Исходный текст программы

```c
#include <math.h>
#include <Sampler.h>

void linfit2(float *x, float *y, float *y_calc,
             float *a, float *b, int n){
    int i;
    float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
    float xi, yi, sxy, syy, sxx;
    float correl_coef, see, sigma_b, sigma_a;
    sum_x = 0.0;
    sum_y = 0.0;
    sum_xy = 0.0;
    sum_x2 = 0.0;
    sum_y2 = 0.0;
    for(i = 0; i < n; i++){
        xi = x[i];
        yi = y[i];
        sum_x = sum_x + xi;
        sum_y = sum_y + yi;
        sum_xy = sum_xy + xi * yi;
        sum_x2 = sum_x2 + xi * xi;
        sum_y2 = sum_y2 + yi * yi;
    }
    sxx = sum_x2 - sum_x * sum_x / n;
    sxy = sum_xy - sum_x * sum_y / n;
    syy = sum_y2 - sum_y * sum_y / n;
    *b = sxy / sxx;
    *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
    correl_coef = sxy / sqrt(sxx * syy);
    see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy)/(n - 2));
    sigma_b = see / sqrt(sxx);
    sigma_a = sigma_b * sqrt(sum_x2 / n);
    for(i = 0; i < n; i++){
        y_calc[i] = (*a) + (*b) * x[i];
    }
}

```

```
38  int main(){
39      float x[1000];
40      float y[1000];
41      float y_calc[1000];
42      float a, b;
43      int i;
44      for(i = 0; i < 1000; i++){
45          x[i] = i / 3.0;
46          y[i] = i * i / 3.0;
47      }
48      linfit2(x, y, y_calc, &a, &b, 1000);
49  }
50
```

Граф управления программы представлен на рисунке 1.

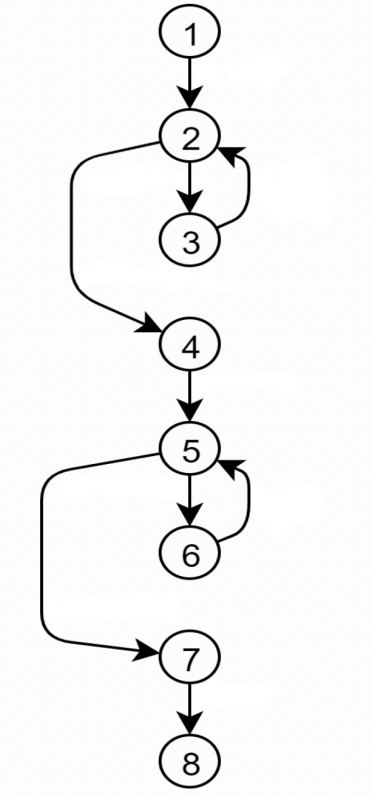


Рисунок 1 – Граф управления

4. Профилирование

Текст программы, подготовленный для профилирования

```c
#include <math.h>
#include <Sampler.h>

void linfit2(float *x, float *y, float *y_calc,
             float *a, float *b, int n){
    SAMPLE;
    int i;
    float sum_x, sum_y, sum_xy, sum_x2, sum_y2;
    float xi, yi, sxy, syy, sxx;
    float correl_coef, see, sigma_b, sigma_a;
    SAMPLE;
    sum_x = 0.0;
    sum_y = 0.0;
    sum_xy = 0.0;
    sum_x2 = 0.0;
    sum_y2 = 0.0;
    SAMPLE;
    for(i = 0; i < n; i++){
        SAMPLE;
        xi = x[i];
        yi = y[i];
        sum_x = sum_x + xi;
        sum_y = sum_y + yi;
        sum_xy = sum_xy + xi * yi;
        sum_x2 = sum_x2 + xi * xi;
        sum_y2 = sum_y2 + yi * yi;
        SAMPLE;
    }
    SAMPLE;
    sxx = sum_x2 - sum_x * sum_x / n;
    sxy = sum_xy - sum_x * sum_y / n;
    syy = sum_y2 - sum_y * sum_y / n;
    *b = sxy / sxx;
    *a = ((sum_x2 * sum_y - sum_x * sum_xy) / n) / sxx;
    correl_coef = sxy / sqrt(sxx * syy);
    see = sqrt((sum_y2 - (*a) * sum_y - (*b) * sum_xy)/(n - 2));
    sigma_b = see / sqrt(sxx);
    sigma_a = sigma_b * sqrt(sum_x2 / n);
    SAMPLE;
    for(i = 0; i < n; i++){
        SAMPLE;
        y_calc[i] = (*a) + (*b) * x[i];
        SAMPLE;
    }
    SAMPLE;
}

int main(){
    float x[1000];
    float y[1000];
    float y_calc[1000];
    float a, b;
    int i;
    for(i = 0; i < 1000; i++){
        x[i] = i / 3.0;
        y[i] = i * i / 3.0;
    }
    SAMPLE;
    linfit2(x, y, y_calc, &a, &b, 1000);
    SAMPLE;
}

```

Результаты профилирования:

Исх.Поз.	Прием.Поз.	Общее время(мкс)	Кол-во прох.	Среднее время(мкс)
1 : 6	1 : 11	0.00	1	0.00

1 : 11	1 : 17	141.64	1	141.64
1 : 17	1 : 19	0.00	1	0.00
1 : 19	1 : 27	15261.35	1000	15.26
1 : 27	1 : 19	256.68	999	0.27
1 : 27	1 : 29	0.00	1	0.00
1 : 29	1 : 39	977.22	1	977.22
1 : 39	1 : 41	0.00	1	0.00
1 : 41	1 : 43	7421.20	1000	7.42
1 : 43	1 : 41	7.54	999	0.01
1 : 43	1 : 45	0.00	1	0.00
1 : 45	1 : 60	0.84	1	0.84
1 : 58	1 : 6	4.19	1	4.19

5. Расчет вероятностей и затрат ресурсов для дуг управляющего графа

	Номера строк	Количество проходов
$L_1 = 145.83$ мкс	58:6, 6:11, 11:17	1, 1, 1
$L_2 = 0.27$ мкс	17:19, 27:19	1, 999
$L_3 = 15.26$ мкс	19:27	1000
$L_4 = 977.22$ мкс	27:29, 29:39	1, 1
$L_5 = 0.0075$ мкс	39:41, 43:41	1, 999
$L_6 = 7.42$ мкс	41:43	1000
$L_7 = 0.84$ мкс	43:45, 45:60	1, 1

6. Операционная графовая модель программы

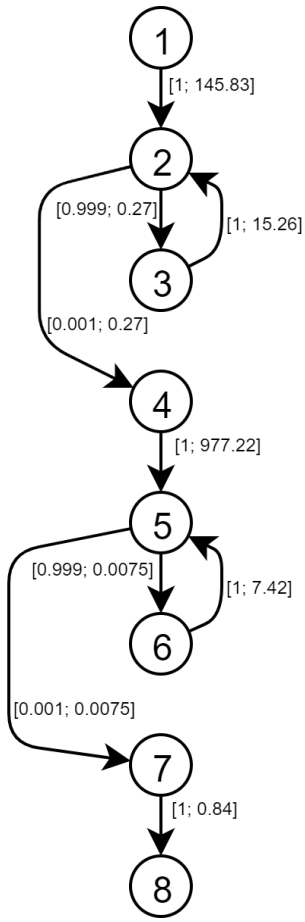


Рисунок 2 – Графовая модель

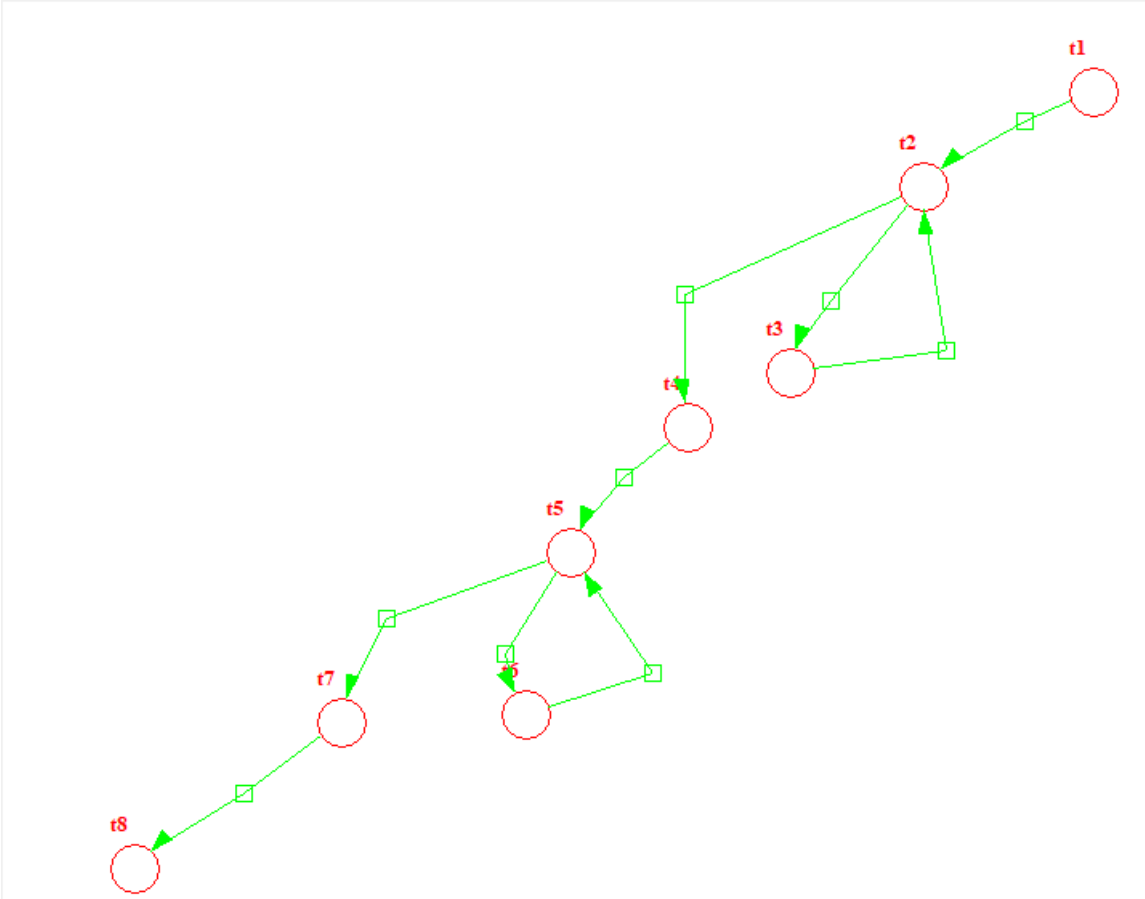
Расчет характеристик эффективности выполнения программы с помощью пакета CSA III методом эквивалентных преобразований

Описание модели

```
<model type = "Objects::AMC::Model" name = "lab5">
    <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>
    <link type = "Objects::AMC::Link" name = "t1-->t2" probability = "1.0"
intensity = "145.83" deviation = "0.0" source = "t1" dest = "t2"></link>
    <link type = "Objects::AMC::Link" name = "t2-->t3" probability = "0.999"
intensity = "0.27" deviation = "0.0" source = "t2" dest = "t3"></link>
    <link type = "Objects::AMC::Link" name = "t2-->t4" probability = "0.001"
intensity = "0.27" deviation = "0.0" source = "t2" dest = "t4"></link>
    <link type = "Objects::AMC::Link" name = "t3-->t2" probability = "1.0"
intensity = "15.26" deviation = "0.0" source = "t3" dest = "t2"></link>
```

```
    <link type = "Objects::AMC::Link" name = "t4-->t5" probability = "1.0"
intensity = "977.22" deviation = "0.0" source = "t4" dest = "t5"></link>
    <link type = "Objects::AMC::Link" name = "t5-->t6" probability = "0.999"
intensity = "0.0075" deviation = "0.0" source = "t5" dest = "t6"></link>
    <link type = "Objects::AMC::Link" name = "t5-->t7" probability = "0.001"
intensity = "0.0075" deviation = "0.0" source = "t5" dest = "t7"></link>
    <link type = "Objects::AMC::Link" name = "t6-->t5" probability = "1.0"
intensity = "7.42" deviation = "0.0" source = "t6" dest = "t5"></link>
    <link type = "Objects::AMC::Link" name = "t7-->t8" probability = "1.0"
intensity = "0.84" deviation = "0.0" source = "t7" dest = "t8"></link>
</model>
```

XML – файл был открыт в программе CSA



t1-->t8 : Objects::AMC::Link

Name	Value
name	t1-->t8
probability	0.999999999999998
intensity	24058.71
deviation	296052307.593749

Согласно расчётам программы, среднее время выполнения процедуры из предыдущей лабораторной работы составляет 24058.71 мкс. При этом по результатам профилирования из предыдущей лабораторной работы, время работы процедуры составляет 24111.20 мкс. Из близости полученных значений следует, что операционная графовая модель программы была построена правильно.

Вывод:

В ходе выполнения лабораторной работы было произведено построение операционной графовой модели программы, был выполнен расчёт характеристик эффективности её выполнения с помощью пакета CSA III.