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. 7304 Шарапенков И.И.

Преподаватель Ефремов М.А.

Санкт-Петербург
2021

Цель работы.

Изучение возможности построения операционной графовой модели программы (ОГМП) и расчета характеристик эффективности ее выполнения методом эквивалентных преобразований.

Постановка задачи.

1. Построение ОГМП.

Для рассмотренного 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.

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.

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).

Ход выполнения.

1. Для выполнения данной лабораторной работы и построения операционной графовой модели программы использовалась программа из третьей лабораторной работы, представленное в Приложении А. Граф управления программы представлен на Рисунке 1:

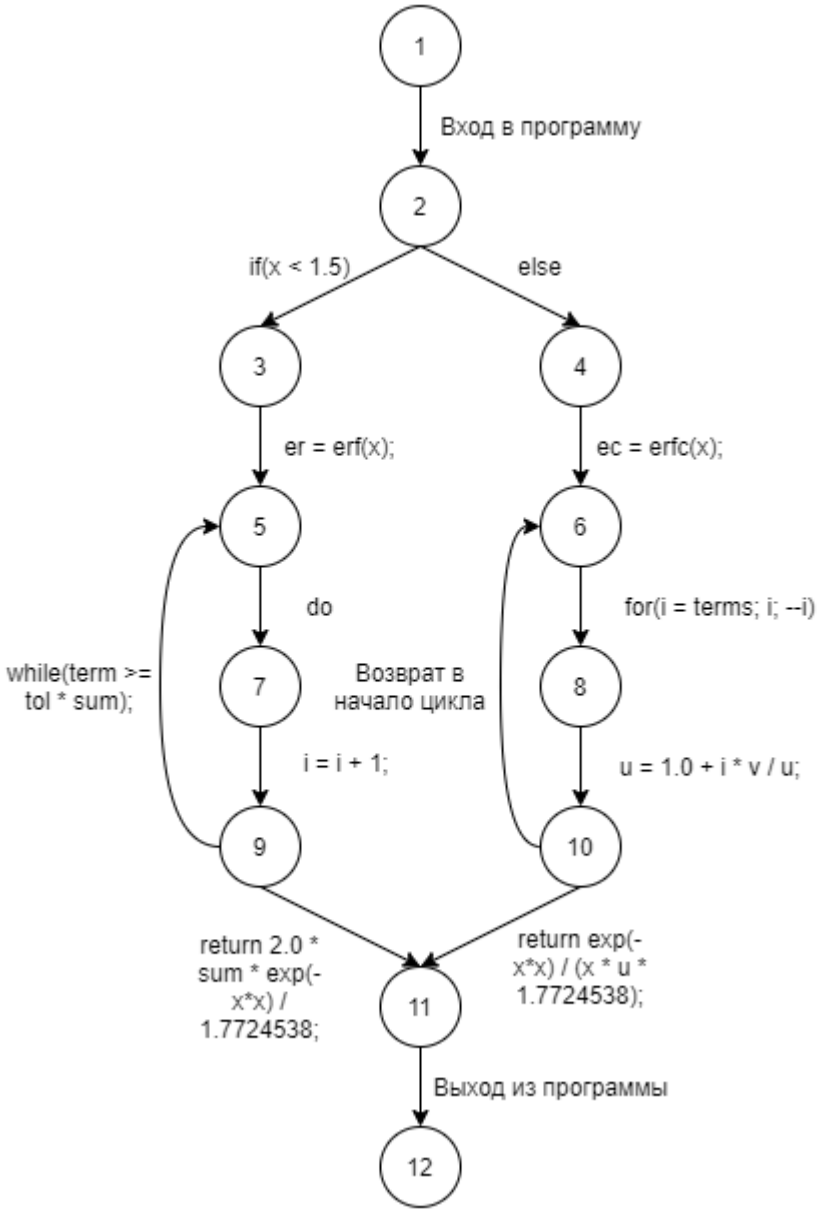

Рисунок 1: Управляющий граф программы

2. На основе текста программы из третьей лабораторной работы был составлен текст программы для профилирования с использованием профилировщика SAMPLER. Текст программы для профилирования представлен в Приложении Б, результаты профилирования представлены на рисунке 1:

1 : 8 1 : 15	87.16	1	87.16
1 : 15 1 : 17	0.84	1	0.84
1 : 17 1 : 21	309.26	7	44.18
1 : 21 1 : 17	112.30	6	18.72
1 : 21 1 : 23	4.19	1	4.19
1 : 23 1 : 54	249.75	1	249.75
1 : 28 1 : 34	207.85	1	207.85
1 : 34 1 : 36	2.51	1	2.51
1 : 36 1 : 38	185.22	12	15.43
1 : 38 1 : 36	17.60	11	1.60
1 : 38 1 : 40	1.68	1	1.68
1 : 40 1 : 59	235.51	1	235.51
1 : 47 1 : 48	1.68	2	0.84
1 : 48 1 : 50	57.83	2	28.91
1 : 50 1 : 52	70.40	1	70.40
1 : 50 1 : 57	4.19	1	4.19
1 : 52 1 : 8	53.64	1	53.64
1 : 54 1 : 62	69.56	1	69.56
1 : 57 1 : 28	54.48	1	54.48
1 : 59 1 : 62	53.64	1	53.64
1 : 62 1 : 63	1.68	2	0.84
1 : 63 1 : 47	1.68	1	1.68

Рисунок 1: Результаты профилирования программы

Суммарное время работы T = 1782,64 мкс.

3. На основании полученных с помощью профилировщика SAMPLER данных о работе программы был проведён расчёт вероятностей и затрат ресурсов для дуг управляющего граф. Результаты расчётов приведены в Таблице 2:

Дуга	**Номера строк**	**Количество проходов**	**Расчет вероятности**	**Затраты ресурсов**
L1 – L2	47 : 48	1	1	0.84
L2 – L3	50 : 52	1	0.5	70.40
L2 – L4	43 : 26	1	0.5	4.19
L3 – L5	52 : 8	1	1	53.64
L4 – L6	57 : 28	1	1	54.48
L5 – L7	15 : 17	1	1	0.84
L7 – L9	17 : 21	1	1	44.18
L9 – L5	21 : 17	1	0.86	18.72
L6 – L8	34 : 36	1	1	2.51
L8 – L10	36 : 38	8	1	15.43
L10 – L6	38 : 36	11	0.92	1.60
L9 – L11	23 : 54	1	0.14	249.75
L10 – L11	40 : 59	1	0.08	235.51
L11 – L12	37 : 39	1	1	0.84

Таблица 2: Расчёт вероятностей и затрат ресурсов

4. На основании полученных расчётов вероятностей и затрат ресурсов был построен операционная графовая модель программы, представленная на Рисунке 2:

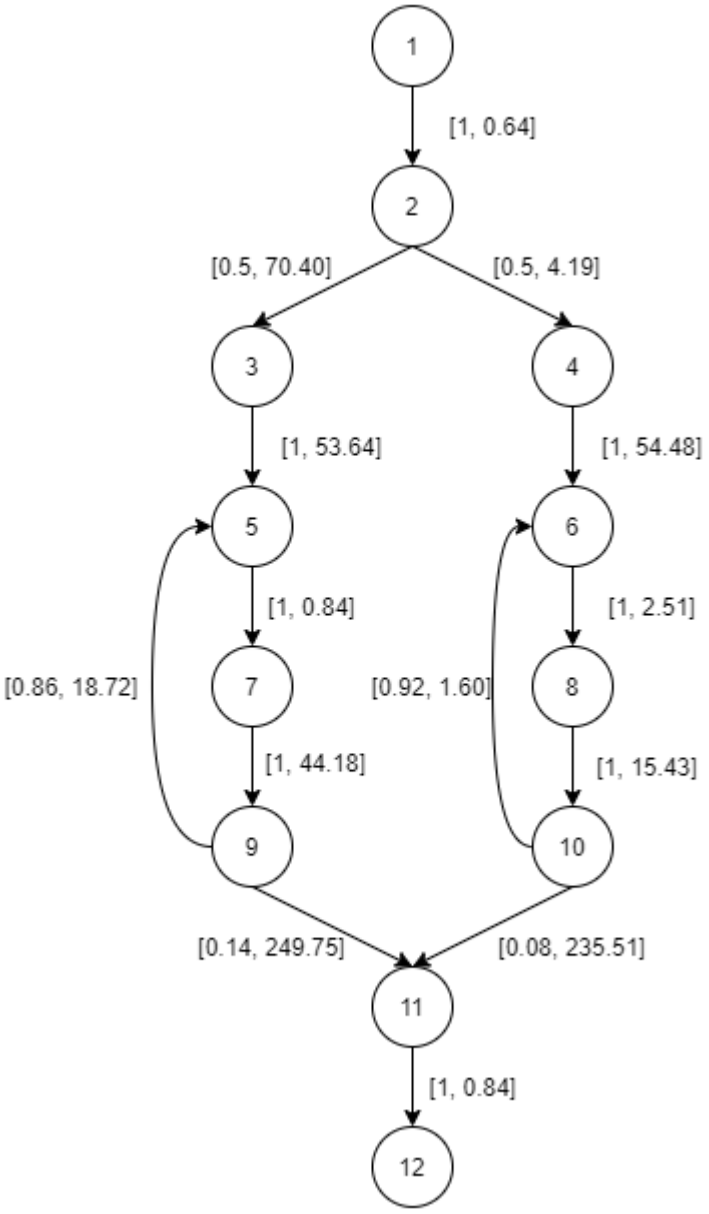

Рисунок 2: Операционная графовая модель программы

5. По полученной ОГМП был создан XML-файл модели программы, приведённый в Приложении В, для расчёта характеристик эффективности выполнения программы методом эквивалентных преобразований с помощью пакета CSA III. Графическое отображение модели представлено на Рисунке 3:

Рисунок 3: Модель программы в CSA III

1. С помощью пакета CSA III были вычислены математическое ожидание и дисперсия времени выполнения программы. Результаты вычислений представлены на Рисунке 4:

Рисунок 4: Результаты вычислений

Выводы.

В ходе выполнения 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 CSAIII были вычислены математическое ожидание и дисперсия времени выполнения программы. Математическое ожидание равно 4894.26, дисперсия – 10865861.82.

Математическое ожидание сильно отличается от полученного суммарного времени работы, равного 1782,64 мкс, что может быть объяснено низкой точностью вычисления вероятностей (шло округление до 2 знаков после запятой).

Приложение А: Исходный код программы.

```
#include <stdio.h>
#include <math.h>

double x, er, ec;

double erf(double x) {
    double x2, sum, term;
    int i;

    sum = x;
    term = x;
    i = 0;
    do {
        i = i + 1;
        term *= 2.0*x*x / (1.0 + 2.0 * i);
        sum += term;
    } while(term >= 1.0E-4 * sum);
    return 2.0 * sum * exp(-x*x) / 1.7724538;
}

double erfc(double x) {
    double u, v;
    int i;

    v = 1.0 / (2.0*x*x);
    u = 1.0 + v * 13.0;
    for(i = 12; i; --i) {
        u = 1.0 + i * v / u;
    }
    return exp(-x*x) / (x * u * 1.7724538);
}

int main() {
```

```
    double val[2] = {1.0, 2.0};
    for (int i = 0; i < 2; i++) {
        x = val[i];
        if(x < 1.5) {
            er = erf(x);
            ec = 1.0 - er;
        } else {
            ec = erfc(x);
            er = 1.0 - ec;
        }
    }
}
```

Приложение Б: Код программы для профилирования.

```
#include <stdio.h>
#include <math.h>
#include "Sampler.h"

double x, er, ec;

double erf(double x) {
    SAMPLE;
    double x2, sum, term;
    int i;

    sum = x;
    term = x;
    i = 0;
    SAMPLE;
    do {
        SAMPLE;
        i = i + 1;
        term *= 2.0*x*x / (1.0 + 2.0 * i);
        sum += term;
        SAMPLE;
    } while(term >= 1.0E-4 * sum);
    SAMPLE;
    return 2.0 * sum * exp(-x*x) / 1.7724538;
}

double erfc(double x) {
    SAMPLE;
    double u, v;
    int i;

    v = 1.0 / (2.0*x*x);
    u = 1.0 + v * 13.0;
    SAMPLE;
    for(i = 12; i; --i) {
        SAMPLE;
        u = 1.0 + i * v / u;
        SAMPLE;
    }
    SAMPLE;
    return exp(-x*x) / (x * u * 1.7724538);
}

int main() {
    double val[2] = {1.0, 2.0};
    for (int i = 0; i < 2; i++) {
        SAMPLE;
        SAMPLE;
        x = val[i];
```

```
            SAMPLE;
            if(x < 1.5) {
                SAMPLE;
                er = erf(x);
                SAMPLE;
                ec = 1.0 - er;
            } else {
                SAMPLE;
                ec = erfc(x);
                SAMPLE;
                er = 1.0 - ec;
            }
            SAMPLE;
            SAMPLE;
        }
    }
```

Приложение B: XML-файл модели программы.

```
<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>
     <node type = "Objects::AMC::Top" name = "t11"></node>
     <node type = "Objects::AMC::Top" name = "t12"></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 = "0.5" intensity = "70.40" deviation = "0.0" source =
"t2" dest = "t3"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t2-->t4"
probability = "0.5" intensity = "4.19" deviation = "0.0" source =
"t2" dest = "t4"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t3-->t5"
probability = "1.0" intensity = "53.64" deviation = "0.0" source =
"t3" dest = "t5"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t4-->t6"
probability = "1.0" intensity = "54.48" deviation = "0.0" source =
"t4" dest = "t6"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t5-->t7"
probability = "1.0" intensity = "0.84" deviation = "0.0" source =
"t5" dest = "t7"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t7-->t9"
probability = "1.0" intensity = "44.18" deviation = "0.0" source =
"t7" dest = "t9"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t9-->t5"
probability = "0.86" intensity = "18.72" deviation = "0.0" source
= "t9" dest = "t5"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t6-->t8"
probability = "1.0" intensity = "2.51" deviation = "0.0" source =
"t6" dest = "t8"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t8-->t10"
probability = "1.0" intensity = "15.43" deviation = "0.0" source =
"t8" dest = "t10"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t10-->t6"
probability = "0.92" intensity = "1.60" deviation = "0.0" source =
"t10" dest = "t6"></link>
     <link   type   =   "Objects::AMC::Link"   name   =   "t9-->t11"
probability = "0.14" intensity = "249.75" deviation = "0.0" source
= "t9" dest = "t11"></link>
```

```
    <link type = "Objects::AMC::Link" name = "t10-->t11"
probability = "0.08" intensity = "235.51" deviation = "0.0" source
= "t10" dest = "t11"></link>
    <link type = "Objects::AMC::Link" name = "t11-->t12"
probability = "1.0" intensity = "0.84" deviation = "0.0" source =
"t11" dest = "t12"></link>
</model>
```