**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Качество и метрология ПО»**

Тема: **Анализ структурной сложности графовых моделей программ**

|  |  |  |
| --- | --- | --- |
| Студент гр.6304 |  | Зыль С.Е. |
| Преподаватель |  | Кирьянчиков В.А. |

Санкт-Петербург

2020

**Задание**

Выполнить оценивание структурной сложности двух программ с помощью критериев:

* Минимального покрытия дуг графа;
* Выбора маршрутов на основе цикломатического числа графа.

Варианты программ:

* Программа с заданной преподавателем структурой управляющего графа;
* Программа из 1-ой лабораторной работы (управляющий граф составить самостоятельно).

Оцениваемые характеристики структурной сложности:

* Число учитываемых маршрутов проверки программы для заданного критерия;
* Цикломатическое число;
* Суммарное число ветвлений по всем маршрутам.

**Заданный граф**

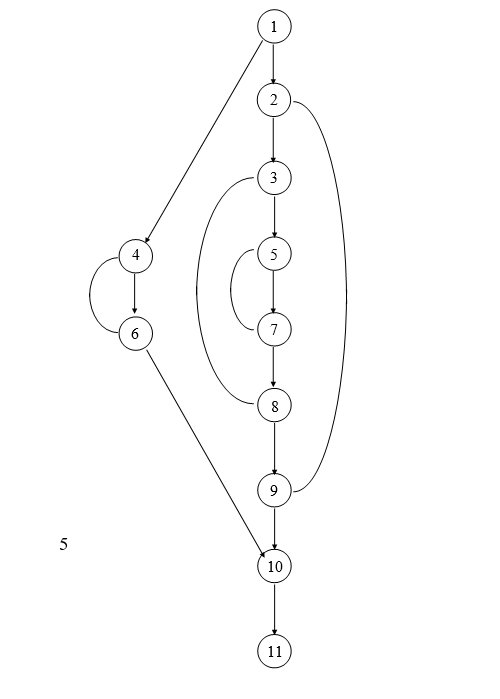


Рисунок 1 – Заданный граф

**Ручной расчет**

*По первому критерию:*

M1: **1** – 2 – 3 – 5 – **7** – 5 – **7** – **8** – 3 – 5 – **7** – **8** – **9** – 2 – 3 – 5 – **7** – **8** – **9** – 10 – 11

M2: **1** – 4– **6** – 4 – **6** – 10 - 11

По первому критерию требуется 2 маршрута М = 2.

Сложность программы по первому критерию:

= 10 + 3 = 13

*По второму критерию:*

**Z** = Y - N + 2\*P = 15 - 11 + 2\*1 = 6

Согласно цикломатическому числу, линейно-независимые маршруты:

M1: **1** – 2 – 3 – 5 – **7** **– 8** – **9** – 10 - 11

M2: **1** – 4 – **6** – 10 – 11

M3: 2 – 3 – 5 – **7 – 8 – 9** – 2

M4: 3 – 5 – **7 – 8** – 3

M5: 4 – **6** – 4

M6: 5–**7**–5

= 4 + 2 + 3 + 2 + 1 + 1= 13

**Автоматический расчет**

На входном языке программы граф имеет вид:

Листинг 1. Описание графа для ways.exe

|  |
| --- |
| Nodes {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}  Top [1]  Last [11]  Arcs {  arc(1, 2);  arc(1, 4);  arc(2, 3);  arc(3, 5);  arc(4, 6);  arc(5, 7);  arc(6, 4);  arc(6, 10);  arc(7, 5);  arc(7, 8);  arc(8, 3);  arc(8, 9);  arc(9, 2);  arc(9, 10);  arc(10, 11);  } |

По первому критерию:

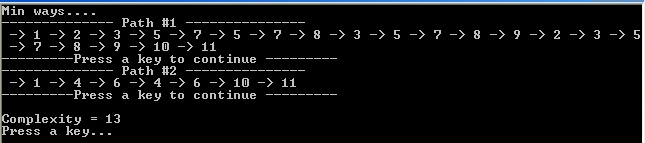


Рисунок 2 – Пути по первому критерию

По второму критерию:

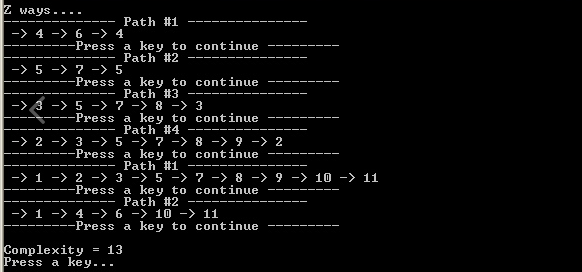


Рисунок 3 – Пути по второму критерию

**Граф программы из первой лабораторной работы (два метода сортировки)**

Листинг 2. Код программы из первой лабораторной работы на языке С

|  |
| --- |
| #include <cstdlib>  #include <ctime>  constint MAX = 1000;  void sort1(double \* a, int n)  {  double hold;  for (inti = 0; i< n; i++) {  for (int j = 0; j < n; j++) {  if (a[j] > a[i]) {  hold = a[i];  a[i] = a[j];  a[j] = hold;  }  }  }  }  void swap(double \* a, int p, int q)  {  double hold = a[p];  a[p] = a[q];  a[q] = hold;  }  void sort2(double \* a, int n)  {  intnoChange = 1;  do {  noChange = 1;  for (inti = 0; i< n - 1; i++) {  if (a[i] > a[i + 1]) {  swap(a, i, i + 1);  noChange = 0;  }  }  } while (noChange != 1);  }  int main(intargc, char const \*argv[])  {  double \* forFirstSort = (double \*)calloc(MAX, sizeof(double));  double \* forSecondSort = (double \*)calloc(MAX, sizeof(double));  srand((unsigned)time(NULL));  double randNuber = 0;  for (inti = 0; i< MAX; i++) {  intrn = (rand() % 1000 + 1);  randNuber = (double) rn;  forFirstSort[i] = randNuber;  forSecondSort[i] = randNuber;  }  sort1(forFirstSort, MAX);  sort2(forSecondSort, MAX);  return 0;  } |

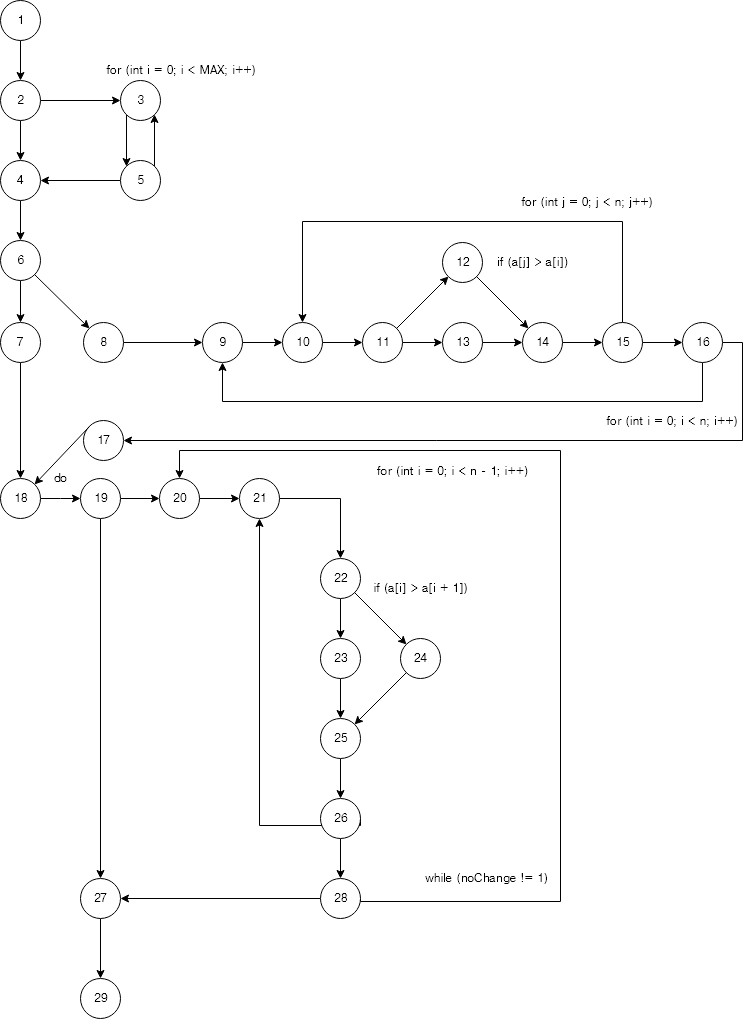


Рисунок 4 – граф из первой лабораторной работы

**Ручные расчёты**

*По первому критерию:*

M1: 1 – **2** – 4 – **6** – 8 – 9 – 10 – **11** – 12 – 13 – 14 – **15** – 10 – **11** – 13 – 14 – **15** – **16** – 9 – 10 – **11** – 13 – 14 – **15** – **16** – 17 – 18 – **19** – 20 – 21 – **22** – 23 – 25 – **26** – 21 – **22** – 24 – 25 – **26** – **28** – 20 – 21 – **22** – 23 – 25 – **26** – **28** – 27 – 29

M2: 1 – **2** – 3 – **5** – 3 – **5** –4 –**6** – 7 – 18 – **19** – 27 – 29

*По второму критерию:*

M1: 1 – **2** – 3 – **5** – 4 – **6** – 7 – 18 – **19** – 27 – 29

M2: 1 – **2** – 3 – **5** – 4 – **6** – 7 – 18 – **19** – 20 – 21 – **22** – 23 – 25 – **26** – **28** – 27 – 29

M3: 1 – **2** – 3 – **5** – 4 – **6** – 7 – 18 – **19** – 20 – 21 – **22** – 24 – 25 – **26** – **28** – 27 – 29

M4: 1 – **2** – 3 – **5** – 4 – **6** – 8 – 9 – 10 – **11** – 12 –14 – **15** – **16** – 17 – 18 – **19** – 27 – 29

M5: 1 – **2** – 3 – **5** – 4 – **6** – 8 – 9 – 10 – **11** – 13 – 14 – **15** – **16** – 17 – 18 – **19** – 27 – 29

M6: 1 – **2** – 4 – **6** – 7 – 18 – **19** – 27 – 29

M7: 3 – **5** – 3

M8: 10 – **11** – 13 – 14 – **15** – 10

M9: 9 – 10 – **11** – 13 – 14 – **15** – **16** – 9

M10:21 – **22** – 23 – 25 – **26** – 21

M11:20 – 21 – **22** – 23 – 25 – **26** – **28** – 20

Описание графа на входном языке программы анализа имеет вид:

**Автоматические расчёты**

Листинг 3. Описание графа для ways.exe

|  |
| --- |
| Nodes {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29}  Top {1}  Last {29}  Arcs {  arc(1, 2);  arc(2, 3);  arc(2, 4);  arc(3, 5);  arc(5, 3);  arc(5, 4);  arc(4, 6);  arc(6, 7);  arc(6, 8);  arc(8, 9);  arc(9, 10);  arc(10, 11);  arc(11, 12);  arc(11, 13);  arc(12, 13);  arc(13, 14);  arc(14, 15);  arc(15, 10);  arc(15, 16);  arc(16, 9);  arc(16, 17);  arc(17, 18);  arc(7, 18);  arc(18, 19);  arc(19, 27);  arc(19, 20);  arc(27, 29);  arc(20, 21);  arc(21, 22);  arc(22, 23);  arc(22, 24);  arc(23, 25);  arc(24, 25);  arc(25, 26);  arc(26, 21);  arc(26, 28);  arc(28, 20);  arc(28, 27);  } |

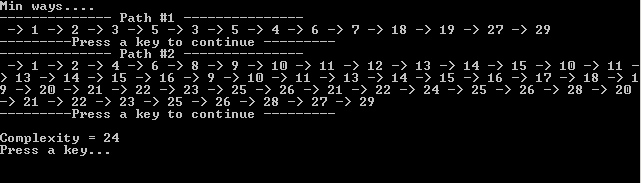
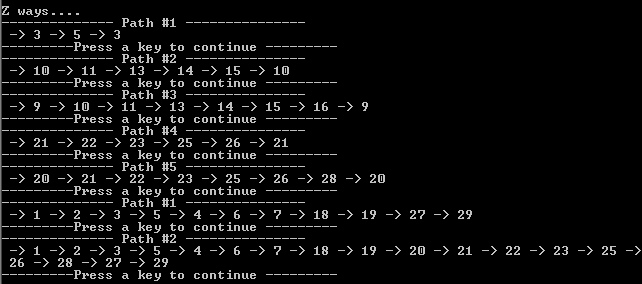


Рисунок 4 – Пути по первому критерию



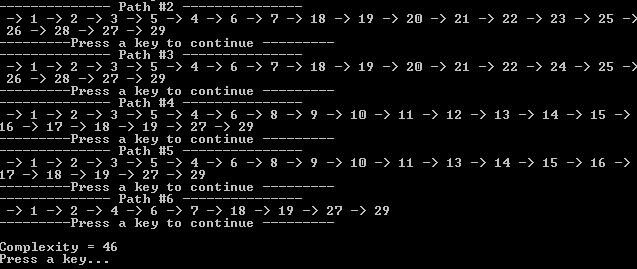


Рисунок 5 – Пути по второму критерию

**Выводы**

В данной лабораторной работе была выполнена оценка структурной сложности двух программ с помощью критериев: минимального покрытия дуг графа и выбора маршрутов на основе цикломатического числа графа. Расчеты были проведены как ручным, так и программным способом.