МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР
ГОУВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра программной инженерии
им. Л.П. Фельдмана

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОЙ РАБОТЕ

НА ТЕМУ: «Построение аналитических моделей алгоритмов и оценка их сложности»

ПО КУРСУ: «Теория алгоритмов и формальных языков»

Выполнил:
студент группы ПИ-21в
Рустамов В.Р.

Руководитель:
Коломойцева И.А.
Серёженко О.А.
Щедрин С.В.

Донецк – 2022

РЕФЕРАТ

Отчет по курсовой работе содержит: 49 страниц, 30 рисунок, 2 таблицы, 3 приложения, 2 источников.

Объект исследования – рекурсивные функции, машины Тьюринга, нормальные алгоритмы Маркова.

Цель – сформировать формальное определение алгоритма в виде трех аналитических моделей, написать программную реализацию машины Тьюринга, распознающей язык $L = \{ wcv \mid w, v \in \{0, 1\}^* \& |w| = 2k \& |v| = 2n+1 \& k \neq n \& k, n \geq 0\}$, построить график временной сложности.

Результат – формальное определение алгоритмов на основе рекурсивных функций, машин Тьюринга и нормальных алгоритмов Маркова, программная реализация машины Тьюринга, распознающей язык $L = \{ wcv \mid w, v \in \{0, 1\}^* \& |w| = 2k \& |v| = 2n+1 \& k \neq n \& k, n \geq 0\}$, график временной сложности машины Тьюринга, файловый вариант протокола работы машины Тьюринга.

МАШИНА ТЬЮРИНГА, ВРЕМЕННАЯ СЛОЖНОСТЬ, АЛФАВИТ, ЛЕНТА, ЯЗЫК, РАСПОЗНАВАНИЕ, ПРОТОКОЛ, ПРИНАДЛЕЖНОСТЬ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1 ОПИСАНИЕ АНАЛИТИЧЕСКОЙ МОДЕЛИ АЛГОРИТМА В ВИДЕ ЭЛЕМЕНТАРНЫХ МАШИН ТЬЮРИНГА И КОМПОЗИЦИИ МТ 5
2 РАЗРАБОТКА АНАЛИТИЧЕСКОЙ И ПРОГРАММНОЙ МОДЕЛИ АЛГОРИТМА ДЛЯ РАСПОЗНАЮЩЕЙ МАШИНЫ ТЬЮРИНГА 10
2.1 Формальное определение распознающей машины Тьюринга 10
2.2 Протоколы работы машины Тьюринга 12
2.3 Программная модель машины Тьюринга 15
2.4 Протоколы работы машины Тьюринга, построенные программно 16
2.5 Расчёт временной сложности (график функции временной сложности) .. 26
3 РАЗРАБОТКА АНАЛИТИЧЕСКОЙ МОДЕЛИ АЛГОРИТМА С ИСОЛЬЗОВАНИЕМ НОРМАЛЬНЫХ АЛГОРИТМОВ МАРКОВА 29
4 ОПИСАНИЕ ФОРМАЛЬНОЙ МОДЕЛИ АЛГОРИТМА НА ОСНОВЕ РЕКУРСИВНЫХ ФУНКЦИЙ 31
ВЫВОДЫ 32
ПЕРЕЧЕНЬ ССЫЛОК 33
ПРИЛОЖЕНИЕ А Техническое задание 34
ПРИЛОЖЕНИЕ Б Руководство пользователя 38
ПРИЛОЖЕНИЕ В Исходный код 39

ВВЕДЕНИЕ

В современной программной инженерии алгоритмы, как методы решения задач, занимают ведущее место по сравнению с традиционной математикой. Причем не важно, существует или нет чистое алгоритмическое решение в абстрактных моделях алгоритмов. Если решение задачи необходимо, широко используется эвристика, а “доказательством” работоспособности алгоритма является успешное его тестирование [1].

Алгоритм – это набор предписаний, однозначно определяющих содержание и последовательность выполнения операций для преобразования варьируемых исходных данных в искомый результат. Алгоритмам характерны определённость, результативность, дискретность, массовость и выполнимость операций.

Теория алгоритмов изучает вопросы существования алгоритмов для решения некоторой задачи и выбора наилучшего из существующих. В ходе данной работы будут рассмотрены следующие формальные модели алгоритмов: машины Тьюринга; нормальные алгоритмы Маркова; рекурсивные функции.

Цель работы – это изучение формальных моделей алгоритмов. Также одной из основных задач является программная реализация одноленточной и многоленточной машин Тьюринга, распознающей язык $L = \{ wcv \mid w, v \in \{0, 1\}^* \ \& \ |w| = 2k \ \& \ |v| = 2n+1 \ \& \ k \neq n \ \& \ k, n \geq 0\}$, и построения графика временной сложности для них. Для решения анной проблемы выбран язык программирования C#, среда программирования Visual Studio 2022.

1 ОПИСАНИЕ АНАЛИТИЧЕСКОЙ МОДЕЛИ АЛГОРИТМА В ВИДЕ ЭЛЕМЕНТАРНЫХ МАШИН ТЬЮРИНГА И КОМПОЗИЦИИ МТ

Машина Тьюринга представляет собой алгоритм как некоторое детерминированное устройство (автомат), реализующий действие над словами. Состоит из бесконечной ленты, разделённой на ячейки. На этой ленте могут быть записаны слова в некотором заранее фиксированном алфавите. По ленте движется управляющее устройство, обозревающее одну из ячеек. В зависимости от состояния машины и содержания обозреваемой ячейки, машина может, в соответствие с командой, изменить (или не изменять) состояние, заменить (или не заменять) содержимое обозреваемой ячейки, сдвинуть (или не сдвигать) на одну ячейку управляющее устройство влево или вправо [2].

Существует несколько способов описания машины Тьюринга:

– система команд;

– функциональная таблица;

– диаграмма состояний (граф переходов).

Опишем тремя способами машину Тьюринга, реализующий функцию циклического сдвига двоичного числа на одну ячейку.

В таблице 1.1 представлена функциональная таблица данной машины.

Таблица 1.1 - Функциональная таблица

	0	1	λ
q_1	q_3 λ R	q_2 λ R	q_z λ E
q_2	q_2 0 R	q_2 1 R	q_4 1 L
q_3	q_3 0 R	q_3 1 R	q_4 0 L
q_4	q_4 0 L	q_4 1 L	q_5 λ R
q_5	q_z 0 E	q_z 1 E	

На рисунке 1.1 представлена система команд рассматриваемой машины Тьюринга.

$q_1\,0 \to q_3\,\lambda\,R$

$q_1\,1 \to q_2\,\lambda\,R$

$q_1\,\lambda \to q_z\,\lambda\,E$

$q_2\,0 \to q_2\,0\,R$

$q_2\,1 \to q_2\,1\,R$

$q_2\,\lambda \to q_4\,1\,L$

$q_3\,0 \to q_3\,0\,R$

$q_3\,1 \to q_3\,1\,R$

$q_3\,\lambda \to q_4\,0\,L$

$q_4\,0 \to q_4\,0\,L$

$q_4\,1 \to q_4\,1\,L$

$q_4\lambda \to q_5\,\lambda\,R$

$q_5\,0 \to q_z\,0\,E$

$q_5\,1 \to q_z\,1\,E$

Рисунок 1.1 – Система команд

На рисунке 1.2 изображен граф переходов (диаграмма состояний), на котором вершины графа – это состояния машины Тьюринга, а дуги – переходы между ними.

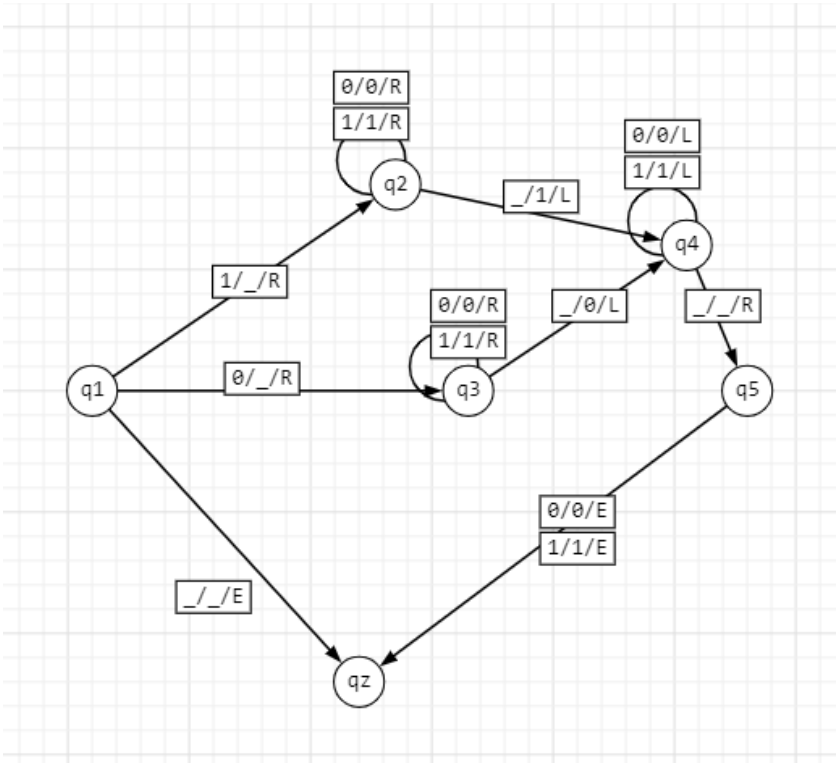

Рисунок 1.2 – Граф переходов

Тестовые примеры работы машины Тьюринга по разным входным данным представлены на рисунках 1.3 – 1.4.

K_0:$q_0$111

K_1:1$q_1$11

K_2: λ1$q_2$1

K_3: λ11q_2

K_4: λ11λq_2

K_5: λ111q_2

K_6: λ11$q_4$1

K_7: λ1$q_4$11

K_8: λ$q_4$111

K_9: λ1$q_5$11

K_{10}: λ1q_z11

Рисунок 1.3 – Пример работы машины Тьюринга (на входе 111)

K_0: λq_0λλ

K_1: λq_1λλ

K_2: λq_zλλ

Рисунок 1.4 – Пример работы машины Тьюринга (на входе пустая лента)

Составим композицию машин Тьюринга на примере следующей задачи. Найти количество полных квадратов до n.

Для более наглядного решения составим блок- схему (см. рис. 1.5).

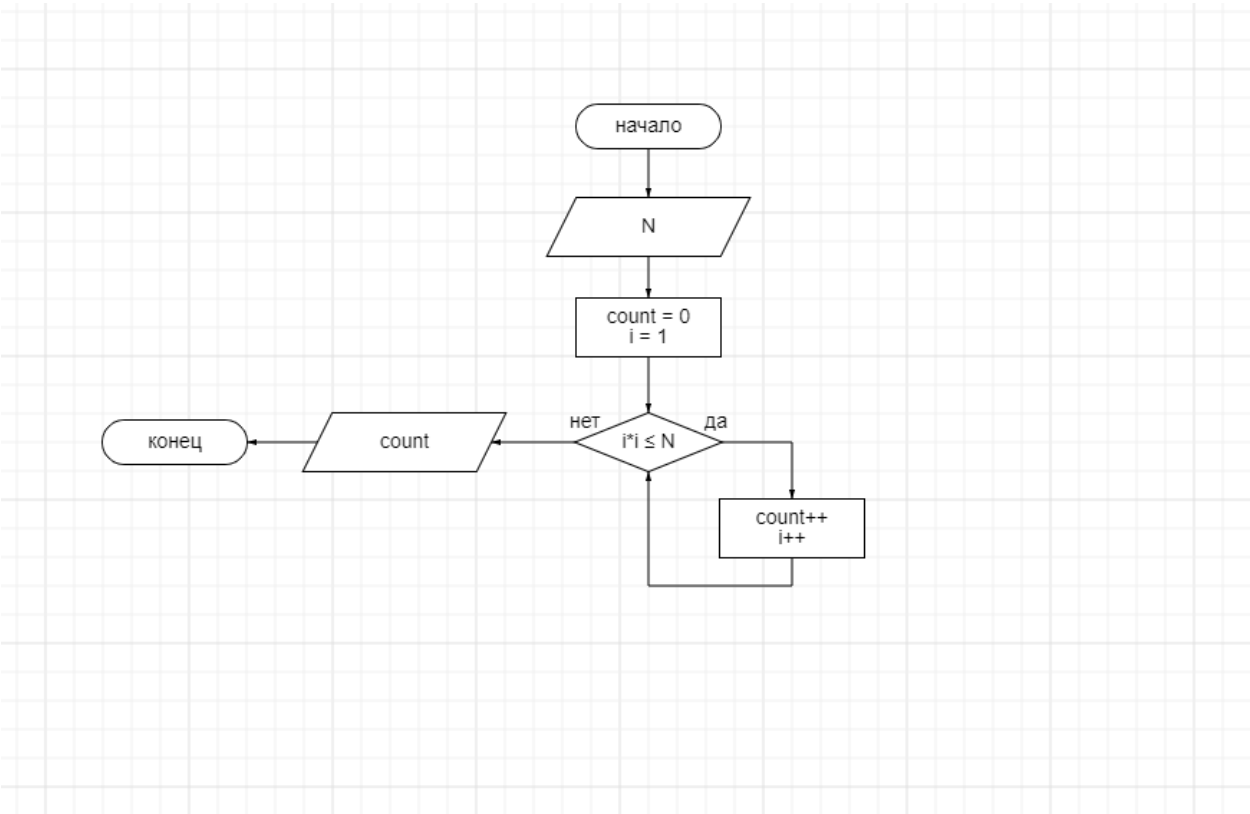

Рисунок 1.5 – Блок-схема алгоритма

Исходя из блок-схемы, составим композицию машин Тьюринга (см. рис. 1.7), где использованы следующие элементарные машины:

1) M_0 – МТ, реализующая копирование входного слова;
2) M^0 – МТ, реализующая функцию установки константы ноль;
3) M_n^i – МТ, реализующая функцию выбора i-того аргумента из n аргументов;
4) M_{+1} – МТ, реализующая функцию инкремента;
5) $M_{<=}$– МТ, реализующая функцию сравнение;
6) $M_{\wedge 2}$ – МТ, реализующая функцию возведения в квадрат.

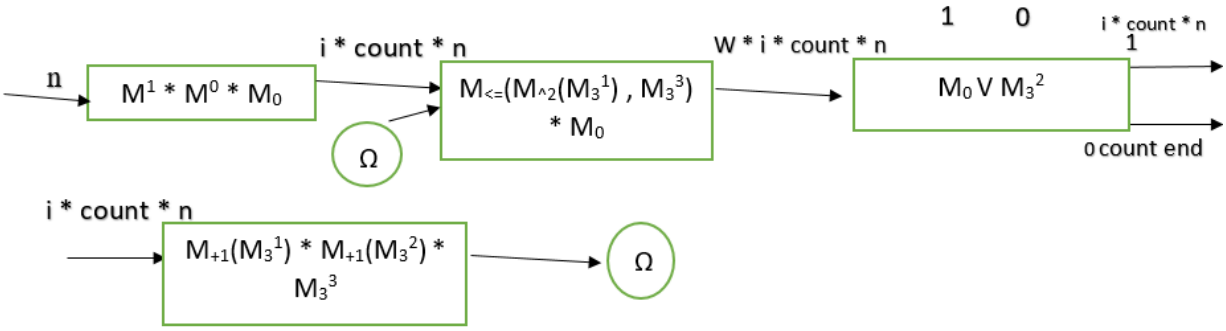

Рисунок 1.6 – Композиция Машин Тьюринга

2 РАЗРАБОТКА АНАЛИТИЧЕСКОЙ И ПРОГРАММНОЙ МОДЕЛИ АЛГОРИТМА ДЛЯ РАСПОЗНАЮЩЕЙ МАШИНЫ ТЬЮРИНГА

2.1 Формальное определение распознающей машины Тьюринга

Для распознавания слов языка L = { wcv|w, v∈{0, 1}* & |w|=2k & |v|=2n+1 & k≠n & k, n≥0} разработано два алгоритма (для одноленточной и двуленточной машин Тьюринга). Рассмотрим подробно.

Система команд для одноленточной машины Тьюринга представлена на рисунке 2.1.

1. $q1\ 0 \to q2\ \#\ R$
2. $q1\ 1 \to q2\ X\ R$
3. $q1\ c \to q6\ c\ R$
4. $q1\ \lambda \to q2\ 1\ R$
5. $q2\ c \to q3\ c\ R$
6. $q2\ 0 \to q2\ 0\ R$
7. $q2\ 1 \to q2\ 1\ R$
8. $q3\ \lambda \to q10\ \lambda\ L$
9. $q3\ 0 \to q4\ \#\ L$
10. $q3\ 1 \to q4\ X\ L$
11. $q3\ X \to q3\ X\ R$
12. $q3\ \# \to q3\ \#\ R$
13. $q4\ c \to q5\ 0\ L$
14. $q4\ X \to q4\ X\ L$
15. $q4\ \# \to q4\ \#\ L$
16. $q5\ 0 \to q5\ 0\ L$
17. $q5\ X \to q1\ X\ R$
18. $q5\ 1 \to q5\ 1\ L$
19. $q5\ \# \to q1\ \#\ R$
20. $q5\ \lambda \to q25\ \lambda\ L$
21. $q6\ 0 \to q7\ 0\ R$
22. $q6\ X \to q6\ X\ R$
23. $q6\ 1 \to q7\ 1\ R$
24. $q6\ \# \to q6\ \#\ R$
25. $q6\ \lambda \to q7\ \lambda\ E$
26. $q7\ 0 \to q10\ 0\ L$
27. $q7\ 1 \to q10\ 1\ L$
28. $q7\ \lambda \to q8\ \lambda\ L$
29. $q8\ 0 \to q8\ \lambda\ L$
30. $q8\ 1 \to q8\ \lambda\ L$
31. $q8\ c \to q8\ c\ L$
32. $q8\ \lambda \to q9\ \lambda\ R$
33. $q8\ \# \to q8\ \lambda\ L$
34. $q8\ X \to q8\ \lambda\ L$
35. $q9\ c \to qz\ 0\ E$
36. $q9\ \lambda \to q9\ \lambda\ R$
37. $q10\ 0 \to q10\ 0\ L$
38. $q10\ 1 \to q10\ 1\ L$
39. $q10\ c \to q10\ c\ L$
40. $q10\ \lambda \to q11\ \lambda\ R$
41. $q10\ \# \to q10\ 0\ L$
42. $q10\ X \to q10\ 1\ L$
43. $q11\ 0 \to q12\ X\ R$
44. $q11\ 1 \to q12\ X\ R$
45. $q11\ c \to q13\ c\ L$
46. $q11\ \lambda \to q14\ 1\ R$
47. $q11\ X \to q11\ \lambda\ R$
48. $q12\ 0 \to q11\ X\ L$
49. $q12\ 1 \to q11\ X\ L$
50. $q12\ c \to q13\ c\ L$
51. $q13\ X \to q14\ 0\ R$
52. $q13\ \lambda \to q14\ 1\ R$
53. $q14\ c \to q15\ c\ R$
54. $q15\ 0 \to q15\ 0\ R$
55. $q15\ 1 \to q15\ 1\ R$
56. $q15\ \lambda \to q16\ \lambda\ L$
57. $q16\ 0 \to q17\ X\ L$
58. $q16\ 1 \to q17\ X\ L$
59. $q16\ c \to q18\ c\ R$
60. $q16\ X \to q16\ \lambda\ L$
61. $q17\ 0 \to q16\ X\ R$
62. $q17\ 1 \to q16\ X\ R$
63. $q17\ c \to q18\ c\ R$
64. $q18\ X \to q19\ 0\ L$
65. $q18\ \lambda \to q19\ 1\ L$
66. $q19\ 0 \to q22\ \lambda\ R$
67. $q19\ 1 \to q20\ \lambda\ R$
68. $q19\ c \to q19\ c\ L$
69. $q20\ 0 \to q21\ \lambda\ L$
70. $q20\ 1 \to q24\ \lambda\ L$
71. $q20\ c \to q20\ c\ R$
72. $q21\ c \to qz\ 1\ E$
73. $q22\ c \to q23\ c\ R$
74. $q23\ 0 \to q24\ \lambda\ L$
75. $q23\ 1 \to q24\ \lambda\ L$
76. $q24\ c \to qz\ 0\ E$
77. $q25\ X \to q17\ X\ L$
78. $q25\ 1 \to q25\ 1\ R$
79. $q25\ c \to q24\ c\ L$
80. $q26\ 0 \to q26\ 0\ R$
81. $q26\ 1 \to q26\ X\ R$
82. $q26\ c \to q27\ c\ R$
83. $q26\ \lambda \to qz\ 0\ E$
84. $q27\ 0 \to q1\ 0\ L$
85. $q27\ 1 \to q1\ 1\ L$
86. $q27\ c \to q8\ c\ L$
87. $q27\ \lambda \to q1\ \lambda\ R$
88. $q28\ c \to q29\ c\ R$
89. $q28\ \lambda \to qz\ 0\ E$
90. $q29\ c \to qz\ 0\ E$
91. $q29\ \lambda \to q1\ \lambda\ L$

Рисунок 2.1 – Система команд для одноленточной машины Тьюринга

На ленту подаётся слово сначала мы проверяем условие (k!=n) для этого мы по очереди передвигаем каретку из слова w,заменяя его на пометку, к слову v,также заменяя его. Если слова w полностью заменено, а слово v имеет символ 1/0 то мы двигаем каретку вправо, и если там пустой символ, значит условие ложно, мы стираем запись и ставим 0, иначе делаем обратную замену и переходим к условию (|w| = 2k && |v| = 2n+1). Здесь мы заменяем символы и стираем их попарно, если следующий символ “c” (для w) или пустой символ(для v), то ставим 0 слева или справа, что означает нечетность, иначе 1, если справа или слева “c” пустой символ. Верное условие будет, когда предфинальный вид слова обретает вид “1c0”, это будет означать, что слова w и v соответствуют условиям четности/нечестности. Далее на основании этих результатов, “c” будет равно либо 1,либо 0.

Система команд для двуленточной машины Тьюринга представлена на рисунке 2.2.

1. $q1\ (0, \lambda) \to q1(\lambda,0;\ R;R)$
2. $q1\ (1, \lambda) \to q1(\lambda,1;\ R;R)$
3. $q1\ (c, \lambda) \to q2(\lambda,\lambda;\ R;L)$
4. $q2\ (1, 0) \to q2(1,0;\ R;R)$
5. $q2\ (1, 1) \to q2(1,1;\ R;R)$
6. $q2\ (0, 0) \to q2(0,0;\ R;R)$
7. $q2\ (0, 1) \to q2(0,1;\ R;R)$
8. $q2\ (1, \lambda) \to q3(1,\lambda;\ R;E)$
9. $q2\ (0, \lambda) \to q3(0,\lambda;\ R;E)$
10. $q2\ (\lambda, \lambda) \to q5(\lambda,\lambda;\ L;L)$
11. $q2\ (\lambda, 1) \to q5(\lambda,1;\ L;L)$
12. $q2\ (\lambda, 0) \to q5(\lambda,0;\ L;L)$
13. $q3\ (0, \lambda) \to q5(0,\lambda;\ L;L)$
14. $q3\ (1, \lambda) \to q5(1,\lambda;\ L;L)$
15. $q3\ (\lambda, \lambda) \to q4(\lambda,\lambda;\ L;L)$
16. $q4\ (1, \lambda) \to q5(\lambda,\lambda;\ L;L)$
17. $q4\ (0, \lambda) \to q4(\lambda,\lambda;\ L;L)$
18. $q4\ (0, 0) \to q4(\lambda,\lambda;\ L;L)$
19. $q4\ (0, 1) \to q4(\lambda,\lambda;\ L;L)$
20. $q4\ (1, 0) \to q4(\lambda,\lambda;\ L;L)$
21. $q4\ (1, 1) \to q4(\lambda,\lambda;\ L;L)$
22. $q4\ (\lambda, \lambda) \to qz(0,\lambda;\ E;E)$
23. $q5\ (0, 0) \to q6(\lambda,\lambda;\ R;R)$
24. $q5\ (0, 1) \to q6(\lambda,\lambda;\ R;R)$
25. $q5\ (1, 0) \to q6(\lambda,\lambda;\ R;R)$
26. $q5\ (1, 1) \to q6(\lambda,\lambda;\ R;R)$
27. $q5\ (\lambda, 1) \to q7(1,\lambda;\ E;R)$
28. $q5\ (\lambda, 0) \to q7(1,\lambda;\ E;R)$
29. $q5\ (0, \lambda) \to q7(\lambda,1;\ R;E)$
30. $q5\ (1, \lambda) \to q7(\lambda,1;\ R;E)$
31. $q6\ (0, 0) \to q5(\lambda,\lambda;\ R;R)$
32. $q6\ (0, 1) \to q5(\lambda,\lambda;\ R;R)$
33. $q6\ (1, 0) \to q5(\lambda,\lambda;\ R;R)$
34. $q6\ (1, 1) \to q5(\lambda,\lambda;\ R;R)$
35. $q6\ (\lambda, 1) \to q7(0,\lambda;\ E;R)$
36. $q6\ (\lambda, 0) \to q7(0,\lambda;\ E;R)$
37. $q6\ (0, \lambda) \to q7(\lambda,0;\ R;E)$
38. $q6\ (1, \lambda) \to q7(\lambda,0;\ R;E)$
39. $q7\ (0, 1) \to qz(1,\lambda;\ E;E)$
40. $q7\ (1, 0) \to qz(0,\lambda;\ E;E)$
41. $q7\ (1, 1) \to qz(0,\lambda;\ E;E)$
42. $q7\ (0, 0) \to qz(0,\lambda;\ E;E)$

Рисунок 2.2 – Система команд для двуленточной машины Тьюринга

На вход подаётся слово, проверяем наличие символов 1/0/c, если их нет, то ставим 0, если символов с больше 1, то 0, иначе записываем на вторую ленту пошагово символы с первой до встречи символа “c”, после мы его стираем. Следующий шаг, проверяем условие (k!=n) одновременно сдвигаем каретки, если достигаем конца слова второй ленты, но не достигаем конца слова первой лент, то проверяем, стоит ли за ним символ, если да, то переходим к проверке слов на четность/нечетность, иначе стираем слова с обеих лент и ставим в первой ленте 0. Проверка на четность/нечетность аналогична одноленточной машине. После проверки, смотрим, если первая лента в результате выдаёт 0, а вторая 1, то стираем вторую, а в первой ставим 1, иначе стираем вторую и в первой ставим 1.

Алгоритмы похожи, однако за счёт дополнительной памяти (второй ленты) двуленточная машина Тьюринга работает быстрее и требует меньше тактов для вычислений.

2.2 Протоколы работы машины Тьюринга

Рассмотрим тестовые примеры работы описанных машин Тьюринга.

Для одноленточной машины подадим на вход слова 1c11 (слово не принадлежит языку) (см. рис. 2.3) и c111 (слово принадлежит языку) (см. рис. 2.4).

1. q_{26}: λ1q_{26}c11λ
2. q_{26}: λ1cq_{26}11λ
3. q_{27}: λ1c1q_{27}1λ
4. q_1: λ1q_1c11λ
5. q_2: λX$cq_2$11λ
6. q_3: λXc1$q_3$1λ
7. q_4: λXcq_4X1λ
8. q_5: λXq_5cX1λ
9. q_1: λXcq_1X1λ
10. q_6: λXcX$q_6$1λ
11. q_6: λXcX1q_6λ
12. q_7: λXcX1λq_7
13. q_8: λXcX1q_8λ
14. q_8: λXcXq_8λλ
15. q_8: λXcq_8λλλ
16. q_8: λXq_8cλλλ
17. q_8: λq_8λcλλλ
18. q_9: λλq_9cλλλ
19. q_9: λλcq_9λλλ
20. q_z: λλ0q_zλλλ

Рисунок 2.3 – Протокол работы машины Тьюринга со словом 1c11

1. q_{26}: λcq_{26}111λ
2. q_{27}: λc1q_{27}11λ
3. q_1: λc$q_1$111λ
4. q_6: λc1$q_6$11λ
5. q_7: λc11$q_7$1λ
6. q_{10}: λc1q_{10}11λ
7. q_{10}: λcq_{10}111λ
8. q_{10}: λq_{10}c111λ
9. q_{11}: λcq_{11}111λ
10. q_{13}: λq_{13}c111λ
11. q_{14}:1cq_{14}111λ
12. q_{15}:1c1q_{15}11λ
13. q_{15}:1c11q_{15}1λ
14. q_{15}:1c111q_{15}λ
15. q_{15}:1c111λq_{15}
16. q_{16}:1c111q_{16}λ
17. q_{17}:1c11q_{17}Xλ
18. q_{16}:1c1XXq_{16}λ
19. q_{16}:1c1Xq_{16}λλ
20. q_{16}:1c1q_{16}λλλ
21. q_{17}:1cq_{17}X λλλ
22. q_{18}:1cXq_{18}λλλ
23. q_{19}:1cq_{19}0λλλ
24. q_{19}:1q_{19}c0λλλ
25. q_{20}: λcq_{20}0λλλ
26. q_{20}: λc0q_{20}λλλ
27. q_{21}: λcq_{21}λλλλ
28. qz: λ1q_zλλλλ

Рисунок 2.4 – Протокол работы машины Тьюринга со словом c111

Для двуленточной машины подадим на вход слова 1c111 (слово не принадлежит языку) (см. рис. 2.5) и c111 (слово принадлежит языку) (см. рис. 2.6).

1. q_1 l1:λ1q_1c111λλλλλλλλλλ
2. q_1 l2:λλq_1λλλλλλλλλλλλλλ
3. q_1 l1:λλc$q_1$111λλλλλλλλλλ
4. q_1 l2:λ1λq_1λλλλλλλλλλλλλ
5. q_2 l1:λλλ1$q_2$11λλλλλλλλλλ
6. q_2 l2:λ1q_2λλλλλλλλλλλλλλ
7. q_2 l1:λλ11$q_2$1λλλλλλλλλλ
8. q_2 l2:λ1λq_2λλλλλλλλλλλλλ
9. q_3 l1:λλλ111q_3λλλλλλλλλλ
10. q_3 l2:λ1λq_3λλλλλλλλλλλλλ
11. q_5 l1:λλλ1$q_5$11λλλλλλλλλλ
12. q_5 l2:λ1q_5λλλλλλλλλλλλλλ
13. q_6 l1:λλλ1$q_6$1λλλλλλλλλλ
14. q_6 l2:λ1q_6λλλλλλλλλλλλλλ
15. q_5 l1:λλλλ1q_6λλλλλλλλλλ
16. q_5 l2:λ1q_6λλλλλλλλλλλλλλ
17. q_6 l1:λλλλλλq_6λλλλλλλλλλ
18. q_6 l2:λ1q_6λλλλλλλλλλλλλλ
19. q_5 l1:λ1q_5λλλλλλλλλλλλλλ
20. q_5 l2:λ1q_5λλλλλλλλλλλλλλ
21. q_6 l1:λλq_6λλλλλλλλλλλλλλ
22. q_6 l2:λλq_6λλλλλλλλλλλλλλ
23. q_7 l1:λλλλλ0q_7λλλλλλλλλλ
24. q_7 l2:λλ0q_7λλλλλλλλλλλλλ
25. q_z l1:λλλλ0q_zλλλλλλλλλλ
26. q_z l2:λλλq_zλλλλλλλλλλλλλ

Рисунок 2.5 – Протокол работы машины Тьюринга со словом 1c111

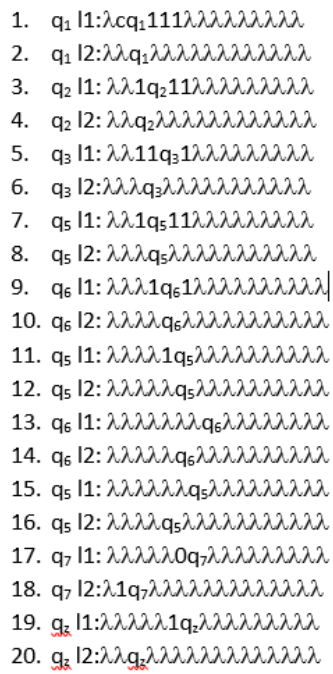

Рисунок 2.6 – Протокол работы машины Тьюринга со словом с111

2.3 Программная модель машины Тьюринга

Разработанная программа демонстрирует работу одноленточной и двуленточной машин Тьюринга. Разработана на языке C# в среде программирования Visual Studio 2022.

Пользователь вводит слово в текстовое поле. После чего слово идёт в конструктор класса StringBuilder. Его преимущества – простой доступ к элементам, а также их редактирование и добавление. Все состояния описаны с помощью конструкций switch case. Состояния реализации движения каретки влево и вправо используется индекс рассматриваемого символа. В зависимости от того, в какую сторону нужно двигаться каретке, к индексу применяется инкремент или декремент. Каждый шаг машины Тьюринга выводиться в текстовое поле в виде текстов. Также протокол работы машины записывается в текстовый файл.

По аналогичному принципу реализована двуленточная машина Тьюринга.

2.4 Протоколы работы машины Тьюринга, построенные программно

Рассмотрим тестовые примеры работы реализованных машин Тьюринга.

Для одноленточной машины подадим на вход слова 11c111 (слово не принадлежит языку) (см. рис. 2.7) и 11c11111 (слово принадлежит языку) (см. рис. 2.8).

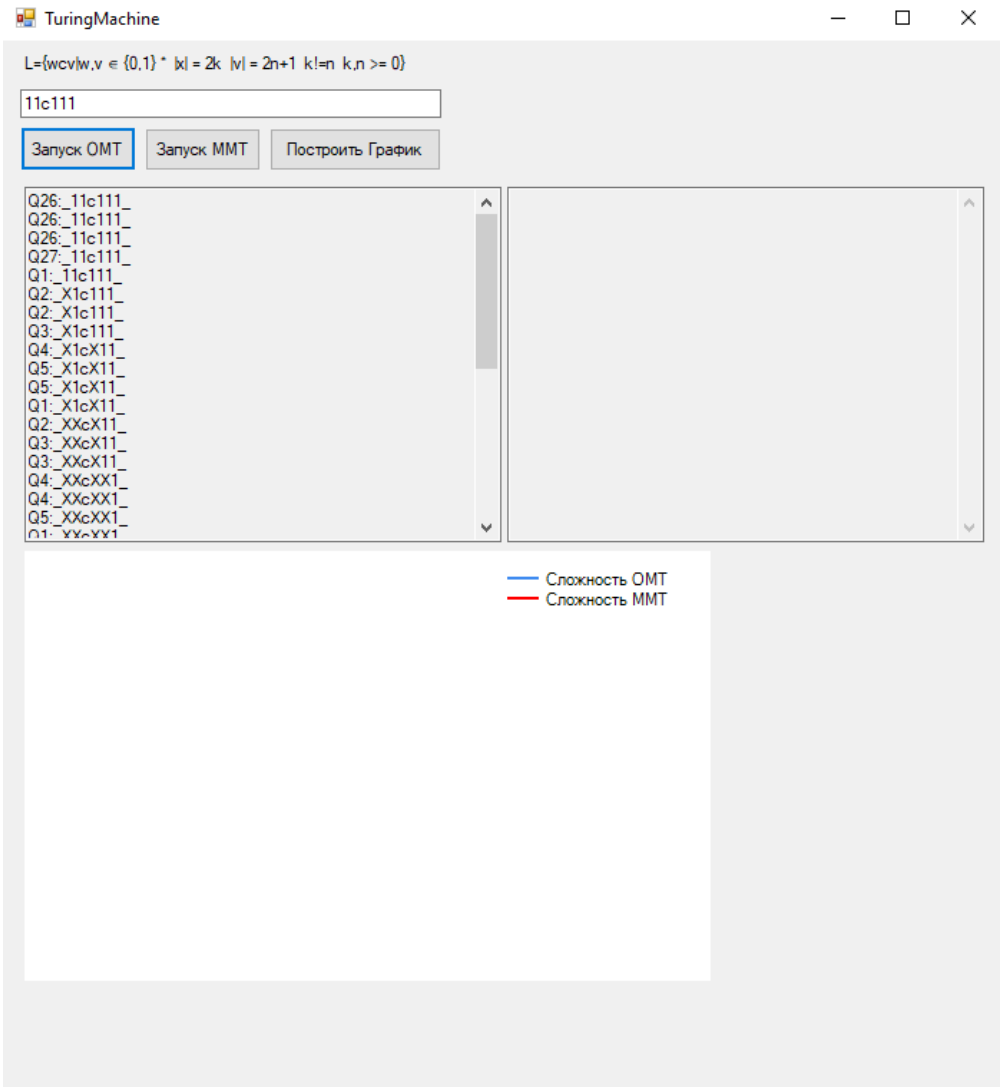

Рисунок 2.7 – Работа машины Тьюринга со словом 11c111

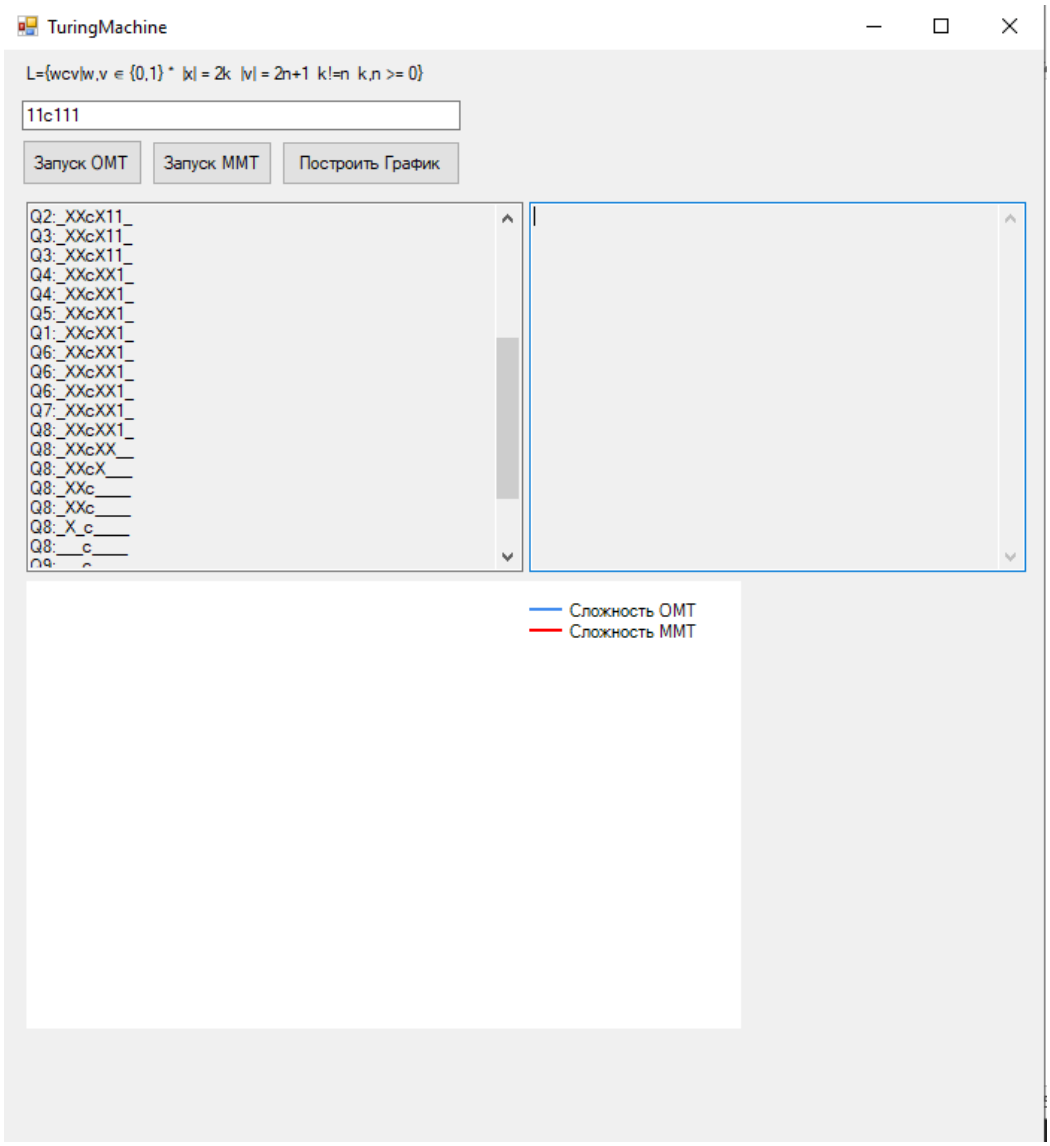

Рисунок 2.7, лист 2

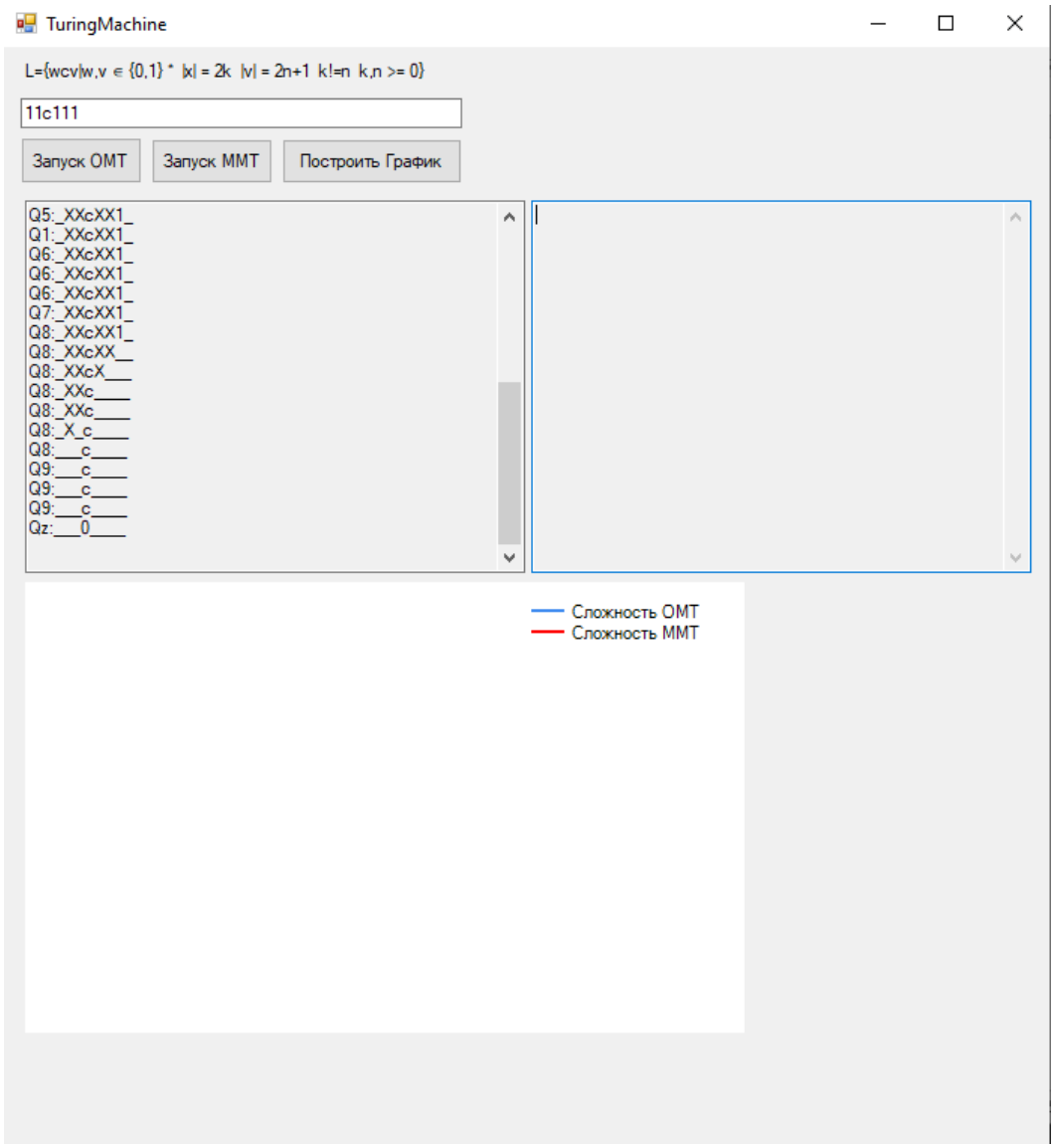

Рисунок 2.7, лист 3

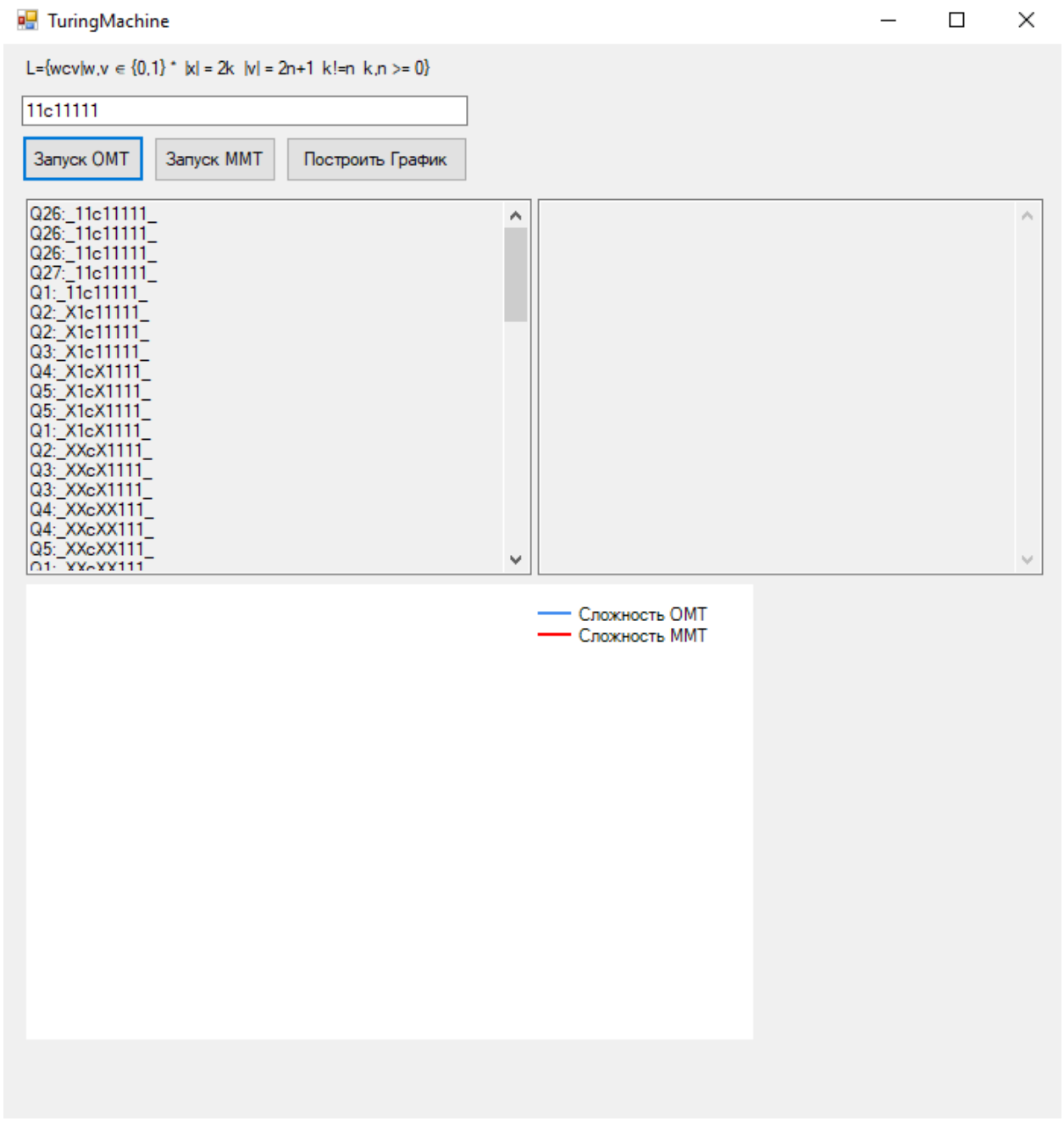

Рисунок 2.8 – Работа машины Тьюринга со словом 11c11111

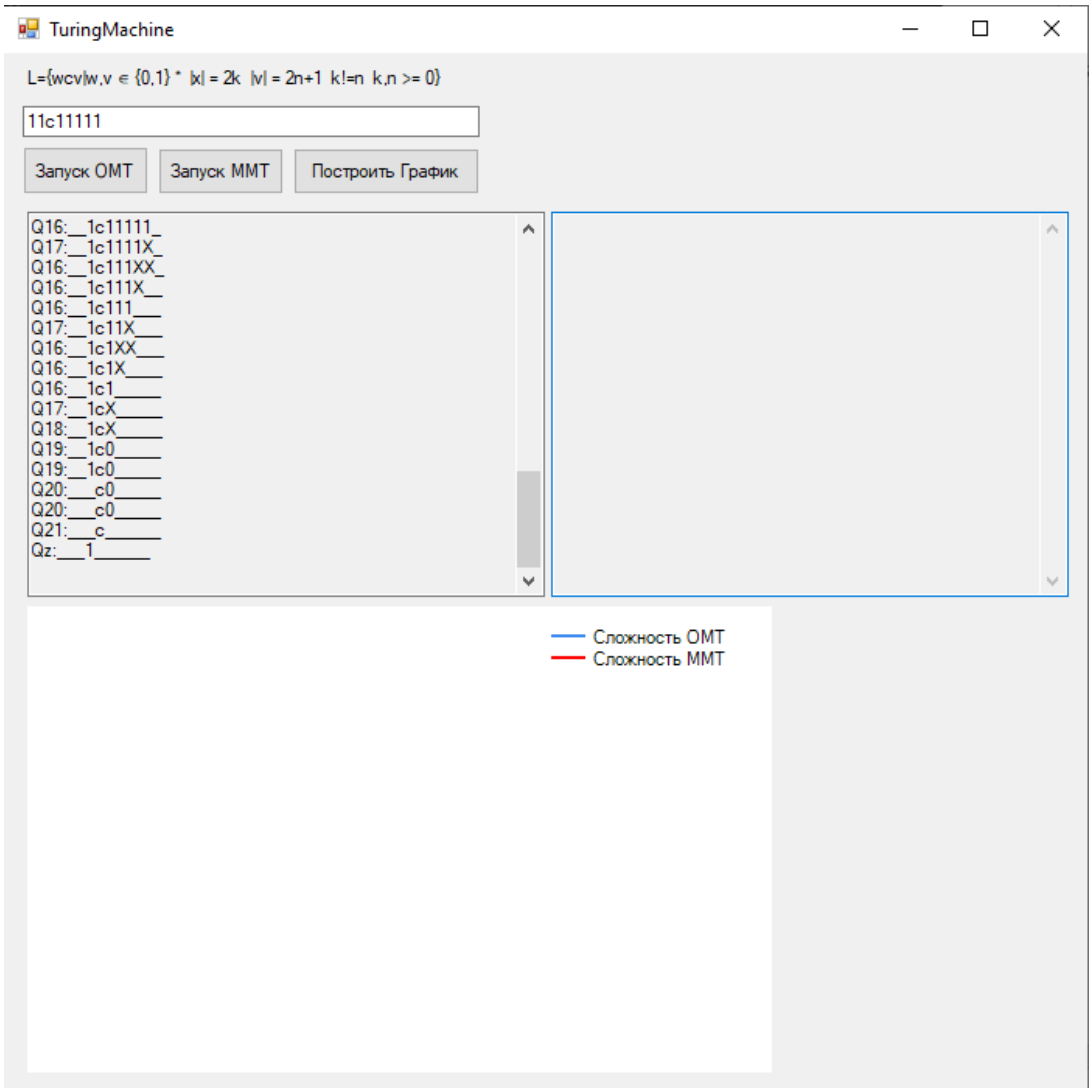

Рисунок 2.8, лист 2

Для двуленточной машины подадим на вход слова 11c111 (не принадлежит языку) (см. рис. 2.9) и 11c11111 (слово принадлежит языку) (см. рис. 2.10).

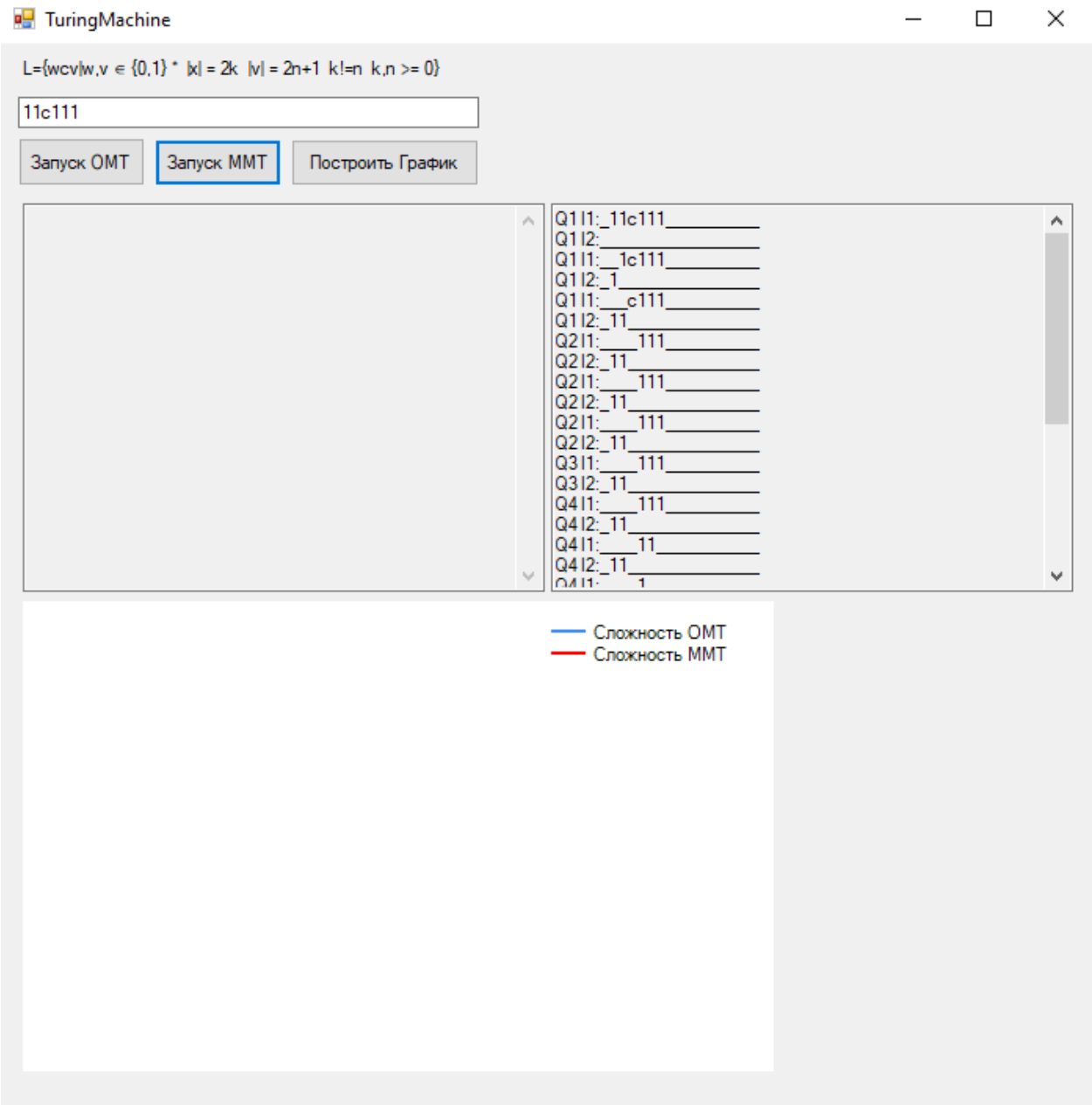

Рисунок 2.9 – Работа машины Тьюринга со словом 11c111

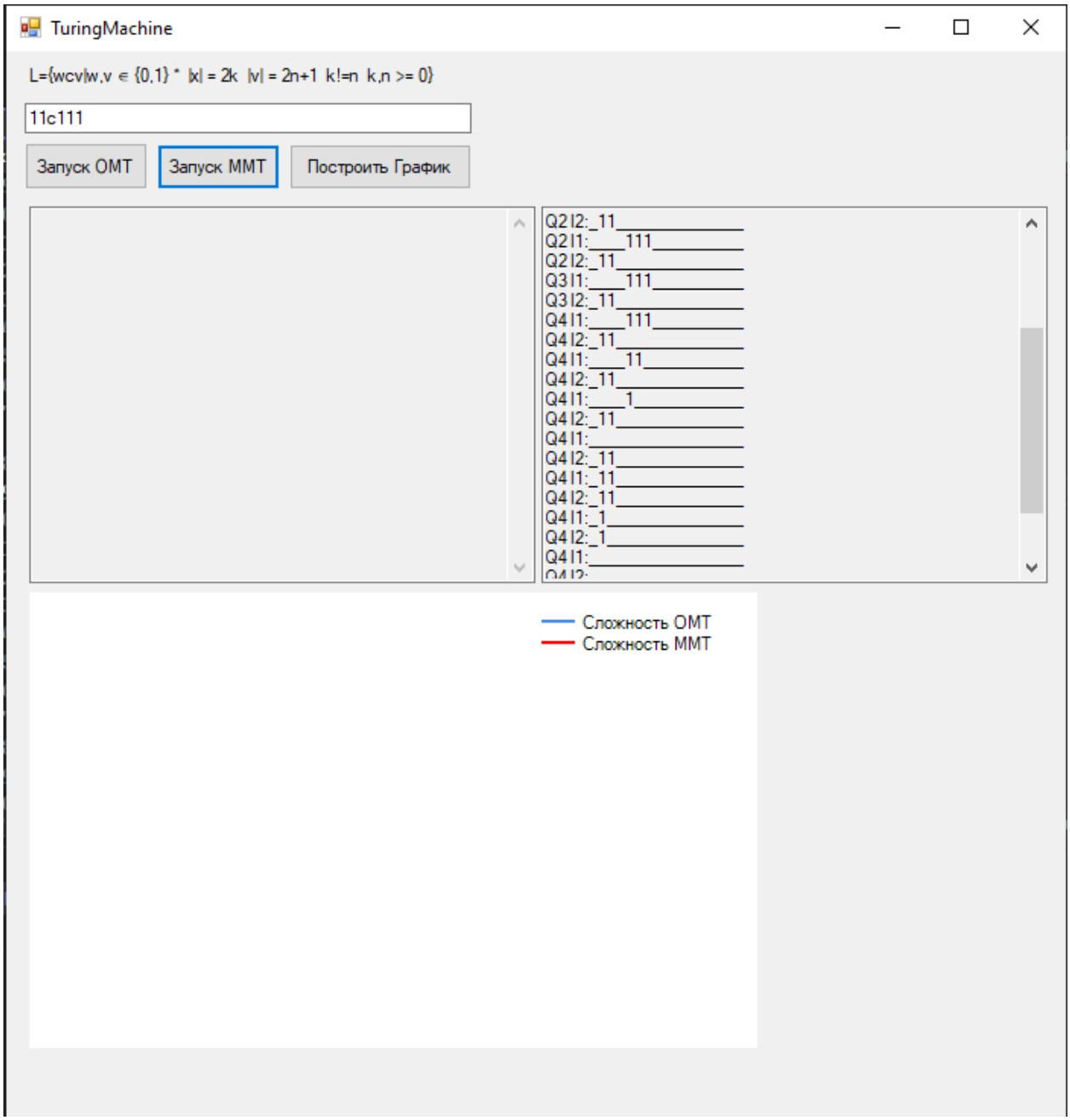

Рисунок 2.9, лист 2

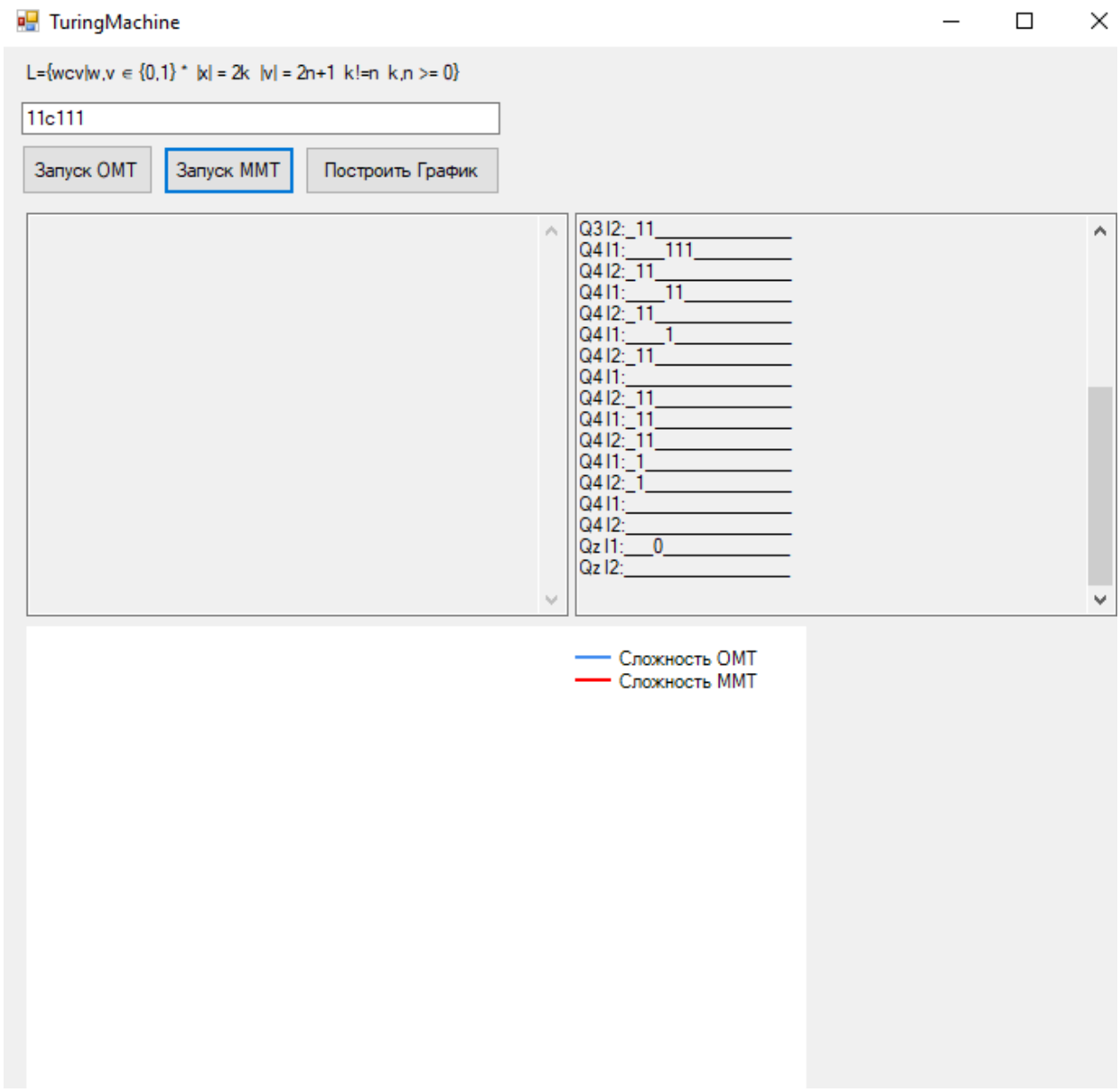

Рисунок 2.9, лист 3

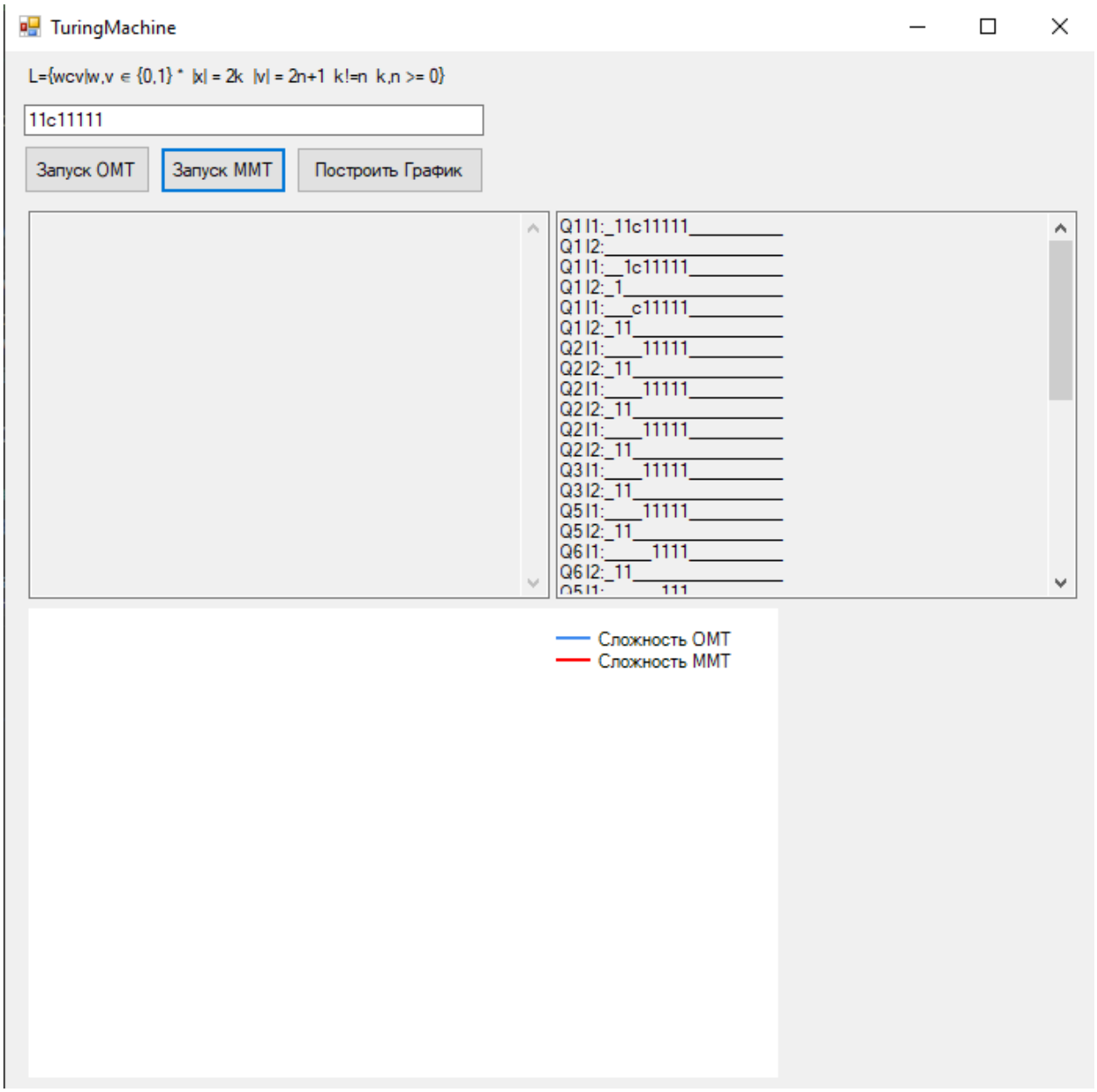

Рисунок 2.10 – Работа машины Тьюринга со словом 11c11111

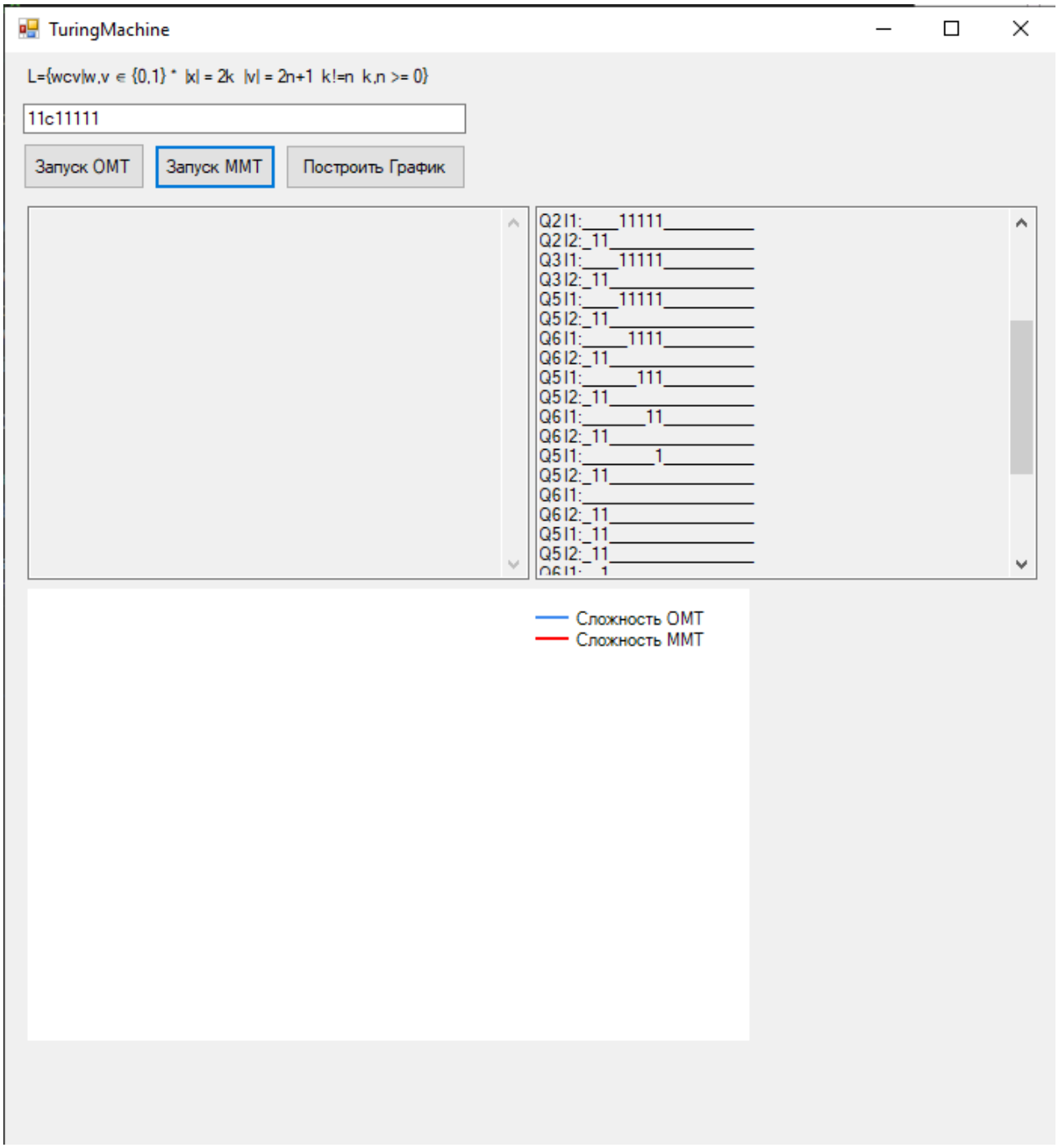

Рисунок 2.10, лист 2

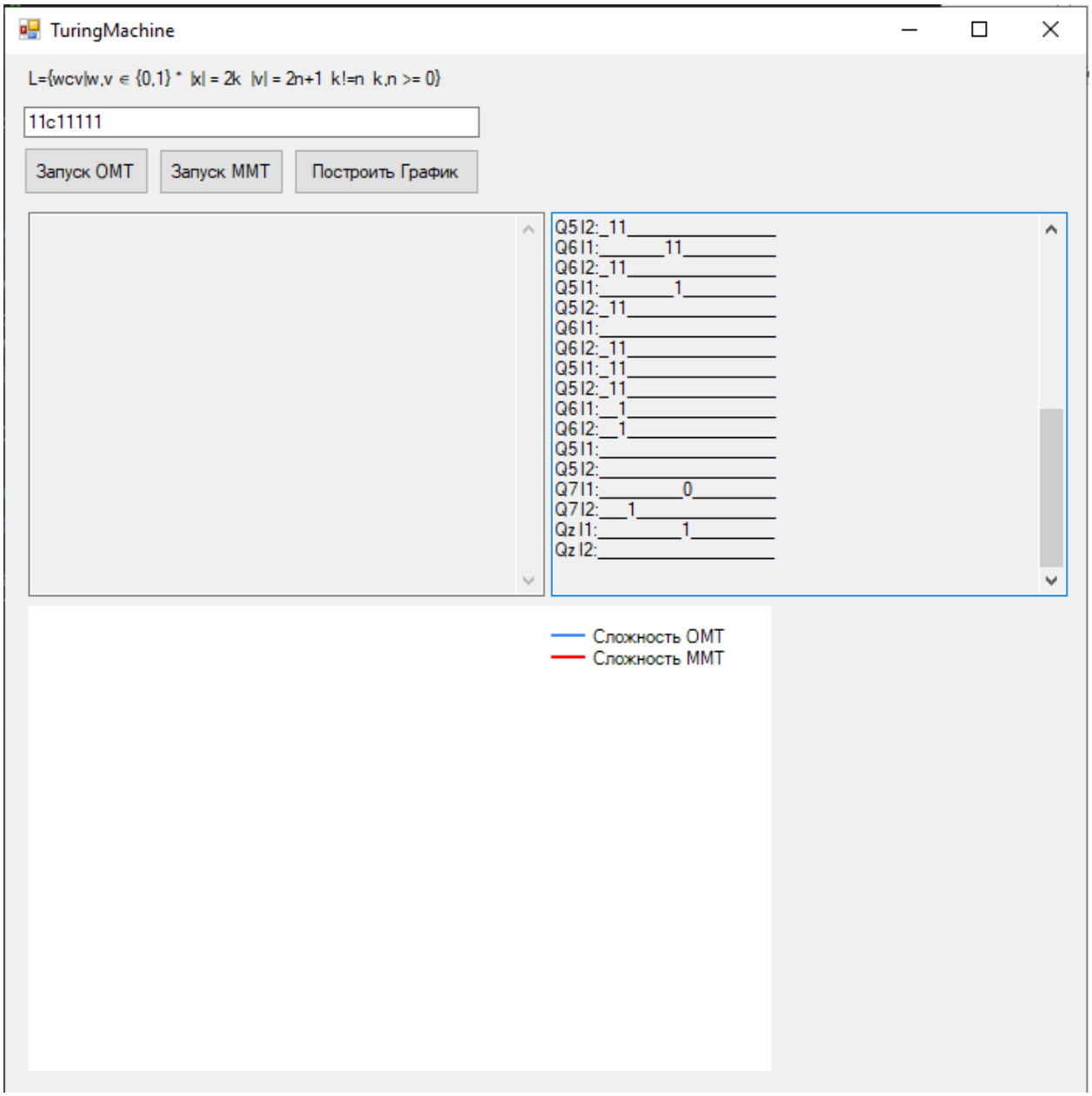

Рисунок 2.10, лист 3

2.5 Расчёт временной сложности (график функции временной сложности)

Временная сложность алгоритма – это зависимость времени выполнения алгоритма от объёма входных данных. В случае машины Тьюринга от количества символов во входном слове зависит количество тактов, за которое алгоритм завершит работу.

Функция для нахождения точек и отрисовки графиков реализованы в асинхронном методе CreateGraph. X – длина слова,Y – количество тактов.

Рассмотрим генерацию слов для машин Тьюринга (см. рис. 2.11).

```
void GenWords(int length, int pos)
{
    int count = 0;
    int numbers = 1 << length;


    for (int i = 0; i < numbers; i++)
    {

        string binary = Convert.ToString(i, 2);
        StringBuilder leading_zeroes = new StringBuilder((new string('0', length)).Substring(0, length - binary.Length));
        StringBuilder result = leading_zeroes;
        Console.WriteLine((result + binary).Insert(pos, "c"));
        listOfGenWords.Add((result + binary).Insert(pos, "c"));
        count++;
    }
    Console.WriteLine($"Количество сгенерированных чисел: {count}");
}
```

Рисунок 2.11 – Генерация слов для машин Тьюринга

Так как алфавит состоит из 1,0,c, разработана функция, которая вычисляет все возможные комбинации числа заданной длины. Суть метода заключается в генерации всех возможных слов длинной N и всеми возможными позициями “c”. Далее, список сгенерированных слов идёт на проход по функции ОМТ и ММТ, где мы выясняем максимальное количество тактов для каждого слова длиной N.

Вызываем CreateGraph, тем самым вызывая методы GenOMTWord и GenMMTWord,после чего из полученных выходных данных строим график сложности.

Функция работает, пока не вычислит максимальное число шагов для слова длинной 10 символов.

Таким образом на рисунке 2.12 показан график для одноленточной машины Тьюринга и для двуленточной машины Тьюринга.

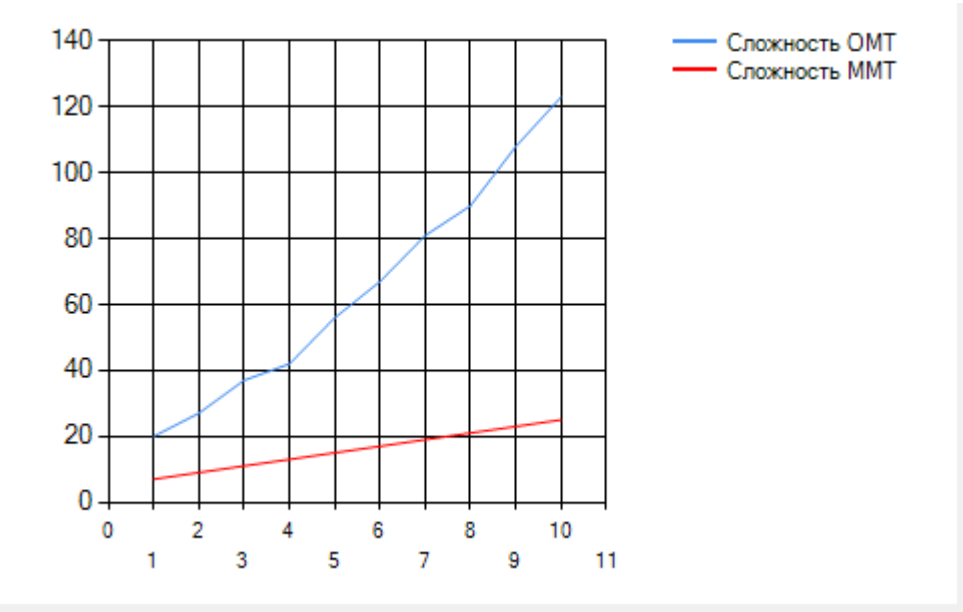

Рисунок 2.12 – График временной сложности для одноленточной и двуленточной машины Тьюринга

3 РАЗРАБОТКА АНАЛИТИЧЕСКОЙ МОДЕЛИ АЛГОРИТМА С ИСОЛЬЗОВАНИЕМ НОРМАЛЬНЫХ АЛГОРИТМОВ МАРКОВА

Нормальные алгоритмы Маркова – это упорядоченный набор элементарных операций над словами.

Марковская подстановка – это операция над словами α, β, γ, заключается в следующем. В исходном слове γ ищется самое левое вхождение слова α, если оно существует, α заменяется на β в слове γ. Полученное слово γ’ является результатом применения марковской подстановки к слову γ.

Рассмотрим особенности алгоритма на примере. Реализовать операцию копирование в алфавите {0,1}, то есть получить из слова *a* слово *a*a*.

Реализация алгоритма показана на рисунке 3.1.

```
1.  0@ -> 0!*
2.  1@ -> 1!*
3.  0! -> ?0a
4.  1! -> ?1b
5.  a0 -> 0a
6.  a1 -> 1a
7.  b0 -> 0b
8.  b1 -> 1b
9.  a* -> *a
10.b* -> *b
11.a -> 0
12.b -> 1
13.? -> !
14.@! -> _.
15.@X0->@0X
16.@X1->@1X
17.X0 -> 0X
18.X1 -> 1X
19.X -> @
20.1->@X1
```

Рисунок 3.1 – Реализация нормального алгоритма Маркова

Тестовые примеры работы описаны на рисунках 3.2 – 3.4.

10 -> @X10 -> @1X0 -> @10X ->@10@ -> @10!* -> @1?0a* -> @1?0 * a ->@1?0*0 ->@1!0 * 0 -> @?1b0 * 0 -> @?10b*0 -> @?10*b0 -> @!10 * 10 -> 10 * 10

Рисунок 3.2 – Тестовый пример работы алгоритма на слове 10

0 -> @X0 -> @0X ->@0@ -> @0!* -> @?0a* -> @?0*a->@!0*a->@!0*0->0 * 0

Рисунок 3.3 – Тестовый пример работы алгоритма на слове 0

10110-> @X10110->@1X0110->…->@10110X->@10110@ -> @10110!* -> @1011?0a*->@1011?0*a -> @1011?0*0->

@1011!0*0->@101?1b0*0->@101?10b*0 -> @101?10*b0->@101?10*10->

@101!10*10->@10?1b10*10->@10?11b0*10->@10?110b*10 -> @10?110*b10

->@10?110*110->@10!110*110->@1?0a110*110->@1?01a10*110->

@1?011a0*110->@1?0110a*110->@1?0110*a110->@1?0110*0110->

@1!0110*0110->@?1b0110*0110->@?10b110*0110->@?101b10*0110->

@?1011b0*0110->@?10110b*0110->@?10110*b0110->@?10110*10110->

@!10110*10110->10110*10110

Рисунок 3.4 – Тестовый пример работы алгоритма на слове 10110

4 ОПИСАНИЕ ФОРМАЛЬНОЙ МОДЕЛИ АЛГОРИТМА НА ОСНОВЕ РЕКУРСИВНЫХ ФУНКЦИЙ

Рекурсивный алгоритм – это алгоритм, который в качестве выполняемых операций может использовать обращение к самому себе.

При построении рекурсивных функций принят традиционный в теории алгоритмов конструктивный подход: задаётся «базис», то есть несколько простейших, очевидным образом вычисляемых функций, и способ построения из них остальных функций с помощью специальных операторов.

Опишем на конкретном примере формальную модель алгоритма на основе рекурсивных функций. Произведение нечетных цифр 16-ричной записи числа n. Формула 4.1 является решением данной задачи.

$$f(n) = \prod_{i=0}^{n} f_{mod10}(n\ div\ 10^{i}\ )^{sg[n\ div\ 10^{i}\ mod\ 2]}$$

Рисунок 4.1 – Рекурсивная формула

Рассмотрим тестовый пример работы рекурсивной функции при n = ED4 (см. рис. 4.2).

$$\begin{aligned} f(ED4) = {} & f_{mod10}(ED4\ div\ 10^{0}\ )^{sg[ED4\ div\ 10^{0}\ mod\ 2]} \\ & * f_{mod10}(ED4\ div\ 10^{1}\ )^{sg[ED4\ div\ 10^{1}\ mod\ 2]} \\ & * f_{mod10}(ED4\ div\ 10^{2}\ )^{sg[ED4\ div\ 10^{2}\ mod\ 2]} \\ & * f_{mod10}(ED4\ div\ 10^{3}\ )^{sg[ED4\ div\ 10^{3}\ mod\ 2]} * \ldots \\ & = 4^{0} * D^{1} * E^{0} * 0^{0} * \ldots = 1 * \mathrm{D} * 1 * 1 = \mathrm{D} \end{aligned}$$

Рисунок 4.2 – Пример работы рекурсивной функции

ВЫВОДЫ

За время выполнения курсовой работы закреплены навыки работы с такими формальными моделями алгоритмов, как машина Тьюринга, нормальные алгоритмы Маркова и рекурсивные функции. Рассмотрен принцип работы каждой модели на конкретных примерах.

Более подробно изучены особенности работы машины Тьюринга, а также оценка временной сложности алгоритмов.

В результате чего разработан программный продукт, имитирующий работу одноленточной и двуленточной машины Тьюринга, для языка L = L = $\{ wcv \mid w, v \in \{0, 1\}^* \& |w| = 2k \& |v| = 2n+1 \& k \neq n \& k, n \geq 0\}$. Результат работы программы – вывод «1» или «0» в зависимости от того принадлежит слово языку или нет соответственно.

Также в программе реализованы: вывод на экран каждого шага работы машины Тьюринга и сохранение протокола работы в текстовом файле.

Построение графика временной сложности происходит в отдельном потоке, при этом точки заносятся на график постепенно, по мере расчёта координат. Для их расчёта генерируется слово длиной n методом полного перебора, для каждого из них запускается машина Тьюринга, после чего получаем максимальное число тактов. Осуществляется построение до длины слов в 10 символов.

ПЕРЕЧЕНЬ ССЫЛОК

1. Поляков, В.И. Основы теории алгоритмов / В.И. Поляков, В.И. Скорубский. - СПб: СПб НИУ ИТМО, 2012. – 51 с.

2. Косовская Т.М. Машины Тьюринга. – URL: http://ipo.spb.ru/journal/content/626/МАШИНЫ%20ТЬЮРИНГА.pdf (Дата обращения: 07.12.22)

ПРИЛОЖЕНИЕ А
ТЕХНИЧЕКОЕ ЗАДАНИЕ

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР
ГОУВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра программной инженерии
им. Л.П. Фельдмана

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
К КУРСОВОЙ РАБОТЕ
НА ТЕМУ: «Построение аналитических моделей алгоритмов и оценка их сложности»

ПО КУРСУ: «Теория алгоритмов и формальных языков»

Выдано:
студенту группы ПИ-21в
Рустамов В.Р.

Руководитель:
Серёженко О.А.
Коломойцева И.А.
Щедрин С.В.

ДОНЕЦК – 2022

1 Основанием для разработки является задание на курсовую работу, выданное кафедрой программной инженерии.

67) $L = \{ wcv \mid w, v \in \{0, 1\}^* \& |w| = 2k \& |v| = 2n+1 \& k \neq n \& k, n \geq 0\}$

2 Целью разработки является создание программной модели машины Тьюринга, распознающий язык Л (L), расчет и экспериментальная проверка расчета временной сложности МТ.

3 Требования к программе:

– при проверке слова на принадлежность языку необходимо запретить ввод с клавиатуры символов не из входного алфавита заданного языка;

– при проверке слова на принадлежность языку выводить на экран каждый шаг работы машины Тьюринга;

– сохранять протокол работы машины Тьюринга в текстовом файле;

– при построении графика временной сложности работы машины Тьюринга значения для графика получить практически, с помощью созданной программной модели машины Тьюринга; для генерации слов длиной n использовать метод полного перебора;

– построение графика временной сложности работы МТ выполняется в отдельном потоке, значения на график заносятся по мере расчёта; построение графика останавливается по требованию пользователя.

4 Требования к программной документации:

пояснительная записка;

руководство пользователя.

5 Этапы разработки

Таблица А.1 – Этапы разработки курсовой работы

№ этапа	Наименование этапа	Срок выполнения
1	Выдача задания, составление ТЗ и его утверждение	2 недели
2	Построение формальных моделей алгоритмов.	2-4 недели
3	Определение структур данных для представления МТ, построение алгоритмов.	4-5 недели
4	Написание программы	5-9 недели
5	Отладка программы	9-12 недели

Продолжение таблицы А.1

6	Написание пояснительной записки	9-13 недели
7	Защита курсовой работы	13-14 недели

Приложение Б
Руководство пользователя

Программа предназначена для демонстрации работы машины Тьюринга на примере языка L = { wcv | w, v∈{0, 1}* & |w|=2k & |v|=2n+1 & k≠n & k, n≥0}Пользователь может ввести слово, проверить принадлежит ли оно языку, при этом данные выводятся в текстовое поле и сохраняются в файл, а также построить график временной сложности.

При запуске программы на форме текстовое поле для слова содержит слово «11c11111» (принадлежит языку). Пользователь может ввести слово, выбрать машину Тьюринга и запустить её, нажав кнопку «Запустить ОМТ». О результатах работы алгоритма можно узнать из вывода «1» или «0» в конце алгоритма, что указывает принадлежит ли введенное слово языку или нет.

Так как 2 машины не могут работать одновременно, некоторые кнопки становятся недоступными. Однако по завершению работы алгоритма кнопки возвращаются в прежнее состояние.

Чтобы построить график, пользователь должен нажать кнопку «Построить график». Для завершения – нужно подождать некоторое время пока график не будет достроен.

Построение графиков не зависит от работы машин Тьюринга. Поэтому работа машины и построение графика может осуществляться асинхронно.

Приложение В
Исходный код

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.CompilerServices;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TuringMachine
{
   public partial class StartButton : Form
   {
      List<string> outOMT = new List<string>();
      List<string> outMMT = new List<string>();
      List<string> listOfGenWords = new
List<string>();
      List<string> listOfOMTresult = new
List<string>();
      List<(string,string)> listOfMMTresult = new
List<(string,string)>();
      List<(int, int)> listGraphOMT = new List<(int,
int)>();
      List<(int, int)> listGraphMMT = new List<(int,
int)>();
      int countOMT = 0;
      int countMMT = 0;
      int maxOMT = 0;
      int maxMMT = 0;
      StringBuilder word;
      StringBuilder word2;
      bool locker = false;
      int indexTemp;
      List<int> indexesTemp = new List<int>();
      double a, b, h;
      double x, y;
      public StartButton()
      {
         InitializeComponent();
      }

      private void StartButton_Load(object sender,
EventArgs e)
      {

      }

      private void InputButton_TextChanged(object
sender, EventArgs e)
      {

      }

      private void button1_Click(object sender,
EventArgs e)
      {
         OutPutBox.Clear();
         Regex regex = new Regex(@"[0-1]?[c][0-
1]?");
         word = new
StringBuilder("_"+InputWordBox.Text+"_");
         //if (!regex.IsMatch(word.ToString()))
         //{
         //    MessageBox.Show("Неверный формат
строки!");
         //}
         //else
         //{
         //    StartSingleTapeMT(word);
         //}
         StartSingleTapeMT(word);
```

```
    }

    private void StartSingleTapeMT(StringBuilder
word)
    {
        outOMT.Clear();
        Qstart1(word, 1);
        outOMT.Add("Qz:" + word);
        for(int i = 0;i < outOMT.Count; i++)
        {
            OutPutBox.Text += outOMT[i] +
Environment.NewLine;
        }
        Thread.Sleep(2000);
        using (var sr = new
StreamWriter(File.Create("logOMT.txt")))
        {
            sr.WriteAsync(OutPutBox.Text);
        }

    }

    private void StartManyTapeMT(StringBuilder
word,StringBuilder word2)
    {
        outMMT.Clear();
        MQ1(word,word2,1, 1);
        if(locker)
            MQ7(word, word2, indexesTemp[0],
indexesTemp[1]);
        outMMT.Add("Qz l1:" + word);
        outMMT.Add("Qz l2:" + word2);
        for(int i = 0;i < outMMT.Count; i++)
        {
            OutPutManyBox.Text += outMMT[i] +
Environment.NewLine;
        }
        Thread.Sleep(2000);
        using (var sr = new
StreamWriter(File.Create("logMMT.txt")))
        {
            sr.WriteAsync(OutPutManyBox.Text);
        }


    }

    #region OMT
    private void Qstart1(StringBuilder word,int
index)
    {
        outOMT.Add("Q26:" + word);
        countOMT++;
        switch (word[index])
        {
            case '0':
                index++;
                Qstart1(word, index);
                break;
            case '1':
                index++;
                Qstart1(word, index);
                break;
            case 'c':
                index++;
                Qstart2(word, index);
                break;
            case '_':
                word.Clear();
                word = new StringBuilder(new string('_',
7));
                word[word.Length / 2] = '0';
                OutPutBox.Text += "Qz:" + word +
Environment.NewLine;
                return;

        }
    }

    private void Qstart2(StringBuilder word, int
index)
    {
        outOMT.Add("Q27:" + word);
        countOMT++;
```

```
            switch (word[index])
            {
                case '0':
                    index = 1;
                    Q1(word, index);
                    break;
                case '1':
                    index = 1;
                    Q1(word, index);
                    break;
                case 'c':
                    word.Clear();
                    word = new StringBuilder(new string('_',
7));
                    word[word.Length / 2] = '0';
                    OutPutBox.Text += "Qz:" + word +
Environment.NewLine;
                    return;
                case '_':
                    index=1;
                    Q1(word, index);
                    break;



            }
        }

        private void Qstart3(StringBuilder word, int
index)
        {
            outOMT.Add("Q28:" + word);
            countOMT++;
            switch (word[index])
            {
                case 'c':
                    index++;
                    Qstart4(word, index);
                    break;
                case '_':
                    word.Clear();
                    word = new StringBuilder(new string('_',
7));
                    word[word.Length / 2] = '0';
                    OutPutBox.Text += "Qz:" + word +
Environment.NewLine;
                    return;



            }
        }
        private void Qstart4(StringBuilder word, int
index)
        {
            outOMT.Add("Q29:" + word);
            countOMT++;
            switch (word[index])
            {
                case 'c':
                    word.Clear();
                    word = new StringBuilder(new string('_',
7));
                    word[word.Length / 2] = '0';
                    OutPutBox.Text += "Qz:" + word +
Environment.NewLine;
                    return;
                case '_':
                    index = 1;
                    Q1(word, index);
                    return;



            }
        }

        private void Q1(StringBuilder word,int index)
        {
            outOMT.Add("Q1:" + word);
            countOMT++;
            switch (word[index])
            {
                case '0':
                    word[index] = '#';
                    index++;
                    Q2(word, index);
```

```
            break;
        case '1':
            word[index] = 'X';
            index++;
            Q2(word, index);
            break;
        case 'c':
            index++;
            Q6(word, index);
            break;
        case '_':
            word[index] = '1';
            index++;
            Q2(word, index);
            break;
    }
}

private void Q2(StringBuilder word,int index)
{
    outOMT.Add("Q2:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            index++;
            Q2(word, index);
            break;
        case '1':
            index++;
            Q2(word, index);
            break;
        case 'c':
            index++;
            Q3(word, index);
            break;
    }
}

private void Q3(StringBuilder word, int index)
{
    outOMT.Add("Q3:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            word[index] = '#';
            index--;
            Q4(word, index);
            break;
        case '1':
            word[index] = 'X';
            index--;
            Q4(word, index);
            break;
        case 'X':
            index++;
            Q3(word, index);
            break;
        case '#':
            index++;
            Q3(word, index);
            break;
        case '_':
            index--;
            Q10(word, index);
            break;
    }
}

private void Q4(StringBuilder word, int index)
{
    outOMT.Add("Q4:" + word);
    countOMT++;
    switch (word[index])
    {
        case 'c':
            index--;
            Q5(word, index);
            break;
        case 'X':
            index--;
            Q4(word, index);
            break;
```

```
        case '#':
            index--;
            Q4(word, index);
            break;
    }
}

private void Q5(StringBuilder word, int index)
{
    outOMT.Add("Q5:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            index--;
            Q5(word, index);
            break;
        case '1':
            index--;
            Q5(word, index);
            break;
        case 'X':
            index++;
            Q1(word, index);
            break;
        case '#':
            index++;
            Q1(word, index);
            break;
        case '_':
            index--;
            Q25(word, index);
            break;
    }
}

private void Q6(StringBuilder word, int index)
{
    outOMT.Add("Q6:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            index++;
            Q7(word, index);
            break;
        case '1':
            index++;
            Q7(word, index);
            break;
        case 'X':
            index++;
            Q6(word, index);
            break;
        case '#':
            index++;
            Q6(word, index);
            break;
        case '_':
            Q7(word, index);
            break;
    }
}

private void Q7(StringBuilder word, int index)
{
    outOMT.Add("Q7:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            index--;
            Q10(word, index);
            break;
        case '1':
            index--;
            Q10(word, index);
            break;
        case '_':
            index--;
            Q8(word, index);
            break;
    }
}
```

```
private void Q8(StringBuilder word, int index)
{
   outOMT.Add("Q8:" + word);
   countOMT++;
   switch (word[index])
   {
      case '0':
         word[index] = '_';
         index--;
         Q8(word, index);
         break;
      case '1':
         word[index] = '_';
         index--;
         Q8(word, index);
         break;
      case 'c':
         index--;
         Q8(word, index);
         break;
      case 'X':
         word[index] = '_';
         index--;
         Q8(word, index);
         break;
      case '#':
         word[index] = '_';
         index--;
         Q8(word, index);
         break;
      case '_':
         index++;
         Q9(word, index);
         break;
   }
}

private void Q9(StringBuilder word, int index)
{
   outOMT.Add("Q9:" + word);
   countOMT++;
   switch (word[index])
   {
      case 'c':
         word[index] = '0';
         return;
      case '_':
         index++;
         Q9(word, index);
         break;
   }
}

private void Q10(StringBuilder word, int index)
{
   outOMT.Add("Q10:" + word);
   countOMT++;
   switch (word[index])
   {
      case '0':
         index--;
         Q10(word, index);
         break;
      case '1':
         index--;
         Q10(word, index);
         break;
      case 'c':
         index--;
         Q10(word, index);
         break;
      case 'X':
         word[index] = '1';
         index--;
         Q10(word, index);
         break;
      case '#':
         word[index] = '0';
         index--;
         Q10(word, index);
         break;
      case '_':
         index++;
```

```
            Q11(word, index);
            break;
    }
}

private void Q11(StringBuilder word, int index)
{
    outOMT.Add("Q11:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            word[index] = 'X';
            index++;
            Q12(word, index);
            break;
        case '1':
            word[index] = 'X';
            index++;
            Q12(word, index);
            break;
        case 'c':
            index--;
            Q13(word, index);
            break;
        case 'X':
            word[index] = '_';
            index++;
            Q11(word, index);
            break;
        case '_':
            word[index] = '1';
            index++;
            Q14(word, index);
            break;
    }
}

private void Q12(StringBuilder word, int index)
{
    outOMT.Add("Q12:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            word[index] = 'X';
            index--;
            Q11(word, index);
            break;
        case '1':
            word[index] = 'X';
            index--;
            Q11(word, index);
            break;
        case 'c':
            index--;
            Q13(word, index);
            break;
    }
}

private void Q13(StringBuilder word, int index)
{
    outOMT.Add("Q13:" + word);
    countOMT++;
    switch (word[index])
    {
        case 'X':
            word[index] = '0';
            index++;
            Q14(word, index);
            break;
        case '_':
            word[index] = '1';
            index++;
            Q14(word, index);
            break;
    }
}

private void Q14(StringBuilder word, int index)
{
    outOMT.Add("Q14:" + word);
    countOMT++;
```

```
    switch (word[index])
    {
        case 'c':
            index++;
            Q15(word, index);
            break;
    }
}

private void Q15(StringBuilder word, int index)
{
    outOMT.Add("Q15:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            index++;
            Q15(word, index);
            break;
        case '1':
            index++;
            Q15(word, index);
            break;
        case '_':
            index--;
            Q16(word, index);
            break;
    }
}

private void Q16(StringBuilder word, int index)
{
    outOMT.Add("Q16:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            word[index] = 'X';
            index--;
            Q17(word, index);
            break;
        case '1':
            word[index] = 'X';
            index--;
            Q17(word, index);
            break;
        case 'c':
            index++;
            Q18(word, index);
            break;
        case 'X':
            word[index] = '_';
            index--;
            Q16(word, index);
            break;
    }
}

private void Q17(StringBuilder word, int index)
{
    outOMT.Add("Q17:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            word[index] = 'X';
            index++;
            Q16(word, index);
            break;
        case '1':
            word[index] = 'X';
            index++;
            Q16(word, index);
            break;
        case 'c':
            index++;
            Q18(word, index);
            break;
    }
}

private void Q18(StringBuilder word, int index)
{
    outOMT.Add("Q18:" + word);
```

```
    countOMT++;
    switch (word[index])
    {
        case 'X':
            word[index] = '0';
            index--;
            Q19(word, index);
            break;
        case '_':
            word[index] = '1';
            index--;
            Q19(word, index);
            break;
    }
}

private void Q19(StringBuilder word, int index)
{
    outOMT.Add("Q19:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            word[index] = '_';
            index++;
            Q22(word, index);
            break;
        case '1':
            word[index] = '_';
            index++;
            Q20(word, index);
            break;
        case 'c':
            index--;
            Q19(word, index);
            break;
    }
}

private void Q20(StringBuilder word, int index)
{
    outOMT.Add("Q20:" + word);
    countOMT++;
    switch (word[index])
    {
        case '0':
            word[index] = '_';
            index--;
            Q21(word, index);
            break;
        case '1':
            word[index] = '_';
            index--;
            Q24(word, index);
            break;
        case 'c':
            index++;
            Q20(word, index);
            break;
    }
}

private void Q21(StringBuilder word, int index)
{
    outOMT.Add("Q21:" + word);
    countOMT++;
    switch (word[index])
    {
        case 'c':
            word[index] = '1';
            return;
    }
}

private void Q22(StringBuilder word, int index)
{
    outOMT.Add("Q22:" + word);
    switch (word[index])
    {
        case 'c':
            index++;
            Q23(word, index);
            break;
    }
```

```
}

private void Q23(StringBuilder word, int index)
{
   outOMT.Add("Q23:" + word);
   countOMT++;
   switch (word[index])
   {
      case '0':
         word[index] = '_';
         index--;
         Q24(word, index);
         break;
      case '1':
         word[index] = '_';
         index--;
         Q24(word, index);
         break;
   }
}

private void Q24(StringBuilder word, int index)
{
   outOMT.Add("Q24:" + word);
   countOMT++;
   switch (word[index])
   {
      case 'c':
         word[index] = '0';
         return;
   }
}

private void Q25(StringBuilder word, int index)
{
   outOMT.Add("Q25:" + word);
   countOMT++;
   switch (word[index])
   {
      case '1':
         index++;
         Q25(word, index);
         break;
      case 'c':
         index--;
         Q24(word, index);
         break;
      case 'X':
         index--;
         Q17(word, index);
         break;
   }
}
#endregion

#region MMT

//private void MQstart1(StringBuilder word,
StringBuilder word2, int index1, int index2)
//{
//   OutPutManyBox.Text += "Q8 l1:" + word +
Environment.NewLine;
//   OutPutManyBox.Text += "Q8 l2:" + word2
+ Environment.NewLine;
//   switch (word[index1])
//   {
//      case '0':
//         index1++;
//         MQstart1(word, word2, index1,
index2);
//         break;
//      case '1':
//         index1++;
//         MQstart1(word, word2, index1,
index2);
//         break;
//      case 'c':
//         index1++;
//         MQstart2(word, word2, index1,
index2);
//         break;
//      case '_':
//         word.Clear();
```

```
        //          word = new StringBuilder(new
string('_', 7));
        //          word[word.Length / 2] = '0';
        //          OutPutManyBox.Text += "Qz:" + word
+ Environment.NewLine;
        //          return;


        //    }
        //}


        //private void MQstart2(StringBuilder word,
StringBuilder word2, int index1, int index2)
        //{
        //    OutPutManyBox.Text += "Q9 l1:" + word +
Environment.NewLine;
        //    OutPutManyBox.Text += "Q9 l2:" + word2
+ Environment.NewLine;
        //    switch (word[index1])
        //    {
        //        case '0':
        //            index1 = 1;
        //            MQ1(word, word2, index1, index2);
        //            break;
        //        case '1':
        //            index1 = 1;
        //            MQ1(word, word2, index1, index2);
        //            break;
        //        case 'c':
        //            word.Clear();
        //            word = new StringBuilder(new
string('_', 15));
        //            word[word.Length / 2] = '0';
        //            OutPutManyBox.Text += "Qz l1:" +
word + Environment.NewLine;
        //            return;
        //        case '_':
        //            index1 = 1;
        //            MQ1(word, word2, index1, index2);
        //            break;



        //    }
        //}
        private void MQ1(StringBuilder
word,StringBuilder word2,int index1,int index2)
        {
            outMMT.Add("Q1 l1:" + word);
            outMMT.Add("Q1 l2:" + word2);
            countMMT++;
            switch (word[index1])
            {
                case '0':
                    word2[index2] = word[index1];
                    word[index1] = '_';
                    index1++;
                    index2++;
                    MQ1(word, word2, index1, index2);
                    break;
                case '1':
                    word2[index2] = word[index1];
                    word[index1] = '_';
                    index1++;
                    index2++;
                    MQ1(word, word2, index1, index2);
                    break;
                case 'c':
                    word[index1] = '_';
                    index1++;
                    indexTemp = index1;
                    index2 = 1;
                    MQ2(word, word2, index1, index2);
                    break;
            }
        }

        private void MQ2(StringBuilder word,
StringBuilder word2, int index1, int index2)
        {
            outMMT.Add("Q2 l1:" + word);
            outMMT.Add("Q2 l2:" + word2);
            countMMT++;
            if ((word[index1] == '1' || word[index1] ==
'0') && (word2[index2] == '1' || word2[index2] ==
'0'))
```

```
        {
            index1++;
            index2++;
            MQ2(word, word2, index1, index2);
        }
        else if (word2[index2] == '_' &&
(word[index1] == '1' || word[index1] == '0'))
        {
            index1++;
            MQ3(word, word2, index1, index2);
        }
        else
        {
            index1 = indexTemp;
            index2 = 1;
            locker = true;
            MQ5(word, index1, 1);
            MQ5(word2, index2, 2);
        }
    }

    private void MQ3(StringBuilder
word,StringBuilder word2,int index1,int index2)
    {
        outMMT.Add("Q3 l1:" + word);
        outMMT.Add("Q3 l2:" + word2);
        countMMT++;
        switch (word[index1])
        {
            case '0':
                index1 = indexTemp;
                index2 = 1;
                locker = true;
                MQ5(word, index1,1);
                MQ5(word2, index2,2);
                break;
            case '1':
                index1 = indexTemp;
                index2 = 1;
                locker = true;
                MQ5(word, index1,1);
                MQ5(word2, index2,2);
                break;
            case '_':
                index1--;
                index2--;
                MQ4(word, index1,1);
                MQ4(word2, index2,2);
                break;
        }
    }

    private void MQ4(StringBuilder word, int
index,int l)
    {
        outMMT.Add("Q4 l1:" + word);
        outMMT.Add("Q4 l2:" + word2);
        countMMT++;
        switch (word[index])
        {
            case '0':
                word[index] = '_';
                index--;
                MQ4(word, index,l);
                break;
            case '1':
                word[index] = '_';
                index--;
                MQ4(word, index,l);
                break;
            case '_':
                word[index] = '0';
                word2[index] = '_';
                return;
        }
    }

    private void MQ5(StringBuilder word, int
index,int l)
    {
        outMMT.Add("Q5 l1:" + word);
        outMMT.Add("Q5 l2:" + word2);
        countMMT++;
        switch (word[index])
```

```
        {
            case '0':
                word[index] = '_';
                index++;
                MQ6(word, index,l);
                break;
            case '1':
                word[index] = '_';
                index++;
                MQ6(word, index,l);
                break;
            case '_':
                word[index] = '1';
                indexesTemp.Add(index);
                return;
        }
    }

    private void MQ6(StringBuilder word, int
index,int l)
    {
        outMMT.Add("Q6 l1:" + word);
        outMMT.Add("Q6 l2:" + word2);
        countMMT++;
        switch (word[index])
        {
            case '0':
                word[index] = '_';
                index++;
                MQ5(word, index, l);
                break;
            case '1':
                word[index] = '_';
                index++;
                MQ5(word, index, l);
                break;
            case '_':
                word[index] = '0';
                indexesTemp.Add(index);
                return;
        }
    }

    private void MQ7(StringBuilder word,
StringBuilder word2, int index1, int index2)
    {
        outMMT.Add("Q7 l1:" + word);
        outMMT.Add("Q7 l2:" + word2);
        countMMT++;
        if (word.ToString().Contains('0') &&
word2.ToString().Contains('1'))
        {
            word.Replace('0', '1');
            word2.Replace('1', '_');
        }
        else
        {
            int size = word.Length;
            word.Replace('0', '_').Replace('1','_');
            word2.Replace('0', '_').Replace('1', '_');
            word[index1] = '0';
            word2[index2] = '_';
        }

    }

    #endregion

    #region Генерация



    void GenAllWords(int length)
    {
        for (int i = 0; i <= length; i++)
        {
            GenWords(length, i);
        }
    }
```

```
        void GenWords(int length, int pos)
        {
            int count = 0;
            int numbers = 1 << length;



            for (int i = 0; i < numbers; i++)
            {

                string binary = Convert.ToString(i, 2);
                StringBuilder leading_zeroes = new
StringBuilder((new string('0', length)).Substring(0,
length - binary.Length));
                StringBuilder result = leading_zeroes;
                Console.WriteLine((result +
binary).Insert(pos, "c"));
                listOfGenWords.Add((result +
binary).Insert(pos, "c"));
                count++;
            }
            Console.WriteLine($"Количество
сгенерированных чисел: {count}");
        }
        #endregion

        private void OutPutBox_TextChanged(object
sender, EventArgs e)
        {

        }

        private void button2_Click(object sender,
EventArgs e)
        {
            OutPutManyBox.Clear();
            Regex regex = new Regex(@"[0-1]?[c][0-
1]?");
            word = new StringBuilder("_" +
InputWordBox.Text + new string('_',10));

            int size = word.Length;
            word2 = new StringBuilder(new
string('_',size));
            if (word.ToString().Contains("cc") ||
!word.ToString().Contains("c"))
            {
                OutPutManyBox.Text += "Q1 l1:" + word
+ Environment.NewLine;
                OutPutManyBox.Text += "Q1 l2:" + word2
+ Environment.NewLine;
                word.Clear();
                word = new StringBuilder(new string('_',
15));
                word[word.Length / 2] = '0';
                OutPutManyBox.Text += "Qz l1:" + word
+ Environment.NewLine;
                OutPutManyBox.Text += "Qz l2:" + word2
+ Environment.NewLine;
                return;
            }
            if (!regex.IsMatch(word.ToString()))
            {
                MessageBox.Show("Неверный формат
строки!");
            }
            else
            {
                StartManyTapeMT(word,word2);
            }
        }

        private void
StartGenSingleTapeMT(StringBuilder word)
        {
            countOMT = 0;
            Qstart1(word, 1);



            listOfOMTresult.Add(word.ToString());
            if (countOMT > maxOMT)
                maxOMT = countOMT;
        }
```

```
        private void
StartGenManyTapeMT(StringBuilder word,
StringBuilder word2)
        {
            countMMT = 0;
            MQ1(word, word2, 1, 1);
            if (locker)
                MQ7(word, word2, indexesTemp[0],
indexesTemp[1]);



            listOfMMTresult.Add((word.ToString(),
word2.ToString()));
            if (countMMT > maxMMT)
                maxMMT = countMMT;
        }

        private async void CreateGraph_Click(object
sender, EventArgs e)
        {
            await Task.Run(() => GenOMTWord());
            Thread.Sleep(1500);
            await Task.Run(() => GenMMTWord());




            int i = 0;

            this.DiffGraph.Series[0].Points.Clear();
            this.DiffGraph.Series[1].Points.Clear();

            while (i != 10)
            {

this.DiffGraph.Series[0].Points.AddXY(listGraphOM
T[i].Item1, listGraphOMT[i].Item2);

this.DiffGraph.Series[1].Points.AddXY(listGraphM
MT[i].Item1, listGraphMMT[i].Item2);
                i++;
            }
        }

        public void GenOMTWord()
        {
            StringBuilder temp = new StringBuilder(' ');
            for(int j = 1;j <= 10; j++)
            {
                GenAllWords(j);
                temp = new StringBuilder(' ');
                for (int i = 0; i < listOfGenWords.Count;
i++)
                {
                    temp.Clear();
                    //int size = listOfGenWords[i].Length;
                    temp = new StringBuilder("_" +
listOfGenWords[i] + "_");
                    StartGenSingleTapeMT(temp);
                }
                listOfOMTresult.Clear();
                listOfGenWords.Clear();
                listGraphOMT.Add((j, maxOMT));
                maxOMT = 0;
            }

        }

        public void GenMMTWord()
        {
            for (int j = 1; j <= 10; j++)
            {
                GenAllWords(j);
                StringBuilder temp = new StringBuilder('
');
                for (int i = 0; i < listOfGenWords.Count;
i++)
                {
                    temp.Clear();
                    int size = listOfGenWords[i].Length;
                    word2 = new StringBuilder(new
string('_', size + 2));
```

```
                temp = new StringBuilder("_" +
listOfGenWords[i] + "_");
                StartGenManyTapeMT(temp, word2);
            }
            listOfMMTresult.Clear();
            listOfGenWords.Clear();
            listGraphMMT.Add((j, maxMMT));
            maxMMT = 0;
        }

    }
  }
}
```