##### Санкт-Петербургский Политехнический Университет Петра Великого Институт компьютерных наук и технологий

##### Высшая школа программной инженерии

Изображение выглядит как текст

Автоматически созданное описание

КУРСОВАЯ РАБОТА

**Использование бинарных решающих диаграмм для решения логических задач.**

по дисциплине «Математическая логика»

|  |  |  |
| --- | --- | --- |
| Выполнил |  |  |
| Студент гр. 3530904/00103 |  | Бабушкин Е.А \_\_\_\_\_ |
| Руководитель |  |  |
| к.т.н. |  | Шошмина И.В. |

Санкт-Петербург  
2022 г.

**Оглавление**

[**Введение. Программирование в ограничениях.** 2](#_Toc121463934)

[**Постановка задачи.** 3](#_Toc121463935)

[**Решение задачи.** 6](#_Toc121463936)

[**Интерпретация задачи.** 9](#_Toc121463937)

[**Задача без склейки** 11](#_Toc121463938)

[**Заключение.** 12](#_Toc121463939)

[**Список литературы.** 13](#_Toc121463940)

[**Приложение А. Исходный код с учетом склейки.** 14](#_Toc121463941)

[**Приложение Б. Ограничения 3 типа без учета склейки.** 21](#_Toc121463942)

# **Введение. Программирование в ограничениях.**

Современное программирование уже давно не похоже на то, что было изначально при зарождении данной отрасли. На сегодняшний день существует огромное количество языков, ответвлений и парадигм программирования, каждое из которых решает свои уникальные задачи. Одной из таких парадигм стало программирование в ограничениях. Данное программирование берет свое начало и может быть выражено в форме программирования логики ограничений, которое встраивает ограничения в логическую программу. Это такой вид программирования, который ориентирован на решение комбинаторных задач, благодаря огромной вариативности методов, таких как: искусственный интеллект, исследований операций, методов информатики и многое другое. В данном типе программирования пользователь устанавливает ограничения на все множество решений поставленной задачи, тем самым сужая область решения.

В данной курсовой работе для решения поставленной задачи, представляющую собой вариант интерпретации задачи Эйнштейна, будет использованы бинарные диаграммы решений (BDD) [4,5]. BDD – это, предложенная Рэндолом Брайантом в 1961 году, форма представления булевой функции в виде ориентированного ациклического графа, состоящий из узлов решений и их потомков. Для реализации бинарных диаграмм в нашем решении задачи мы воспользовались свободно распространяющейся библиотекой BuDDy [2], написанной на языке программирования C++.

# **Постановка задачи.**

Имеется 9 объектов и каждый элемент представляет собой позицию в матрице 3 на 3. “Соседские” отношения между объектами определяются по варианту №21, склейка по левой-правой границам.

|  |  |  |
| --- | --- | --- |
| 0 |  |  |
| \* |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | 0 |
|  |  | \* |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | 0 |  |
|  | \* |  |
|  |  |  |

Таким образом, для объекта, помеченного как “0”, соседями являются объекты, помеченные “\*”. Все варианты соседских отношений с учетом склейки представлены с 1 по 9 таблицу.

|  |  |  |
| --- | --- | --- |
|  |  | \* |
| 0 |  |  |
| \* |  |  |

Таблица 1.   
Вариант соседских отношений

Таблица 2.   
Вариант соседских отношений

Таблица 3.   
Вариант соседских отношений

|  |  |  |
| --- | --- | --- |
| \* |  |  |
|  | 0 |  |
|  | \* |  |

|  |  |  |
| --- | --- | --- |
|  | \* |  |
|  |  | 0 |
|  |  | \* |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | \* |
| 0 |  |  |

Таблица 4.   
Вариант соседских отношений

Таблица 5.   
Вариант соседских отношений

Таблица 6.   
Вариант соседских отношений

отношений

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | \* |  |
|  |  | 0 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| \* |  |  |
|  | 0 |  |

Таблица 7.   
Вариант соседских отношений

Таблица 8.   
Вариант соседских отношений

Таблица 9.   
Вариант соседских отношений

|  |  |  |
| --- | --- | --- |
| 0 |  |  |
| \* |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  |  | 0 |
|  |  | \* |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | 0 |  |
|  | \* |  |
|  |  |  |

Все варианты соседских отношений без учета склейки представлены с 10 по 18 таблицу.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 0 |  |  |
| \* |  |  |

Таблица 10.   
Вариант соседских отношений

Таблица 11.   
Вариант соседских отношений

Таблица 12.   
Вариант соседских отношений

|  |  |  |
| --- | --- | --- |
| \* |  |  |
|  | 0 |  |
|  | \* |  |

|  |  |  |
| --- | --- | --- |
|  | \* |  |
|  |  | 0 |
|  |  | \* |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
| 0 |  |  |

Таблица 13.   
Вариант соседских отношений

Таблица 14.   
Вариант соседских отношений

Таблица 15.   
Вариант соседских отношений

отношений

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | \* |  |
|  |  | 0 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| \* |  |  |
|  | 0 |  |

Таблица 16.   
Вариант соседских отношений

Таблица 17.   
Вариант соседских отношений

Таблица 18.   
Вариант соседских отношений

Имеется 4 свойства, принимающих 9 различных значений [1,3]. Для получения решения требуется задать:

* 9 ограничения типа 1: устанавливает свойству k1 конкретного объекта i1 значение j1.  
  ***F: = F*** ∧ ***p (k1, i1, j1).***
* 6 ограничения типа 2: устанавливает соответствие между двумя свойствами какого-либо объекта.  
   ***F: = F***∧ ***(p (k1, i, j1)⬄ p (k2, i, j2))***
* 3 ограничения типа 3: ограничение расположения объектов – расположение в соответствии с таблицами 1–9. Это ограничения типа объект, у которого свойство k1 имеет значение j1, расположен слева снизу/ слева сверху от объекта, у которого свойство k2 имеет значение j2.   
  ***F: = F***∧ ***(p (k1, i1, j1)⬄ p (k2, i2, j2))***
* 2 ограничения типа 4: ограничение расположения объектов – расположение «рядом», дизъюнкция ограничений типа 3

***F: = F*** ∧ ***[( p(k1, i1, j1) ⬄ p(k2, i2, j2) ) ∨ ( p(k2, i1, j2) ⬄ p(k1, i2, j1))]***

Для полученного решения придумать физическую интерпретацию задачи. Если задача не имеет решений или же имеет слишком много решений, следует убрать, добавить и/или изменить некоторые ограничения так, чтобы задача имела 10–12 решений.

# **Решение задачи.**

Для начала добавим заданное количество ограничений каждого типа.

Ограничения типа 1:

1. Свойство 0 у объекта с номером 0 равно 0.
2. Свойство 0 у объекта с номером 1 равно 1.
3. Свойство 0 у объекта с номером 2 равно 2.
4. Свойство 0 у объекта с номером 3 равно 3.
5. Свойство 0 у объекта с номером 4 равно 4.
6. Свойство 0 у объекта с номером 5 равно 5.

Ограничения типа 2:

1. Если у объекта свойство 1 имеет значение 2, то его свойство 2 имеет значение 4, и наоборот.
2. Если у объекта свойство 2 имеет значение 2, то его свойство 1 имеет значение 3, и наоборот.
3. Если у объекта свойство 3 имеет значение 3, то его свойство 1 имеет значение 4, и наоборот.
4. Если у объекта свойство 0 имеет значение 2, то его свойство 1 имеет значение 5, и наоборот.
5. Если у объекта свойство 1 имеет значение 3, то его свойство 3 имеет значение 4, и наоборот.
6. Если у объекта свойство 2 имеет значение 3, то его свойство 1 имеет значение 0, и наоборот.
7. Если у объекта свойство 0 имеет значение 3, то его свойство 2 имеет значение 6, и наоборот.
8. Если у объекта свойство 2 имеет значение 4, то его свойство 3 имеет значение 5, и наоборот.

Ограничения типа 3:

1. Если у объекта свойство 1 имеет значение 2, то объект, имеющий свойство 2 со значением 3, стоит сверху слева от этого объекта.
2. Если у объекта свойство 3 имеет значение 1, то объект, имеющий свойство 3 со значением 4, стоит снизу от этого объекта.
3. Если у объекта свойство 0 имеет значение 5, то объект, имеющий свойство 3 со значением 2, стоит снизу от этого объекта.
4. Если у объекта свойство 1 имеет значение 1, то объект, имеющий свойство 2 со значением 2, стоит снизу от этого объекта.
5. Если у объекта свойство 2 имеет значение 5, то объект, имеющий свойство 3 со значением 7, стоит снизу от этого объекта.

Ограничение типа 4:

1. Если у объекта свойство 0 имеет значение 1, то он стоит сверху слева или снизу от объекта, имеющего свойство 2 со значением 0.
2. Если у объекта свойство 3 имеет значение 6, то он стоит слева сверху или снизу от объекта, имеющего свойство 1 со значением 6.
3. Если у объекта свойство 2 имеет значение 0, то он стоит слева сверху или снизу от объекта, имеющего свойство 0 со значением 6.

Для того, чтобы получить 10–12 решений, введем дополнительные ограничения.

Дополнительные ограничения типа 1:

1. Свойство 3 у объекта с номером 3 равно 3.
2. Свойство 2 у объекта с номером 2 равно 1.
3. Свойство 2 у объекта с номером 2 равно 1.

Дополнительные ограничения типа 2:

* Если у объекта свойство 2 имеет значение 8, то его свойство 1 имеет значение 8, и наоборот.
* Если у объекта свойство 3 имеет значение 6, то его свойство 1 имеет значение 7, и наоборот.
* Если у объекта свойство 3 имеет значение 7, то его свойство 2 имеет значение 7, и наоборот.

В результате работы программы получили 10 решений. Одно из 10 решений представлено в таблице 10.

|  |
| --- |
| 0: 0 6 7 7  1: 1 0 3 2  2: 2 5 1 8  3: 3 4 6 3  4: 4 1 0 1  5: 5 2 4 5  6: 7 7 5 6  7: 6 3 2 4  8: 8 8 8 0 |

Таблица 10.   
Одно из 10 решений

# **Интерпретация задачи.**

На парковке есть 9 мест, все 9 мест заняты. Каждое место занято машиной. У каждой машины есть 4 свойства: цвет, марка, водитель, страна производства. На въезде на парковку стоит контролер, который записывает данные о машинах в тетрадь. К сожалению, контролер пролил кофе на свою тетрадь. Необходимо восстановить данные о каждой машине.

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Таблица 11.   
Расположение посылок на стеллаже.

Так как контролер очень любит кофе, он часто портит свои записи, поэтому у него хорошая память. Что ему удалось вспомнить:

* 1 место заняла красная машина
* 2 место заняла черная машина
* 3 место заняла белая машина
* 4 место заняла синяя машина
* 5 место заняла зеленая машина
* 6 место заняла желтая машина
* Машиной марки Ferrari владеет Аня
* Машиной марки Toyota владеет Денис
* Машина марки Lada была произведена в Италии
* Машина марки BMW белого цвета
* Машина марки Toyota произведена в Китае
* Машиной марки Kia владеет Ксюша
* Машиной синего цвета владеет Сережа
* Аня владеет машиной из Японии
* Машина марки Ferrari стоит слева сверху от машины Ксюши
* Машина из Германии стоит снизу от машины из Китая
* Желтая машина стоит сверху слева от машина из Франции
* Машина марки Kia стоит снизу от машина Дениса
* Машина Руслана стоит снизу от машины из США
* Черная машина стоит снизу или слева сверху от машины Романа
* Машина из Кореи стоит снизу или слева сверху от машины марки Mercedes
* Машина Романа стоит снизу или слева сверху от розовой машины

Подумав еще, контролер вспомнил еще некоторые детали:

* 4 место занято машиной из Италии
* 3 место занято Женей
* Таня владеет машиной марки Suzuki
* Машина марки Mazda произведена в Корее
* Саша владеет машиной из США

|  |
| --- |
| 0: 0 6 7 7  1: 1 0 3 2  2: 2 5 1 8  3: 3 4 6 3  4: 4 1 0 1  5: 5 2 4 5  6: 7 7 5 6  7: 6 3 2 4  8: 8 8 8 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Место № | *Свойство 1.*  Цвет | *Свойство 2.*  Марка | *Свойство 3.*  Водитель | *Свойство 4.*  Страна производства |
| 1 | Красный | Mercedes | Саша | США |
| 2 | Черный | Kia | Ксюша | Франция |
| 3 | Белый | BMW | Женя | Великобритания |
| 4 | Синий | Lada | Сережа | Италия |
| 5 | Зеленый | Honda | Рома | Германия |
| 6 | Желтый | Ferrari | Аня | Япония |
| 7 | Серый | Mazda | Руслан | Корея |
| 8 | Розовый | Toyota | Денис | Китай |
| 9 | Коричневый | Suzuki | Таня | РФ |

Таблица 12. Информация о машинах

# **Задача без склейки**

Контролеру не удалось восстановить данные. Подумав еще немного, он вспомнил, что:

* 8 место заняла машина из Кореи

|  |
| --- |
| 0: 0 6 7 7  1: 1 0 3 2  2: 2 5 1 8  3: 3 4 6 3  4: 4 1 0 1  5: 5 2 4 5  6: 7 8 8 0  7: 6 3 2 4  8: 8 7 5 6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Место № | *Свойство 1.*  Цвет | *Свойство 2.*  Марка | *Свойство 3.*  Водитель | *Свойство 4.*  Страна производства |
| 1 | Красный | Mercedes | Саша | США |
| 2 | Черный | Kia | Ксюша | Франция |
| 3 | Белый | BMW | Женя | Великобритания |
| 4 | Синий | Lada | Сережа | Италия |
| 5 | Зеленый | Honda | Рома | Германия |
| 6 | Желтый | Ferrari | Аня | Япония |
| 7 | Серый | Suzuki | Таня | РФ |
| 8 | Розовый | Toyota | Денис | Китай |
| 9 | Коричневый | Mazda | Руслан | Корея |

# **Заключение.**

Подводя итог, хочу сказать, что в результате работы мы познакомились и изучили библиотеку BuDDy, которая является самой доступной реализацией бинарных диаграмм решений. Данная библиотека заметно упростила поиск решения задачи. Благодаря этой библиотеки была реализована программа, написанная на языке C++, в среде CLion, которая вычисляет решения поставленной задачи, являющейся модификацией головоломки “Задача Эйнштейна”. Как итог, мы получили масштабируемую реализацию программы с поставленными ограничениями, которая выдала 10–12 решений, удовлетворяющей задаче ограниченного программирования. Одно из решений было физически интерпретировано и наглядно показано.

# **Список литературы.**

1. Программирование в ограничениях: [Электронный ресурс]: URL: <https://ru.wikipedia.org/wiki/Программирование_в_ограничениях>

1. BuDDy: A BDD package [Электронный ресурс].

URL: <http://buddy.sourceforge.net/manual/main.html>

1. Программирование в ограничениях - Constraint programming: [Электронный ресурс]:

URL: <https://ru.qaz.wiki/wiki/Constraint_programming>

1. Модуль 4. Бинарные решающие диаграммы: [Электронный ресурс]: URL: <https://courses.openedu.ru/courses/course-v1:spbstu+MATLOG+fall_2021/courseware/c6a8ff97af6f4b25864ef013fbbb6971/b066bb9741d844df83f03e323bf703e2/1?activate_block_id=block-v1%3Aspbstu%2BMATLOG%2Bfall_2021%2Btype%40vertical%2Bblock%40a62e15d3faf34df795506671018ae8af>
2. Бинарная диаграмма решений: [Электронный ресурс]:URL: [https://ru.wikipedia.org/wiki/Бинарная\_диаграмма\_решений.](https://ru.wikipedia.org/wiki/Бинарная_диаграмма_решений)

# **Приложение А. Исходный код с учетом склейки.**

**main.cxx**

#include "bdd.h"

#include <iostream>

#include <string.h>

#include <fstream>

#include <cmath>

constexpr int log2(int x)

{

return x < 3 ? 1 : 1 + log2(x - x / 2);

}

// число объектов

constexpr int N = 9;

// число свойств

constexpr int M = 4;

// логарифм по основанию 2, взятый с потолком

constexpr int LOG\_N = log2(N);

// число булевых переменных

constexpr int N\_VAR = N \* M \* LOG\_N;

// корень квадратный из N

const int SQRT\_N = std::sqrt(N);

// входной файл

const char\* INPUT\_FILE = "C:\\Users\\zheny\\source\\repos\\CourseWorkLogic\\input.txt";

char var[N\_VAR];

std::ostream\* out\_stream;

// p-property o-object v-propVal

struct pov

{

pov() : propNum(-1), objNum(-1), propVal(-1) {}

//номер свойства

int propNum;

// номер объекта

int objNum;

//значение свойства

int propVal;

};

void limit1(bdd&, bdd p[M][N][N], pov);

void limit2(bdd&, bdd p[M][N][N], pov, pov);

bdd limit3\_UpLeft(bdd p[M][N][N], pov, pov);

bdd limit3\_Down(bdd p[M][N][N], pov, pov);

void limit4(bdd&, bdd p[M][N][N], pov, pov);

void limit5(bdd&, bdd p[M][N][N]);

void limit6(bdd&, bdd p[M][N][N]);

void print\_func(char\*, int);

int main()

{

// инициализация BuDDy

bdd\_init(100000, 10000);

bdd\_setvarnum(N\_VAR);

// Введение функции p(prop, obj, val)

bdd p[M][N][N];

for (unsigned propNum = 0; propNum < M; ++propNum)

{

for (unsigned objNum = 0; objNum < N; ++objNum)

{

for (unsigned propVal = 0; propVal < N; ++propVal)

{

p[propNum][objNum][propVal] = bddtrue;

for (unsigned t = 0; t < LOG\_N; ++t)

{

const unsigned index = LOG\_N \* M \* objNum + LOG\_N \* propNum + t;

p[propNum][objNum][propVal] &= (((propVal >> t) & 1) ? bdd\_ithvar(index) : bdd\_nithvar(index));

}

}

}

}

// БДД, являющаяся решением задачи

bdd solution = bddtrue;

// Ввод и применение ограничений

try

{

std::ifstream in;

in.open(INPUT\_FILE);

if (!in.is\_open())

{

throw std::invalid\_argument("ERROR: can`t open file.");

}

else

{

char command[4];

while (!in.eof())

{

pov prop1;

pov prop2;

in >> command;

if (in.bad())

{

throw std::invalid\_argument("ERROR: bad file.");

}

// Ограничение 1 типа

if (strcmp(command, "L1") == 0)

{

in >> prop1.propNum >> prop1.objNum >> prop1.propVal;

if (!in.bad())

{

limit1(solution, p, prop1);

}

else

{

throw std::invalid\_argument("ERROR: bad file.");

}

}

// Ограничение 2 типа

else if (strcmp(command, "L2") == 0)

{

in >> prop1.propNum >> prop1.propVal;

in >> prop2.propNum >> prop2.propVal;

if (!in.bad())

{

limit2(solution, p, prop1, prop2);

}

else

{

throw std::invalid\_argument("ERROR: bad file.");

}

}

// Ограничение 3 типа

else if (strcmp(command, "L3") == 0)

{

char dir[5];

in >> prop1.propNum >> prop1.propVal;

in >> dir;

in >> prop2.propNum >> prop2.propVal;

if (!in.bad())

{

if (strcmp(dir, "UP") == 0)

{

solution &= limit3\_UpLeft(p, prop1, prop2);

}

else if (strcmp(dir, "DOWN") == 0)

{

solution &= limit3\_Down(p, prop1, prop2);

}

else

{

throw std::invalid\_argument("ERROR: bad file.");

}

}

}

// Ограничение 4 типа

else if (strcmp(command, "L4") == 0)

{

in >> prop1.propNum >> prop1.propVal;

in >> prop2.propNum >> prop2.propVal;

if (!in.bad())

{

limit4(solution, p, prop1, prop2);

}

else

{

throw std::invalid\_argument("ERROR: bad file.");

}

}

// Конец ввода ограничений

else if (strcmp(command, "EOF") == 0)

{

break;

}

else

{

throw std::invalid\_argument("ERROR: bad file.");

}

if (in.get() != '\n')

{

throw std::invalid\_argument("ERROR: bad file.");

}

}

}

limit6(solution, p);

limit5(solution, p);

}

catch (std::invalid\_argument& e)

{

std::cerr << e.what();

return 1;

}

// Вывод результатов

std::ofstream out("output.txt");

auto satCount = bdd\_satcount(solution);

out << satCount << " solution(s):\n";

std::cout << satCount << " solution(s):\n";

out\_stream = &out;

if (satCount)

{

bdd\_allsat(solution, print\_func);

}

out\_stream = &std::cout;

if (satCount)

{

bdd\_allsat(solution, print\_func);

}

out.close();

if (satCount) std::cout << "Possible solution(s) in \"output.txt\"\n";

// Завершение работы с библиотекой BuDDy

bdd\_done();

return 0;

}

void limit1(bdd& solution, bdd p[M][N][N], pov prop)

{

solution &= p[prop.propNum][prop.objNum][prop.propVal];

}

void limit2(bdd& solution, bdd p[M][N][N], pov lProp, pov rProp)

{

for (unsigned obj = 0; obj < N; ++obj)

{

bdd l = p[lProp.propNum][obj][lProp.propVal];

bdd r = p[rProp.propNum][obj][rProp.propVal];

solution &= (!l & !r) | (l & r);

}

}

bdd limit3\_Down(bdd p[M][N][N], pov lProp, pov rProp)

{

bdd temp = bddtrue;

//Все, кроме нижней строки

for (unsigned obj = 0; obj < N - SQRT\_N; ++obj)

{

//dlObj - номер объекта снизу

auto dObj = (obj + SQRT\_N);

bdd l = p[lProp.propNum][obj][lProp.propVal];

bdd r = p[rProp.propNum][dObj][rProp.propVal];

temp &= ((!l) & (!r)) | (l & r);

}

return temp;

}

bdd limit3\_UpLeft(bdd p[M][N][N], pov lProp, pov rProp)

{

bdd temp = bddtrue;

//Все, кроме верхней строки

for (unsigned obj = SQRT\_N; obj < N; ++obj)

{

//ulObj - номер объекта слева сверху

if (obj % N == 0)

{

auto ulObj = obj - 1;

bdd l = p[lProp.propNum][obj][lProp.propVal];

bdd r = p[rProp.propNum][ulObj][rProp.propVal];

temp &= ((!l) & (!r)) | (l & r);

}

else

{

auto ulObj = obj - (SQRT\_N + 1);

bdd l = p[lProp.propNum][obj][lProp.propVal];

bdd r = p[rProp.propNum][ulObj][rProp.propVal];

temp &= ((!l) & (!r)) | (l & r);

}

}

return temp;

}

void limit4(bdd& solution, bdd p[M][N][N], pov lProp, pov rProp)

{

solution &= (limit3\_UpLeft(p, lProp, rProp)

| limit3\_Down(p, lProp, rProp));

}

// Проверка, что никакое значение одного свойства не встерчается дважды

void limit5(bdd& solution, bdd p[M][N][N])

{

for (unsigned prop = 0; prop < M; ++prop)

{

for (unsigned obj1 = 0; obj1 < N - 1; ++obj1)

{

for (unsigned obj2 = obj1 + 1; obj2 < N; ++obj2)

{

for (unsigned val = 0; val < N; ++val)

{

solution &= (!p[prop][obj1][val] | !p[prop][obj2][val]);

}

}

}

}

}

// Проверка, что все propVal меньше N

void limit6(bdd& solution, bdd p[M][N][N])

{

for (unsigned obj = 0; obj < N; ++obj)

{

bdd temp = bddtrue;

for (unsigned prop = 0; prop < M; ++prop)

{

bdd tempSecond = bddfalse;

for (unsigned val = 0; val < N; ++val)

{

tempSecond |= p[prop][obj][val];

}

temp &= tempSecond;

}

solution &= temp;

}

}

void print()

{

for (unsigned i = 0; i < N; ++i)

{

(\*out\_stream) << i << ": ";

for (unsigned j = 0; j < M; ++j)

{

unsigned J = i \* M \* LOG\_N + j \* LOG\_N;

unsigned num = 0;

for (unsigned k = 0; k < LOG\_N; ++k)

{

num += (unsigned)(var[J + k] << k);

}

(\*out\_stream) << num << ' ';

}

(\*out\_stream) << '\n';

}

(\*out\_stream) << '\n';

}

void build(char\* varset, unsigned n, unsigned I)

{

if (I == n - 1)

{

if (varset[I] >= 0)

{

var[I] = varset[I];

print();

return;

}

var[I] = 0;

print();

var[I] = 1;

print();

return;

}

if (varset[I] >= 0)

{

var[I] = varset[I];

build(varset, n, I + 1);

return;

}

var[I] = 0;

build(varset, n, I + 1);

var[I] = 1;

build(varset, n, I + 1);

}

void print\_func(char\* varset, int size)

{

build(varset, size, 0);

}

**input.txt**

L1 0 0 0

L1 0 1 1

L1 0 2 2

L1 0 3 3

L1 0 4 4

L1 0 5 5

L2 1 2 2 4

L2 2 2 1 3

L2 3 3 1 4

L2 0 2 1 5

L2 1 3 3 4

L2 2 3 1 0

L2 0 3 2 6

L2 2 4 3 5

L3 1 2 UP 2 3

L3 3 1 DOWN 3 4

L3 0 5 UP 3 2

L3 1 1 DOWN 2 2

L3 2 5 DOWN 3 7

L4 0 1 2 0

L4 3 6 1 6

L4 2 0 0 6

L1 3 3 3

L1 2 2 1

L2 2 8 1 8

L2 3 6 1 7

L2 3 7 2 7

EOF

# **Приложение Б. Ограничения 3 типа без учета склейки.**

Остальной код не отличается от исходного

bdd L3\_UpLeft(bdd p[M][N][N], pov lProp, pov rProp)

{

bdd temp = bddtrue;

//Все, кроме верхней строки и левого столбца

for (unsigned obj = SQRT\_N; obj < N; ++obj)

{

//ulObj - номер объекта слева сверху

if (obj % N != 0)

{

auto ulObj = obj - 1;

bdd l = p[lProp.propNum][obj][lProp.propVal];

bdd r = p[rProp.propNum][ulObj][rProp.propVal];

temp &= ((!l) & (!r)) | (l & r);

}

}

return temp;

}

bdd L3\_Down(bdd p[M][N][N], pov lProp, pov rProp)

{

bdd temp = bddtrue;

//Все, кроме нижней строки

for (unsigned obj = 0; obj < N - SQRT\_N; ++obj)

{

//dlObj - номер объекта снизу

auto dObj = (obj + SQRT\_N);

bdd l = p[lProp.propNum][obj][lProp.propVal];

bdd r = p[rProp.propNum][dObj][rProp.propVal];

temp &= ((!l) & (!r)) | (l & r);

}

return temp;

}

В текстовый файл была добавлена строка

L1 3 8 6