

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. В. Г. ШУХОВА» (БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Лабораторная работа №7

по дисциплине: «Исследование операций»

Вариант 23

Выполнил: ст. группы ПВ-211

Чувилко Илья Романович

Проверил:

Куртова Лилиана Николаевна

Вирченко Юрий Петрович

Белгород 2023 г.

Тема: Решение полностью целочисленных задач с помощью первого алгоритма Гомори, а также методом ветвей и границ

Цель работы: освоить метод отсечения Гомори для полностью целочисленных задач. Изучить алгоритм этого метода. Программно реализовать этот алгоритм.

Ход работы:

$$z = 10x_1 - 5x_2 + 4x_4 \rightarrow \max$$

$$\begin{cases} 9x_1 + 3x_2 + x_3 = 75 \\ 18x_1 - 4x_2 - 3x_4 = 22 \\ 2x_1 - 10x_2 + x_5 = 18 \end{cases}$$

$$x_i \geq 0, x_i - \text{целые } (i = \overline{1,5})$$

1. Изучить возможные постановки задач целочисленного и частичноцелочисленного программирования.
2. Ознакомиться с методами решения таких задач, в частности, с методами отсечения и методом ветвей и границ.
3. Выяснить для каких задач применяется первый алгоритм Гомори. Изучить этот алгоритм и написать реализующую его программу для ПЭВМ. В качестве тестовых данных использовать, решенную вручную одну из нижеследующих задач.

$$\begin{aligned} z &= 34x_1 - \frac{31}{3}x_2 - \frac{88}{3} \rightarrow \max \\ \begin{cases} 9x_1 + 3x_2 + x_3 = 75 \\ -6x_1 + \frac{4}{3}x_2 + x_4 = -\frac{22}{3} \\ 2x_1 - 10x_2 + x_5 = 18 \end{cases} \\ x_i &\geq 0, x_i - \text{целые } (i = \overline{1,5}) \end{aligned}$$

Отбрасывая условия целочисленности, решаем обобщенным двойственным симплекс методом задачу 3_0 :

$$\begin{aligned} z &= 34x_1 - \frac{31}{3}x_2 - \frac{88}{3} \rightarrow \max \\ \begin{cases} 9x_1 + 3x_2 + x_3 = 75 \\ -6x_1 + \frac{4}{3}x_2 + x_4 = -\frac{22}{3} \\ 2x_1 - 10x_2 + x_5 = 18 \end{cases} \\ x_i &\geq 0 \quad (i = \overline{1,5}) \end{aligned}$$

Таблица 1

Б	С	x1↓	x2	x3	x4	x5	Отн
x3	75	9	3	1	0	0	8 1/3
←x4	-7 1/3	-6	1 1/3	0	1	0	1 2/9
x5	18	2	-10	0	0	1	9
z	-29 1/3	-34	10 1/3	0	0	0	

Таблица 2

Б	С	x1	x2	x3	x4↓	x5	Отн
←x3	64	0	5	1	1 1/2	0	42 2/3
x1	1 2/9	1	- 2/9	0	- 1/6	0	-7 1/3
x5	15 5/9	0	-9 5/9	0	1/3	1	46 2/3
z	12 2/9	0	2 7/9	0	-5 2/3	0	

Таблица 3

Б	С	x1	x2	x3	x4	x5
x4	42 2/3	0	3 1/3	2/3	1	0
x1	8 1/3	1	1/3	1/9	0	0
x5	1 1/3	0	-10 2/3	- 2/9	0	1
z	254	0	21 2/3	3 7/9	0	0

Из последней симплекс-таблицы получаем оптимальное решение задачи Z_0 :

$$\bar{X}_0 = \{8\frac{1}{3}; 0; 0; 42\frac{2}{3}; 1\frac{1}{3}\}$$

Это решение не является целочисленным, построим задачу Z_1 , для этого запишем сечение Гомори по первой строке:

$$\frac{2}{3} - \frac{1}{3}x_2 - \frac{2}{3}x_3 \leq 0$$

Преобразуя и уравнивая неравенство, получим:

$$-\frac{1}{3}x_2 - \frac{2}{3}x_3 + u_1 = -\frac{2}{3}$$

Добавляя это ограничение к ограничениям задачи Z_0 , получим задачу Z_1

Таблица 4

Б	С	x1	x2	x3↓	x4	x5	u1
x4	42 2/3	0	3 1/3	2/3	1	0	0
x1	8 1/3	1	1/3	1/9	0	0	0
x5	1 1/3	0	-10 2/3	- 2/9	0	1	0
←u1	- 2/3	0	- 1/3	- 2/3	0	0	1
z	254	0	21 2/3	3 7/9	0	0	0
Отн			65	5 2/3			

Таблица 5

Б	С	x1	x2	x3	x4	x5	u1
x4	42	0	3	0	1	0	1
x1	8 2/9	1	5/18	0	0	0	1/6
x5	1 5/9	0	-10 5/9	0	0	1	- 1/3
x3	1	0	1/2	1	0	0	-1 1/2
z	250 2/9	0	19 7/9	0	0	0	5 2/3

Полученное решение не является целочисленным. Построим задачу Z_2 , для этого запишем сечение Гомори по третьей строке:

$$\frac{5}{9} - \frac{4}{9}x_2 - \frac{2}{3}u_1 \leq 0$$

Преобразуя и уравнивая неравенство, получим:

$$-\frac{4}{9}x_2 - \frac{2}{3}u_1 + u_2 = -\frac{5}{9}$$

Добавляя это ограничение к ограничениям задачи Z_1 , получим задачу Z_2 .

Таблица 6

Б	С	x1	x2	x3	x4	x5	u1↓	u2
x4	42	0	3	0	1	0	1	0
x1	8 2/9	1	5/18	0	0	0	1/6	0
x5	1 5/9	0	-10 5/9	0	0	1	- 1/3	0
x3	1	0	1/2	1	0	0	-1 1/2	0
←u2	- 5/9	0	- 4/9	0	0	0	- 2/3	1
z	250 2/9	0	19 7/9	0	0	0	5 2/3	0
Отн			44 1/2				8 1/2	

Таблица 7

Б	С	x1	x2	x3	x4	x5	u1	u2
x4	41 1/6	0	2 1/3	0	1	0	0	1 1/2
x1	8 1/12	1	1/6	0	0	0	0	1/4
x5	1 5/6	0	-10 1/3	0	0	1	0	- 1/2
x3	2 1/4	0	1 1/2	1	0	0	0	-2 1/4
u1	5/6	0	2/3	0	0	0	1	-1 1/2
z	245 1/2	0	16	0	0	0	0	8 1/2

Полученное решение не является целочисленным. Построим задачу Z_3 , для этого запишем сечение Гомори по третьей строке:

$$\frac{5}{6} - \frac{2}{3}x_2 - \frac{1}{2}u_2 \leq 0$$

Преобразуя и уравнивая неравенство, получим:

$$-\frac{2}{3}x_2 - \frac{1}{2}u_2 + u_3 = -\frac{5}{6}$$

Добавляя это ограничение к ограничениям задачи Z_2 , получим задачу Z_3

Таблица 8

Б	C	x1	x2	x3	x4	x5	u1	u2↓	u3
x4	41 1/6	0	2 1/3	0	1	0	0	1 1/2	0
x1	8 1/12	1	1/6	0	0	0	0	1/4	0
x5	1 5/6	0	-10 1/3	0	0	1	0	- 1/2	0
x3	2 1/4	0	1 1/2	1	0	0	0	-2 1/4	0
u1	5/6	0	2/3	0	0	0	1	-1 1/2	0
←u3	- 5/6	0	- 2/3	0	0	0	0	- 1/2	1
z	245 1/2	0	16	0	0	0	0	8 1/2	0
Отн			24					17	

Таблица 9

Б	C	x1	x2	x3	x4	x5	u1	u2	u3
x4	38 2/3	0	1/3	0	1	0	0	0	3
x1	7 2/3	1	- 1/6	0	0	0	0	0	1/2
x5	2 2/3	0	-9 2/3	0	0	1	0	0	-1
x3	6	0	4 1/2	1	0	0	0	0	-4 1/2
u1	3 1/3	0	2 2/3	0	0	0	1	0	-3
u2	1 2/3	0	1 1/3	0	0	0	0	1	-2
z	231 1/3	0	4 2/3	0	0	0	0	0	17

Полученное решение не является целочисленным. Построим задачу Z_4 , для этого запишем сечение Гомори по первой строке:

$$\frac{2}{3} - \frac{1}{3}x_2 \leq 0$$

Преобразуя и уравнивая неравенство, получим:

$$-\frac{1}{3}x_2 + u_4 = -\frac{2}{3}$$

Добавляя это ограничение к ограничениям задачи Z_3 , получим задачу Z_4 .

Таблица 10

Б	С	x1	x2↓	x3	x4	x5	u1	u2	u3	u4
x4	38 2/3	0	1/3	0	1	0	0	0	3	0
x1	7 2/3	1	- 1/6	0	0	0	0	0	1/2	0
x5	2 2/3	0	-9 2/3	0	0	1	0	0	-1	0
x3	6	0	4 1/2	1	0	0	0	0	-4 1/2	0
u1	3 1/3	0	2 2/3	0	0	0	1	0	-3	0
u2	1 2/3	0	1 1/3	0	0	0	0	1	-2	0
←u4	- 2/3	0	- 1/3	0	0	0	0	0	0	1
z	231 1/3	0	4 2/3	0	0	0	0	0	17	0
Отн			14							

Таблица 11

Б	С	x1	x2	x3	x4	x5	u1	u2	u3↓	u4
x4	38	0	0	0	1	0	0	0	3	1
x1	8	1	0	0	0	0	0	0	1/2	- 1/2
x5	22	0	0	0	0	1	0	0	-1	-29
←x3	-3	0	0	1	0	0	0	0	-4 1/2	13 1/2
u1	-2	0	0	0	0	0	1	0	-3	8
u2	-1	0	0	0	0	0	0	1	-2	4
x2	2	0	1	0	0	0	0	0	0	-3
z	222	0	0	0	0	0	0	0	17	14
Отн									3 7/9	

Таблица 12

Б	С	x1	x2	x3	x4	x5	u1	u2	u3	u4
x4	36	0	0	2/3	1	0	0	0	0	10
x1	7 2/3	1	0	1/9	0	0	0	0	0	1
x5	22 2/3	0	0	- 2/9	0	1	0	0	0	-32
u3	2/3	0	0	- 2/9	0	0	0	0	1	-3
u1	0	0	0	- 2/3	0	0	1	0	0	-1
u2	1/3	0	0	- 4/9	0	0	0	1	0	-2
x2	2	0	1	0	0	0	0	0	0	-3
z	210 2/3	0	0	3 7/9	0	0	0	0	0	65

Полученное решение не является целочисленным. Построим задачу Z_5 , для этого запишем сечение Гомори по второй строке:

$$\frac{2}{3} - \frac{1}{9}x_3 \leq 0$$

Преобразуя и уравнивая неравенство, получим:

$$-\frac{1}{9}x_3 + u_5 = -\frac{2}{3}$$

Добавляя это ограничение к ограничениям задачи Z_4 , получим задачу Z_5 .

Таблица 13

Б	С	x1	x2	x3↓	x4	x5	u1	u2	u3	u4	u5
x4	36	0	0	2/3	1	0	0	0	0	10	0
x1	7 2/3	1	0	1/9	0	0	0	0	0	1	0
x5	22 2/3	0	0	- 2/9	0	1	0	0	0	-32	0
u3	2/3	0	0	- 2/9	0	0	0	0	1	-3	0
u1	0	0	0	- 2/3	0	0	1	0	0	-1	0
u2	1/3	0	0	- 4/9	0	0	0	1	0	-2	0
x2	2	0	1	0	0	0	0	0	0	-3	0
←u5	- 2/3	0	0	- 1/9	0	0	0	0	0	0	1
z	210 2/3	0	0	3 7/9	0	0	0	0	0	65	0
Отн				34							

Таблица 14

Б	С	x1	x2	x3	x4	x5	u1	u2	u3	u4	u5
x4	32	0	0	0	1	0	0	0	0	10	6
x1	7	1	0	0	0	0	0	0	0	1	1
x5	24	0	0	0	0	1	0	0	0	-32	-2
u3	2	0	0	0	0	0	0	0	1	-3	-2
u1	4	0	0	0	0	0	1	0	0	-1	-6
u2	3	0	0	0	0	0	0	1	0	-2	-4
x2	2	0	1	0	0	0	0	0	0	-3	0
x3	6	0	0	1	0	0	0	0	0	0	-9
z	188	0	0	0	0	0	0	0	0	65	34

Полученное решение является целочисленным. Решение исходной задачи целочисленного программирования:

$z_{\max} = 188$; точка максимума: (7; 2; 6; 32; 24)

Программная реализация первого алгоритма Гомори:

```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
#include <cmath>
#include <numeric>

#define DBL_EPSILON_IN_MY_CASE 7.7e-10
using SimplexTable = std::vector<std::pair<std::string,
std::vector<double>>>>;

void outputSimplexTable(const SimplexTable &simplexTable) {
    std::cout << "BV\FV\t";
    for (size_t i{1}; i < simplexTable.at(0).second.size(); ++i) {
        std::cout << "x" << i << "\t";
    }
    std::cout << "\n";
    for (size_t i{1}; i < simplexTable.size(); ++i) {
        std::cout << simplexTable.at(i).first << "\t";
        for (size_t j{1}; j < simplexTable.at(i).second.size();
            ++j) {
            std::cout << simplexTable.at(i).second.at(j) << "\t";
        }
        std::cout << "\n";
    }
    std::cout << "\n";
}

void derivationOfTheOptimumPoint(const SimplexTable
    &simplexTable, const std::vector<std::string> integralVar) {
    std::cout << "Optimum point : (" ;
    for (size_t i{1}; i < integralVar.size(); ++i) {
```

```

bool findVar{false};
for (size_t j{}; j < simplexTable.size() - 1; ++j) {
    if (simplexTable.at(j).first == integralVar.at(i)) {
        findVar = true;
        if (simplexTable.at(j).second.at(0) <=
            DBL_EPSILON_IN_MY_CASE) {
            std::cout << 0 << ' ';
        } else {
            std::cout << simplexTable.at(j).second.at(0)
                << ' ';
        }
    }
}
if (!findVar) {
    std::cout << 0 << ' ';
}
std::cout << "\b)\n";
}

bool objFunctionHasNegative(const SimplexTable &simplexTable,
    size_t &minNegativeIndex) {
    size_t rowIndex{simplexTable.size() - 1};
    size_t minIndex{};
    bool findNegative{false};
    for (size_t i{1}; i < simplexTable.at(0).second.size(); ++i) {
        if (simplexTable.at(rowIndex).second.at(i) < (-
            DBL_EPSILON_IN_MY_CASE)) {
            findNegative = true;
            if (simplexTable.at(rowIndex).second.at(i) <
                simplexTable.at(rowIndex).second.at(minIndex)) {
                minIndex = i;
            }
        }
    }
    minNegativeIndex = minIndex;
    return findNegative;
}

bool exHasPositiveCoeff(const SimplexTable &simplexTable, const
size_t &colIndex, size_t &minCoeffIndex) {
    size_t minCoeffIn{};
    double minCoeff{std::numeric_limits<double>::max()};
    bool findPositive{false};
    for (size_t i{}; i < simplexTable.size() - 1; ++i) {
        if (simplexTable.at(i).second.at(0) /
            simplexTable.at(i).second.at(colIndex) > DBL_EPSILON_IN_MY_CASE) {
            findPositive = true;
            const double coeff{simplexTable.at(i).second.at(0) /
                simplexTable.at(i).second.at(colIndex)};
            if (coeff < minCoeff) {
                minCoeff = coeff;
                minCoeffIn = i;
            }
        }
    }
    minCoeffIndex = minCoeffIn;
    return findPositive;
}

void maxValueOfTheObjFunctionWithTableDisplay(SimplexTable
    &simplexTable) {
    size_t minIndex{};

```



```

size_t minCoeffIndex{};
while (objFunctionHasNegative(simplexTable, minIndex)) {
    std::cout << "Simplex table :\n";
    outputSimplexTable(simplexTable);
    if (exHasPositiveCoeff(simplexTable, minIndex,
        minCoeffIndex)) {
        const double
            divider{simplexTable.at(minCoeffIndex).second.at(minIndex)};
        for (size_t i{}; i <
            simplexTable.at(minCoeffIndex).second.size(); ++i) {
            simplexTable.at(minCoeffIndex).second.at(i) /=
                divider;
        }
        for (size_t i{}; i < simplexTable.size(); ++i) {
            if (i != minCoeffIndex) {
                const double divide{-
                    simplexTable.at(i).second.at(minIndex) /
                    simplexTable.at(minCoeffIndex).second.at(minIndex)};
                for (size_t j{}; j <
                    simplexTable.at(i).second.size(); ++j) {
                    simplexTable.at(i).second.at(j) +=
                        (divide * simplexTable.at(minCoeffIndex).second.at(j));
                }
            }
        }
        simplexTable.at(minCoeffIndex).first = "x" +
            std::to_string(minIndex);
    } else {
        std::cout << "The problem does not have solution(The objective "
            "function is unbounded on the range of admissible "
            "values of solutions)";
        std::exit(1);
    }
}
std::cout << "Simplex table :\n";
outputSimplexTable(simplexTable);
}

```

```

bool hasNegativeFree(const SimplexTable &simplexTable, size_t
&minFreeIndex) {
    size_t minFreeIn{};
    double minFree{std::numeric_limits<double>::max()};
    bool findNegative{false};
    for (size_t i{}; i < simplexTable.size() - 1; ++i) {
        if (simplexTable.at(i).second.at(0) < (-
            DBL_EPSILON_IN_MY_CASE)) {
            findNegative = true;
            if (simplexTable.at(i).second.at(0) < minFree) {
                minFree = simplexTable.at(i).second.at(0);
                minFreeIn = i;
            }
        }
    }
    minFreeIndex = minFreeIn;
    return findNegative;
}

```

```

bool hasNegativeCoeff(const SimplexTable &simplexTable, const
size_t &rowIndex, size_t &minColIndex) {
    bool hasNegative{false};
    double minimum{std::numeric_limits<double>::max()};
    size_t minIndex{0};
    for (size_t i{1}; i <

```

```

        simplexTable.at(rowIndex).second.size(); ++i) {
if (simplexTable.at(rowIndex).second.at(i) < (-
    DBL_EPSILON_IN_MY_CASE)) {
    hasNegative = true;
    const double coeff{-
        simplexTable.at(simplexTable.size() - 1).second.at(i) /
        simplexTable.at(rowIndex).second.at(i)};
    if (coeff < minimum) {
        minimum = coeff;
        minIndex = i;
    }
}
}
minColIndex = minIndex;
return hasNegative;
}

double generalizedSimplexMethod(SimplexTable &simplexTable) {
    maxValueOfTheObjFunctionWithTableDisplay(simplexTable);
    size_t minFreeIndex{};
    while (hasNegativeFree(simplexTable, minFreeIndex)) {
        size_t minColIndex{};
        if (hasNegativeCoeff(simplexTable, minFreeIndex,
            minColIndex)) {
            const double
                divide{simplexTable.at(minFreeIndex).second.at(minColIndex)};
            for (size_t i{}; i <
                simplexTable.at(minFreeIndex).second.size(); ++i) {
                simplexTable.at(minFreeIndex).second.at(i) /=
                    divide;
            }
            for (size_t i{}; i < simplexTable.size(); ++i) {
                if (i != minFreeIndex) {
                    const double divider{-
                        simplexTable.at(i).second.at(minColIndex) /
                        simplexTable.at(minFreeIndex).second.at(minColIndex)};
                    for (size_t j{}; j <
                        simplexTable.at(i).second.size(); ++j) {
                        simplexTable.at(i).second.at(j) +=
                            (divider * simplexTable.at(minFreeIndex).second.at(j));
                    }
                }
            }
            simplexTable.at(minFreeIndex).first = "x" +
                std::to_string(minColIndex);
        } else {
            std::cout << "The problem has no solution due to the absence "
                "of admissible solutions to the system of constraints\n";
            std::exit(1);
        }
        std::cout << "Simplex table :\n";
        outputSimplexTable(simplexTable);
    }
    return simplexTable.at(simplexTable.size() - 1).second.at(0);
}

double fractionPart(const double &value) {
    if (std::fabs(value) <= DBL_EPSILON_IN_MY_CASE) {
        return 0;
    }
    return value - std::floor(value);
}

```

```

bool isInteger(const double &value) {
    return (fractionPart(value) <= DBL_EPSILON_IN_MY_CASE);
}

bool resultIsInteger(const SimplexTable &simplexTable, const
std::vector<std::string> &integerVar) {
    for (size_t i{}; i < integerVar.size(); ++i) {
        for (size_t j{}; j < simplexTable.size() - 1; ++j) {
            if (simplexTable.at(j).first == integerVar.at(i) &&
                !isInteger(simplexTable.at(j).second.at(0))) {
                return false;
            }
        }
    }
    return true;
}

size_t rowIndexOfMaxFraction(const SimplexTable &simplexTable) {
    size_t maxIndex{};
    double maxValue{std::numeric_limits<double>::min()};
    for (size_t i{}; i < simplexTable.size() - 1; ++i) {
        double
            fracPart{fractionPart(simplexTable.at(i).second.at(0))};
        if (fracPart - maxValue > DBL_EPSILON_IN_MY_CASE) {
            maxValue = fracPart;
            maxIndex = i;
        }
    }
    return maxIndex;
}

double GomorysFirstAlgorithm(SimplexTable &simplexTable, const
std::vector<std::string> &integerVar) {
    double result{};
    while (true) {
        result = generalizedSimplexMethod(simplexTable);
        if (resultIsInteger(simplexTable, integerVar)) {
            return result;
        }
    }
    /* build Section of Gomory
    size_t maxRow{rowIndexOfMaxFraction(simplexTable)};
    simplexTable.push_back(std::make_pair(std::string("x" +
                                                std::to_string(simplexTable.at(0).second.size())),
                                                std::vector<double>{}));
    for (size_t i{}; i <
        simplexTable.at(maxRow).second.size(); ++i) {
        simplexTable.at(simplexTable.size() -
            1).second.push_back(-
                fractionPart(simplexTable.at(maxRow).second.at(i)));
    }
    for (size_t i{}; i < simplexTable.size(); ++i) {
        if (i != simplexTable.size() - 1) {
            simplexTable.at(i).second.push_back(0);
        } else {
            simplexTable.at(i).second.push_back(1);
        }
    }
    std::swap(simplexTable.at(simplexTable.size() - 1),
        simplexTable.at(simplexTable.size() - 2));
    }
}

int main(int argc, char **argv) {

```

```

size_t numberOfIntegerVar{};
std::cout << "Number of integer variables = ";
std::cin >> numberOfIntegerVar;
std::cout << "Enter variables that are integers (In ascending "
    "order of indices) : ";
std::vector<std::string>
    integerVariables(numberOfIntegerVar);
for (size_t i{}; i < numberOfIntegerVar; ++i) {
    std::cin >> integerVariables.at(i);
}
size_t numberOfRows{};
size_t numberOfCols{};
std::cout << "Number of rows in simplex table = ";
std::cin >> numberOfRows;
std::cout << "Number of cols in simplex table = ";
std::cin >> numberOfCols;
std::cout << "Enter simplex table(with the names of basic "
    "variables) : \n";
SimplexTable simplexTable(numberOfRows);
for (size_t i{}; i < numberOfRows; ++i) {
    std::string basisVarName{};
    std::cin >> basisVarName;
    simplexTable.at(i).first = basisVarName;
    for (size_t j{}; j < numberOfCols; ++j) {
        double value{};
        std::cin >> value;
        simplexTable.at(i).second.push_back(value);
    }
}
std::cout << std::setprecision(3);
double maxFunctValue{GomorysFirstAlgorithm(simplexTable,
    integerVariables)};
std::cout << "Max function value = " << maxFunctValue <<
    "\n\t";
derivationOfTheOptimumPoint(simplexTable, integerVariables);
/*
std::boolalpha(std::cout);
std::cout << resultIsInteger(simplexTable, integerVariables);
*/
return 0;
}

```

Результат работы программы:

```
C:\BGTU\BGTU\IsOp\Lab7\Code\cmake-build-debug\Code.exe
Number of integer variables =5
Enter variables that are integers (In ascending order of indices) :x1 x2 x3 x4 x5
Number of rows in simp
lex table =4
Number of cols in simplex table =6
Enter simplex table(with the names of basic variables) :
x3 110 9 4 1 0 0
x4 -24 -11 3 0 1 0
x5 -15 -2 7 0 0 1
z 0 -7 -1 0 0 0
Simplex table :
BV      FV      x1      x2      x3      x4      x5
x3      110      9      4      1      0      0
x4      -24      -11     3      0      1      0
x5      -15      -2      7      0      0      1
z        0      -7     -1      0      0      0

Simplex table :
BV      FV      x1      x2      x3      x4      x5
x3      90.4      0      6.45     1      0.818     0
x1      2.18      1     -0.273     -0     -0.0909     -0
x5     -10.6      0      6.45      0     -0.182      1
z       15.3      0     -2.91      0     -0.636      0

Simplex table :
BV      FV      x1      x2      x3      x4      x5
x2      14      0      1      0.155     0.127      0
x1       6      1      0      0.0423    -0.0563      0
x5     -101      0      0      -1      -1      1
z       56      0      0      0.451    -0.268      0

Simplex table :
BV      FV      x1      x2      x3      x4      x5
x2      1.2      0      1      0.0282     0      0.127
x1     11.7      1      0      0.0986      0     -0.0563
x4     101     -0     -0      1      1      -1
z      83      0      0      0.718      0     -0.268

Simplex table :
BV      FV      x1      x2      x3      x4      x5
x5      9.44      0      7.89     0.222      0      1
x1     12.2      1      0.444     0.111      0      0
x4     110      0      7.89     1.22      1      0
z     85.6      0      2.11     0.778      0      0

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6
x5      9.44      0      7.89     0.222      0      1      0
x1     12.2      1      0.444     0.111      0      0      0
x4     110      0      7.89     1.22      1      0      0
x6     -0.444     -0     -0.889    -0.222     -0     -0      1
z     85.6      0      2.11     0.778      0      0      0
```

```

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6
x5      5.5      0      0      -1.75    0      1      8.87
x1      12      1      0      9.71e-17  0      0      0.5
x4      107      0      0      -0.75    1      0      8.87
x2      0.5      0      1      0.25     0      0      -1.12
z       84.5     0      0      0.25     0      0      2.37

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7
x5      5.5      0      0      -1.75    0      1      8.87    0
x1      12      1      0      9.71e-17  0      0      0.5      0
x4      107      0      0      -0.75    1      0      8.87    0
x2      0.5      0      1      0.25     0      0      -1.12    0
x7      -0.5     -0     -0     -0.25    -0     -0     -0.875    1
z       84.5     0      0      0.25     0      0      2.37     0

Simplex table :
BV      FV      x1      x2      x3      x4      x5      x6      x7
x5      9      0      0      0      0      1      15     -7
x1      12      1      0      0      0      0      0.5    3.89e-16
x4      108     0      0      0      1      0      11.5   -3
x2      -7.7e-14  0      1      0      0      0      0     -2      1
x3      2      0      0      1      0      0      3.5    -4
z       84      0      0      0      0      0      1.5     1

Max function value = 84,      Optimum point : (12;0;2;108;9)

Process finished with exit code 0

```

Вывод: Освоил метод сечения Гомори для полностью целочисленных задач.
Изучил алгоритм этого метода. Программно реализовал этот алгоритм.