# МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Національний університет “Львівська політехніка”**



**Інститут післядипломної освіти**

**ЗВІТ**

**Про виконання лабораторної роботи №1**

**«Розв’язування задачі лінійного програмування симплекс-методом»**

**з дисципліни «Дослідження операцій»**

Виконав:

слухач групи ПЗС-21

Гринчук Тарас

Прийняла:

ст. викл. Мельник Н. Б.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2014 р.

∑ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЛЬВІВ – 2014

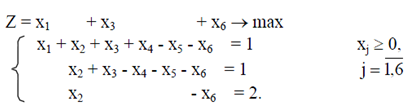
**Тема роботи**: Розв’язування задачі лінійного програмування симплекс-методом.

**Мета роботи**: Навчитись на практиці розв’язувати задачі лінійного програмування симплекс-методом.

## Індивідуальне завдання

## (Варіант 4)

Знайти рішення задачі лінійного програмування симплекс-методом:



**2. Текст програми на мові програмування JAVA**

package simplex;

import simplex.source.\*;

import simplex.gui.\*;

/\*\*

\* Точка входу в програму

\* @author Taras

\*/

public class SimplexMethod {

/\*\*

\* Точка входу

\* @param args the command line arguments

\*/

public static void main(String[] args) {

if(args.length == 0)

//запуск графічного інтерфейсу

MainWindow.main(null);

else if(args[0].toLowerCase().equals("con"))

// запуск з консолі

Simplex.main(null);

else

System.err.println("Помилковий формат командноi строки !");

}

}

package simplex.source;

import java.util.\*;

/\*\*

\* <b>Клас для обчислення симплекс методу</b>

\* @see Rational

\* @author Тарас Гринчук

\* @version 1.0

\*/

public class Simplex {

/\*\*

\* Обчислення max c(x), при A\*x менших(рівних) b, x - відмінні від нуля

\* @param A матриця обмежень

\* @param b стовбець вільних членів

\* @param c стовбець коефіцієнтів функціі

\* @param x стовбець розв'язків

\* @return Максимальне значення функціі

\*/

public static Rational simplex(Rational[][] A, Rational[] b, Rational[] c, Rational[] x) {

int m = A.length;

int n = A[0].length + 1;

int[] index = new int[n + m];

for (int i = 0; i < n + m; i++) {

index[i] = i;

}

Rational[][] a = new Rational[m + 2][n + 1];

for (Rational[] a1 : a) {

Arrays.fill(a1, Rational.ZERO);

}

int L = m;

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

for (int j = 0; j < n - 1; j++) {

a[i][j] = A[i][j].negate();

}

a[i][n - 1] = Rational.ONE;

a[i][n] = b[i];

if (a[L][n].compareTo(a[i][n]) > 0) {

L = i;

}

}

for (int j = 0; j < n - 1; j++) {

a[m][j] = c[j];

}

a[m + 1][n - 1] = Rational.ONE.negate();

for (int E = n - 1;;) {

if (L < m) {

int t = index[E];

index[E] = index[L + n];

index[L + n] = t;

a[L][E] = a[L][E].reciprocal();

for (int j = 0; j <= n; j++) {

if (j != E) {

a[L][j] = a[L][j].times(a[L][E].negate());

}

}

for (int i = 0; i <= m + 1; i++) {

if (i != L) {

for (int j = 0; j <= n; j++) {

if (j != E) {

a[i][j] = a[i][j].plus(a[L][j].times(a[i][E]));

}

}

a[i][E] = a[i][E].times(a[L][E]);

}

}

}

E = -1;

for (int j = 0; j < n; j++) {

if (E < 0 || index[E] > index[j]) {

if (a[m + 1][j].toDouble() > 0 || a[m + 1][j].toDouble() == 0 && a[m][j].toDouble() > 0) {

E = j;

}

}

}

if (E < 0) {

break;

}

L = -1;

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

if (a[i][E].toDouble() < 0) {

Rational d;

if (L < 0 || (d = a[L][n].divides(a[L][E]).minus(a[i][n].divides(a[i][E]))).toDouble() < 0 || d.toDouble() == 0

&& index[L + n] > index[i + n]) {

L = i;

}

}

}

if (L < 0) {

return Rational.POSITIVE\_INFINITY;

}

}

if (a[m + 1][n].toDouble() < 0) {

return null;

}

if (x != null) {

Arrays.fill(x, Rational.ZERO);

for (int i = 0; i < m; i++)

if (index[n + i] < n - 1)

x[index[n + i]] = a[i][n];

}

return a[m][n];

}

/\*\*

\* Приклад розв'язку

\* @param args командна строка

\*/

public static void main(String[] args) {

System.out.println("================================== Варіант 4");

long[][] a;

long[] b;

long[] c;

a = new long[][] { { 1, 1, 1, 1, -1, -1 }, { 0, 1, 1, -1, -1, -1 },

{ 0, 1, 0, 0, 0, -1 } };

b = new long[] { 1, 1, 2 };

c = new long[] { 1, 0, 1, 0, 0, 1 };

Rational[] x = new Rational[c.length];

Rational res = simplex(cnv(a), cnv(b), cnv(c), x);

System.out.println(Arrays.toString(x));

System.out.println("Інші приклади:");

System.out.println("=============================================");

a = new long[][] { { 3, 4, -3 }, { 5, -4, -3 }, { 7, 4, 11 } };

b = new long[] { 23, 10, 30 };

c = new long[] { -1, 1, 2 };

x = new Rational[c.length];

res = simplex(cnv(a), cnv(b), cnv(c), x);

System.out.println(new Rational(57, 8).equals(res));

System.out.println(Arrays.toString(x));

System.out.println("=============================================");

// нема допустимих рішень розв'язків (в розвязку присутні штучні змінні)

a = new long[][] { { 4, -1 }, { 2, 1 }, { -5, 2 } };

b = new long[] { 8, -10, 2 };

c = new long[] { 1, 1 };

res = simplex(cnv(a), cnv(b), cnv(c), null);

System.out.println(null == res);

System.out.println("=============================================");

//Останній рядок містить від'ємні елементи. Простір допустимих рішень необмежено. Рішення не існує.

a = new long[][] { { -4, 1 }, { -2, -1 }, { -5, 2 } };

b = new long[] { -8, -10, 2 };

c = new long[] { 1, 1 };

res = simplex(cnv(a), cnv(b), cnv(c), null);

System.out.println(Rational.POSITIVE\_INFINITY == res);

System.out.println("=============================================");

// нема допустимих рішень

a = new long[][] { { 1 }, { -1 } };

b = new long[] { 1, -2 };

c = new long[] { 0 };

res = simplex(cnv(a), cnv(b), cnv(c), null);

System.out.println(null == res);

System.out.println("=============================================");

// безліч розв'язків

a = new long[][] { { 1, 1 } };

b = new long[] { 0 };

c = new long[] { 1, 1 };

x = new Rational[c.length];

res = simplex(cnv(a), cnv(b), cnv(c), x);

System.out.println(Arrays.toString(x));

}

//створення одновимірного масиву дробів

public static Rational[] cnv(long[] a) {

Rational[] res = new Rational[a.length];

for (int i = 0; i < a.length; i++) {

res[i] = new Rational((int) a[i]);

}

return res;

}

//створення двовимірного масиву дробів

public static Rational[][] cnv(long[][] a) {

Rational[][] res = new Rational[a.length][];

for (int i = 0; i < a.length; i++) {

res[i] = cnv(a[i]);

}

return res;

}

}

package simplex.source;

/\*\*

\* <b>Клас для роботи з раціональними числоми у вигляді дробів</b>

\*/

public class Rational implements Comparable<Rational> {

/\*\*

\* <i>Нуль у вигляды дробу</i>

\* @see ONE

\* @see POSITIVE\_INFINITY

\*/

public static final Rational ZERO = new Rational(0, 1);

/\*\*

\* <i>Одиниця у вигляды дробу</i>

\*/

public static final Rational ONE = new Rational(1, 1);

/\*\*

\* <i>Додатня нескінченність у вигляды дробу</i>

\*/

public static final Rational POSITIVE\_INFINITY = new Rational(1, 0);

private int num; // чисельник

private int den; // знаменник

/\*\* <i>Конструктор створення дробу з чисельником та знаменником</i>

\* @param numerator чисельник

\* @param denominator значенник

\*/

public Rational(int numerator, int denominator) {

// скорочення дробу

int g = gcd(numerator, denominator);

num = numerator / g;

den = denominator / g;

// перенесення мінуса в чисельник

if (den < 0) { den = -den; num = -num; }

}

/\*\* <i>Конструктор створення дробу тільки за чисельником</i>

\* @param numerator чисельник

\*/

public Rational(int numerator) {

this(numerator, 1);

}

/\*\*

\* @return Чисельник

\*/

public int numerator() { return num; }

/\*\*

\* @return Знаменник

\*/

public int denominator() { return den; }

/\*\*

\* <i>Переведення дробу в Double</i>

\* @return Десяткове представлення дробу

\*/

public double toDouble() {

return (double) num / den;

}

/\*\*

\* @return Представлення у вигляды строки

\*/

@Override

public String toString() {

if (den == 1) return num + "";

else return num + "/" + den;

}

/\*\*

\* @param b інший дріб

\* @return -1, 0, +1

\*/

@Override

public int compareTo(Rational b) {

Rational a = this;

int lhs = a.num \* b.den;

int rhs = a.den \* b.num;

if (lhs < rhs) return -1;

if (lhs > rhs) return +1;

return 0;

}

/\*\*

\* Чи даний дріб є рівний <b>y</b>

\* @param y інший дріб

\* @return true/false

\*/

@Override

public boolean equals(Object y) {

if (y == null) return false;

if (y.getClass() != this.getClass()) return false;

Rational b = (Rational) y;

return compareTo(b) == 0;

}

/\*\*

\* Пов'язано з перевизначенням equals

\* @see equals

\* @return Хеш-код дробу

\*/

@Override

public int hashCode() {

return this.toString().hashCode();

}

/\*\*

\* Створює новий дріб: (r.num + s.num) / (r.den + s.den)

\* @param r Чисельник

\* @param s Знаменник

\* @return Результуючий дріб

\*/

public static Rational mediant(Rational r, Rational s) {

return new Rational(r.num + s.num, r.den + s.den);

}

/\*\*

\* Найбільший спільний дільник

\* @param m перше число

\* @param n друге число

\* @return Найбільший спільний дільник двох чисел

\*/

public static int gcd(int m, int n) {

if (m < 0) m = -m;

if (n < 0) n = -n;

if (0 == n) return m;

else return gcd(n, m % n);

}

/\*\*

\* Найменше спільне кратне

\* @param m перше число

\* @param n друге число

\* @return Найменше спільне кратне двох чисел

\*/

public static int lcm(int m, int n) {

if (m < 0) m = -m;

if (n < 0) n = -n;

return m \* (n / gcd(m, n));

}

/\*\*

\* добуток a \* b

\* @param b інший дріб

\* @return добуток двох дробів

\*/

public Rational times(Rational b) {

Rational a = this;

Rational c = new Rational(a.num, b.den);

Rational d = new Rational(b.num, a.den);

return new Rational(c.num \* d.num, c.den \* d.den);

}

/\*\*

\* сума a + b

\* @param b інший дріб

\* @return сума двох дробів

\*/

public Rational plus(Rational b) {

Rational a = this;

if (a.compareTo(ZERO) == 0) return b;

if (b.compareTo(ZERO) == 0) return a;

int f = gcd(a.num, b.num);

int g = gcd(a.den, b.den);

Rational s = new Rational((a.num / f) \* (b.den / g) + (b.num / f) \* (a.den / g),

lcm(a.den, b.den));

s.num \*= f;

return s;

}

/\*\*

\* -Дріб

\* @return дріб з мінусом

\*/

public Rational negate() {

return new Rational(-num, den);

}

/\*\*

\* a - b

\* @param b інший дріб

\* @return різниця

\*/

public Rational minus(Rational b) {

Rational a = this;

return a.plus(b.negate());

}

/\*\*

\* Обертання дробу

\* @return обернений дріб

\*/

public Rational reciprocal() { return new Rational(den, num); }

/\*\*

\* a / b

\* @param b інший дріб

\* @return часка з ділення

\*/

public Rational divides(Rational b) {

Rational a = this;

return a.times(b.reciprocal());

}

/\*\*

\* тестування

\* @param args командна строка

\*/

public static void main(String[] args) {

Rational x, y, z;

// 1/2 + 1/3 = 5/6

x = new Rational(1, 2);

y = new Rational(1, 3);

z = x.plus(y);

System.out.println(z);

// 8/9 + 1/9 = 1

x = new Rational(8, 9);

y = new Rational(1, 9);

z = x.plus(y);

System.out.println(z);

// 1/200000000 + 1/300000000 = 1/120000000

x = new Rational(1, 200000000);

y = new Rational(1, 300000000);

z = x.plus(y);

System.out.println(z);

// 1073741789/20 + 1073741789/30 = 1073741789/12

x = new Rational(1073741789, 20);

y = new Rational(1073741789, 30);

z = x.plus(y);

System.out.println(z);

// 4/17 \* 17/4 = 1

x = new Rational(4, 17);

y = new Rational(17, 4);

z = x.times(y);

System.out.println(z);

// 3037141/3247033 \* 3037547/3246599 = 841/961

x = new Rational(3037141, 3247033);

y = new Rational(3037547, 3246599);

z = x.times(y);

System.out.println(z);

// 1/6 - -4/-8 = -1/3

x = new Rational( 1, 6);

y = new Rational(-4, -8);

z = x.minus(y);

System.out.println(z);

}

}

**3. Протокол виконання програми**

Запустимо програму на виконання (рис. 3.1).

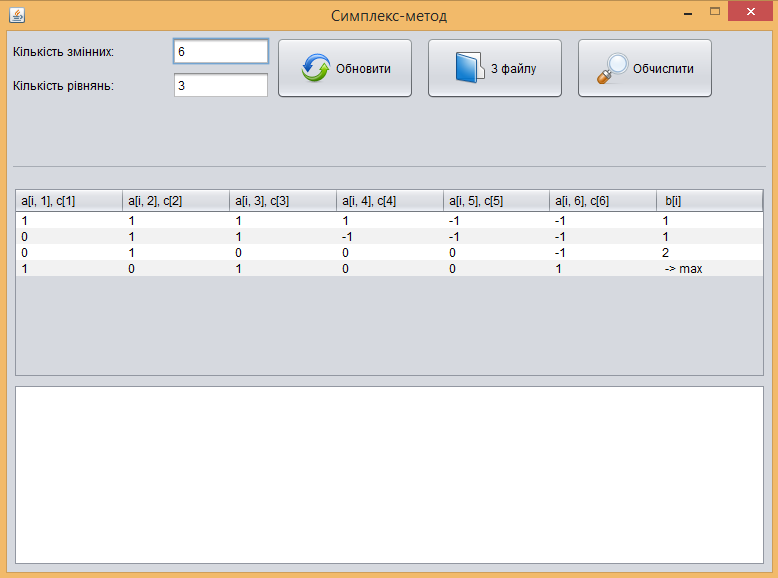


Рис. 3.1. Запуск програми

Натиснемо кнопку «Обчислити» (рис. 3.2).

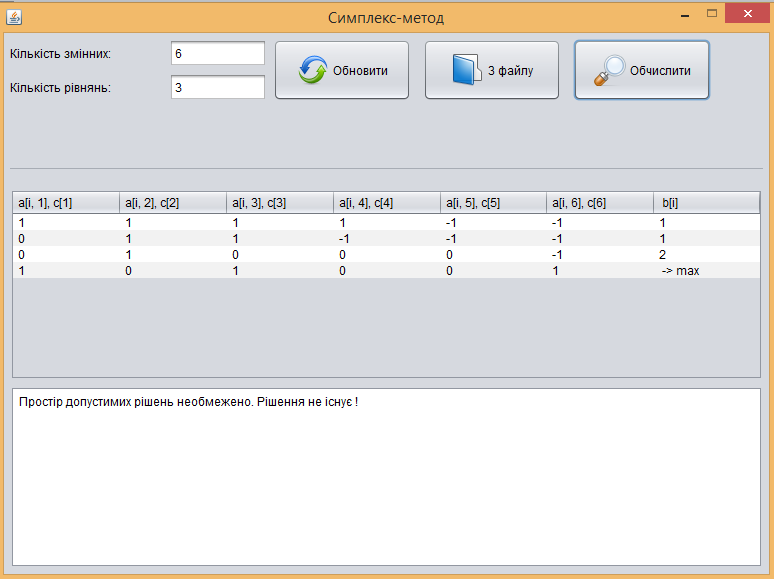


Рис. 3.2. Виконання індивідуального завдання

Перевіримо результати за допомогою сервісу: <http://math.semestr.ru/simplex/simplex.php> (рис 3.3):

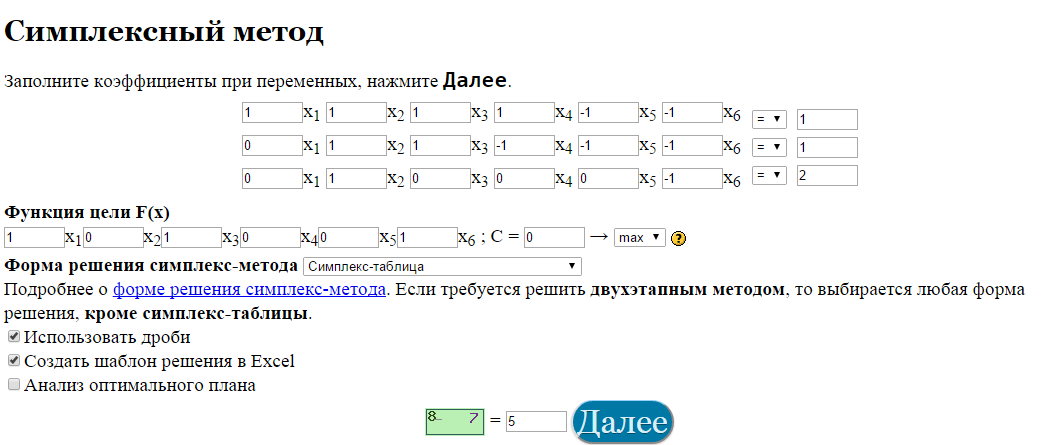


Рис. 3.3. Перевірка результатів

Натиснемо кнопку «Далее» для отримання результату: (рис 3.4):

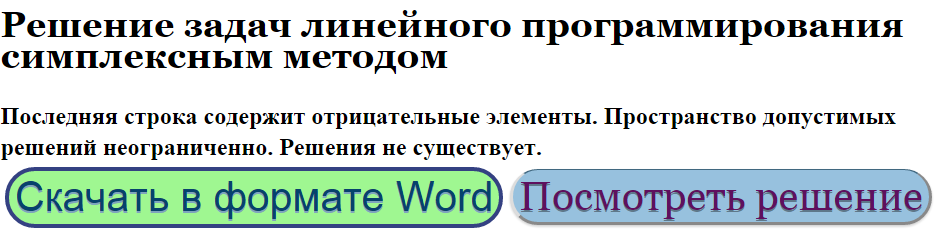


Рис. 3.4. Перевірка результатів. Рішення – вірне.

## ВИСНОВКИ

В ході даної лабораторної роботи я навчився на практиці розв’язувати задачі лінійного програмування симплекс-методом. А також програмуванню цього методу на мові програмування Java.