**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э.БАУМАНА**

**Факультет**: Информатика и системы управления  
**Кафедра**: ИУ8 - Информационная безопасность

**МЕТОДЫ ОПТИМИЗАЦИИ**

**Лабораторная работа №3**  
«Целочисленное линейное программирование. Метод ветвей и границ»

**Вариант** 3

**Преподаватель:**   
Коннова Н.С.

**Студент:**   
Волков Г.А.

**Группа:**   
ИУ8-35

Москва, 2019

**СОДЕРЖАНИЕ**

[**ОСНОВНАЯ ЧАСТЬ** 3](#_Toc21702608)

[Цель работы 3](#_Toc21702609)

[Постановка задачи 3](#_Toc21702610)

[Ход работы 3](#_Toc21702611)

[**ЗАКЛЮЧЕНИЕ** 18](#_Toc21702612)

[**ПРИЛОЖЕНИЕ** 19](#_Toc21702613)

# **ОСНОВНАЯ ЧАСТЬ**

## Цель работы

Изучить постановку задачи целочисленного линейного программирования; получить решение задачи ЦЛП методом ветвей и границ.

## Постановка задачи

Требуется найти решение следующей задачи:

xi∈Z0

Необходимо найти вектор удовлетворяющий условиям, при котором целевой функции (ЦФ) стремится к максимуму. Здесь:

c – вектор коэффициентов ЦФ,  
A – матрица системы ограничений,  
b– вектор правой части системы ограничений.

По условию:

## Ход работы

Приведём исходные данные задачи к каноническому виду:

Выразим базисные переменные в уравнениях:

Тогда исходная симплекс-таблица будет иметь вид:

Default symplic table

4.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 0.00 X5

2.00 0.00 0.50 1.00 X6

0.00 2.00 8.00 3.00 F

S X1 X2 X3

Razr stolb X1

Razr stroka X4

Symplic table 1

2.00 0.50 0.50 0.50 X1

4.00 -0.50 1.50 -0.50 X5

2.00 0.00 0.50 1.00 X6

-4.00 -1.00 7.00 2.00 F

S X4 X2 X3

Razr stolb X3

Razr stroka X6

Symplic table 2

1.00 0.50 0.25 -0.50 X1

5.00 -0.50 1.75 0.50 X5

2.00 0.00 0.50 1.00 X3

-8.00 -1.00 6.00 -2.00 F

S X4 X2 X6

Razr stolb X2

Razr stroka X5

Symplic table 3

0.29 0.57 -0.14 -0.57 X1

2.86 -0.29 0.57 0.29 X2

0.57 0.14 -0.29 0.86 X3

-25.14 0.71 -3.43 -3.71 F

S X4 X5 X6

Razr stolb X4

Razr stroka X1

Symplic table 4

0.50 1.75 -0.25 -1.00 X4

3.00 0.50 0.50 0.00 X2

0.50 -0.25 -0.25 1.00 X3

-25.50 -1.25 -3.25 -3.00 F

S X1 X5 X6

Symplic table

0.50 1.75 -0.25 -1.00 X4

3.00 0.50 0.50 0.00 X2

0.50 -0.25 -0.25 1.00 X3

-25.50 -1.25 -3.25 -3.00 F

S X1 X5 X6

Начнём ветвление x3<=0

Start X3 <= 0.0

def tab

4.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 0.00 X5

2.00 0.00 0.50 1.00 X6

0.00 0.00 0.00 1.00 X7

0.00 2.00 8.00 3.00 F

S X1 X2 X3

Razr stolb X1

Razr stroka X4

Symplic table 1

2.00 0.50 0.50 0.50 X1

4.00 -0.50 1.50 -0.50 X5

2.00 0.00 0.50 1.00 X6

0.00 0.00 0.00 1.00 X7

-4.00 -1.00 7.00 2.00 F

S X4 X2 X3

Razr stolb X3

Razr stroka X7

Symplic table 2

2.00 0.50 0.50 -0.50 X1

4.00 -0.50 1.50 0.50 X5

2.00 0.00 0.50 -1.00 X6

0.00 0.00 0.00 1.00 X3

-4.00 -1.00 7.00 -2.00 F

S X4 X2 X7

Razr stolb X2

Razr stroka X5

Symplic table 3

0.67 0.67 -0.33 -0.67 X1

2.67 -0.33 0.67 0.33 X2

0.67 0.17 -0.33 -1.17 X6

0.00 0.00 -0.00 1.00 X3

-22.67 1.33 -4.67 -4.33 F

S X4 X5 X7

Razr stolb X4

Razr stroka X1

Symplic table 4

1.00 1.50 -0.50 -1.00 X4

3.00 0.50 0.50 0.00 X2

0.50 -0.25 -0.25 -1.00 X6

0.00 -0.00 0.00 1.00 X3

-24.00 -2.00 -4.00 -3.00 F

S X1 X5 X7

Symplic table

1.00 1.50 -0.50 -1.00 X4

3.00 0.50 0.50 0.00 X2

0.50 -0.25 -0.25 -1.00 X6

0.00 -0.00 0.00 1.00 X3

-24.00 -2.00 -4.00 -3.00 F

S X1 X5 X7

Получен ответ:

Answer:

F = 24.000

X1 = 0.00

X2 = 3.00

X3 = 0.00

End X3 <= 0.0

Начнём ветвление X3 >= 1

Start X3 >= 1.0

def tab

4.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 0.00 X5

2.00 0.00 0.50 1.00 X6

-1.00 0.00 0.00 -1.00 X7

0.00 2.00 8.00 3.00 F

S X1 X2 X3

Найдём опорное решение:

Opor Symplics table 1

3.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 -0.00 X5

1.00 0.00 0.50 1.00 X6

1.00 -0.00 -0.00 -1.00 X3

-3.00 2.00 8.00 3.00 F

S X1 X2 X7

Razr stolb X1

Razr stroka X4

Symplic table 1

1.50 0.50 0.50 0.50 X1

4.50 -0.50 1.50 -0.50 X5

1.00 -0.00 0.50 1.00 X6

1.00 0.00 0.00 -1.00 X3

-6.00 -1.00 7.00 2.00 F

S X4 X2 X7

Razr stolb X7

Razr stroka X6

Symplic table 2

1.00 0.50 0.25 -0.50 X1

5.00 -0.50 1.75 0.50 X5

1.00 -0.00 0.50 1.00 X7

2.00 0.00 0.50 1.00 X3

-8.00 -1.00 6.00 -2.00 F

S X4 X2 X6

Razr stolb X2

Razr stroka X7

Symplic table 3

0.50 0.50 -0.50 -1.00 X1

1.50 -0.50 -3.50 -3.00 X5

2.00 -0.00 2.00 2.00 X2

1.00 0.00 -1.00 0.00 X3

-20.00 -1.00 -12.00 -14.00 F

S X4 X7 X6

Symplic table

0.50 0.50 -0.50 -1.00 X1

1.50 -0.50 -3.50 -3.00 X5

2.00 -0.00 2.00 2.00 X2

1.00 0.00 -1.00 0.00 X3

-20.00 -1.00 -12.00 -14.00 F

S X4 X7 X6

Начнём ветвление по переменной X1.

Start X1 <= 0.0

def tab

4.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 0.00 X5

2.00 0.00 0.50 1.00 X6

-1.00 0.00 0.00 -1.00 X7

0.00 1.00 0.00 0.00 X8

0.00 2.00 8.00 3.00 F

S X1 X2 X3

Найдём опорное решение:

Opor Symplics table 1

3.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 -0.00 X5

1.00 0.00 0.50 1.00 X6

1.00 -0.00 -0.00 -1.00 X3

0.00 1.00 0.00 -0.00 X8

-3.00 2.00 8.00 3.00 F

S X1 X2 X7

End opor

Razr stolb X1

Razr stroka X8

Symplic table 1

3.00 -2.00 1.00 1.00 X4

6.00 -1.00 2.00 0.00 X5

1.00 -0.00 0.50 1.00 X6

1.00 0.00 0.00 -1.00 X3

0.00 1.00 0.00 -0.00 X1

-3.00 -2.00 8.00 3.00 F

S X8 X2 X7

Razr stolb X7

Razr stroka X6

Symplic table 2

2.00 -2.00 0.50 -1.00 X4

6.00 -1.00 2.00 -0.00 X5

1.00 -0.00 0.50 1.00 X7

2.00 0.00 0.50 1.00 X3

0.00 1.00 0.00 0.00 X1

-6.00 -2.00 6.50 -3.00 F

S X8 X2 X6

Razr stolb X2

Razr stroka X7

Symplic table 3

1.00 -2.00 -1.00 -2.00 X4

2.00 -1.00 -4.00 -4.00 X5

2.00 -0.00 2.00 2.00 X2

1.00 0.00 -1.00 0.00 X3

0.00 1.00 -0.00 0.00 X1

-19.00 -2.00 -13.00 -16.00 F

S X8 X7 X6

Symplic table

1.00 -2.00 -1.00 -2.00 X4

2.00 -1.00 -4.00 -4.00 X5

2.00 -0.00 2.00 2.00 X2

1.00 0.00 -1.00 0.00 X3

0.00 1.00 -0.00 0.00 X1

-19.00 -2.00 -13.00 -16.00 F

S X8 X7 X6

Получен ответ:

Answer:

F = 19.000

X1 = 0.00

X2 = 2.00

X3 = 1.00

End X1 <= 0.0

Продолжаем ветвление по переменной X1:

Start X1 >= 1.0

def tab

4.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 0.00 X5

2.00 0.00 0.50 1.00 X6

-1.00 0.00 0.00 -1.00 X7

-1.00 -1.00 0.00 0.00 X8

0.00 2.00 8.00 3.00 F

S X1 X2 X3

Найдём опорное решение:

Opor Symplics table 1

3.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 -0.00 X5

1.00 0.00 0.50 1.00 X6

1.00 -0.00 -0.00 -1.00 X3

-1.00 -1.00 0.00 -0.00 X8

-3.00 2.00 8.00 3.00 F

S X1 X2 X7

Opor Symplics table 2

1.00 2.00 1.00 1.00 X4

5.00 1.00 2.00 -0.00 X5

1.00 0.00 0.50 1.00 X6

1.00 -0.00 -0.00 -1.00 X3

1.00 -1.00 -0.00 0.00 X1

-5.00 2.00 8.00 3.00 F

S X8 X2 X7

End opor

Razr stolb X8

Razr stroka X4

Symplic table 1

0.50 0.50 0.50 0.50 X8

4.50 -0.50 1.50 -0.50 X5

1.00 -0.00 0.50 1.00 X6

1.00 0.00 0.00 -1.00 X3

1.50 0.50 0.50 0.50 X1

-6.00 -1.00 7.00 2.00 F

S X4 X2 X7

Razr stolb X7

Razr stroka X8

Symplic table 2

1.00 1.00 1.00 2.00 X7

5.00 0.00 2.00 1.00 X5

0.00 -1.00 -0.50 -2.00 X6

2.00 1.00 1.00 2.00 X3

1.00 0.00 0.00 -1.00 X1

-8.00 -3.00 5.00 -4.00 F

S X4 X2 X8

Razr stolb X2

Razr stroka X7

Symplic table 3

1.00 1.00 1.00 2.00 X2

3.00 -2.00 -2.00 -3.00 X5

0.50 -0.50 0.50 -1.00 X6

1.00 0.00 -1.00 0.00 X3

1.00 0.00 -0.00 -1.00 X1

-13.00 -8.00 -5.00 -14.00 F

S X4 X7 X8

Symplic table

1.00 1.00 1.00 2.00 X2

3.00 -2.00 -2.00 -3.00 X5

0.50 -0.50 0.50 -1.00 X6

1.00 0.00 -1.00 0.00 X3

1.00 0.00 -0.00 -1.00 X1

-13.00 -8.00 -5.00 -14.00 F

S X4 X7 X8

Получили решение:

Answer:

F = 13.000

X1 = 1.00

X2 = 1.00

X3 = 1.00

End X1 >= 1.0

End X3 >= 1.0

Итоговый ответ:

Fmax = 24.0

X1 = 0.00

X2 = 3.00

X3 = 0.00

Начнём находить максимум методом прямого перебора. Выводятся только значения, превышающие предыдущий максимум.

Start brute force

0 [0, 0, 0]

3 [0, 0, 1]

6 [0, 0, 2]

8 [0, 1, 0]

11 [0, 1, 1]

16 [0, 2, 0]

19 [0, 2, 1]

24 [0, 3, 0]

Получен ответ:

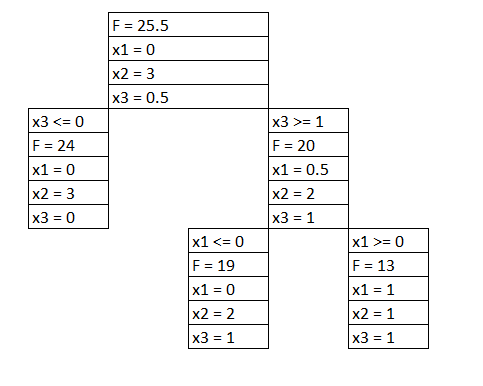
Fmax = 24

X1 = 0.00

X2 = 3.00

X3 = 0.00

Представим ход работы нашей программы для данной задачи в виде дерева:



# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения лабораторной работы мы изучили основные приёмы использования метода ветвей и границ. Научились ставить и решать прямую задачу линейного программирования с помощью этого метода, а так же узнали, что он является наиболее универсальным из вычислительных методов для целочисленного линейного программирования.

Также мы убедились, что метод прямого перебора даёт такой же ответ, как и метод ветвей и границ.

Код, реализующий симплекс-метод, написан на языке Python версии 3.\* и представлен в приложении.

# **ПРИЛОЖЕНИЕ**

**import** copy  
  
flEnd = **False**base = set()  
bbase = set()  
answ = dict()  
  
  
**def** printWithXInTable(tab):  
 m = len(tab)  
 n = len(tab[0])  
 print()  
 **for** i **in** range(m - 1):  
 **for** j **in** range(n - 1):  
 print(**'{:>7.2f}'**.format(tab[i][j]), end=**""**)  
 pam = str(int(tab[i][-1]))  
 **if** pam == **"88"**:  
 pam = **" F"  
 elif** pam == **"99"**:  
 pam = **""  
 else**:  
 pam = **" X"** + pam  
 print(pam)  
 **for** i **in** range(n):  
 pam = str(int(tab[-1][i]))  
 **if** pam == **"77"**:  
 pam = **" S"  
 elif** pam == **"99"**:  
 pam = **""  
 else**:  
 pam = **" X"** + pam  
 print(**'{:>7s}'**.format(pam), end=**""**)  
 print()  
  
  
**def** printAnswer(n, flag, tab):  
 **if** flag == **"max"**:  
 zn = -1  
 **else**:  
 zn = 1  
 *# print("\n" + "Answer:\n" + "F" + str(flag) + " = " + " " + '{:<9.3f}'.format(tab[-2][0] \* zn))* print(**"\n"** + **"Answer:\n"** + **"F"** + **" = "** + **" "** + **'{:<9.3f}'**.format(tab[-2][0] \* zn))  
 le = len(bbase)  
 pam = [0] \* (le + 1)  
 **for** i **in** range(le):  
 pam[i] = 0  
 m = len(tab)  
 **for** i **in** range(m):  
 **if** tab[i][-1] **in** bbase:  
 pam[tab[i][-1]] = tab[i][0]  
 **for** i **in** range(1, le + 1):  
 print(**"X"** + str(i) + **" = "** + **'{:<7.2f}'**.format(pam[i]))  
 print()  
 answ[tab[-2][0] \* zn] = pam  
  
  
**def** printOpor(tab):  
 print(**"\n"** + **"Opor answer:\n"** + **"F"** + **" = "** + **" "** + **'{:<9.3f}'**.format(-tab[-2][0]))  
 n = len(bbase)  
 pam = [0] \* (n + 1)  
  
 **for** i **in** range(n + 1):  
 pam[i] = 0  
 **for** i **in** range(len(tab)):  
 **if** tab[i][-1] **in** bbase:  
 pam[tab[i][-1]] = tab[i][0]  
 **for** i **in** range(1, len(bbase) + 1):  
 print(**"X"** + str(i) + **" = "** + **'{:<7.2f}'**.format(pam[i]))  
  
 answ[tab[-2][0] \* (-1)] = pam  
 print()  
 print(\*answ)  
  
  
**def** createSimplexTable(flag, n, a, b, c):  
 tab = [[0] \* (n + 2) **for** i **in** range(n + 2)]  
 **if** flag == **"max"**:  
 flag = 1  
 **else**:  
 flag = -1  
 **for** i **in** range(n):  
 tab[i][0] = b[i] \* flag  
 **for** j **in** range(1, n + 1):  
 tab[i][j] = a[i][j - 1] \* flag  
 tab[n][i + 1] = flag \* c[i] *# +-* tab[n][0] = 0 *# F(s)* **for** i **in** range(n + 1):  
 tab[i][n + 1] = i + n + 1  
 tab[n + 1][i] = i  
 tab[n + 1][0] = 77 *# S* tab[n][n + 1] = 88 *# F* tab[n + 1][n + 1] = 99 *# empty* **return** tab  
  
  
**def** findOpor(tabA):  
 print(**"Start opor"**)  
 kol = 0  
 m = len(tabA)  
 n = len(tabA[0])  
 **while True**:  
 kol += 1  
 ma = -1  
 k = -1  
 fl = **True  
 for** i **in** range(0, m - 2):  
 **if** tabA[i][0] < 0:  
 pam = i  
 fl = **False  
 break  
 if** fl:  
 **break  
 for** i **in** range(1, n - 1):  
 **if** tabA[pam][i] < 0:  
 k = i  
 **break  
 if** k == -1:  
 print(**'{:>28s}'**.format(**"Error"**))  
 **break** mi = 10000000  
 r = -1  
 **for** i **in** range(0, m - 2):  
 **if** tabA[i][k] != 0:  
 **if** tabA[i][0] / tabA[i][k] < mi **and** tabA[i][0] / tabA[i][k] >= 0:  
 mi = tabA[i][0] / tabA[i][k]  
 r = i  
 **if** r == -1:  
 *# print('{:>28s}'.format("No more positive in S/ Xk"))* **break** *# print('{:<28s}'.format("Razr stolb " + "X" + str(tabA[-1][k])))  
 # print('{:<28s}'.format("Razr stroka " + "X" + str(tabA[r][-1])), end="\n\n")* tab2 = copy.deepcopy(tabA)  
 tabA[r][k] = 1 / tab2[r][k]  
 **for** i **in** range(m - 1):  
 **for** j **in** range(n - 1):  
 **if** j != k:  
 tabA[r][j] = tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tabA[i][j] = tab2[i][j] - tab2[i][k] \* tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tabA[i][k] = -tab2[i][k] / tab2[r][k]  
 tabA[-1][k], tabA[r][-1] = tabA[r][-1], tabA[-1][k]print(**'{:>28s}'**.format(**"Opor Symplics table "** + str(kol)))  
 printWithXInTable(tabA)print(**"End opor"**)  
 **return** tabA  
  
  
**def** calculateSimplexTable(tabB):  
 pam = copy.deepcopy(tabB)  
 tabB = findOpor(copy.deepcopy(tabB))kol = 0  
 m = len(tabB)  
 n = len(tabB[0])  
 flEnd = **False  
 while True**:  
 kol += 1  
 mi = 10000000  
 k = -1  
 **for** i **in** range(1, n - 1):  
 **if** tabB[-2][i] >= 0 **and** tabB[-2][i] < mi:  
 mi = tabB[-2][i]  
 k = i  
 **if** k == -1:  
 *# print('{:>28s}'.format("No more positive in F"))* flEnd = **True  
 break** mi = 10000000  
 r = -1  
 **for** i **in** range(0, m - 2):  
 **if** tabB[i][k] != 0:  
 **if** tabB[i][0] / tabB[i][k] < mi **and** tabB[i][0] / tabB[i][k] >= 0 **and** tabB[i][k] > 0:  
 mi = tabB[i][0] / tabB[i][k]  
 r = i  
 **if** r == -1:  
 *# print('{:>28s}'.format("No more positive in S/ Xk"))* flEnd = **True  
 break** print(**'{:<28s}'**.format(**"Razr stolb "** + **"X"** + str(tabB[-1][k])))  
 print(**'{:<28s}'**.format(**"Razr stroka "** + **"X"** + str(tabB[r][-1])), end=**"\n\n"**)  
 tab2 = copy.deepcopy(tabB)  
 tabB[r][k] = 1 / tab2[r][k]  
 **for** i **in** range(m - 1):  
 **for** j **in** range(n - 1):  
 **if** j != k:  
 tabB[r][j] = tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tabB[i][j] = tab2[i][j] - tab2[i][k] \* tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tabB[i][k] = -tab2[i][k] / tab2[r][k]  
 tabB[-1][k], tabB[r][-1] = tabB[r][-1], tabB[-1][k]  
 print(**'{:>28s}'**.format(**"Symplic table "** + str(kol)))  
 printWithXInTable(tabB)  
 print(**'{:>28s}'**.format(**"Symplic table "**), end=**""**)  
 printWithXInTable(tabB)  
 flEnd = **True  
 return** tabB  
  
  
**def** checkFree(tabC):  
 m = len(tabC)  
 n = len(tabC[0])  
 **for** i **in** range(0, m):  
 **if** (tabC[i][0] % 1 != 0) **and** (tabC[i][-1] **in** base):  
 **return** i  
 **return** -1  
  
  
**def** checkCalcTable(tab):  
 m = len(tab)  
 n = len(tab[0])  
 fl\_e = **True  
 for** i **in** range(n - 1):  
 **if** tab[-2][i] >= 0:  
 fl\_e = **False  
 break  
 if** fl\_e:  
 **for** i **in** range(m - 2):  
 **if** tab[i][0] < 0:  
 **return True  
 return False  
  
  
def** addInDefTab(fl, nameX, num, tabIn):  
 m = len(tabIn)  
 n = len(tabIn[0])  
 tab2 = [[0] \* (n) **for** i **in** range(m + 1)]  
 **if** fl == **"<"**:  
 fl = 1  
 **else**:  
 fl = -1  
 **for** i **in** range(m - 2):  
 **for** j **in** range(n):  
 tab2[i][j] = tabIn[i][j]  
 **for** i **in** range(n):  
 tab2[-3][i] = 0  
 **if** tabIn[-1][i] == nameX:  
 tab2[-3][i] = fl  
 tab2[-3][0] = num \* fl  
 tab2[-3][-1] = m + n - 3  
 **for** j **in** range(n):  
 tab2[-2][j] = tabIn[-2][j]  
 **for** j **in** range(1, n):  
 tab2[-1][j] = tabIn[-1][j]  
 tab2[-1][0] = 77 *# S* tab2[-2][-1] = 88 *# F* tab2[-1][-1] = 99 *# empty* **return** tab2  
  
  
**def** branchAndBound(tabI, dTab):  
 i = checkFree(tabI)  
 **if** i == -1:  
 printAnswer(len(tabI[0]), **"max"**, tabI)  
 **return  
 if** checkCalcTable(tabI):  
 print(**"exit"**)  
 **return** num1 = tabI[i][0] - tabI[i][0] % 1  
 print(**"\n"**, **"Start X"**, tabI[i][-1], **" <= "**, num1, sep=**''**)  
 dTab1 = addInDefTab(**"<"**, tabI[i][-1], num1, copy.deepcopy(dTab))  
 print(**"def tab"**, end=**""**)  
 printWithXInTable(dTab1)  
 tab1 = calculateSimplexTable(copy.deepcopy(dTab1))branchAndBound(copy.deepcopy(tab1), copy.deepcopy(dTab1))  
 print(**"End "**, **"X"**, tabI[i][-1], **" <= "**, num1, sep=**''**)  
 print(**"\n"**, **"Start X"**, tabI[i][-1], **" >= "**, num1 + 1, sep=**''**)  
 **if** checkCalcTable(tabI):  
 print(**"exit"**)  
 **return** dTab2 = addInDefTab(**">"**, tabI[i][-1], num1 + 1, copy.deepcopy(dTab))  
 print(**"def tab"**, end=**""**)  
 printWithXInTable(dTab2)  
 tab2 = calculateSimplexTable(copy.deepcopy(dTab2))branchAndBound(copy.deepcopy(tab2), copy.deepcopy(dTab2))  
 print(**"End "**, **"X"**, tabI[i][-1], **" >= "**, num1 + 1, sep=**''**)  
  
  
**def** bruteForce():  
 a = getA()  
 b = getB()  
 c = getC()  
 len1 = len(a)  
 d = dict()  
 ma = -10000000  
 print()  
 print(**"Start brute force"**)  
 **for** i **in** range(100):  
 **for** j **in** range(100):  
 **for** k **in** range(100):  
 fl = **True** pam = 0  
 **for** ii **in** range(len1):  
 pam += a[ii][0] \* i + a[ii][1] \* j + a[ii][2] \* k  
 **if** pam > b[ii]:  
 fl = **False  
 break** pam = 0  
 **if** fl:  
 **if** c[0] \* i + c[1] \* j + c[2] \* k > ma:  
 ma = c[0] \* i + c[1] \* j + c[2] \* k  
 d[ma] = [i, j, k]  
 print(ma, d[ma])  
 print(**"Fmax ="**, ma)  
 **for** i **in** range(1, len(bbase) + 1):  
 print(**"X"** + str(i) + **" = "** + **'{:<7.2f}'**.format(d[ma][i - 1]))  
  
  
**def** getA():  
 a = [[2, 1, 1],  
 [1, 2, 0],  
 [0, 0.5, 1]]  
 **return** a  
  
  
**def** getC():  
 c = [2, 8, 3]  
 **return** c  
  
  
**def** getB():  
 b = [4, 6, 2]  
 **return** b  
  
  
**def** main():  
 n = 3  
 m = 3  
 flag = **"max"** tab = createSimplexTable(flag, n, getA(), getB(), getC()) **for** i **in** range(1, n + 1):  
 base.add(i)  
 bbase.add(i)  
 print(**'{:>28s}'**.format(**"Default symplic table"**), end=**""**)  
 printWithXInTable(tab)  
 num = 0  
 pam = calculateSimplexTable(copy.deepcopy(tab))  
 branchAndBound(copy.deepcopy(pam), copy.deepcopy(tab))  
 print()  
 pam = sorted(answ)[-1]  
 print(**"F"** + str(flag) + **" = "** + str(pam))  
 pam = answ[pam]  
 **for** i **in** range(1, len(bbase) + 1):  
 print(**"X"** + str(i) + **" = "** + **'{:<7.2f}'**.format(pam[i]))  
 bruteForce()  
  
  
main()