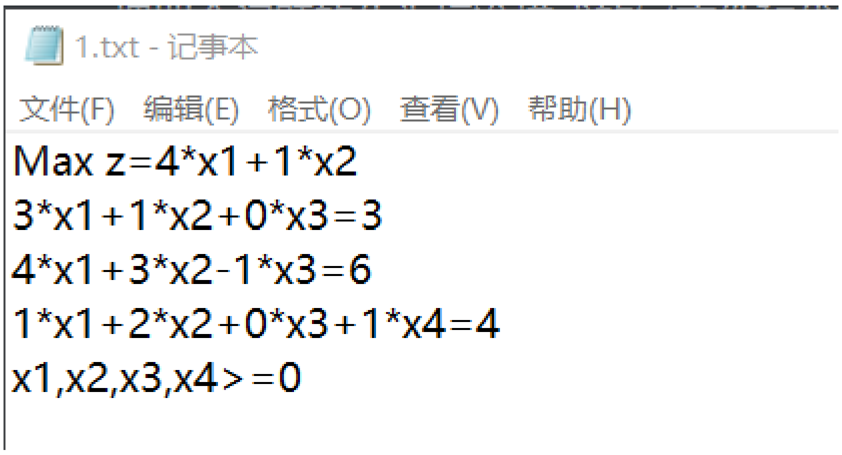
# 单纯形表法的代码实践

> 车宇庚 大数据202 2029730202

使用pycharm IDE进行样例求解 使用python的numpy库进行数学计算 (代码附在最后)

使用固定的格式把题写在文本文档中使用python的的IO流读取,分离数据元素

如图:



```
Max z=4*x1+1*x2
3*x1+1*x2+0*x3=3
4*x1+3*x2-1*x3=6
1*x1+2*x2+0*x3+1*x4=4
x1,x2,x3,x4>=0
```

四个样例:

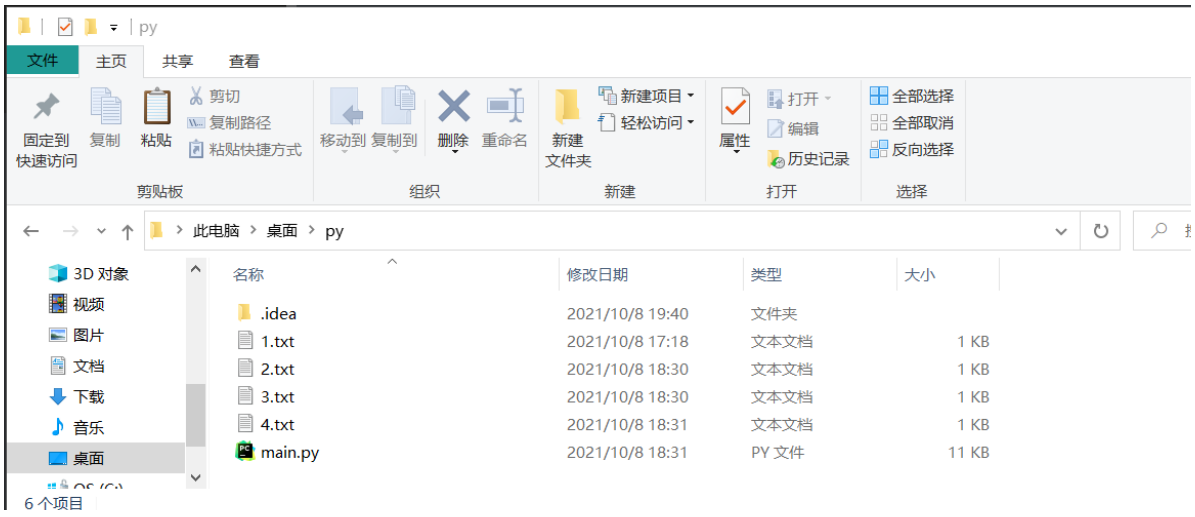
$$maxZ = 3x_1 - x_2 + 2x_3 \qquad minZ = 2x_1 + 3x_2 + x_3$$

$$s.t. = \begin{cases} x_1 + x_2 + x_3 \geq 6 \\ -2x_1 \quad + x_3 \geq 2 \\ 2x_2 - x_3 = 0 \\ x_j \geq 0 (j = 1, 2, 3) \end{cases} \qquad s.t. = \begin{cases} x_1 + 4x_2 + 2x_3 \geq 8 \\ 3x_1 + 2x_3 \quad \geq 6 \\ x_j \geq 0 (j = 1, 2) \end{cases}$$

$$maxZ = 10x_1 + 15x_2 + 12x_3 \qquad maxZ = 4x_1 + x_2$$

$$s.t. = \begin{cases} 5x_1 + 3x_2 + x_3 \leq 9 \\ -5x_1 + 6x_2 + 15x_3 \leq 15 \\ 2x_1 + x_2 + x_3 \geq 5 \\ x_j \geq 0 (j = 1, 2, 3) \end{cases} \qquad s.t. = \begin{cases} 3x_1 + x_2 \quad = 3 \\ 4x_1 + 3x_2 - x_3 = 6 \\ x_1 + 2x_2 \quad + x_4 = 4 \\ x_j \geq 0 (j = 1, 2, 3, 4) \end{cases}$$

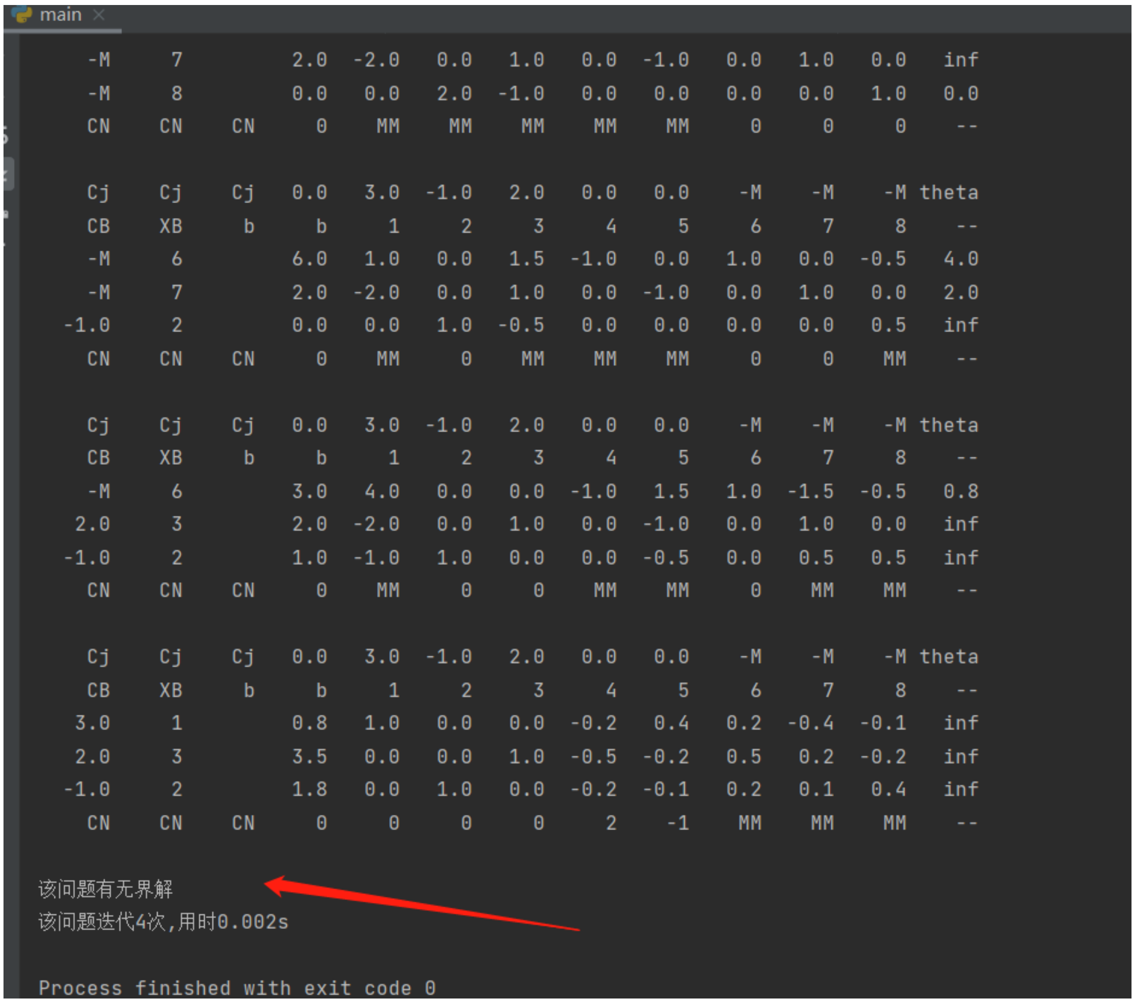
把四个问题化为标准模式的 txt 文件和代码一同放在一个文件夹中运行程序就可以自动解决单纯形问题



```
名称        修改日期          类型      大小
.idea       2021/10/8 19:40   文件夹
1.txt       2021/10/8 17:18   文本文档  1 KB
2.txt       2021/10/8 18:30   文本文档  1 KB
3.txt       2021/10/8 18:30   文本文档  1 KB
4.txt       2021/10/8 18:31   文本文档  1 KB
main.py     2021/10/8 18:31   PY 文件   11 KB
```

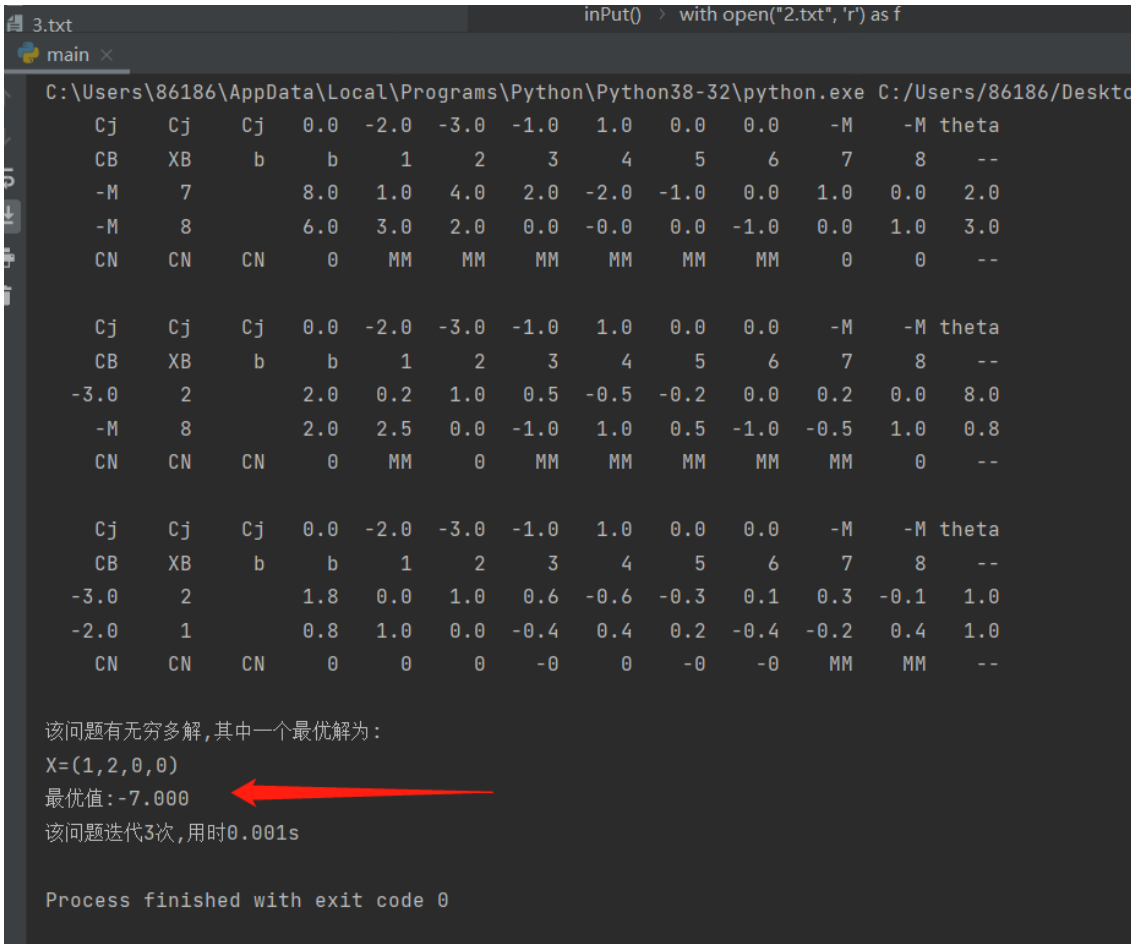
得到的结果

- NO.1



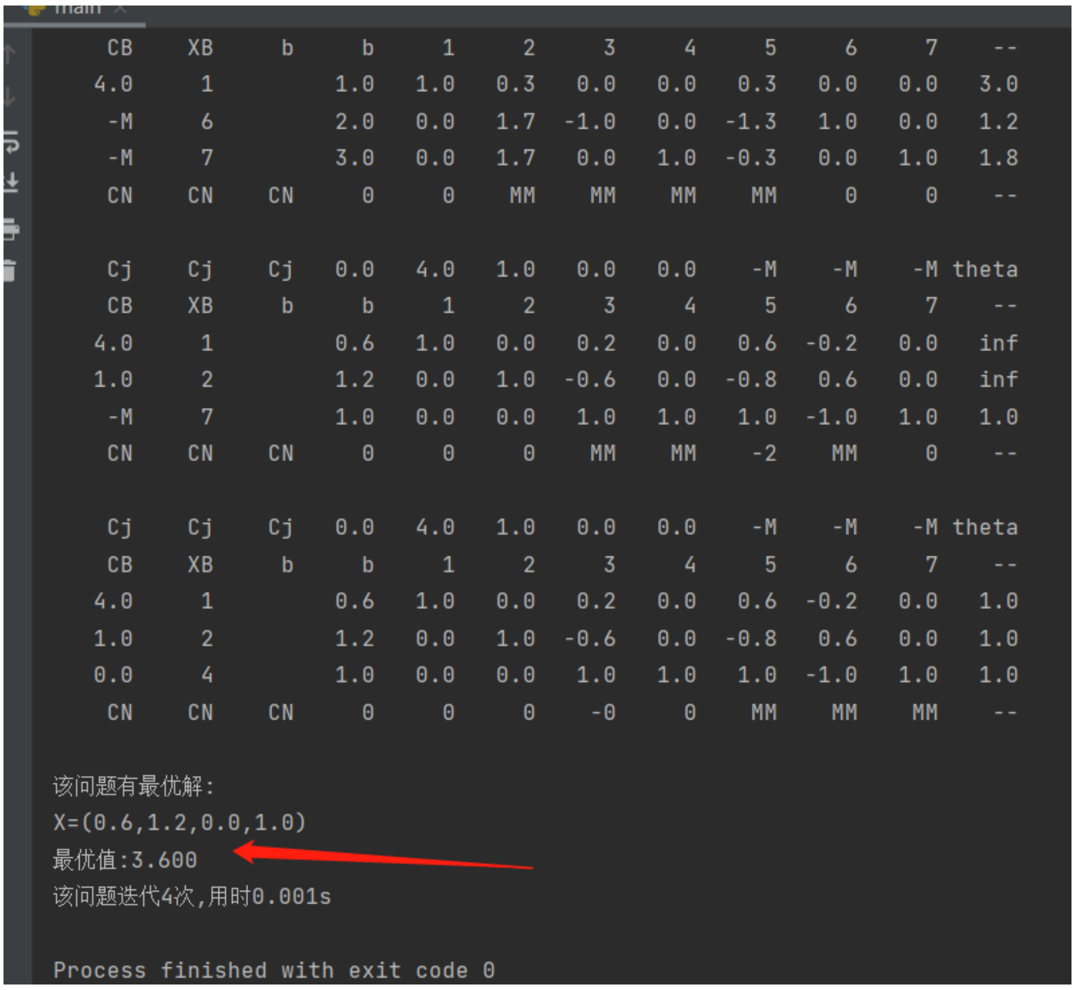
```
   -M     7         2.0  -2.0   0.0   1.0   0.0  -1.0   0.0   1.0   0.0   inf
   -M     8         0.0   0.0   2.0  -1.0   0.0   0.0   0.0   0.0   1.0   0.0
   CN    CN    CN     0    MM    MM    MM    MM    MM     0     0     0    --

   Cj    Cj    Cj   0.0   3.0  -1.0   2.0   0.0   0.0    -M    -M    -M theta
   CB    XB     b     b     1     2     3     4     5     6     7     8    --
   -M     6         6.0   1.0   0.0   1.5  -1.0   0.0   1.0   0.0  -0.5   4.0
   -M     7         2.0  -2.0   0.0   1.0   0.0  -1.0   0.0   1.0   0.0   2.0
 -1.0     2         0.0   0.0   1.0  -0.5   0.0   0.0   0.0   0.0   0.5   inf
   CN    CN    CN     0    MM     0    MM    MM    MM     0     0    MM    --

   Cj    Cj    Cj   0.0   3.0  -1.0   2.0   0.0   0.0    -M    -M    -M theta
   CB    XB     b     b     1     2     3     4     5     6     7     8    --
   -M     6         3.0   4.0   0.0   0.0  -1.0   1.5   1.0  -1.5  -0.5   0.8
  2.0     3         2.0  -2.0   0.0   1.0   0.0  -1.0   0.0   1.0   0.0   inf
 -1.0     2         1.0  -1.0   1.0   0.0   0.0  -0.5   0.0   0.5   0.5   inf
   CN    CN    CN     0    MM     0     0    MM    MM     0    MM    MM    --

   Cj    Cj    Cj   0.0   3.0  -1.0   2.0   0.0   0.0    -M    -M    -M theta
   CB    XB     b     b     1     2     3     4     5     6     7     8    --
  3.0     1         0.8   1.0   0.0   0.0  -0.2   0.4   0.2  -0.4  -0.1   inf
  2.0     3         3.5   0.0   0.0   1.0  -0.5  -0.2   0.5   0.2  -0.2   inf
 -1.0     2         1.8   0.0   1.0   0.0  -0.2  -0.1   0.2   0.1   0.4   inf
   CN    CN    CN     0     0     0     0     2    -1    MM    MM    MM    --

该问题有无界解
该问题迭代4次,用时0.002s

Process finished with exit code 0
```

- NO.2



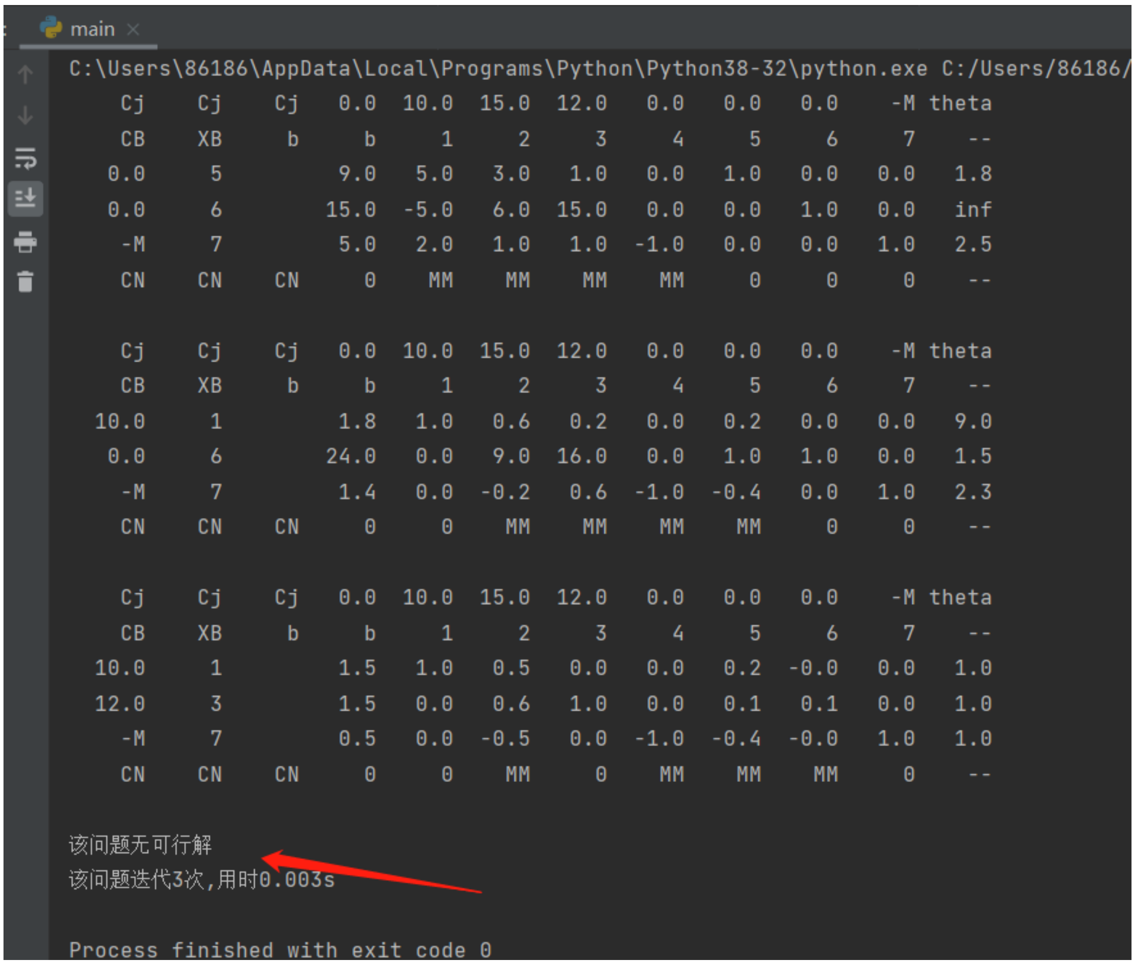
```
C:\Users\86186\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/86186/Deskto
    Cj    Cj    Cj   0.0  -2.0  -3.0  -1.0   1.0   0.0   0.0    -M    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7     8    --
    -M     7         8.0   1.0   4.0   2.0  -2.0  -1.0   0.0   1.0   0.0   2.0
    -M     8         6.0   3.0   2.0   0.0  -0.0   0.0  -1.0   0.0   1.0   3.0
    CN    CN    CN     0    MM    MM    MM    MM    MM    MM     0     0    --

    Cj    Cj    Cj   0.0  -2.0  -3.0  -1.0   1.0   0.0   0.0    -M    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7     8    --
  -3.0     2         2.0   0.2   1.0   0.5  -0.5  -0.2   0.0   0.2   0.0   8.0
    -M     8         2.0   2.5   0.0  -1.0   1.0   0.5  -1.0  -0.5   1.0   0.8
    CN    CN    CN     0    MM     0    MM    MM    MM    MM    MM     0    --

    Cj    Cj    Cj   0.0  -2.0  -3.0  -1.0   1.0   0.0   0.0    -M    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7     8    --
  -3.0     2         1.8   0.0   1.0   0.6  -0.6  -0.3   0.1   0.3  -0.1   1.0
  -2.0     1         0.8   1.0   0.0  -0.4   0.4   0.2  -0.4  -0.2   0.4   1.0
    CN    CN    CN     0     0     0    -0     0    -0    -0    MM    MM    --

该问题有无穷多解,其中一个最优解为:
X=(1,2,0,0)
最优值:-7.000
该问题迭代3次,用时0.001s

Process finished with exit code 0
```

- NO.3



```
    CB    XB     b     b     1     2     3     4     5     6     7    --
   4.0     1         1.0   1.0   0.3   0.0   0.0   0.3   0.0   0.0   3.0
    -M     6         2.0   0.0   1.7  -1.0   0.0  -1.3   1.0   0.0   1.2
    -M     7         3.0   0.0   1.7   0.0   1.0  -0.3   0.0   1.0   1.8
    CN    CN    CN     0     0    MM    MM    MM    MM     0     0    --

    Cj    Cj    Cj   0.0   4.0   1.0   0.0   0.0    -M    -M    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7    --
   4.0     1         0.6   1.0   0.0   0.2   0.0   0.6  -0.2   0.0   inf
   1.0     2         1.2   0.0   1.0  -0.6   0.0  -0.8   0.6   0.0   inf
    -M     7         1.0   0.0   0.0   1.0   1.0   1.0  -1.0   1.0   1.0
    CN    CN    CN     0     0     0    MM    MM    -2    MM     0    --

    Cj    Cj    Cj   0.0   4.0   1.0   0.0   0.0    -M    -M    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7    --
   4.0     1         0.6   1.0   0.0   0.2   0.0   0.6  -0.2   0.0   1.0
   1.0     2         1.2   0.0   1.0  -0.6   0.0  -0.8   0.6   0.0   1.0
   0.0     4         1.0   0.0   0.0   1.0   1.0   1.0  -1.0   1.0   1.0
    CN    CN    CN     0     0     0    -0     0    MM    MM    MM    --

该问题有最优解:
X=(0.6,1.2,0.0,1.0)
最优值:3.600
该问题迭代4次,用时0.001s

Process finished with exit code 0
```

- NO.4



```
C:\Users\86186\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/86186/
    Cj    Cj    Cj   0.0  10.0  15.0  12.0   0.0   0.0   0.0    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7    --
   0.0     5         9.0   5.0   3.0   1.0   0.0   1.0   0.0   0.0   1.8
   0.0     6        15.0  -5.0   6.0  15.0   0.0   0.0   1.0   0.0   inf
    -M     7         5.0   2.0   1.0   1.0  -1.0   0.0   0.0   1.0   2.5
    CN    CN    CN     0    MM    MM    MM    MM     0     0     0    --

    Cj    Cj    Cj   0.0  10.0  15.0  12.0   0.0   0.0   0.0    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7    --
  10.0     1         1.8   1.0   0.6   0.2   0.0   0.2   0.0   0.0   9.0
   0.0     6        24.0   0.0   9.0  16.0   0.0   1.0   1.0   0.0   1.5
    -M     7         1.4   0.0  -0.2   0.6  -1.0  -0.4   0.0   1.0   2.3
    CN    CN    CN     0     0    MM    MM    MM    MM     0     0    --

    Cj    Cj    Cj   0.0  10.0  15.0  12.0   0.0   0.0   0.0    -M theta
    CB    XB     b     b     1     2     3     4     5     6     7    --
  10.0     1         1.5   1.0   0.5   0.0   0.0   0.2  -0.0   0.0   1.0
  12.0     3         1.5   0.0   0.6   1.0   0.0   0.1   0.1   0.0   1.0
    -M     7         0.5   0.0  -0.5   0.0  -1.0  -0.4  -0.0   1.0   1.0
    CN    CN    CN     0     0    MM     0    MM    MM    MM     0    --

该问题无可行解
该问题迭代3次,用时0.003s

Process finished with exit code 0
```

dd

可以得到答案与手算的答案相同代码验证是对的

- **第二题由于x_3没有约束条件所以要用把他换成x_3-x_4的形式(x_3,x_4 >= 0),同时把min换算成max**
- **第三题第目标方程的形缺少两个变量所以要在标准形式中加上**

下面列出函数的名字

```python
# 读取线性规划模型，并返回字符串列表
def inPut():
# 得到价值向量c
def find_C(model_str):
# 求系数矩阵+资源向量
def find_A(model_str):
# 由以上三个函数可得到 1.系数+资源向量矩阵 2.价值向量    A c b  三个变量

# 为进行资源向量非负化，约束条件等式化，记录每个约束的形式(  =记为0，>记为1,<记为-1)
def constraintCondition(model_str):
# 对A,b进行标准化
def change_A(str, A, c, yueshu):
# 对价值向量c进行标准化
def change_c(str, A, c, yueshu):
# 基变量对应的价值系数
def C_B(x_b, c, m):
# 计算检验数
def CN(A, c, x_b, c_b, m, n):
    # 判断解的情况
def judge(A, cn, m, n, x_b, c_b):
```

```
# 计算theta
def theTa(cn, A, m):
# 进行基变换
def baseChange(A, cn, theta, m, n):
# 进行基的换入与换出
def X_B(cn, theta, x_b):
# 计算最优值
def Value(c_b, A, m):
# 打印单纯形表
def Excel(A, c_b, x_b, c, theta, cn, m, n):

def outPut(result, value, x_b, A, X):
```

下面是完整的代码:

```
import os
import re
import numpy as np
import time

"""
输入线性规划模型到一个文档中，程序读取线性规划模型，化其为标准型。
找到初始可行基，建立单纯形表，迭代求解，判断解的类型，输出结果。
"""



# 读取线性规划模型，并返回字符串列表
def inPut():
    #  读取线性规划模型
    model_str = []
    with open("4.txt", 'r') as f:  # 建立一个文本文档  储存线性规划模型
        lines = f.readlines()
    for element in lines:
        element = element[:-1]
        model_str.append(element)
    return (model_str)


# 得到价值向量c
def find_C(model_str):
    x_temple = re.findall(r"-?\d+\.?\d*", model_str[-1])
    x_temple = [int(i) for i in x_temple]
    x_max = max(x_temple)
    c_temple = re.findall(r"-?\d+\.?\d*", model_str[0])
    c_temple = [float(i) for i in c_temple]
    c_list_1 = c_temple[1::2]
    c_list_2 = c_temple[::2]
    c = []
    for i in range(x_max + 1):
        c.append(0)
    for i in range(len(c_list_1)):
        c[int(c_list_1[i])] = c_list_2[i]
    c = np.array(c)
    return c


# 求系数矩阵+资源向量
```

```python
def find_A(model_str):
    b = []
    x_temple = re.findall(r"-?\d+\.?\d*", model_str[-1])
    x_temple = [int(i) for i in x_temple]
    x_max = max(x_temple)
    A = [[0 for i in range(x_max + 1)] for i in range(len(model_str) - 2)]
    k = 0
    for element in model_str[1:-1]:
        x_temple = re.findall(r"-?\d+\.?\d*", element)
        x_temple = [float(i) for i in x_temple]
        x_list_1 = x_temple[1::2]
        x_list_2 = x_temple[:-1:2]
        x = []
        b.append(x_temple[-1])
        for i in range(x_max + 1):
            x.append(0)
        for i in range(len(x_list_1)):
            x[int(x_list_1[i])] = x_list_2[i]
        for num in range(len(x)):
            A[k][num] = x[num]
        k += 1
    for num in range(len(b)):
        A[num][0] = b[num]
    A = np.array(A, dtype=np.float_)
    return A


# 由以上三个函数可得到 1.系数+资源向量矩阵 2.价值向量    A c b  三个变量

# 为进行资源向量非负化，约束条件等式化，记录每个约束的形式(  =记为0，>记为1,<记为-1)
def constraintCondition(model_str):
    yueshu = []
    for element in model_str[1:-1]:
        if '<' in element:
            yueshu.append(-1)
        elif '>' in element:
            yueshu.append(1)
        else:
            yueshu.append(0)
    yueshu = np.array(yueshu)
    return yueshu


# 对A,b进行标准化
def change_A(str, A, c, yueshu):
    # 约束条件等式化
    for i in range(A.shape[0]):
        if A[i, 0] < 0:
            yueshu[i] *= -1
            A[i] *= -1
    for i in range(A.shape[0]):
        if yueshu[i] == 1:
            # 添加松弛变量
            temple_list = [0 for j in range(A.shape[0])]
            temple_list[i] = -1
            A = np.column_stack((A, temple_list))

    for i in range(A.shape[0]):
        if yueshu[i] == -1:
            # 添加剩余变量
            temple_list = [0 for j in range(A.shape[0])]
            temple_list[i] = 1
            A = np.column_stack((A, temple_list))
        elif yueshu[i] == 1:
            # 添加人工变量
            temple_list = [0 for j in range(A.shape[0])]
            temple_list[i] = 1
            A = np.column_stack((A, temple_list))
        elif yueshu[i] == 0:
            #  添加人工变量
            temple_list = [0 for j in range(A.shape[0])]
            temple_list[i] = 1
            A = np.column_stack((A, temple_list))
    return A


# 对价值向量c进行标准化
def change_c(str, A, c, yueshu):
    if 'in' in str[0]:
        c = c * -1
    # 约束条件等式化
    for i in range(A.shape[0]):
        if A[i, 0] < 0:
            yueshu[i] *= -1

    for i in range(A.shape[0]):
        if yueshu[i] == 1:
            # 添加松弛变量
            c = list(c)
            c.append(0)
            c = np.array(c)

    for i in range(A.shape[0]):
        if yueshu[i] == -1:
            # 添加剩余变量
            c = list(c)
            c.append(0)
            c = np.array(c)

        elif yueshu[i] == 1:
            # 添加人工变量
            c = list(c)
            c.append(-pow(10, 9))
            c = np.array(c)
        elif yueshu[i] == 0:
            #  添加人工变量
            c = list(c)
            c.append(-pow(10, 9))
            c = np.array(c)
    return c


# 基变量对应的价值系数
def C_B(x_b, c, m):
    c_b = []
    for element in x_b:
        c_b.append(c[element])
    return c_b


# 计算检验数
def CN(A, c, x_b, c_b, m, n):
    cn = [0]
    for num_1 in range(1, n):
        cj = c[num_1]
        for num_2 in range(m):
            cj = cj - c_b[num_2] * A[num_2, num_1]
        if 0 < cj and cj < 0.0001:
            cj = 0
        cn.append(cj)
    return cn


# 判断解的情况
def judge(A, cn, m, n, x_b, c_b):
    # 最优解
    num = cn.count(0)
    if max(cn) <= 0:
        for j in range(len(c_b)):
            if c_b[j] == -pow(10, 9):
                if A[j, 0] != 0:
                    return 4  # 无可行解
        if num == m + 1:
            return 1  # 有最优解
        else:
            return 2  # 有无穷多最优解
    else:
        for num in range(len(cn)):
            if cn[num] > 0:
                lis = []
                for i in range(m):
                    lis.append(A[i, num])
                if max(lis) < 0:
                    for j in range(len(c_b)):
                        if c_b[j] == -pow(10, 9):
                            if A[j, 0] != 0:
                                return 4  # 无可行解
                    return 3  # 有无界解
    return 0  # 继续迭代


# 计算theta
def theTa(cn, A, m):
    theta = []
    j = cn.index(max(cn))
    for num in range(m):
        if A[num, j] > 0:
            theta.append(A[num, 0] / A[num, j])
        else:
            theta.append(float("inf"))

    return theta
```

```python
# 进行基变换
def baseChange(A, cn, theta, m, n):
    # 找出中心元素
    j = cn.index(max(cn))
    i = theta.index(min(theta))
    main_elem = A[i, j]

    A[i] = A[i] / main_elem
    for num in range(m):
        if num != i:
            A[num] = A[num] - A[num][j] * A[i]
    return A


# 进行基的换入与换出
def X_B(cn, theta, x_b):
    j = cn.index(max(cn))
    i = theta.index(min(theta))
    x_b[i] = j
    return x_b


# 计算最优值
def Value(c_b, A, m):
    value = 0
    for num in range(m):
        value += c_b[num] * A[num, 0]
    return value


# 打印单纯形表
def Excel(A, c_b, x_b, c, theta, cn, m, n):
    print("%6s" % "Cj", end="")
    print("%6s" % "Cj", end="")
    print("%6s" % "Cj", end="")
    for i in c:
        if i != -pow(10, 9):
            print("%6.1f" % i, end="")
        else:
            print("%6s" % "-M", end="")
    print("%6s" % "theta")
    # 第二行
    print("%6s" % "CB", end="")
    print("%6s" % "XB", end="")
    print("%6s" % "b", end="")
    print("%6s" % "b", end="")
    for i in range(1, n):
        print("%6d" % i, end="")
    print("%6s" % "--")

    # 打印数字
    for i in range(m):
        if c_b[i] != -pow(10, 9):
            print("%6.1f" % c_b[i], end="")
        else:
            print("%6s" % "-M", end="")
        print("%6d" % x_b[i], end="")
        print("%6s" % "", end="")
        for j in range(n):
            print("%6.1f" % A[i, j], end="")
        print("%6.1f" % theta[i])
    # 打印检验数
    print("%6s" % "CN", end="")
    print("%6s" % "CN", end="")
    print("%6s" % "CN", end="")
    for i in range(n):
        if cn[i] > 10000 or cn[i] < -10000:
            print("%6s" % "MM", end="")
        else:
            print("%6.f" % cn[i], end="")
    print("%6s" % "--")
    print("")


def outPut(result, value, x_b, A, X):
    if result == 1:
        num = 0
        for i in x_b:
            if i <= len(X):
                if A[num, 0] < 0:
                    print("该问题无可行解:")
                    return
                X[i - 1] = A[num, 0]
            num += 1
        print("该问题有最优解:")
        print("X=(", end="")
        for i in range(len(X)):
            if i == len(X) - 1:
                print("%.1f" % X[i], end="")
            else:
                print("%.1f," % X[i], end="")
        print(")")
        print("最优值:%.3f" % value)
    elif result == 2:
        print("该问题有无穷多解,其中一个最优解为:")
        num = 0
        for i in x_b:
            if i <= len(X):
                if A[num, 0] < 0:
                    return 4
                X[i - 1] = A[num, 0]
            num += 1
        print("X=(", end="")
        for i in range(len(X)):
            if i == len(X) - 1:
                print("%.0f" % X[i], end="")
            else:
                print("%.0f," % X[i], end="")
        print(")")
        print("最优值:%.3f" % value)

    elif result == 3:
        print("该问题有无界解")

    elif result == 4:
        print("该问题无可行解")
```

```python
# 输入线性规划模型，找出变量
def solve():
    str = inPut()
    startTime = time.time()
    yueshu = constraintCondition(str)
    c = find_C(str)
    A = find_A(str)
    X = [0 for i in range(A.shape[1] - 1)]
    # 标准化
    A = change_A(str, A, c, yueshu)
    c = change_c(str, A, c, yueshu)

    # 得到矩阵的维数
    m = A.shape[0]
    n = A.shape[1]
    # 确定初始可行基X的下标,x_b存储可行基X的下标
    x_b = []
    for i in range(m):
        x_b.append(n - m + i)

    # X_B对应的价值系数c_b
    c_b = C_B(x_b, c, m)
    # 初始单纯形表建立

    # 1.计算检验数cn, theta
    # 2.判断解的情况 若未达到终止条件 找到主元素 改变可行基 进行单位变换  进行第一步
    # 3.若达到终止条件，输出结果
    num = 0
    while True:
        cn = CN(A, c, x_b, c_b, m, n)
        theta = theTa(cn, A, m)
        num += 1
        Excel(A, c_b, x_b, c, theta, cn, m, n)
        result = judge(A, cn, m, n, x_b, c_b)
        if result != 0:
            break
        theta = theTa(cn, A, m)
        A = baseChange(A, cn, theta, m, n)
        x_b = X_B(cn, theta, x_b)
        c_b = C_B(x_b, c, m)
    value = Value(c_b, A, m)
    if "in" in str[0]:
        value *= -1
    outPut(result, value, x_b, A, X)
    endTime = time.time()
    time_1 = endTime - startTime
    print("该问题迭代%d次,用时%.3fs" % (num, time_1))


solve()
```