《运筹学》实验报告

第 一 次实验	日期：9 月 19 日	得分：
学号：	姓名：	专业：智能科学技术

一、实验目的

理解、熟悉单纯形法的计算过程和运算方法

二、算法设计

完全按照单纯形法的数学思路，将其转换为代码进行运算即可。

三、核心程序代码

主要代码

```
class 单纯形表:
    def __init__(self, 目标函数: list, 增广矩阵: list):
        self.c = 目标函数
        self.增广矩阵 = 增广矩阵
        self.base_i = []  # 基变量
        self.x_num = len(self.c)  # 有多少个x
        self.iter = 0
        self.sigma = [0] * self.x_num  # 各个x的检验数
        for i in range(len(self.c)):
            if self.c[i] == 0:
                self.base_i.append(i)
        self.calcSigma()
        self.printSheet()

    def printSheet(self):
        """
        输出单纯形表
        :return:
        """
        print("单纯形表" + str(self.iter) + ":")
        print("c_j→", self.c)
        print(["C_B", "基", "b"] + ["x_" + str(i) for i in
range(len(self.c))])
        for i in range(len(self.base_i)):
            print(
                [self.c[self.base_i[i]], "x_" + str(self.base_i[i]),
```

```
(self.增广矩阵[i])[-1]] + (self.增广矩阵[i])[:-1])
        print(["sigma:"] + ['{:.4f}'.format(i) for i in self.sigma])

    def calcSigma(self):
        """
        计算Sigma值以确定换入变量
        :return:
        """
        for i in range(self.x_num):
            if i in self.base_i:
                self.sigma[i] = 0
            else:
                self.sigma[i] = self.c[i] - sum(
                    [self.c[self.base_i[j]] * self.增广矩阵[j][i] for j in
range(len(self.base_i))])

    def _calcTheta(self, inIndex):
        """
        计算Theta值以确定换出变量
        :param inIndex:
        :return:
        """
        b = [i[-1] for i in self.增广矩阵]
        theta = [0] * len(b)
        for i in range(len(self.base_i)):
            if self.增广矩阵[i][inIndex] > 0:
                theta[i] = b[i] / self.增广矩阵[i][inIndex]
            else:
                theta[i] = 1.7976931348623157e+308  # 其实对于分母为0时是无
法运算的，这里取得是float的最大值，为了能在下面的min中排除
        return self.base_i[theta.index(min(theta))]  # 要换出的变量

    def iterCalc(self):
        """
        迭代计算
        :return:
        """
        inIndex = self.sigma.index(max(self.sigma))  # 要换入的变量
        outIndex = self._calcTheta(inIndex)
        print("换入x_" + str(inIndex) + ", 换出x_" + str(outIndex))
        self.base_i[self.base_i.index(outIndex)] = inIndex
        self.增广矩阵[self.base_i.index(inIndex)] = [i / self.增广矩阵
[self.base_i.index(inIndex)][inIndex] for i in
                                                self.增广矩阵
```

```
[self.base_i.index(inIndex)]]
        for i in range(len(self.增广矩阵)):
            if i != self.base_i.index(inIndex):
                self.增广矩阵[i] = [self.增广矩阵[i][j] - self.增广矩阵
[i][inIndex] * self.增广矩阵[self.base_i.index(inIndex)][j] for
                                    j in range(len(self.增广矩阵[i]))]
        self.calcSigma()
        self.iter += 1
        self.printSheet()
        if max(self.sigma) > 0:
            self.iterCalc()
        else:
            x = [0] * self.x_num
            b = [i[-1] for i in self.增广矩阵]
            for i in range(self.x_num):
                if i not in self.base_i:
                    x[i] = 0
                else:
                    x[i] = b[self.base_i.index(i)]
            z = sum([self.c[i] * x[i] for i in range(self.x_num)])
            print("最优解为:", x)
            print("目标函数值为:", z)
```

函数调用

```
目标函数 = [2, 3, 0, 0, 0]
增广矩阵 = [[2, 2, 1, 0, 0, 12],
        [4, 0, 0, 1, 0, 16],
        [0, 5, 0, 0, 1, 15]]

a = 单纯形表(目标函数=目标函数, 增广矩阵=增广矩阵)
a.iterCalc()
```

四、测试及结果（给出测试用例及测试结果）

(1) 测试用例：

使用课本 P23 例 5 作为测试用例

$$maxz = 2x_1 + 3x_2 + 0x_3 + 0x_4 + 0x_5$$

$$s.t.\begin{cases} 2x_1 + 2x_2 + x_3 = 12 \\ 4x_1 + x_4 = 16 \\ 5x_2 + x_5 = 15 \\ x_j \geq 0 (j = 1,2,\dots,5) \end{cases}$$

其目标函数即记为$[2,3,0,0,0]$，

等式的限制条件记为$\begin{vmatrix} 2 & 2 & 1 & 0 & 0 & 12 \\ 4 & 0 & 0 & 1 & 0 & 16 \\ 0 & 5 & 0 & 0 & 1 & 15 \end{vmatrix}$

随后调用上述代码进行计算。

(2) 运算结果：

```
D:\anaconda3\python.exe E:/OneDrive/作业暂存/运筹学/实验1/单纯形法.py
单纯形表0:
c_j→ [2, 3, 0, 0, 0]
['C_B', '基', 'b', 'x_0', 'x_1', 'x_2', 'x_3', 'x_4']
[0, 'x_2', 12, 2, 2, 1, 0, 0]
[0, 'x_3', 16, 4, 0, 0, 1, 0]
[0, 'x_4', 15, 0, 5, 0, 0, 1]
['sigma:', '2.0000', '3.0000', '0.0000', '0.0000', '0.0000']
换入x_1, 换出x_4
单纯形表1:
c_j→ [2, 3, 0, 0, 0]
['C_B', '基', 'b', 'x_0', 'x_1', 'x_2', 'x_3', 'x_4']
[0, 'x_2', 6.0, 2.0, 0.0, 1.0, 0.0, -0.4]
[0, 'x_3', 16.0, 4.0, 0.0, 0.0, 1.0, 0.0]
[3, 'x_1', 3.0, 0.0, 1.0, 0.0, 0.0, 0.2]
['sigma:', '2.0000', '0.0000', '0.0000', '0.0000', '-0.6000']
换入x_0, 换出x_2
单纯形表2:
c_j→ [2, 3, 0, 0, 0]
['C_B', '基', 'b', 'x_0', 'x_1', 'x_2', 'x_3', 'x_4']
[2, 'x_0', 3.0, 1.0, 0.0, 0.5, 0.0, -0.2]
[0, 'x_3', 4.0, 0.0, 0.0, -2.0, 1.0, 0.8]
[3, 'x_1', 3.0, 0.0, 1.0, 0.0, 0.0, 0.2]
['sigma:', '0.0000', '0.0000', '-1.0000', '0.0000', '-0.2000']
最优解为: [3.0, 3.0, 0, 4.0, 0]
目标函数值为: 15.0

进程已结束,退出代码0
```

由于 Python 中矩阵的初始下标为 0，所以计算结果中，所有下标均比课本上少 1，如 x_0 表示x_1，x_1 表示x_2，依此类推。

计算结果和步骤与课本一致，测试通过。