Otimização Linear 2021/2022

Relatório do Projeto 1

Programação do Método Simplex

Ana Almeida 99431

Francisca Coimbra 99528

Francisca Ferreira 98695

Tiago Ferreira 78106

Preâmbulo

A programação linear é um ramo da programação matemática que faz parte dos métodos quantitativos de apoio à tomada de decisão. Normalmente, os problemas de programação matemática dizem respeito à afetação de recursos escassos a usos alternativos de forma a satisfazer um objetivo sujeito a um conjunto de condições ou restrições. A solução que satisfaz simultaneamente a função objetivo e as restrições é chamada a solução ótima do problema. Assim, esta programação é aplicável quando o objetivo e as restrições do problema podem ser traduzidos por funções lineares.

Para este projeto, optámos por usar o Python para programar o Método Simplex. Deste modo, adquirimos novas e melhores competências nesta linguagem de programação através do desenvolvimento de um algoritmo para o método escolhido.

Introdução ao algoritmo

Em primeiro lugar, criámos um programa para estarmos aptos a implementar o Método Simplex. Assim, recorremos a funções do tipo *Boolean* e colocámos várias escolhas base com a finalidade de podermos realizar o *input*.

A nossa introdução ao Método consistiu na abordagem de verificar se o que é colocado como *input* são dígitos. Seguidamente, fizemos o estudo se o problema é de maximização ou minimização, tendo depois colocado os coeficientes da função a otimizar e o número de restrições. Uma vez que as restrições são inequações, o programa também tem de reconhecer os sinais de desigualdade.

A seguir, tivemos em conta o Critério de Otimalidade e fizemos *print* de todos os resultados que constituem as tabelas. Para uma melhor abordagem, apresentamos em seguida *prints* do nosso programa.

```
7   if __name__ == "__main__":
8
9
10
11      simplex = Simplex()
12
13      def isNotNumber(n: str) -> bool:
14          if(n==''):
15              return False
16
17          elif (n.isnumeric()):
18              return False
19
20          elif (n[0]=='-' and n[1:len(n)].isnumeric()):
21              return False
22
23          elif ( '/' in n):
24
25
26              if (n[0] == '-' and n[1:n.index('/')].isnumeric() and n[n.index('/')+1:len(n)].isnumeric()):
27                  return False
28
29              elif (n[0:n.index('/')].isnumeric() and n[n.index('/')+1:len(n)].isnumeric()):
30                  return False
31
32              else:
33                  return True
34
35          elif ('.' in n):
36              return False
37
38          else:
39              return True
```

```
41      print('Max - M \t Min - m')
42      m = input()
43
44      while (m != 'M' and m != 'm'):
45          print('escolha uma das duas opções M/m')
46          m = input()
47
48      if (m == 'M'):
49          m = 1
50
51      else:
52          m = -1
53
54      z = []
55      variable = []
56      count = 1
57
58      print('escreva os coeficientes da funçao a otimizar:')
59
60      while True:
61          n = input(f'x{count}:')
62
63          while (isNotNumber(n)):
64              |
65              print('introdusa um valor numerico')
66              n = input(f'x{count}:')
67
68          if (n==''):
69              z = [float(i)*m for i in z ]
70              print('  ',variable)
71              print('z=',z)
72              count -=1
73              print()
74              print(count)
75              break
```

```
77          else:
78              if('/' in n):
79                  n = int(n[0:n.index('/')])/int(n[n.index('/')+1:len(n)])
80
81              z.append(n)
82              variable.append('x'+ str(count))
83              count += 1
84
85
86
87      rest = []
88      sinals = []
89      c_ind = []
90      count2 =1
91      operation = ['=', '<=', '=<', '=>', '>=']
92      n_rest =int(input('Numero de restrições:'))
93
94      while (len(rest)<n_rest):
95          print(f'R{count2}:')
96
97          r = []
98
99          for i in range(0,count,1):
100             var = input(f'x{i+1}:')
101
102             while (isNotNumber(var)):
103                 print('introdusa um valor numerico')
104                 var = input(f'x{count2}:')
105
106             if('/' in var):
107                 var = int(var[0:var.index('/')])/int(var[var.index('/')+1:len(var)])
108
109             elif (var == ''):
110                var = 0
111             r.append(var)
112
113         r = [float(i) for i in r ]
```

```
115             print('Escolha um sinal ente: = <= =>')
116             s = input('sinal:')
117 
118             while (s not in operation):
119                 print('Escolha um sinal ente: = <= =>')
120                 s = input('sinal:')
121 
122             sinals.append(s)
123 
124             b = input(f'b{count2}:')
125 
126             while (isNotNumber(b)):
127                 print('introdusa um valor numerico')
128                 b = input(f'b{count2}:')
129 
130             c_ind.append(float(b))
131             print()
132 
133             rest.append(r)
134             count2 +=1
135             r = []
136 
137         for i in range(0,len(rest),1):
138             print(f'R{i+1}:',rest[i],sinals[i],c_ind[i])
139 
140         print()
141         # for i in range(0,len(rest),1):
142         #     rest[i].insert(0,float(0))
143 
144         for i in range(0,len(c_ind),1):
145 
146             if (c_ind[i] < 0):
147                 c_ind[i] *= -1
148                 rest[i] = [-1* value for value in rest[i]]
149 
150                 if (sinals[i] in ['=>','>=']):
151                     sinals = '<='
```

```
153                 elif(sinals[i] in ['<=','=<']):
154                     sinals = '=>'
155 
156                 else:
157                     pass
158 
159             else:
160                 pass
161 
162         for i in range(0,len(rest),1):
163             print(f'R{i+1}:',rest[i],sinals[i],c_ind[i])
164 
165     # criterio de nao negatividade
166 
167         print('\n')
168         print('criterio de nao negatividade das variaveis')
169 
170         answer = ''
171 
172         for i in range(0,len(variable),1):
173             print(variable[i],':\n 1 - positiva\n 2- ilimitada\n 3- maior que uma constsnte positiva\n')
174             answer = int(input('Resposta:'))
175 
176             while (answer not in [1,2,3]):
177                 print('Só pode escolher ente 1, 2 e 3!')
178                 answer = int(input('Resposta:'))
179 
180             if (answer == 2):
181                 variable[i] = 'x'+ str(i)+'+'
182                 variable.insert(i+1,'x'+ str(i)+'-')
183                 z.insert(i+1,-1*z[i])
184 
185                 for j in range (0,len(rest),1):
186                     rest[j].insert(i+1,-1*rest[j][i])
187 
188             if (answer == 3):
```

```
    # metedo big M

    idx_aVar =[]

    for i in range(0,len(variable),1):
        if ('a' in variable[i]):
            idx_aVar.append(i)

    for i in range(0,len(rest),1):

        for j in range(0,len(idx_aVar),1):

            if(rest[i][idx_aVar[j]]==1):
                z = simplex.bigmMet(big_M,z,rest[i])



    print('-----------------------------------------------------------------------------------------------------\n')
    print('  ',variable)
    print('z= ',z)

    for i in range(0,len(rest),1):
        print(f'R{i+1}:',rest[i])

    print()
    print()
    print('-----------------------------------------------------------------------------------------------------\n\n')
    print()
    print()

    simplex.objectiveFunction(z)

    for i in range(0,len(rest),1):
        simplex.addRest(rest[i])

    simplex.solve()
```

```
                    while (len(rest[j]) < len(z)):
                        rest[j].append(float(0))

                count3 += 1


            elif(sinals[i] in ['=>', '>=']):
                variable.append('f'+ str(count3))
                z.append(float(0))
                rest[i].append(float(-1))

                for j in range(0,len(sinals),1):

                    while (len(rest[j]) < len(z)):
                        rest[j].append(float(0))

                count3 += 1

            else:
                pass



    # introdução das variaveis artificiais

        big_M =simplex.bigM(z,m)

        count3 = 1

        for i in range(0,len(sinals),1):

            if(sinals[i] in ['=>', '>=']):
                variable.append('a'+ str(count3))
                z.append(big_M)
                rest[i].append(float(1))
```

```
        if (answer == 3):
            const = input('escreva a constante:')

            while (isNotNumber(const)):
                print('introdusa um valor numerico')
                const = input('escreva a constante:')

            if('/' in const):
                const = int(const[0:const.index('/')])/int(const[const.index('/')+1:len(const)])

            variable[i] = 'x'+'~'+ str(i)

            for j in range (0,len(c_ind),1):
                c_ind[j] -= rest[j]*const



#########################################################################################################################
#                                                  Forma Standart                                                       #
#########################################################################################################################

# introdução das variaveis de folga


    count3 = 1

    for i in range(0,len(sinals),1):

        if(sinals[i] in ['<=', '=<']):
            variable.append('f'+ str(count3))
            z.append(float(0))
            rest[i].append(float(1))

            for j in range(0,len(sinals),1):
```

```
                for j in range(0,len(sinals),1):

                    while (len(rest[j]) < len(z)):
                        rest[j].append(float(0))

                count3 += 1

            elif(sinals[i] == '=' ):
                variable.append('a'+ str(count3))
                z.append(float(big_M))
                rest[i].append(float(1))

                for j in range(0,len(sinals),1):

                    while (len(rest[j]) < len(z)):
                        rest[j].append(float(0))

                count3 += 1

            else:
                pass

    variable.append('b')
    z.append(float(0))

    for i in range(0,len(c_ind),1):
        rest[i].append(c_ind[i])

    print('\n\n')
    print(variable)
    print(' ',z)
    for i in range(0,len(rest),1):
        print(f'R{i+1}:',rest[i])

    print()
    print()
```

Desenvolvimento do algoritmo

Vamos agora expor, de forma detalhada, como construímos o algoritmo em linguagem Python. Iremos fazer *print* de todo o processo da linha de comandos e, por fim, analisar o output.

```
class  Simplex :

    def __init__(self):
        self.table = []
        self.variabels = []


 def objectiveFunction (self, fo: list):
     self.table.append(fo)

 def addRest (self, sa: list):
     self.table.append(sa)

 def addVariabels (self, variabel: str):
     self.variabels.append(variabel)
```

Começamos por iniciar com uma tabela vazia, a partir da qual vamos adicionar uma linha com a função objetivo (fo). De seguida, colocamos a lista com as restrições (sa). Fizemos também uma função para calcular as variáveis básicas designada *basicVariables*.

```
def basicVariavels (self) -> list:

    basic_variabels = []

    for col in range(len(self.variabels)):

        if (self.table[0][col] == 0):
            value = []

            for line in range(len(self.table)):

                if (line != 0):
                    value.append(self.table[line][col])

                else:
                    pass


            if (sum(value) == 1 and (1 or 0 in value)):
                basic_variabels.append(col)

    return basic_variabels
```

Neste momento, recorremos à função *getEntryVar*, que é a função que escolhe a variável que entra, isto é, averigua qual o valor máximo da última linha e guarda o seu índice, retornando-o.

```
def getEntryVar (self) -> int:
    fo = []

    for col in range(len(self.table[0])-1):
        fo.append(self.table[0][col])

    c_p = max(fo)
    idx = self.table[0].index(c_p)

    return idx
```

Após estar definida a linha pivotal, procedemos à escolha da variável de saída. É, então, implementada a função *getExitVar*, onde implementamos um algoritmo para prevenir a ocorrência de *cycling*, recorrendo à Regra Lexicográfica.

```
def getExitVar (self, entry_var: int) -> int:
    result = {}
    vidx = []

    for line in range(len(self.table)):

        if (line > 0):

            if (self.table[line][entry_var] > 0):
                division = self.table[line][-1] / self.table[line][entry_var]
                result[line] = division

            else:
                pass

    idx = min(result, key=result.get)

    for line in result.keys():

        if (result[line] == min(result.values())):
            vidx.append(int(line))

    if (len(vidx)==1):
        return idx
```

```
else:
    basicv = self.basicVariavels()
    k = 0

    while (len(vidx)!=1):
        vidx = []

        for line in result.keys():

            if (line > 0):

                if (self.table[line][entry_var] > 0):
                    division = self.table[line][basicv[k]] / self.table[line][entry_var]
                    result[line] = division

                else:
                    pass

        k += 1
        idx = min(result, key=result.get)

        for line in result.keys():

            if (result[line] == min(result.values())):
                vidx.append(int(line))

    idx = vidx[0]

    return idx
```

A função recebe a variável que entra. Abrimos um dicionário contém dois valores, a chave e o valor associado a esta chave, e ainda uma lista.

No ciclo *for*, temos o objetivo de calcular os rácios. Assim, o *if* imputado serve para garantir que não recorremos à linha 0, onde está o z e garante que os elementos não são negativos nem zero. Procedemos à divisão entre o bbarra e os elementos da coluna pivotal, este resultado é guardado no dicionário associado a cada linha. De seguida, pretendemos calcular o mínimo, onde recorremos à Regra Lexicográfica.

Implementamos, novamente, um ciclo *for*. Verificamos se existe alguma linha com valor igual ao mínimo e, caso exista, esse valor é colocado com *vidx*. Depois, verificamos se o vetor só tem 1 elemento, para salvaguardar que o mínimo é único. Caso isto aconteça irá retornar o índice da linha pivô.

Tudo isto, irá ocorrer sem que *vidx* for diferente de 1, pois o pretendido é percorrer as colunas e fazer a comparação lexicográfica.

Posto isto, criámos novamente, um ciclo *for*, pois só queremos ver os vetores que estão empatados. Depois, o código segue implementando novamente a condição de não negatividade.

Em *result[line]* entram os valores das colunas. Entretanto, o ciclo vai avançando e passa para a seguinte coluna, depois verifica se o mínimo é único e adiciona ao vetor *vidx*. Enquanto o vetor não tem um único elemento (tamanho único) não sai do ciclo. Após sair, o índice é retornado.

A função *calNewPivotLine* executa a nova linha pivotal, fazendo, por isso, a divisão entre a linha anterior e o pivô. A função *calNewLine* calcula as novas linhas. Assim, uma vez que a divisão já estava guardada na variável *newpivotline*, resta colocar na lista a soma da linha da tabela anterior com a *resultLine*.

```
def calNewPivotLine (self, entry_var: int, exit_var: int) -> list:
    line = self.table[exit_var]

    pivot = line[entry_var]

    new_pivot_line = [value / pivot for value in line]

    return new_pivot_line
```

```
def calNewLine(self, line: list, entry_var: int, pivotLine: list) -> list:
    pivot = float(line[entry_var] * -1.0)

    resultLine = [value * pivot for value in pivotLine]

    newLine = []

    for col in range(len(resultLine)):
        sumValue = line[col] + resultLine[col]
        newLine.append(sumValue)

    return newLine
```

Decidimos utilizar o Método do Big-M. Inicialmente, procedemos à função *BigM* para gerar um número suficientemente grande, elevando o max(z) a 10 e caso o max(z) seja igual a 1, este passaria a ser 2, e depois este seria elevado a 10.

```
def bigM(self,z: list, m: int) -> float:
        maxZ = max(z)

        if ( abs(maxZ) == 1):
            maxZ =2

        big_M = float(-m*maxZ**10)

        return big_M
```

Isto garante que o valor é elevado em relação aos outros coeficientes. Na iteração seguinte, temos uma nova variável, *resultline*, que é a multiplicação entre o valor mencionado que foi guardado na função anterior. De seguida, foi criado um ciclo que irá percorrer de novo os valores todos da linha elegida pelo Big-M e procedemos à sua soma. Por fim, o valor é mostrado após a compilação e guardado.

```
def bigmMet(self,bigM: float, z: list, artVarLine: list) -> list:
    resultLine = [value * bigM for value in artVarLine]

    newZ = []

    for i in range(len(resultLine)):
        sumValue = z[i] - resultLine[i]
        newZ.append(sumValue)

    return newZ
```

Posto isto, definimos o critério de paragem. Temos de verificar se o Critério de Dantzig está satisfeito, ou seja, se cumpre o Critério de Otimalidade. Deste modo, a função *boolean* retorna positivo caso as variáveis da solução sejam positivas e repete todo o processo mencionado até o retorno final ser falso.

```
def dantzigCriterion (self) -> bool:
    positive = list(filter(lambda x: x > 0, self.table[0]))

    return False if len(positive) > 0 else True
```

Criou-se uma função ShowTable, para que fossem impressas as sucessivas tabelas do método simplex.

```
def showTable (self):
    max_ = []

    for col in range(len(self.table[0])):
        maxbycol = 0

        for line in range(len(self.table)):
            maxbycol = max(maxbycol,len(f"{self.table[line][col]:.2f}"))

        max_.append(maxbycol)

    for col in range(len(self.variabels)):
        print(' '*(int(max_[col]/8 +1)*8-len(f"{self.variabels[col]}")) + f"{self.variabels[col]}", end = "")

    print('  |'+' '*(int(max_[-1]/8 +1)*8-3) +'RHS\n')

    for col in range(len(self.table[0])):
        print('-'*(int(max_[col]/8 +1)*8), end = '')

    print('---')

    for line in range(len(self.table)):

        if (line == 1):

            for col in range(len(self.table[0])):
                print('-'*(int(max_[col]/8 +1)*8), end = '')
            print('---')

        for col in range(len(self.table[0])):

            if ( col == len(self.table[0])-1):
                print('  |', end='')

            print(' '*(int(max_[col]/8 +1)*8-len(f"{self.table[line][col]:.2f}")) + f"{self.table[line][col]:.2f}",end = '')
        print("\n")
```

Em seguida, é feita uma nova tabela, onde os valores da tabela anterior são substituídos pelos valores calculados recentemente. Uma vez mais, caso o retorno na função *boolean* seja positivo, é criado um ciclo para os valores, que são percorridos um a um, não saiam deste até à otimalidade ser satisfeita. Assim, será sempre impressa uma nova tabela atualizada. Isto é o que ocorre nas funções *calculate* e *solve*.

```
def calculate(self):
    entryVar = self.getEntryVar()
    firstExitLine = self.getExitVar(entryVar)
    pivotLine = self.calNewPivotLine(entryVar,firstExitLine)
    self.table[firstExitLine] = pivotLine
    tableCopy = self.table.copy()
    idx = 0

    while idx < len (self.table):

        if (idx != firstExitLine):
            line = tableCopy[idx]
            newLine = self.calNewLine(line, entryVar, pivotLine)
            self.table[idx] = newLine

        idx += 1
```

```
def solve(self):
    print('Quadro 0\n')
    self.showTable()
    z = None
    x = []
    vb = []
    v = self.basicVariavels()

    for col in range(len(v)):
        vb.append(self.variabels[v[col]])

    print('Base basica:', vb)
    print(f'variavel a sair: {vb[self.getExitVar(self.getEntryVar())-1]}',f'variavel a entrar: {self.variabels[self.getEntryVar()]}', sep= ' --> ')
    vb.pop(self.getExitVar(self.getEntryVar())-1)
    vb.insert(self.getExitVar(self.getEntryVar())-1,self.variabels[self.getEntryVar()])
    self.calculate()
    print('_____________________________________________________________________________________________\n')
    print('Quadro 1\n')
    self.showTable()

    count = 2

    while not self.dantzigCriterion():
        print('Base basica:', vb)
        print(f'variavel a sair: {vb[self.getExitVar(self.getEntryVar())-1]}',f'variavel a entrar: {self.variabels[self.getEntryVar()]}', sep= ' --> ')
        vb.pop(self.getExitVar(self.getEntryVar())-1)
        vb.insert(self.getExitVar(self.getEntryVar())-1,self.variabels[self.getEntryVar()])
        self.calculate()
        print('_____________________________________________________________________________________________\n')
        print(f'Quadro {count}\n')
        print()
        self.showTable()
        count += 1
print('Base basica:', vb)
z = self.table[0][-1]

for col in range(len(self.variabels)):

    if ( self.variabels[col] in vb):
        x.append(self.table[vb.index(self.variabels[col])+1][-1])

    else:
        x.append(float(0))

print('z(x) =',z)
print('x =',x)
```

Resolução dos problemas teste

1. Input:

```
Max - M          Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:1
x2:9
x3:1
x4:
   ['x1', 'x2', 'x3']
z= [1.0, 9.0, 1.0]

Numero de variaveis: 3

Numero de restrições:2
R1:
x1:1
x2:2
x3:3
Escolha um sinal entre: = <= =>
sinal:<=
b1:9

R2:
x1:3
x2:2
x3:2
Escolha um sinal entre: = <= =>
sinal:<=
b2:15

R1: [1.0, 2.0, 3.0] <= 9.0
R2: [3.0, 2.0, 2.0] <= 15.0

R1: [1.0, 2.0, 3.0] <= 9.0
R2: [3.0, 2.0, 2.0] <= 15.0


criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
```

```
Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1



['x1', 'x2', 'x3', 'f1', 'f2', 'b']
  [1.0, 9.0, 1.0, 0.0, 0.0, 0.0]
R1: [1.0, 2.0, 3.0, 1.0, 0.0, 9.0]
R2: [3.0, 2.0, 2.0, 0.0, 1.0, 15.0]


--------------------------------------------------------------------------------


   ['x1', 'x2', 'x3', 'f1', 'f2', 'b']
z=  [1.0, 9.0, 1.0, 0.0, 0.0, 0.0]
R1: [1.0, 2.0, 3.0, 1.0, 0.0, 9.0]
R2: [3.0, 2.0, 2.0, 0.0, 1.0, 15.0]


--------------------------------------------------------------------------------
```

Output:

```
Quadro 0

      x1      x2      x3      f1      f2  |     RHS

-----------------------------------------------------
    1.00    9.00    1.00    0.00    0.00  |    0.00

-----------------------------------------------------
    1.00    2.00    3.00    1.00    0.00  |    9.00
```

```
Quadro 1

      x1      x2      x3      f1      f2  |     RHS

-----------------------------------------------------
   -3.50    0.00  -12.50   -4.50    0.00  |  -40.50

-----------------------------------------------------
    0.50    1.00    1.50    0.50    0.00  |    4.50

    2.00    0.00   -1.00   -1.00    1.00  |    6.00

Base basica: ['x2', 'f2']
z(x) = -40.5
x = [0.0, 4.5, 0.0, 0.0, 6.0]
PS C:\Users\reymy>
```

2. Input:

```
Max - M          Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:4500
x2:5000
x3:3000
x4:
   ['x1', 'x2', 'x3']
z= [4500.0, 5000.0, 3000.0]

Numero de variaveis: 3

Numero de restrições:3
R1:
x1:5000
x2:4000
x3:2500
Escolha um sinal entre: = <= =>
sinal:<=
b1:5000

R2:
x1:400
x2:600
x3:300
Escolha um sinal entre: = <= =>
sinal:<=
b2:500

R3:
x1:1
x2:1
x3:1
Escolha um sinal entre: = <= =>
sinal:<=
b3:1

R1: [5000.0, 4000.0, 2500.0] <= 5000.0
R2: [400.0, 600.0, 300.0] <= 500.0
R3: [1.0, 1.0, 1.0] <= 1.0

R1: [5000.0, 4000.0, 2500.0] <= 5000.0
R2: [400.0, 600.0, 300.0] <= 500.0
R3: [1.0, 1.0, 1.0] <= 1.0
```

```
criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:3
escreva a constante:0.2


['x1', 'x2', 'x3~', 'f1', 'f2', 'f3', 'b']
  [4500.0, 5000.0, 3000.0, 0.0, 0.0, 0.0, 0.0]
R1: [5000.0, 4000.0, 2500.0, 1.0, 0.0, 0.0, 4999.8]
R2: [400.0, 600.0, 300.0, 0.0, 1.0, 0.0, 499.8]
R3: [1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.8]


--------------------------------------------------------------------------------------------------

   ['x1', 'x2', 'x3~', 'f1', 'f2', 'f3', 'b']
z=  [4500.0, 5000.0, 3000.0, 0.0, 0.0, 0.0, 0.0]
R1: [5000.0, 4000.0, 2500.0, 1.0, 0.0, 0.0, 4999.8]
R2: [400.0, 600.0, 300.0, 0.0, 1.0, 0.0, 499.8]
R3: [1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.8]


--------------------------------------------------------------------------------------------------
```

Output:

```
Quadro 0

      x1      x2     x3~      f1      f2      f3  |     RHS

-------------------------------------------------------------
 4500.00 5000.00 3000.00    0.00    0.00    0.00  |    0.00

-------------------------------------------------------------
 5000.00 4000.00 2500.00    1.00    0.00    0.00  | 4999.80

  400.00  600.00  300.00    0.00    1.00    0.00  |  499.80

    1.00    1.00    1.00    0.00    0.00    1.00  |    0.80

Base basica: ['f1', 'f2', 'f3']
variavel a sair: f3 --> variavel a entrar: x2
-------------------------------------------------------------------------------------------------
Quadro 1

      x1      x2             x3~      f1      f2             f3  |            RHS

-------------------------------------------------------------------------------------------
 -500.00    0.00        -2000.00    0.00    0.00        -5000.00  |        -4000.00

-------------------------------------------------------------------------------------------
 1000.00    0.00        -1500.00    1.00    0.00        -4000.00  |         1799.80

 -200.00    0.00         -300.00    0.00    1.00         -600.00  |           19.80

    1.00    1.00            1.00    0.00    0.00            1.00  |            0.80

Base basica: ['f1', 'f2', 'x2']
z(x) = -4000.0
x = [0.0, 0.8, 0.0, 1799.8000000000002, 19.80000000000001, 0.0]
PS C:\Users\reymy>
```

3. Input:

```
Max - M          Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:0
x2:0
x3:0
x4:0
x5:-1
x6:
   ['x1', 'x2', 'x3', 'x4', 'x5']
z= [0.0, 0.0, 0.0, 0.0, -1.0]

Numero de variaveis: 5

Numero de restrições:5
R1:
x1:3
x2:-2
x3:-4
x4:6
x5:-1
Escolha um sinal entre: = <= =>
sinal:<=
b1:0

R2:
x1:-4
x2:2
x3:-1
x4:-8
x5:-1
Escolha um sinal entre: = <= =>
sinal:<=
b2:0

R3:
x1:0
x2:-3
x3:-2
x4:-1
x5:-1
Escolha um sinal entre: = <= =>
sinal:<=
b3:0

R4:
x1:1
x2:1
x3:1
x4:1
x5:0
Escolha um sinal entre: = <= =>
sinal:<=
b4:1

R5:
x1:-1
x2:-1
x3:-1
x4:1
x5:0
Escolha um sinal entre: = <= =>
sinal:<=
b5:-1

R1: [3.0, -2.0, -4.0, 6.0, -1.0] <= 0.0
R2: [-4.0, 2.0, -1.0, -8.0, -1.0] <= 0.0
R3: [0.0, -3.0, -2.0, -1.0, -1.0] <= 0.0
R4: [1.0, 1.0, 1.0, 1.0, 0.0] <= 1.0
R5: [-1.0, -1.0, -1.0, 1.0, 0.0] <= -1.0

R1: [3.0, -2.0, -4.0, 6.0, -1.0] <= 0.0
R2: [-4.0, 2.0, -1.0, -8.0, -1.0] <= 0.0
R3: [0.0, -3.0, -2.0, -1.0, -1.0] <= 0.0
R4: [1.0, 1.0, 1.0, 1.0, 0.0] <= 1.0
R5: [1.0, 1.0, 1.0, -1.0, -0.0] => 1.0


criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva
```

```
Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x4 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x5 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:2



['x1', 'x2', 'x3', 'x4', 'x5+', 'x5-', 'f1', 'f2', 'f3', 'f4', 'f5', 'a1', 'b']
  [0.0, 0.0, 0.0, 0.0, -1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1024.0, 0.0]
R1: [3.0, -2.0, -4.0, 6.0, -1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
R2: [-4.0, 2.0, -1.0, -8.0, -1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
R3: [0.0, -3.0, -2.0, -1.0, -1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
R4: [1.0, 1.0, 1.0, 1.0, 0.0, -0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0]
R5: [1.0, 1.0, 1.0, -1.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0, 1.0, 1.0]


--------------------------------------------------------------------------------------------------


   ['x1', 'x2', 'x3', 'x4', 'x5+', 'x5-', 'f1', 'f2', 'f3', 'f4', 'f5', 'a1', 'b']
z=  [1024.0, 1024.0, 1024.0, -1024.0, -1.0, 1.0, 0.0, 0.0, 0.0, 0.0, -1024.0, 0.0, 1024.0]
R1: [3.0, -2.0, -4.0, 6.0, -1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
R2: [-4.0, 2.0, -1.0, -8.0, -1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
R3: [0.0, -3.0, -2.0, -1.0, -1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0]
R4: [1.0, 1.0, 1.0, 1.0, 0.0, -0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0]
R5: [1.0, 1.0, 1.0, -1.0, -0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -1.0, 1.0, 1.0]


--------------------------------------------------------------------------------------------------
```

Output:

```
Quadro 0
      x1      x2      x3            x4     x5+     x5-      f1      f2      f3      f4            f5      a1 |     RHS
-------------------------------------------------------------------------------------------------------------------------
 1024.00 1024.00 1024.00       -1024.00   -1.00    1.00    0.00    0.00    0.00    0.00       -1024.00    0.00 | 1024.00
-------------------------------------------------------------------------------------------------------------------------
    3.00   -2.00   -4.00           6.00   -1.00    1.00    1.00    0.00    0.00    0.00           0.00    0.00 |    0.00
   -4.00    2.00   -1.00          -8.00   -1.00    1.00    0.00    1.00    0.00    0.00           0.00    0.00 |    0.00
    0.00   -3.00   -2.00          -1.00   -1.00    1.00    0.00    0.00    1.00    0.00           0.00    0.00 |    0.00
    1.00    1.00    1.00           1.00    0.00   -0.00    0.00    0.00    0.00    1.00           0.00    0.00 |    1.00
    1.00    1.00    1.00          -1.00   -0.00    0.00    0.00    0.00    0.00    0.00          -1.00    1.00 |    1.00
Base basica: ['f1', 'f2', 'f3', 'f4', 'a1']
variavel a sair: f1 --> variavel a entrar: x1
_________________________________________________________________________________________________________________________

Quadro 1
      x1      x2      x3            x4     x5+     x5-      f1      f2      f3      f4            f5      a1 |     RHS
-------------------------------------------------------------------------------------------------------------------------
    0.00 1706.67 2389.33       -3072.00  340.33 -340.33 -341.33    0.00    0.00    0.00       -1024.00    0.00 | 1024.00
-------------------------------------------------------------------------------------------------------------------------
    1.00   -0.67   -1.33           2.00   -0.33    0.33    0.33    0.00    0.00    0.00           0.00    0.00 |    0.00
    0.00   -0.67   -6.33           0.00   -2.33    2.33    1.33    1.00    0.00    0.00           0.00    0.00 |    0.00
    0.00   -3.00   -2.00          -1.00   -1.00    1.00    0.00    0.00    1.00    0.00           0.00    0.00 |    0.00
    0.00    1.67    2.33          -1.00    0.33   -0.33   -0.33    0.00    0.00    1.00           0.00    0.00 |    1.00
    0.00    1.67    2.33          -3.00    0.33   -0.33   -0.33    0.00    0.00    0.00          -1.00    1.00 |    1.00
Base basica: ['x1', 'f2', 'f3', 'f4', 'a1']
variavel a sair: a1 --> variavel a entrar: x3
_________________________________________________________________________________________________________________________
```

Quadro 2

x1	x2	x3	x4	x5+	x5-	f1	f2	f3	f4	f5	a1	RHS
0.00	0.00	0.00	0.00	-1.00	1.00	0.00	0.00	0.00	0.00	0.00	-1024.00	0.00
1.00	0.29	0.00	0.29	-0.14	0.14	0.14	0.00	0.00	0.00	-0.57	0.57	0.57
0.00	3.86	0.00	-8.14	-1.43	1.43	0.43	1.00	0.00	0.00	-2.71	2.71	2.71
0.00	-1.57	0.00	-3.57	-0.71	0.71	-0.29	0.00	1.00	0.00	-0.86	0.86	0.86
0.00	0.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	-1.00	0.00
0.00	0.71	1.00	-1.29	0.14	-0.14	-0.14	0.00	0.00	0.00	-0.43	0.43	0.43

Base basica: ['x1', 'f2', 'f3', 'f4', 'x3']
variavel a sair: f3 --> variavel a entrar: x5-

Quadro 3

x1	x2	x3	x4	x5+	x5-	f1	f2	f3	f4	f5	a1	RHS
0.00	2.20	0.00	5.00	0.00	0.00	0.40	0.00	-1.40	0.00	1.20	-1025.20	-1.20
1.00	0.60	0.00	1.00	0.00	0.00	0.20	0.00	-0.20	0.00	-0.40	0.40	0.40
0.00	7.00	0.00	-1.00	0.00	0.00	1.00	1.00	-2.00	0.00	-1.00	1.00	1.00
0.00	-2.20	0.00	-5.00	-1.00	1.00	-0.40	0.00	1.40	0.00	-1.20	1.20	1.20
0.00	0.00	0.00	2.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	-1.00	0.00
0.00	0.40	1.00	-2.00	0.00	0.00	-0.20	0.00	0.20	0.00	-0.60	0.60	0.60

Base basica: ['x1', 'f2', 'x5-', 'f4', 'x3']
variavel a sair: f4 --> variavel a entrar: x4

Quadro 4

x1	x2	x3	x4	x5+	x5-	f1	f2	f3	f4	f5	a1	RHS
0.00	2.20	0.00	0.00	0.00	0.00	0.40	0.00	-1.40	-2.50	-1.30	-1022.70	-1.20
1.00	0.60	0.00	0.00	0.00	0.00	0.20	0.00	-0.20	-0.50	-0.90	0.90	0.40
0.00	7.00	0.00	0.00	0.00	0.00	1.00	1.00	-2.00	0.50	-0.50	0.50	1.00
0.00	-2.20	0.00	0.00	-1.00	1.00	-0.40	0.00	1.40	2.50	1.30	-1.30	1.20
0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50	-0.50	0.00
0.00	0.40	1.00	0.00	0.00	0.00	-0.20	0.00	0.20	1.00	0.40	-0.40	0.60

Base basica: ['x1', 'f2', 'x5-', 'x4', 'x3']
variavel a sair: f2 --> variavel a entrar: x2

Quadro 5

x1	x2	x3	x4	x5+	x5-	f1	f2	f3	f4	f5	a1	RHS
0.00	0.00	0.00	0.00	0.00	0.00	0.09	-0.31	-0.77	-2.66	-1.14	-1022.86	-1.51
1.00	0.00	0.00	0.00	0.00	0.00	0.11	-0.09	-0.03	-0.54	-0.86	0.86	0.31
0.00	1.00	0.00	0.00	0.00	0.00	0.14	0.14	-0.29	0.07	-0.07	0.07	0.14
0.00	0.00	0.00	0.00	-1.00	1.00	-0.09	0.31	0.77	2.66	1.14	-1.14	1.51
0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50	-0.50	0.00
0.00	0.00	1.00	0.00	0.00	0.00	-0.26	-0.06	0.31	0.97	0.43	-0.43	0.54

Base basica: ['x1', 'x2', 'x5-', 'x4', 'x3']
variavel a sair: x2 --> variavel a entrar: f1

```
      x1     x2     x3     x4    x5+    x5-     f1     f2     f3     f4     f5            a1 |    RHS
-------------------------------------------------------------------------------------------------------
    0.00   0.00   0.00   0.00   0.00   0.00   0.09  -0.31  -0.77  -2.66  -1.14       -1022.86 |  -1.51
-------------------------------------------------------------------------------------------------------
    1.00   0.00   0.00   0.00   0.00   0.00   0.11  -0.09  -0.03  -0.54  -0.86           0.86 |   0.31
    0.00   1.00   0.00   0.00   0.00   0.00   0.14   0.14  -0.29   0.07  -0.07           0.07 |   0.14
    0.00   0.00   0.00   0.00  -1.00   1.00  -0.09   0.31   0.77   2.66   1.14          -1.14 |   1.51
    0.00   0.00   0.00   1.00   0.00   0.00   0.00   0.00   0.00   0.50   0.50          -0.50 |   0.00
    0.00   0.00   1.00   0.00   0.00   0.00  -0.26  -0.06   0.31   0.97   0.43          -0.43 |   0.54

Base basica: ['x1', 'x2', 'x5-', 'x4', 'x3']
variavel a sair: x2 --> variavel a entrar: f1
_____________________________________________________________________________________________

Quadro 6

      x1     x2     x3     x4    x5+    x5-     f1     f2     f3     f4     f5            a1 |    RHS
-------------------------------------------------------------------------------------------------------
    0.00  -0.60   0.00   0.00   0.00   0.00   0.00  -0.40  -0.60  -2.70  -1.10       -1022.90 |  -1.60
-------------------------------------------------------------------------------------------------------
    1.00  -0.80   0.00   0.00   0.00   0.00   0.00  -0.20   0.20  -0.60  -0.80           0.80 |   0.20
    0.00   7.00   0.00   0.00   0.00   0.00   1.00   1.00  -2.00   0.50  -0.50           0.50 |   1.00
    0.00   0.60   0.00   0.00  -1.00   1.00   0.00   0.40   0.60   2.70   1.10          -1.10 |   1.60
    0.00   0.00   0.00   1.00   0.00   0.00   0.00   0.00   0.00   0.50   0.50          -0.50 |   0.00
    0.00   1.80   1.00   0.00   0.00   0.00   0.00   0.20  -0.20   1.10   0.30          -0.30 |   0.80

Base basica: ['x1', 'f1', 'x5-', 'x4', 'x3']
z(x) = -1.6000000000000005
x = [0.19999999999999996, 0.0, 0.8000000000000003, 0.0, 0.0, 1.6000000000000005, 1.0000000000000007, 0.0, 0.0, 0.0, 0.0, 0.0]
PS C:\Users\reymy>
```

4. Input:

```
Max - M          Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:1
x2:-2
x3:1
x4:
   ['x1', 'x2', 'x3']
z= [1.0, -2.0, 1.0]

Numero de variaveis: 3

Numero de restrições:3
R1:
x1:1
x2:2
x3:1
Escolha um sinal entre: = <= =>
sinal:<=
b1:12

R2:
x1:2
x2:1
x3:-1
Escolha um sinal entre: = <= =>
sinal:<=
b2:6

R3:
x1:-1
x2:3
x3:0
Escolha um sinal entre: = <= =>
sinal:<=
b3:9

R1: [1.0, 2.0, 1.0] <= 12.0
R2: [2.0, 1.0, -1.0] <= 6.0
R3: [-1.0, 3.0, 0.0] <= 9.0

R1: [1.0, 2.0, 1.0] <= 12.0
R2: [2.0, 1.0, -1.0] <= 6.0
R3: [-1.0, 3.0, 0.0] <= 9.0
```

```
criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1



['x1', 'x2', 'x3', 'f1', 'f2', 'f3', 'b']
  [1.0, -2.0, 1.0, 0.0, 0.0, 0.0, 0.0]
R1: [1.0, 2.0, 1.0, 1.0, 0.0, 0.0, 12.0]
R2: [2.0, 1.0, -1.0, 0.0, 1.0, 0.0, 6.0]
R3: [-1.0, 3.0, 0.0, 0.0, 0.0, 1.0, 9.0]


--------------------------------------------------------------------------------------------------

   ['x1', 'x2', 'x3', 'f1', 'f2', 'f3', 'b']
z=  [1.0, -2.0, 1.0, 0.0, 0.0, 0.0, 0.0]
R1: [1.0, 2.0, 1.0, 1.0, 0.0, 0.0, 12.0]
R2: [2.0, 1.0, -1.0, 0.0, 1.0, 0.0, 6.0]
R3: [-1.0, 3.0, 0.0, 0.0, 0.0, 1.0, 9.0]


--------------------------------------------------------------------------------------------------
```

Output:

```
Quadro 0

      x1     x2     x3     f1     f2     f3  |     RHS

 ------------------------------------------------------------
    1.00   -2.00   1.00   0.00   0.00   0.00  |   0.00

 ------------------------------------------------------------
    1.00   2.00   1.00   1.00   0.00   0.00  |  12.00

    2.00   1.00  -1.00   0.00   1.00   0.00  |   6.00

   -1.00   3.00   0.00   0.00   0.00   1.00  |   9.00

Base basica: ['f1', 'f2', 'f3']
variavel a sair: f2 --> variavel a entrar: x1
 ------------------------------------------------------------

Quadro 1

      x1     x2     x3     f1     f2     f3  |     RHS

 ------------------------------------------------------------
    0.00   -2.50   1.50   0.00  -0.50   0.00  |  -3.00

 ------------------------------------------------------------
    0.00   1.50   1.50   1.00  -0.50   0.00  |   9.00

    1.00   0.50  -0.50   0.00   0.50   0.00  |   3.00

    0.00   3.50  -0.50   0.00   0.50   1.00  |  12.00

Base basica: ['f1', 'x1', 'f3']
variavel a sair: f1 --> variavel a entrar: x3
 ------------------------------------------------------------

Quadro 2


      x1     x2     x3     f1     f2     f3  |     RHS

 ------------------------------------------------------------
    0.00   -4.00   0.00  -1.00   0.00   0.00  |  -12.00
```

```
    2.00   1.00  -1.00   0.00   1.00   0.00  |   6.00

   -1.00   3.00   0.00   0.00   0.00   1.00  |   9.00

Base basica: ['f1', 'f2', 'f3']
variavel a sair: f2 --> variavel a entrar: x1
 ------------------------------------------------------------
```

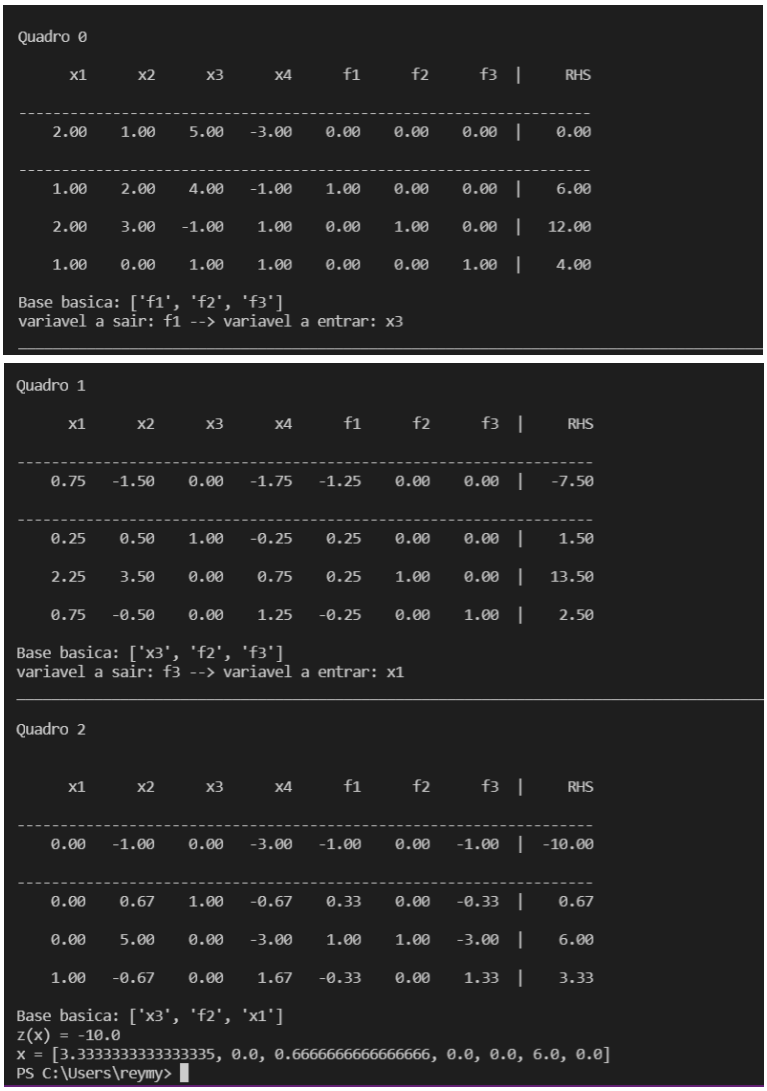
5. Input:

```
Max - M          Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:2
x2:1
x3:5
x4:-3
x5:
   ['x1', 'x2', 'x3', 'x4']
z= [2.0, 1.0, 5.0, -3.0]

Numero de variaveis: 4

Numero de restrições:3
R1:
x1:1
x2:2
x3:4
x4:-1
Escolha um sinal entre: = <= =>
sinal:<=
b1:6

R2:
x1:2
x2:3
x3:-1
x4:1
Escolha um sinal entre: = <= =>
sinal:<=
b2:12

R3:
x1:1
x2:0
x3:1
x4:1
Escolha um sinal entre: = <= =>
sinal:<=
b3:4

R1: [1.0, 2.0, 4.0, -1.0] <= 6.0
R2: [2.0, 3.0, -1.0, 1.0] <= 12.0
R3: [1.0, 0.0, 1.0, 1.0] <= 4.0
```

```
R1: [1.0, 2.0, 4.0, -1.0] <= 6.0
R2: [2.0, 3.0, -1.0, 1.0] <= 12.0
R3: [1.0, 0.0, 1.0, 1.0] <= 4.0


criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x4 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1



['x1', 'x2', 'x3', 'x4', 'f1', 'f2', 'f3', 'b']
  [2.0, 1.0, 5.0, -3.0, 0.0, 0.0, 0.0, 0.0]
R1: [1.0, 2.0, 4.0, -1.0, 1.0, 0.0, 0.0, 6.0]
R2: [2.0, 3.0, -1.0, 1.0, 0.0, 1.0, 0.0, 12.0]
R3: [1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 4.0]
```

```
   ['x1', 'x2', 'x3', 'x4', 'f1', 'f2', 'f3', 'b']
z=  [2.0, 1.0, 5.0, -3.0, 0.0, 0.0, 0.0, 0.0]
R1: [1.0, 2.0, 4.0, -1.0, 1.0, 0.0, 0.0, 6.0]
R2: [2.0, 3.0, -1.0, 1.0, 0.0, 1.0, 0.0, 12.0]
R3: [1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 4.0]
```

Output:



```
Quadro 0

      x1     x2     x3     x4     f1     f2     f3  |    RHS

-------------------------------------------------------------
    2.00   1.00   5.00  -3.00   0.00   0.00   0.00  |   0.00

-------------------------------------------------------------
    1.00   2.00   4.00  -1.00   1.00   0.00   0.00  |   6.00

    2.00   3.00  -1.00   1.00   0.00   1.00   0.00  |  12.00

    1.00   0.00   1.00   1.00   0.00   0.00   1.00  |   4.00

Base basica: ['f1', 'f2', 'f3']
variavel a sair: f1 --> variavel a entrar: x3
-------------------------------------------------------------
Quadro 1

      x1     x2     x3     x4     f1     f2     f3  |    RHS

-------------------------------------------------------------
    0.75  -1.50   0.00  -1.75  -1.25   0.00   0.00  |  -7.50

-------------------------------------------------------------
    0.25   0.50   1.00  -0.25   0.25   0.00   0.00  |   1.50

    2.25   3.50   0.00   0.75   0.25   1.00   0.00  |  13.50

    0.75  -0.50   0.00   1.25  -0.25   0.00   1.00  |   2.50

Base basica: ['x3', 'f2', 'f3']
variavel a sair: f3 --> variavel a entrar: x1
-------------------------------------------------------------
Quadro 2


      x1     x2     x3     x4     f1     f2     f3  |    RHS

-------------------------------------------------------------
    0.00  -1.00   0.00  -3.00  -1.00   0.00  -1.00  | -10.00

-------------------------------------------------------------
    0.00   0.67   1.00  -0.67   0.33   0.00  -0.33  |   0.67

    0.00   5.00   0.00  -3.00   1.00   1.00  -3.00  |   6.00

    1.00  -0.67   0.00   1.67  -0.33   0.00   1.33  |   3.33

Base basica: ['x3', 'f2', 'x1']
z(x) = -10.0
x = [3.3333333333333335, 0.0, 0.6666666666666666, 0.0, 0.0, 6.0, 0.0]
PS C:\Users\reymy>
```

6. Input:

```
Max - M          Min - m
m
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:3
x2:-1
x3:
   ['x1', 'x2']
z= [-3.0, 1.0]

Numero de variaveis: 2

Numero de restrições:4
R1:
x1:1
x2:-3
Escolha um sinal entre: = <= =>
sinal:>=
b1:-3

R2:
x1:2
x2:1
Escolha um sinal entre: = <= =>
sinal:>=
b2:-2

R3:
x1:2
x2:1
Escolha um sinal entre: = <= =>
sinal:<=
b3:8

R4:
x1:4
x2:-1
Escolha um sinal entre: = <= =>
sinal:<=
b4:16

R1: [1.0, -3.0] >= -3.0
R2: [2.0, 1.0] >= -2.0
R3: [2.0, 1.0] <= 8.0
R4: [4.0, -1.0] <= 16.0
```

```
R1: [-1.0, 3.0] <= 3.0
R2: [-2.0, -1.0] <= 2.0
R3: [2.0, 1.0] <= 8.0
R4: [4.0, -1.0] <= 16.0


criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:2
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:2



['x1+', 'x1-', 'x2+', 'x2-', 'f1', 'f2', 'f3', 'f4', 'b']
  [-3.0, 3.0, 1.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
R1: [-1.0, 1.0, 3.0, -3.0, 1.0, 0.0, 0.0, 0.0, 3.0]
R2: [-2.0, 2.0, -1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 2.0]
R3: [2.0, -2.0, 1.0, -1.0, 0.0, 0.0, 1.0, 0.0, 8.0]
R4: [4.0, -4.0, -1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 16.0]


-------------------------------------------------------------------------------------------------------

   ['x1+', 'x1-', 'x2+', 'x2-', 'f1', 'f2', 'f3', 'f4', 'b']
z=  [-3.0, 3.0, 1.0, -1.0, 0.0, 0.0, 0.0, 0.0, 0.0]
R1: [-1.0, 1.0, 3.0, -3.0, 1.0, 0.0, 0.0, 0.0, 3.0]
R2: [-2.0, 2.0, -1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 2.0]
R3: [2.0, -2.0, 1.0, -1.0, 0.0, 0.0, 1.0, 0.0, 8.0]
R4: [4.0, -4.0, -1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 16.0]
```

Output:

```
Quadro 0
     x1+     x1-     x2+     x2-      f1      f2      f3      f4 |     RHS
-------------------------------------------------------------------------
   -3.00    3.00    1.00   -1.00    0.00    0.00    0.00    0.00 |    0.00
-------------------------------------------------------------------------
   -1.00    1.00    3.00   -3.00    1.00    0.00    0.00    0.00 |    3.00
   -2.00    2.00   -1.00    1.00    0.00    1.00    0.00    0.00 |    2.00
    2.00   -2.00    1.00   -1.00    0.00    0.00    1.00    0.00 |    8.00
    4.00   -4.00   -1.00    1.00    0.00    0.00    0.00    1.00 |   16.00
Base basica: ['f1', 'f2', 'f3', 'f4']
variavel a sair: f2 --> variavel a entrar: x1-
_________________________________________________________________________

Quadro 1
     x1+     x1-     x2+     x2-      f1      f2      f3      f4 |     RHS
-------------------------------------------------------------------------
    0.00    0.00    2.50   -2.50    0.00   -1.50    0.00    0.00 |   -3.00
-------------------------------------------------------------------------
    0.00    0.00    3.50   -3.50    1.00   -0.50    0.00    0.00 |    2.00
   -1.00    1.00   -0.50    0.50    0.00    0.50    0.00    0.00 |    1.00
    0.00    0.00    0.00    0.00    0.00    1.00    1.00    0.00 |   10.00
    0.00    0.00   -3.00    3.00    0.00    2.00    0.00    1.00 |   20.00
Base basica: ['f1', 'x1-', 'f3', 'f4']
variavel a sair: f1 --> variavel a entrar: x2+
_________________________________________________________________________
```

```
Quadro 2

     x1+     x1-     x2+     x2-      f1      f2      f3      f4 |     RHS
-------------------------------------------------------------------------
    0.00    0.00    0.00    0.00   -0.71   -1.14    0.00    0.00 |   -4.43
-------------------------------------------------------------------------
    0.00    0.00    1.00   -1.00    0.29   -0.14    0.00    0.00 |    0.57
   -1.00    1.00    0.00    0.00    0.14    0.43    0.00    0.00 |    1.29
    0.00    0.00    0.00    0.00    0.00    1.00    1.00    0.00 |   10.00
    0.00    0.00    0.00    0.00    0.86    1.57    0.00    1.00 |   21.71
Base basica: ['x2+', 'x1-', 'f3', 'f4']
z(x) = -4.428571428571429
x = [0.0, 1.2857142857142856, 0.5714285714285714, 0.0, 0.0, 0.0, 10.0, 21.714285714285715]
PS C:\Users\reymy>
```

7. Input:

```
Max - M         Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:-3
x2:2
x3:-1
x4:1
x5:
   ['x1', 'x2', 'x3', 'x4']
z= [-3.0, 2.0, -1.0, 1.0]

Numero de variaveis: 4

Numero de restrições:3
R1:
x1:2
x2:-3
x3:-1
x4:1
Escolha um sinal entre: = <= =>
sinal:<=
b1:8

R2:
x1:-1
x2:2
x3:2
x4:-3
Escolha um sinal entre: = <= =>
sinal:<=
b2:10

R3:
x1:-1
x2:1
x3:-4
x4:1
Escolha um sinal entre: = <= =>
sinal:<=
b3:3

R1: [2.0, -3.0, -1.0, 1.0] <= 8.0
R2: [-1.0, 2.0, 2.0, -3.0] <= 10.0
R3: [-1.0, 1.0, -4.0, 1.0] <= 3.0
```

```
R1: [2.0, -3.0, -1.0, 1.0] <= 8.0
R2: [-1.0, 2.0, 2.0, -3.0] <= 10.0
R3: [-1.0, 1.0, -4.0, 1.0] <= 3.0


criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x4 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1



['x1', 'x2', 'x3', 'x4', 'f1', 'f2', 'f3', 'b']
  [-3.0, 2.0, -1.0, 1.0, 0.0, 0.0, 0.0, 0.0]
R1: [2.0, -3.0, -1.0, 1.0, 1.0, 0.0, 0.0, 8.0]
R2: [-1.0, 2.0, 2.0, -3.0, 0.0, 1.0, 0.0, 10.0]
R3: [-1.0, 1.0, -4.0, 1.0, 0.0, 0.0, 1.0, 3.0]
```

```
--------------------------------------------------------------------------------


   ['x1', 'x2', 'x3', 'x4', 'f1', 'f2', 'f3', 'b']
z=  [-3.0, 2.0, -1.0, 1.0, 0.0, 0.0, 0.0, 0.0]
R1: [2.0, -3.0, -1.0, 1.0, 1.0, 0.0, 0.0, 8.0]
R2: [-1.0, 2.0, 2.0, -3.0, 0.0, 1.0, 0.0, 10.0]
R3: [-1.0, 1.0, -4.0, 1.0, 0.0, 0.0, 1.0, 3.0]
```

Output:

```
Quadro 0

      x1      x2      x3      x4      f1      f2      f3  |     RHS

-------------------------------------------------------------------------
   -3.00    2.00   -1.00    1.00    0.00    0.00    0.00  |    0.00

-------------------------------------------------------------------------
    2.00   -3.00   -1.00    1.00    1.00    0.00    0.00  |    8.00

   -1.00    2.00    2.00   -3.00    0.00    1.00    0.00  |   10.00

   -1.00    1.00   -4.00    1.00    0.00    0.00    1.00  |    3.00

Base basica: ['f1', 'f2', 'f3']
variavel a sair: f3 --> variavel a entrar: x2
```

```
Quadro 1

      x1      x2      x3      x4      f1      f2      f3  |     RHS

-------------------------------------------------------------------------
   -1.00    0.00    7.00   -1.00    0.00    0.00   -2.00  |   -6.00

-------------------------------------------------------------------------
   -1.00    0.00  -13.00    4.00    1.00    0.00    3.00  |   17.00

    1.00    0.00   10.00   -5.00    0.00    1.00   -2.00  |    4.00

   -1.00    1.00   -4.00    1.00    0.00    0.00    1.00  |    3.00

Base basica: ['f1', 'f2', 'x2']
variavel a sair: f2 --> variavel a entrar: x3
_________________________________________________________________________

Quadro 2

      x1      x2      x3      x4      f1      f2      f3  |     RHS

-------------------------------------------------------------------------
   -1.70    0.00    0.00    2.50    0.00   -0.70   -0.60  |   -8.80

-------------------------------------------------------------------------
    0.30    0.00    0.00   -2.50    1.00    1.30    0.40  |   22.20

    0.10    0.00    1.00   -0.50    0.00    0.10   -0.20  |    0.40

   -0.60    1.00    0.00   -1.00    0.00    0.40    0.20  |    4.60

Base basica: ['f1', 'x3', 'x2']
Traceback (most recent call last):
  File "c:\Users\reymy\OneDrive\Área de Trabalho\Simplex\iniciarSilplex.py", line 380, in <module>
    simplex.solve()
  File "c:\Users\reymy\OneDrive\Área de Trabalho\Simplex\simplex.py", line 234, in solve
    print(f'variavel a sair: {vb[self.gettxitVar(self.getEntryVar())-1]}',f'variavel a entrar: {self.variabels[self.getEntryVar()]}', sep= ' --> ')
  File "c:\Users\reymy\OneDrive\Área de Trabalho\Simplex\simplex.py", line 67, in getExitVar
    idx = min(result, key=result.get)
ValueError: min() arg is an empty sequence
PS C:\Users\reymy>
```

8. Input:

```
Max - M          Min - m
m
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:-3/4
x2:20
x3:-1/2
x4:6
x5:
   ['x1', 'x2', 'x3', 'x4']
z= [0.75, -20.0, 0.5, -6.0]

Numero de variaveis: 4

Numero de restrições:3
R1:
x1:1/4
x2:-8
x3:-1
x4:9
Escolha um sinal entre: = <= =>
sinal:<=
b1:0

R2:
x1:1/2
x2:-12
x3:-1/2
x4:3
Escolha um sinal entre: = <= =>
sinal:<=
b2:0

R3:
x1:0
x2:0
x3:1
x4:0
Escolha um sinal entre: = <= =>
sinal:<=
b3:1

R1: [0.25, -8.0, -1.0, 9.0] <= 0.0
R2: [0.5, -12.0, -0.5, 3.0] <= 0.0
R3: [0.0, 0.0, 1.0, 0.0] <= 1.0
```

```
R1: [0.25, -8.0, -1.0, 9.0] <= 0.0
R2: [0.5, -12.0, -0.5, 3.0] <= 0.0
R3: [0.0, 0.0, 1.0, 0.0] <= 1.0


criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x4 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1



['x1', 'x2', 'x3', 'x4', 'f1', 'f2', 'f3', 'b']
  [0.75, -20.0, 0.5, -6.0, 0.0, 0.0, 0.0, 0.0]
R1: [0.25, -8.0, -1.0, 9.0, 1.0, 0.0, 0.0, 0.0]
R2: [0.5, -12.0, -0.5, 3.0, 0.0, 1.0, 0.0, 0.0]
R3: [0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0]
```

Output

```
Quadro 0

      x1      x2      x3      x4      f1      f2      f3  |     RHS

-----------------------------------------------------------------------
    0.75  -20.00    0.50   -6.00    0.00    0.00    0.00  |    0.00

-----------------------------------------------------------------------
    0.25   -8.00   -1.00    9.00    1.00    0.00    0.00  |    0.00

    0.50  -12.00   -0.50    3.00    0.00    1.00    0.00  |    0.00

    0.00    0.00    1.00    0.00    0.00    0.00    1.00  |    1.00

Base basica: ['f1', 'f2', 'f3']
variavel a sair: f2 --> variavel a entrar: x1
```

```
Quadro 1

      x1      x2      x3      x4      f1      f2      f3  |     RHS

-----------------------------------------------------------------
    0.00   -2.00    1.25  -10.50    0.00   -1.50    0.00  |    0.00

-----------------------------------------------------------------
    0.00   -2.00   -0.75    7.50    1.00   -0.50    0.00  |    0.00

    1.00  -24.00   -1.00    6.00    0.00    2.00    0.00  |    0.00

    0.00    0.00    1.00    0.00    0.00    0.00    1.00  |    1.00

Base basica: ['f1', 'x1', 'f3']
variavel a sair: f3 --> variavel a entrar: x3
```

```
Quadro 2


      x1      x2      x3      x4      f1      f2      f3  |     RHS

-----------------------------------------------------------------
    0.00   -2.00    0.00  -10.50    0.00   -1.50   -1.25  |   -1.25

-----------------------------------------------------------------
    0.00   -2.00    0.00    7.50    1.00   -0.50    0.75  |    0.75

    1.00  -24.00    0.00    6.00    0.00    2.00    1.00  |    1.00

    0.00    0.00    1.00    0.00    0.00    0.00    1.00  |    1.00

Base basica: ['f1', 'x1', 'x3']
z(x) = -1.25
x = [1.0, 0.0, 1.0, 0.0, 0.75, 0.0, 0.0]
PS C:\Users\reymy>
```

9. Input:

```
Max - M         Min - m
m
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:-2
x2:2
x3:1
x4:1
x5:
   ['x1', 'x2', 'x3', 'x4']
z= [2.0, -2.0, -1.0, -1.0]

Numero de variaveis: 4

Numero de restrições:3
R1:
x1:1
x2:2
x3:1
x4:1
Escolha um sinal entre: = <= =>
sinal:<=
b1:2

R2:
x1:1
x2:-1
x3:1
x4:2
Escolha um sinal entre: = <= =>
sinal:>=
b2:3

R3:
x1:2
x2:-1
x3:1
x4:0
Escolha um sinal entre: = <= =>
sinal:>=
b3:2

R1: [1.0, 2.0, 1.0, 1.0] <= 2.0
R2: [1.0, -1.0, 1.0, 2.0] >= 3.0
R3: [2.0, -1.0, 1.0, 0.0] >= 2.0
```

```
R1: [1.0, 2.0, 1.0, 1.0] <= 2.0
R2: [1.0, -1.0, 1.0, 2.0] >= 3.0
R3: [2.0, -1.0, 1.0, 0.0] >= 2.0


criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x4 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1



['x1', 'x2', 'x3', 'x4', 'f1', 'f2', 'f3', 'a1', 'a2', 'b']
  [2.0, -2.0, -1.0, -1.0, 0.0, 0.0, 0.0, 1024.0, 1024.0, 0.0]
R1: [1.0, 2.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0]
R2: [1.0, -1.0, 1.0, 2.0, 0.0, -1.0, 0.0, 1.0, 0.0, 3.0]
R3: [2.0, -1.0, 1.0, 0.0, 0.0, 0.0, -1.0, 0.0, 1.0, 2.0]
```

Output:

```
Quadro 0

             x1      x2             x3             x4      f1      f2      f3      a1      a2  |             RHS

-------------------------------------------------------------------------------------------------------------------
        -3070.00 2046.00        -2049.00       -2049.00    0.00 1024.00 1024.00    0.00    0.00  |        -5120.00

-------------------------------------------------------------------------------------------------------------------
           1.00    2.00           1.00           1.00    1.00    0.00    0.00    0.00    0.00  |           2.00

           1.00   -1.00           1.00           2.00    0.00   -1.00    0.00    1.00    0.00  |           3.00

           2.00   -1.00           1.00           0.00    0.00    0.00   -1.00    0.00    1.00  |           2.00

Base basica: ['f1', 'a1', 'a2']
variavel a sair: f1 --> variavel a entrar: x2
```

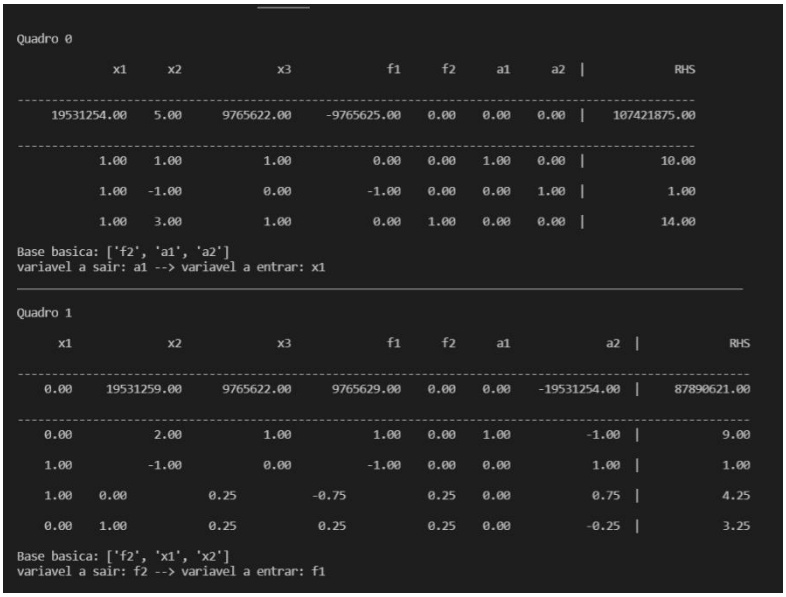
```
Quadro 1

           x1      x2             x3             x4             f1      f2      f3      a1      a2  |             RHS

-------------------------------------------------------------------------------------------------------------------
       -4093.00    0.00       -3072.00       -3072.00       -1023.00 1024.00 1024.00    0.00    0.00  |        -7166.00

-------------------------------------------------------------------------------------------------------------------
          0.50    1.00           0.50           0.50           0.50    0.00    0.00    0.00    0.00  |           1.00

          1.50    0.00           1.50           2.50           0.50   -1.00    0.00    1.00    0.00  |           4.00

          2.50    0.00           1.50           0.50           0.50    0.00   -1.00    0.00    1.00  |           3.00

Base basica: ['x2', 'a1', 'a2']
Traceback (most recent call last):
  File "c:\Users\reymy\OneDrive\Área de Trabalho\Simplex\iniciarSilplex.py", line 380, in <module>
    simplex.solve()
  File "c:\Users\reymy\OneDrive\Área de Trabalho\Simplex\simplex.py", line 234, in solve
    print(f'variavel a sair: {vb[self.getExitVar(self.getEntryVar())-1]}',f'variavel a entrar: {self.variabels[self.getEntryVar()]}', sep= ' --> ')
  File "c:\Users\reymy\OneDrive\Área de Trabalho\Simplex\simplex.py", line 67, in getExitVar
    idx = min(result, key=result.get)
ValueError: min() arg is an empty sequence
PS C:\Users\reymy>
```

10. Input:

```
Max - M          Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:4
x2:5
x3:-3
x4:
   ['x1', 'x2', 'x3']
z= [4.0, 5.0, -3.0]

Numero de variaveis: 3

Numero de restrições:3
R1:
x1:1
x2:1
x3:1
Escolha um sinal entre: = <= =>
sinal:=
b1:10

R2:
x1:1
x2:-1
x3:0
Escolha um sinal entre: = <= =>
sinal:>=
b2:1

R3:
x1:1
x2:3
x3:1
Escolha um sinal entre: = <= =>
sinal:<=
b3:14

R1: [1.0, 1.0, 1.0] = 10.0
R2: [1.0, -1.0, 0.0] >= 1.0
R3: [1.0, 3.0, 1.0] <= 14.0

R1: [1.0, 1.0, 1.0] = 10.0
R2: [1.0, -1.0, 0.0] >= 1.0
R3: [1.0, 3.0, 1.0] <= 14.0
```

```
criterio de nao negatividade das variaveis
x1 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x2 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1
x3 :
 1 - positiva
 2- ilimitada
 3- maior que uma constante positiva

Resposta:1



['x1', 'x2', 'x3', 'f1', 'f2', 'a1', 'a2', 'b']
  [4.0, 5.0, -3.0, 0.0, 0.0, -9765625.0, -9765625.0, 0.0]
R1: [1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 10.0]
R2: [1.0, -1.0, 0.0, -1.0, 0.0, 0.0, 1.0, 1.0]
R3: [1.0, 3.0, 1.0, 0.0, 1.0, 0.0, 0.0, 14.0]


--------------------------------------------------------------------------------


   ['x1', 'x2', 'x3', 'f1', 'f2', 'a1', 'a2', 'b']
z=  [19531254.0, 5.0, 9765622.0, -9765625.0, 0.0, 0.0, 0.0, 107421875.0]
R1: [1.0, 1.0, 1.0, 0.0, 0.0, 1.0, 0.0, 10.0]
R2: [1.0, -1.0, 0.0, -1.0, 0.0, 0.0, 1.0, 1.0]
R3: [1.0, 3.0, 1.0, 0.0, 1.0, 0.0, 0.0, 14.0]
```

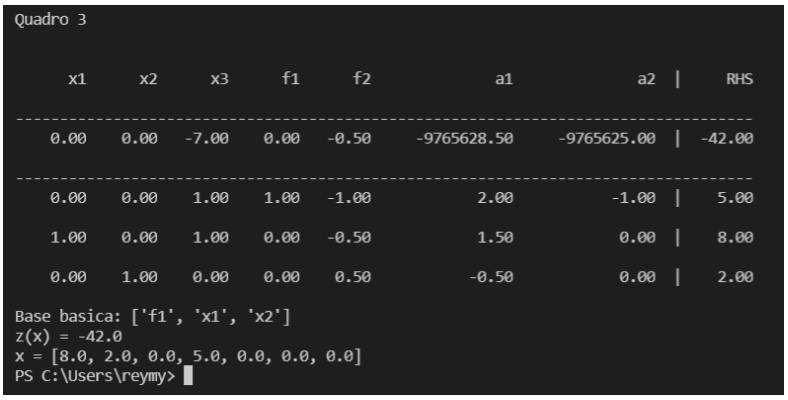
Output:



```
Quadro 0

            x1     x2            x3            f1     f2     a1     a2  |           RHS

-------------------------------------------------------------------------------------------

     19531254.00    5.00     9765622.00    -9765625.00    0.00   0.00   0.00  |   107421875.00

-------------------------------------------------------------------------------------------

           1.00    1.00           1.00           0.00   0.00   1.00   0.00  |          10.00

           1.00   -1.00           0.00          -1.00   0.00   0.00   1.00  |           1.00

           1.00    3.00           1.00           0.00   1.00   0.00   0.00  |          14.00

Base basica: ['f2', 'a1', 'a2']
variavel a sair: a1 --> variavel a entrar: x1
```



```
Quadro 1

      x1            x2            x3            f1     f2     a1            a2  |           RHS

-------------------------------------------------------------------------------------------

    0.00     19531259.00     9765622.00     9765629.00   0.00   0.00   -19531254.00  |    87890621.00

-------------------------------------------------------------------------------------------

    0.00           2.00           1.00           1.00   0.00   1.00          -1.00  |           9.00

    1.00          -1.00           0.00          -1.00   0.00   0.00           1.00  |           1.00

    1.00    0.00           0.25          -0.75           0.25   0.00           0.75  |           4.25

    0.00    1.00           0.25           0.25           0.25   0.00          -0.25  |           3.25

Base basica: ['f2', 'x1', 'x2']
variavel a sair: f2 --> variavel a entrar: f1
```



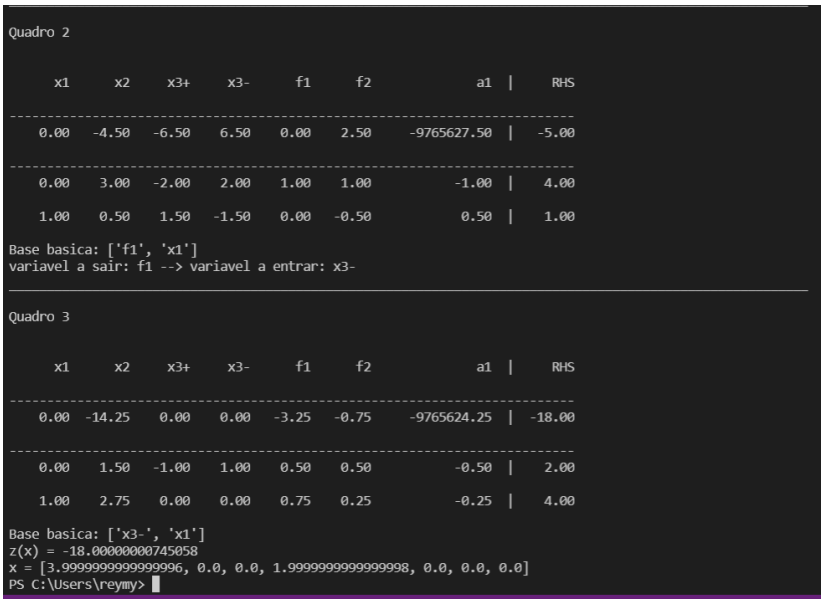
```
Quadro 3

      x1      x2      x3      f1      f2             a1             a2  |     RHS

-------------------------------------------------------------------------------------

    0.00    0.00   -7.00    0.00   -0.50     -9765628.50     -9765625.00  |  -42.00

-------------------------------------------------------------------------------------

    0.00    0.00    1.00    1.00   -1.00            2.00           -1.00  |    5.00

    1.00    0.00    1.00    0.00   -0.50            1.50            0.00  |    8.00

    0.00    1.00    0.00    0.00    0.50           -0.50            0.00  |    2.00

Base basica: ['f1', 'x1', 'x2']
z(x) = -42.0
x = [8.0, 2.0, 0.0, 5.0, 0.0, 0.0, 0.0]
PS C:\Users\reymy>
```

11. Input:

```
Max - M          Min - m
M
escreva os coeficientes da funçao a otimizar (carregue no enter para acabar o ciclo):
x1:5
x2:-2
x3:1
x4:
   ['x1', 'x2', 'x3']
z= [5.0, -2.0, 1.0]

Numero de variaveis: 3

Numero de restrições:2
R1:
x1:2
x2:4
x3:1
Escolha um sinal entre: = <= =>
sinal:<=
b1:6

R2:
x1:2
x2:1
x3:3
Escolha um sinal entre: = <= =>
sinal:>=
b2:2

R1: [2.0, 4.0, 1.0] <= 6.0
R2: [2.0, 1.0, 3.0] >= 2.0

R1: [2.0, 4.0, 1.0] <= 6.0
R2: [2.0, 1.0, 3.0] >= 2.0

 criterio de nao negatividade das variaveis
 x1 :
  1 - positiva
  2- ilimitada
  3- maior que uma constante positiva

 Resposta:1
 x2 :
  1 - positiva
  2- ilimitada
  3- maior que uma constante positiva

 Resposta:1
 x3 :
  1 - positiva
  2- ilimitada
  3- maior que uma constante positiva

 Resposta:2



['x1', 'x2', 'x3+', 'x3-', 'f1', 'f2', 'a1', 'b']
  [5.0, -2.0, 1.0, -1.0, 0.0, 0.0, -9765625.0, 0.0]
R1: [2.0, 4.0, 1.0, -1.0, 1.0, 0.0, 0.0, 6.0]
R2: [2.0, 1.0, 3.0, -3.0, 0.0, -1.0, 1.0, 2.0]


 ----------------------------------------------------------------------------------------------------


   ['x1', 'x2', 'x3+', 'x3-', 'f1', 'f2', 'a1', 'b']
z=  [19531255.0, 9765623.0, 29296876.0, -29296876.0, 0.0, -9765625.0, 0.0, 19531250.0]
R1: [2.0, 4.0, 1.0, -1.0, 1.0, 0.0, 0.0, 6.0]
R2: [2.0, 1.0, 3.0, -3.0, 0.0, -1.0, 1.0, 2.0]
```

Output:

```
Quadro 0

            x1            x2           x3+           x3-     f1            f2      a1  |           RHS

-------------------------------------------------------------------------------------------------------------

     19531255.00     9765623.00    29296876.00   -29296876.00    0.00    -9765625.00    0.00  |    19531250.00

-------------------------------------------------------------------------------------------------------------

           2.00           4.00           1.00          -1.00    1.00           0.00    0.00  |           6.00

           2.00           1.00           3.00          -3.00    0.00          -1.00    1.00  |           2.00

Base basica: ['f1', 'a1']
variavel a sair: a1 --> variavel a entrar: x3+
_____________________________________________________________________________________________

Quadro 1

      x1     x2     x3+     x3-      f1      f2            a1  |     RHS

-----------------------------------------------------------------------

    4.33   -2.33    0.00    0.00    0.00    0.33    -9765625.33  |   -0.67

-----------------------------------------------------------------------

    1.33    3.67    0.00    0.00    1.00    0.33          -0.33  |    5.33

    0.67    0.33    1.00   -1.00    0.00   -0.33           0.33  |    0.67

Base basica: ['f1', 'x3+']
variavel a sair: x3+ --> variavel a entrar: x1
_____________________________________________________________________________________________
```


Quadro 2
x1 x2 x3+ x3- f1 f2 a1 | RHS
0.00 -4.50 -6.50 6.50 0.00 2.50 -9765627.50 | -5.00
0.00 3.00 -2.00 2.00 1.00 1.00 -1.00 | 4.00
1.00 0.50 1.50 -1.50 0.00 -0.50 0.50 | 1.00
Base basica: ['f1', 'x1']
variavel a sair: f1 --> variavel a entrar: x3-
Quadro 3
x1 x2 x3+ x3- f1 f2 a1 | RHS
0.00 -14.25 0.00 0.00 -3.25 -0.75 -9765624.25 | -18.00
0.00 1.50 -1.00 1.00 0.50 0.50 -0.50 | 2.00
1.00 2.75 0.00 0.00 0.75 0.25 -0.25 | 4.00
Base basica: ['x3-', 'x1']
z(x) = -18.00000000745058
x = [3.9999999999999996, 0.0, 0.0, 1.9999999999999998, 0.0, 0.0, 0.0]
PS C:\Users\reymy>