# Trabalho prático

**Implementação do método Simplex**

Data de entrega: 10/01/2024

O objetivo deste trabalho é resolver PLs gerais, a serem fornecidas e cujo formato será especificado abaixo.

(i) A implementação deve ser feita em Python 3.10 ou superior. É permitido o uso da biblioteca `numpy` para gerenciar as matrizes, mas você deve implementar as operações de pivoteamento.

(ii) Tome cuidado com o condicionamento da matriz. Sugiro transformar números pequenos o suficiente em 0.

(iii) Seu programa deve receber um argumento em linha de comando, um arquivo de entrada com a PL. A saída deve ser impressa no `stdout`.

(iv) Para o formato do arquivo texto de entrada, considere o seguinte exemplo:

$$\begin{aligned} \max \quad & x_1 + x_2 \\ \text{sujeito a} \quad & x_1 - x_2 \leq 2 \\ & x_1 + x_2 \geq 1 \\ & x_1 \geq 0 \end{aligned}$$

Ela será representada assim no arquivo de entrada:

```
2
2
1 0
1 1
1 -1 <= 2
1 1 >= 1
```

A primeira linha representa o número de variáveis e a segunda representa o número de restrições (excluindo as restrições de não negatividade). A terceira linha indica se as variáveis são não-negativas ou livres:

- 1 para variáveis não negativas,
- -1 para variáveis não positivas,
- 0 para variáveis livres.

A quarta linha representa os coeficientes na função objetivo e as demais linhas representam as restrições, incluindo coeficientes, sinal (<=, >= ou ==) e o lado direito.

(v) Para simplificar, você pode remover linhas linearmente dependentes da matriz $\mathring{A}$ antes do Simplex começar. Você pode, se quiser, chamar a função `makeMatrixFullRank(A)` que disponibilizei junto com este trabalho.

(vi) A ideia não é apenas resolver o problema, mas fazer um código caprichado: deve imprimir o passo a passo no console, tente faze-lo de forma organizada e legível. Para imprimir a saída no console, utilize indexação das variáveis começando de zero ($x_0, x_1$, etc.).

O seu programa deve receber os seguintes argumentos na linha de comando (você pode usar a biblioteca `argparse` para gerenciar essas opções, se quiser):

```
positional arguments:
  filename      Nome do arquivo lp de entrada

options:
  --decimals    N. de casas decimais para imprimir valores numéricos.
  --digits      N. total de dígitos para imprimir valores numéricos.
  --policy      Valores válidos: 'largest' (default), 'bland', 'smallest'
```

Considere o exemplo abaixo:

```
python3 meusimplex.py lpDeTeste.txt --decimals 3 --digits 7 --policy bland
```

Neste exemplo, todo número deve ser impresso com 7 dígitos no total e 3 casas decimais, como por exemplo `"%*.*f" % (7, 3, numero)` . Também neste exemplo, deve-se usar a regra de Bland. As outras opções são largest (maior valor, Bland em caso de empate) e smallest (menor valor, Bland em caso de empate).

(vii) Na execução, seu código deve:

(a) Ler a entrada

(b) Transforma-la em FPI

(c) Rodar a PL auxiliar para encontrar uma base e verificar se o problema é viável (este passo pode ser pulado caso você identifique uma base óbvia).

(d) Se o problema for viável, rodar o Simplex e ou encontrar a solução ótima, ou verificar que o problema é ilimitado.

(e) Além do passo a passo, escrever o resultado final no console, como nos exemplos abaixo.

(viii) Em caso de PL inviável:

```
Status: inviavel
```

(ix) Em caso de PL ilimitada:

```
Status: ilimitado
```

(x) Caso possua uma única solução ótima:

```
Status: otimo
Objetivo: z
Solucao:
v1 v2 v3 ...
Dual:
w1 w2 w3 ...
```

O valor da função objetivo é `z`, os valores das variáveis da primal são `v1 v2 v3 ...` os valores das variáveis da dual são `w1 w2 w3 ....`

(xi) Caso a PL possua múltiplas soluções ótimas, execute mais uma iteração do Simplex e produza duas soluções ótimas distintas:

```
Status: otimo (multiplos)
Objetivo: z
Solucoes:
v1 v2 v3 ...
u1 u2 u3 ...
Dual:
w1 w2 w3 ...
```

A segunda solução ótima é `u1 u2 u3 ....`

(xii) O trabalho deve ser submetido via Moodle como um único arquivo com extensão `.py`.