

Trabalho Prático 2

Como será a entrega?

Os alunos poderão escolher se desejam fazer o trabalho prático em C/C++, python ou Julia. Existe 1 tarefa no Moodle para a entrega. Os alunos deverão enviar **UM APENAS UM ARQUIVO .py ou .c ou .cpp ou jl** nesta tarefa. A submissão de qualquer outro formato de arquivo ou de mais de um arquivo implicará em **ZERO**. O arquivo também deve ter uma nomenclatura específica. Para cada tarefa, o arquivo submetido pelo aluno de matrícula xxxxxx deve se chamar `tp2_xxxxxx.py` ou `tp2_xxxxxx.c` ou `tp2_xxxxxx.cpp` ou `tp2_xxxxxx.jl`, a depender da escolha do aluno.

É proibido o uso de bibliotecas para manipulação de vetores, matrizes ou de álgebra linear e algoritmos, como a **numpy** em python e a **algorithms.h** em c++

O que faremos?

O objetivo deste trabalho é resolver PLs gerais, a serem fornecidas e cujo formato será especificado abaixo. Em outras palavras, vamos fazer uma aplicação do método simplex.

Resolva a programação linear definida por

$$\begin{aligned} \max \quad & \mathbf{c}^T \mathbf{x} \\ \text{sujeito a} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq 0 \end{aligned}$$

e encontre o certificado que comprove seu resultado.

$$\mathbf{A} = \begin{pmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,m} \\ a_{2,1} & a_{2,2} & \dots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,m} \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \quad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{pmatrix}$$

Entrada

A primeira linha da entrada contém dois inteiros n e m , o número de restrições e variáveis respectivamente.

A segunda linha contém m inteiros, c_i , que formam o vetor de custo.

Cada uma das n linhas seguintes contém $m + 1$ inteiros que representam as restrições. Para a i -ésima linha, os m primeiros números são $a_{i,1}, a_{i,2}, \dots, a_{i,m}$ enquanto o último é b_i . Repare que esses valores, incluindo b_i , podem ser **negativos**.

Uma entrada genérica é da forma:

n	m			
c_1	c_2	\dots	c_m	
$a_{1,1}$	$a_{1,2}$	\dots	$a_{1,m}$	b_1
$a_{2,1}$	$a_{2,2}$	\dots	$a_{2,m}$	b_2
\vdots	\vdots	\ddots	\vdots	\vdots
$a_{n,1}$	$a_{n,2}$	\dots	$a_{n,m}$	b_n

onde

$$1 \leq n \leq 1000$$

$$1 \leq m \leq 1000$$

$$\forall i, 1 \leq i \leq n, \forall j, 1 \leq j \leq m, |a_{i,j}| \leq 1000$$

$$\forall i, 1 \leq i \leq n, |b_i| \leq 1000$$

$$\forall i, 1 \leq i \leq m, |c_i| \leq 1000$$

Saídas

Escreva o resultado da programação linear de acordo com as especificações seguintes:

- Para o caso em que a PL possui valor ótimo, escreva, na primeira linha, **ótima**. Na segunda linha, o valor objetivo atingido.
- Para o caso em que a PL é inviável, escreva, na primeira linha, **inviavel**.
- Para o caso em que a PL é ilimitada, escreva, na primeira linha, **ilimitada**.

Todos os números devem ser escritos com, no máximo, 3 casas decimais.

Sua resposta será considerada correta se o erro absoluto ou relativo entre o valor calculado pelo seu certificado o valor ideal não ultrapassar 10^{-3} . Em termos práticos, isso significa que, assumindo que o valor calculado seja a e o valor ideal seja b , sua resposta será considerada correta se $\frac{|a-b|}{\max(1,b)} \leq 10^{-3}$.

Exemplos

Para auxiliar na execução do TP estão disponíveis no Moodle alguns exemplos de arquivos de entrada e de saída. Os arquivos de entrada estão nomeados da seguinte forma EN-TRADA_exemplo_X e a saída correspondente está nomeada como SAIDA_exemplo_X.

AVISO: as instâncias ótimas de exemplos possuem os valores das variáveis na solução ótima para conferência. Não é preciso fazer isso no programa.

Execução

Os códigos serão executados em uma máquina Ubuntu 20.04. Seu código deve executar nessa configuração, caso ele não execute ou produza uma erro será atribuída a nota 0.

Os códigos escritos em **C** serão compilados com a seguinte linha de código:

```
gcc -m(32/64) -o3 codigo.c -o codigo
```

E executados com a linha

```
./codigo ENTRADA.txt SAIDA.txt
```

Os códigos escritos em **C++** serão compilados com a seguinte linha de código:

```
g++ -m(32/64) -o3 codigo.cpp -o codigo
```

E executados com a linha

```
./codigo ENTRADA.txt SAIDA.txt
```

Os códigos escritos em **Python** serão executados com a linha

```
python3 codigo.py ENTRADA.txt SAIDA.txt
```

Como será a avaliação?

Serão testadas um total de 10 instâncias, cada uma delas valendo 1 ponto. Se o programa executar corretamente o aluno recebe 1, caso contrário o aluno recebe 0. não existirá pontuação intermediária.

Neste trabalho teremos também uma pontuação extra para os códigos mais *rápidos*. Mais precisamente, para cada instância de teste i o código de cada aluno será executado 3 vezes e será computada uma média do tempo de execução. Os três códigos com melhor tempo de execução receberão 0.5 pontos e isto será feito para cada uma das 10 instâncias, possibilitando aos alunos conseguirem até 5 pontos extras.