UTFPR - Especialização em Métodos Matemáticos Aplicados

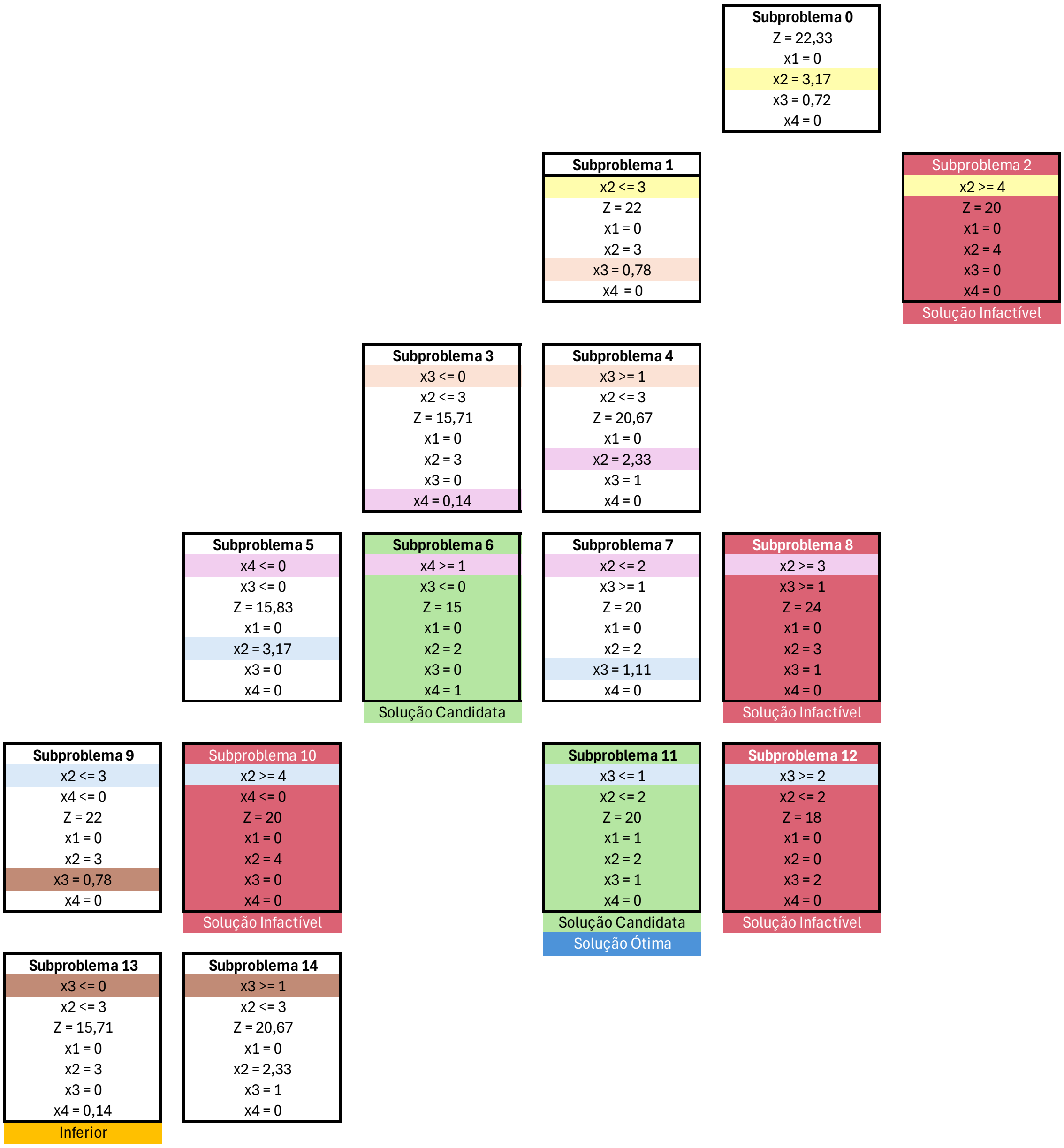
Disciplina: Programação Linear Inteira

Discente: Cintia Izumi Shinoda

Parte 1 - Construção da Árvore Branch-and-Bound (B&B)

Exercício 1

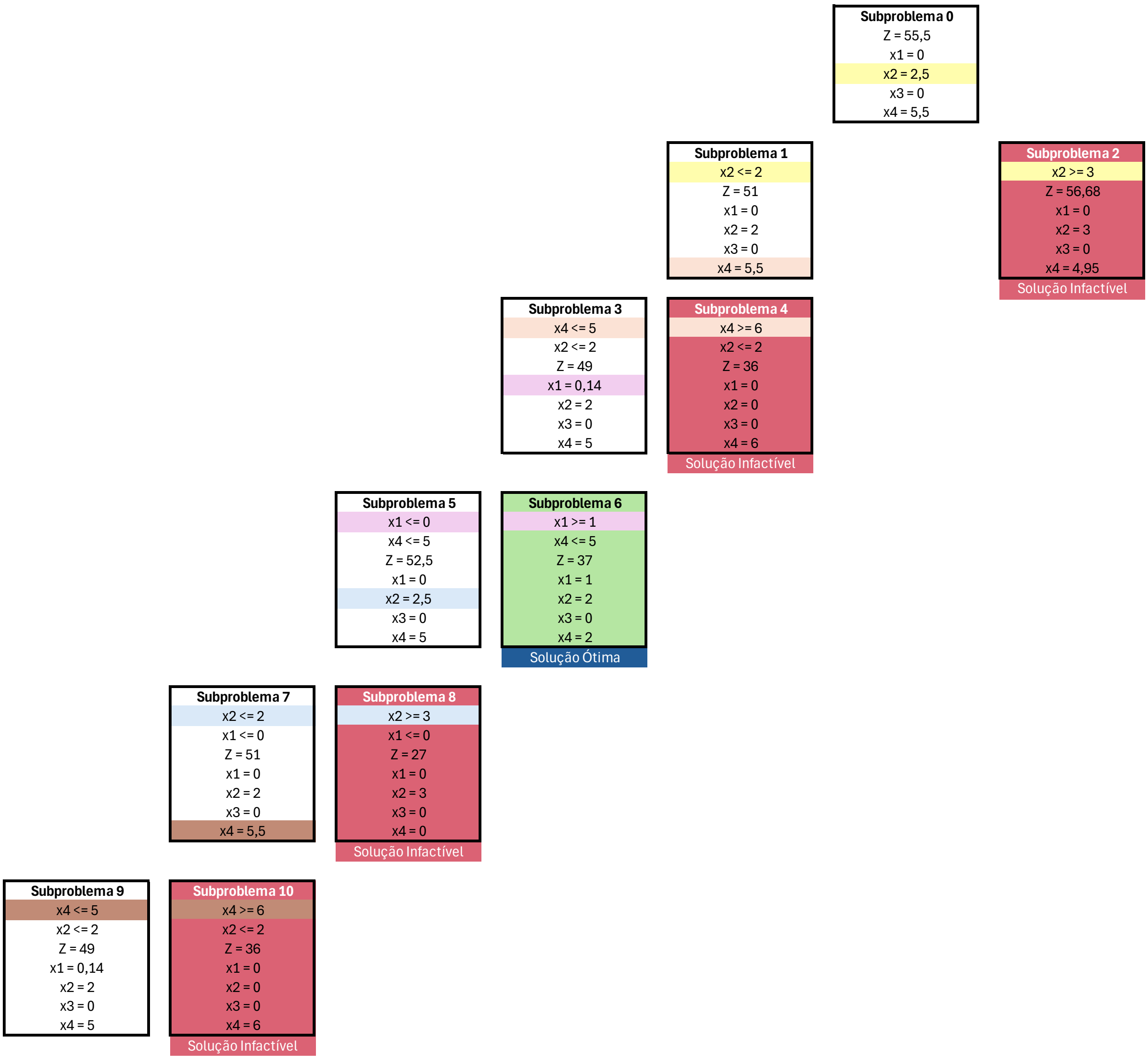
Estratégia de busca: em largura



Nó	Restrições	z	x_1	x_2	x_3	x_4	
S_0	-	22,33	0	3,17	0,72	0	
S_1	$x_2 \leq 3$	22	0	3	0,78	0	
S_2	$x_2 \geq 4$	20	0	4	0	0	Solução Infactível
S_3	$x_3 \leq 0$ $x_2 \leq 3$	15,71	0	3	0	0,14	
S_4	$x_3 \geq 1$ $x_2 \leq 3$	20,67	0	2,33	1	0	
S_5	$x_4 \leq 0$ $x_3 \leq 0$	15,83	0	3,17	0	0	
S_6	$x_4 \geq 1$ $x_3 \leq 0$	15	0	2	0	1	Solução Candidata
S_7	$x_2 \leq 2$ $x_3 \geq 1$	20	0	2	1,11	0	
S_8	$x_2 \geq 3$ $x_3 \geq 1$	24	0	3	1	0	Solução Infactível
S_9	$x_2 \leq 3$ $x_4 \leq 0$	22	0	3	0,78	0	
S_{10}	$x_2 \geq 4$ $x_4 \leq 0$	20	0	4	0	0	Solução Infactível
S_{11}	$x_3 \leq 1$ $x_2 \leq 2$	20	1	2	1	0	Solução Ótima
S_{12}	$x_3 \geq 2$ $x_2 \leq 2$	18	0	0	2	0	Solução Infactível
S_{13}	$x_3 \leq 0$ $x_2 \leq 3$	15,71	0	3	0	0,14	Solução Inferior à Melhor Obtida
S_{14}	$x_3 \geq 1$ $x_2 \leq 3$	20,67	0	2,33	1	0	

Exercício 2

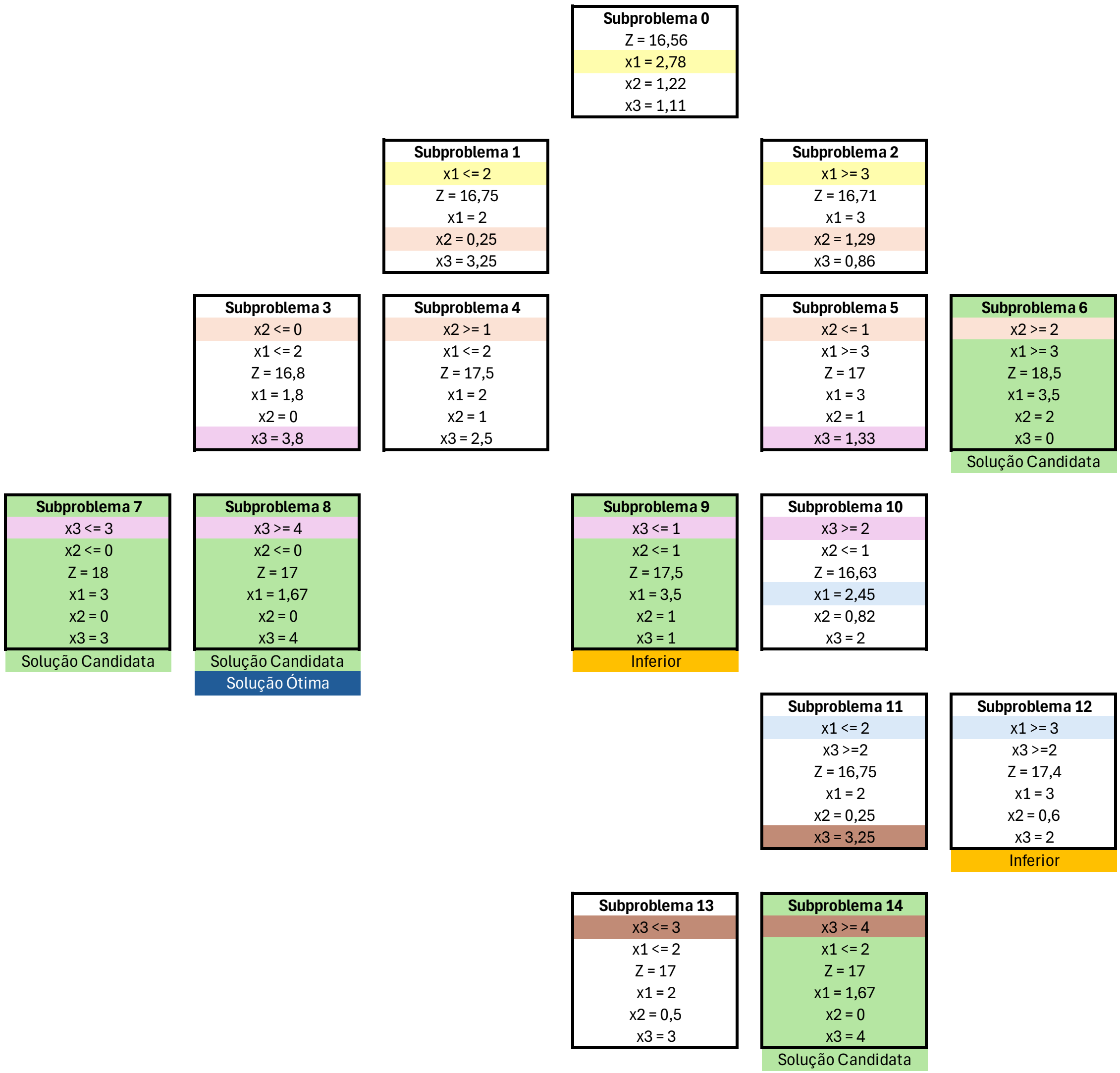
Estratégia de busca: em largura



Nó	Restrições	z	$\mathbf{x_1}$	$\mathbf{x_2}$	$\mathbf{x_3}$	$\mathbf{x_4}$	
S_0	-	55,5	0	2,5	0	5,5	
S_1	$x_2 \leq 2$	51	0	2	0	5,5	
S_2	$x_2 \geq 3$	56,68	0	3	0	4,95	Solução Infactível
S_3	$x_4 \leq 5$ $x_2 \leq 2$	49	0,14	2	0	5	
S_4	$x_4 \geq 6$ $x_2 \leq 2$	36	0	0	0	6	Solução Infactível
S_5	$x_1 \leq 0$ $x_4 \leq 5$	52,5	0	2,5	0	5	
S_6	$x_1 \geq 1$ $x_4 \leq 5$	37	1	2	0	2	Solução Ótima
S_7	$x_2 \leq 2$ $x_1 \leq 0$	51	0	2	0	5,5	
S_8	$x_2 \geq 3$ $x_1 \leq 0$	27	0	3	0	0	Solução Infactível
S_9	$x_4 \leq 5$ $x_2 \leq 2$	49	0,14	2	0	5	
S_{10}	$x_4 \geq 6$ $x_2 \leq 2$	36	0	0	0	6	Solução Infactível

Exercício 3

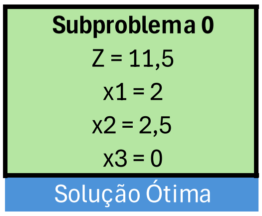
Estratégia de busca: em largura



Nó	Restrições	z	x_1	x_2	x_3	
S_0	-	16,56	2,78	1,22	1,11	
S_1	$x_1 \leq 2$	16,75	2	0,25	3,25	
S_2	$x_1 \geq 3$	16,71	3	1,29	0,86	
S_3	$x_2 \leq 0$ $x_1 \leq 2$	16,8	1,8	0	3,8	
S_4	$x_2 \geq 1$ $x_1 \leq 2$	17,5	2	1	2,5	
S_5	$x_2 \leq 1$ $x_1 \geq 3$	17	3	1	1,33	
S_6	$x_2 \geq 2$ $x_1 \geq 3$	18,5	3,5	2	0	Solução Candidata
S_7	$x_3 \leq 3$ $x_2 \leq 0$	18	3	0	3	Solução Candidata
S_8	$x_3 \geq 4$ $x_2 \leq 0$	17	1,67	0	4	Solução Ótima
S_9	$x_3 \leq 1$ $x_2 \leq 1$	17,5	3,5	1	1	Solução Inferior à Melhor Obtida
S_{10}	$x_3 \geq 2$ $x_2 \leq 1$	16,63	2,45	0,82	2	
S_{11}	$x_1 \leq 2$ $x_3 \geq 2$	16,75	2	0,25	3,25	
S_{12}	$x_1 \geq 3$ $x_3 \geq 2$	17,4	3	0,6	2	Solução Inferior à Melhor Obtida
S_{13}	$x_3 \leq 3$ $x_1 \leq 2$	17	2	0,5	3	
S_{14}	$x_3 \geq 4$ $x_1 \leq 2$	17	1,67	0	4	

Exercício 4

Estratégia de busca: não foi necessário definir



Nó	Restrições	z	x_1	x_2	x_3	
S_0	-	11,5	2	2,5	0	Solução Ótima

Parte 2 - Formulação de Problemas de PLI

Exercício 1

Variáveis de decisão:

x_{ij} : Quantidade de carga do tipo i alocada no compartimento j

i	Tipos de carga	i ∈ {1; 2; 3; 4; 5}
j	Compartimentos	j ∈ {1; 2; 3; 4}

Função objetivo:

$$Maximizar \quad Z = \sum_{i=1}^{5} \sum_{j=1}^{4} L_i . x_{ij}$$

L_i	Lucro

Restrições:

1. Peso por compartimento:

$$\sum_{i=1}^{5} pu_i . x_{ij} \leq PesoMax_j$$

pu	Peso Unitário da Carga
$PesoMax_j$	Peso Máximo no compartimento

2. Volume por compartimento (exceto cargas a granel):

$$\sum_{j=1}^{3} vu_i \cdot x_{ij} \leq EM_j$$

vu	Volume por unidade
EM	Espaço máximo

3. Proporção da distribuição do peso para equilíbrio:

$$5F : 7Ce : 6\,Ca : 7P$$

F	Peso Máximo no compartimento Frontal
Ce	Peso Máximo no compartimento Central
Ca	Peso Máximo no compartimento da Cauda
P	Peso Máximo no Porão

4. Cargas a granel são restritas ao porão (porão: $j = 4$):

$$i \in \{4, 5\}$$

$$x_{i1} = 0; \quad x_{i2} = 0; \quad x_{i3} = 0$$

5. Não negatividade

$$x_{ij} \geq 0$$

Exercício 2

Variáveis de decisão:

Variável	Descrição
x_1	Quantidade de horas de operação do helicóptero AH-1
x_2	Quantidade de horas de operação do avião tanque
x_3	Quantidade de horas de operação do avião B67

Função objetivo:

$$Minimizar \quad Z = 2000x_1 + 4000x_2 + 10000x_3$$

Restrições:

1. Cobertura da área total:

$$15000\mathrm{x}_1 + 40000\mathrm{x}_2 + 85000\mathrm{x}_3 \geq 3000000$$

2. Restrição do número de pilotos:
    a. Pilotos de helicópteros: $2x_1 \leq 10$
    b. Pilotos de avião: $2x_2 + 2x_3 \leq 14$
3. Restrição do número de operadores:

$$x_2 + 3x_3 \leq 22$$

4. Restrição de tempo de operação:

$$x_1 \leq 3; \quad x_2 \leq 3; \quad x_3 \leq 3$$

5. Não negatividade:

$$x_1 \geq 0; \quad x_2 \geq 0; \quad x_3 \geq 0$$

Exercício 3

Mês	**Necessidades (aeromoças-horas-de-voo)**
Janeiro	8000
Fevereiro	9000
Março	7000
Abril	10000
Maio	9000
Junho	11000

Experiência	**Custo**
Experiente	\$850
Em treinamento	\$450

Variáveis de decisão:

x_t: número de aeromoças que começam o treinamento no mês t

y_t: número de aeromoças experientes disponíveis para voar no mês t

z_t: horas de voo excedentes disponíveis no mês t

Função objetivo:

$$Minimizar \quad \sum_{t=1}^{6} 450x_t + 850y_t$$

Restrições:

1. Disponibilidade de horas de voo:

$$150y_t - 100x_{t-1} \geq Necessidade\ de\ horas\ de\ voo\ no\ mês\ t$$

x_{t-1} : aeromoças que iniciaram o treinamento no mês anterior

2. Evolução das aeromoças experientes:

$$y_t = 0.9y_{t-1} + x_{t-1}$$

- $0.9y_{t-1}$: no mês seguinte, apenas 90% das aeromoças experientes permanecem.
- x_{t-1} : aeromoças que terminaram o treinamento no mês anterior $(t-1)$, se tornam experientes no mês atual (t).

3. Horas excedentes:

$$x_t \geq 0$$

4. Conservação da força de trabalho:

$$y_t \geq 0$$

$$x_t \geq 0$$

Adicionando julho ao horizonte de planejamento:

A solução pode mudar, pois a quantidade de horas necessárias em julho, influenciará o número de aeromoças contratadas em meses anteriores para atender à demanda.

Haveria necessidade de adição das variáveis x_7 e y_7, inclusão das restrições de demanda para julho e ajustamento da restrição das aeromoças experientes até $t = 7$.

Exercício 4

Variáveis de decisão:

$$x_i \in \{0,1\}$$

i : número da mudança no projeto

$x_i = 1$: a mudança i será implementada

$x_i = 0$: a mudança i não será implementada

Função objetivo:

$$Minimizar \quad Z = 130000x_1 + 110000x_2 + 120000x_3 + 150000x_4 + 80000x_5 + 80000x_{x_6} + 360000x_7 + 400000x_8 + 160000x_9 + 120000x_{10} + 200000x_{11} + 160000x_{12}$$

Restrições:

1. Redução de peso com as mudanças escolhidas:

$$30x_1 + 20x_2 + 25_x3 + 40x_4 + 15x_5 + 10x_6 + 60x_7 + 80x_8 + 40x_9 + 30x_{10} + 50x_{11} + 35x_{12} \geq 180$$

2. Integridade:

$$x_i \in \{0,1\}$$

$$i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9,10,11,12\}$$

Exercício 5

Variáveis de decisão:

x_{ij}: Quantidade de material transportada do local i para o distribuidor j

y_i: Variável binária que indica se o local i foi selecionado ($y_i = 1$) ou não ($y_i = 0$)

c_{ij}: Custo unitário de transporte do local i para o distribuidor j

f_i: Custo fixo do local i

v_i: Custo variável no local i

d_j: Demanda do distribuidor j

a_i: Capacidade máxima do local i

Função objetivo:

$$Minimizar \quad Z = \sum_i f_i y_i + \sum_i \sum_j (c_{ij}\, x_{ij} + v_i x_{ij})$$

Restrições:

1. Demanda dos distribuidores:

$$\sum_i x_{ij} = d_j$$

2. Capacidade do local:

$$\sum_j x_{ij} \leq a_i y_i$$

3. Integridade:

$$x_{ij} \geq 0$$

$$y_i \in \{0,1\}$$

Parte 3 - Problemas de transporte, transbordo e designação

Exercício 1

```python
from pulp import LpProblem, LpMinimize, LpVariable, lpSum

# dados
custos = [
[10, 7, 5, 6],
[12, 7, 6, 4],
[13, 6, 3, 5]]

ofertas = [220, 180, 230]
demandas = [150, 165, 210, 90]

fabrica = range(len(ofertas))
mercado = range(len(demandas))


# Modelo
modelo = LpProblem("Exercicio-1", LpMinimize)


# Variáveis de Decisão
x = [[LpVariable(f"x_{i}_{j}", lowBound=0) for j in mercado] for i in fabrica]


# Função Objetivo
modelo += lpSum(custos[i][j] * x[i][j] for i in fabrica for j in mercado)


# Restrição Oferta
for i in fabrica:
    modelo += lpSum(x[i][j] for j in mercado) <= ofertas[i], f"Oferta_Fabrica_{i}"


# Restrição Demanda
for j in mercado:
    modelo += lpSum(x[i][j] for i in fabrica) == demandas[j], f"Demanda_Mercado_{j}"


modelo.solve()

# Resultados
print("Status:", modelo.status)   # Status = 1: A solução é ótima
print("Custo Total:", modelo.objective.value())
for i in fabrica:
    for j in mercado:
        print(f"Fábrica {i+1} para Mercado {j+1}: {x[i][j].value()} toneladas")
```

```
Status: 1
Custo Total: 3625.0
Fábrica 1 para Mercado 1: 150.0 toneladas
Fábrica 1 para Mercado 2: 70.0 toneladas
Fábrica 1 para Mercado 3: 0.0 toneladas
Fábrica 1 para Mercado 4: 0.0 toneladas
Fábrica 2 para Mercado 1: 0.0 toneladas
Fábrica 2 para Mercado 2: 75.0 toneladas
Fábrica 2 para Mercado 3: 0.0 toneladas
Fábrica 2 para Mercado 4: 90.0 toneladas
Fábrica 3 para Mercado 1: 0.0 toneladas
Fábrica 3 para Mercado 2: 20.0 toneladas
Fábrica 3 para Mercado 3: 210.0 toneladas
Fábrica 3 para Mercado 4: 0.0 toneladas
```

Exercício 2

Variáveis de Decisão:

x_{ij}: Quantidade de pneus

i	Terminal	$\in \{Curitiba, Londrina, Cascavel, Campo\ Mourão\}$
j	Revendedor	$\in \{A;\ B;\ C\}$

Função Objetivo:

$$Minimizar\ \ Z = \sum_{i,j} c_{i,j}.x_{ij}$$

c_{ij}: custo unitário do pneu do revendedor j para o terminal i

Restrições:

1. Demanda de cada terminal:

Curitiba	$x_{Curitiba,A} + x_{Curitiba,B} + x_{Curitiba,C} = 4000$
Londrina	$x_{Londrina,A} + x_{Londrina,B} + x_{Londrina,C} = 8000$
Cascavel	$x_{Cascavel,A} + x_{Cascavel,B} + x_{Cascavel,C} = 3000$
Campo Mourão	$x_{CampoMourao,A} + x_{CampoMourao,B} + x_{CampoMourao,C} = 5000$

2. Estoque de cada revendedor:

Revendedor A	$x_{Curitiba,A} + x_{Londrina,A} + x_{Cascavel,A} + x_{CampoMourao,A} \leq 12000$
Revendedor B	$x_{Curitiba,B} + x_{Londrina,B} + x_{Cascavel,B} + x_{CampoMourao,B} \leq 6000$

Revendedor C	$x_{Curitiba,C} + x_{Londrina,C} + x_{Cascavel,C} + x_{CampoMourao,C} \leq 4000$

3. Não negatividade

$$x_{ij} \geq 0, \text{ para todo } i, j$$

```python
from pulp import LpProblem, LpMinimize, LpVariable, lpSum

# Dados:
custos = {
('Curitiba', 'A'): 70, ('Curitiba', 'B'): 64, ('Curitiba', 'C'): 68,
('Londrina', 'A'): 74, ('Londrina', 'B'): 62, ('Londrina', 'C'): 65,
('Cascavel', 'A'): 62, ('Cascavel', 'B'): 68, ('Cascavel', 'C'): 64,
('Campo Mourão', 'A'): 62, ('Campo Mourão', 'B'): 72, ('Campo Mourão', 'C'): 66
}

demanda = {
'Curitiba': 4000,
'Londrina': 8000,
'Cascavel': 3000,
'Campo Mourão': 5000
}

estoque = {
'A': 12000,
'B': 6000,
'C': 4000
}

# Modelo:
modelo = LpProblem("Exercicio-2", LpMinimize)

# Variáveis de Decisão:
x = LpVariable.dicts("x", [(i, j) for i in demanda for j in estoque], lowBound=0)

# Função Objetivo:
modelo += lpSum(custos[i, j] * x[i, j] for i in demanda for j in estoque)

# Restrições Demanda:
for i in demanda:
    modelo += lpSum(x[i, j] for j in estoque) == demanda[i], f"Demanda_{i}"

# Restrições Estoque:
for j in estoque:
    modelo += lpSum(x[i, j] for i in demanda) <= estoque[j], f"Estoque_{j}"

modelo.solve()

# Exibe Resultados:
print("Status:", modelo.status)  # Status: 1 = é a solução ótima
print("Custo Total:", modelo.objective.value())
for i in demanda:
    for j in estoque:
        print(f"Quantidade de pneus de {j} para {i}: {x[i, j].value()}")
```

```
Status: 1
Custo Total: 1272000.0
Quantidade de pneus de A para Curitiba: 2000.0
Quantidade de pneus de B para Curitiba: 2000.0
Quantidade de pneus de C para Curitiba: 0.0
Quantidade de pneus de A para Londrina: 0.0
Quantidade de pneus de B para Londrina: 4000.0
Quantidade de pneus de C para Londrina: 4000.0
Quantidade de pneus de A para Cascavel: 3000.0
Quantidade de pneus de B para Cascavel: 0.0
Quantidade de pneus de C para Cascavel: 0.0
Quantidade de pneus de A para Campo Mourão: 5000.0
Quantidade de pneus de B para Campo Mourão: 0.0
Quantidade de pneus de C para Campo Mourão: 0.0
```

Exercício 3 (não consta na lista)

Exercício 4

Variáveis de decisão:

x_{ij} : Quantidade de galões fornecida pelo fornecedor i para o aeroporto j

i	Fornecedores	$i \in \{1, 2, 3\}$
j	Aeroportos	$j \in \{1, 2, 3\}$

Função Objetivo:

$$Minimizar \quad Z = 92x_{11} + 89_{12} + 90_{13} + 91_{21} + 91_{22} + 95_{23} + 87_{31} + 90_{32} + 92_{33}$$

Restrições:

1. Demanda dos aeroportos:

a) Aeroporto 1: $x_{11} + x_{21} + x_{31} = 100000$

b) Aeroporto 2: $x_{12} + x_{22} + x_{32} = 180000$

c) Aeroporto 3: $x_{13} + x_{23} + x_{33} = 30000$

2. Capacidade dos fornecedores:

a) Fornecedor 1: $x_{11} + x_{12} + x_{13} = 320000$

b) Fornecedor 2: $x_{21} + x_{22} + x_{23} = 270000$

c) Fornecedor 3: $x_{31} + x_{32} + x_{33} = 150000$

3. Não negatividade

$$x_{ij} \geq 0$$

```python
from pulp import LpProblem, LpMinimize, LpVariable, lpSum

# Custos:
custos = {
(1, 1): 92, (1, 2): 89, (1, 3): 90,
(2, 1): 91, (2, 2): 91, (2, 3): 95,
(3, 1): 87, (3, 2): 90, (3, 3): 92
}

# Restrição de demanda dos aeroportos:
demanda = {1: 100000, 2: 180000, 3: 300000}

# Restrição de capacidade dos fornecedores:
capacidade = {1: 320000, 2: 270000, 3: 150000}


# Modelo:
modelo = LpProblem("Exercicio-4", LpMinimize)

# Variáveis de decisão:
x = {(i, j): LpVariable(f"x_{i}_{j}", lowBound=0) for i in range(1, 4) for j in range(1, 4)}

# Função Objetivo:
modelo += lpSum(custos[i, j] * x[i, j] for i in range(1, 4) for j in range(1, 4)), "Custo Total"

# Restrições de demanda nos aeroportos:
for j in range(1, 4):
    modelo += lpSum(x[i, j] for i in range(1, 4)) == demanda[j], f"Demanda Aeroporto_{j}"

# Restrições de capacidade dos fornecedores:
for i in range(1, 4):
    modelo += lpSum(x[i, j] for j in range(1, 4)) <= capacidade[i], f"Capacidade Fornecedor_{i}"

modelo.solve()

# Resultados:
print("Status:", modelo.status)
print("Custo Total Mínimo:", modelo.objective.value())
for i in range(1, 4):
    for j in range(1, 4):
        print(f"[{i}][{j}] = {x[i, j].value()} galões")
```

```
Status: 1
Custo Total Mínimo: 51990000.0
[1][1] = 0.0 galões
[1][2] = 20000.0 galões
[1][3] = 300000.0 galões
[2][1] = 0.0 galões
[2][2] = 110000.0 galões
[2][3] = 0.0 galões
[3][1] = 100000.0 galões
[3][2] = 50000.0 galões
[3][3] = 0.0 galões
```

Exercício 5

Variáveis de decisão:

c_{ij} : custo de transporte

x_{ij}: volume transportado

i	Centro
j	Armazém

Função Objetivo:

$$Minimizar \quad Z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

```python
import pulp

# Dados:
custo_por_km = 0.50

custos = [
[10, 22, 29, 45, 11, 31, 42, 61, 36, 21, 45],
[25, 35, 17, 38, 9, 17, 65, 45, 42, 5, 41],
[18, 19, 22, 29, 24, 54, 39, 78, 51, 14, 38]
]

capacidades = [500, 750, 400]
demandas = [112, 85, 138, 146, 77, 89, 101, 215, 53, 49, 153]

# Modelo
modelo = pulp.LpProblem("Exercicio-5", pulp.LpMinimize)

# Variáveis de Decisão:
x = [[pulp.LpVariable(f"x_{i}_{j}", lowBound=0, cat='Continuous') for j in range(len(demandas))] for i in range(len(capacidades))]

# Função Objetivo:
modelo += pulp.lpSum(custos[i][j] * custo_por_km * x[i][j] for i in range(len(capacidades)) for j in range(len(demandas)))

# Restrições de capacidade dos centros:
for i in range(len(capacidades)):
    modelo += pulp.lpSum(x[i][j] for j in range(len(demandas))) <= capacidades[i], f"Capacidade Centro_{i+1}"

# Restrições de demanda dos armazéns:
for j in range(len(demandas)):
    modelo += pulp.lpSum(x[i][j] for i in range(len(capacidades))) == demandas[j], f"Demanda Armazem_{j+1}"

modelo.solve()

# Exibir os resultados
print("Status:", pulp.LpStatus[modelo.status])
print("Custo Total Mínimo:", pulp.value(modelo.objective))

for i in range(len(capacidades)):
    for j in range(len(demandas)):
        print(f"[{i+1}][{j+1}] = {x[i][j].varValue}")
```

```
Status: Optimal
Custo Total Mínimo: 16678.5
[1][1] = 112.0
[1][2] = 0.0
[1][3] = 0.0
[1][4] = 0.0
[1][5] = 0.0
[1][6] = 0.0
[1][7] = 0.0
[1][8] = 0.0
[1][9] = 53.0
[1][10] = 0.0
[1][11] = 0.0
[2][1] = 0.0
[2][2] = 0.0
[2][3] = 138.0
[2][4] = 0.0
[2][5] = 77.0
[2][6] = 89.0
[2][7] = 0.0
[2][8] = 215.0
[2][9] = 0.0
[2][10] = 49.0
[2][11] = 85.0
[3][1] = 0.0
[3][2] = 85.0
[3][3] = 0.0
[3][4] = 146.0
[3][5] = 0.0
[3][6] = 0.0
[3][7] = 101.0
[3][8] = 0.0
[3][9] = 0.0
[3][10] = 0.0
[3][11] = 68.0
```

Exercício 6

Variáveis de decisão:

x_{ij}: variável binária ($x_{ij} = 1$ se o trabalhador for designado para a tarefa ou $x_{ij} = 0$, se não for designado.)

c_{ij}: tempo do trabalhador

i	Trabalhador
j	Tarefa

Função Objetivo:

$$Minimizar \quad Z = \sum_{i=1}^{6}\sum_{j=1}^{6} c_{ij}x_{ij}$$

```python
import pulp

# Definição dos dados do problema
custos = [
    [13, 22, 19, 21, 16, 20],
    [18, 17, 24, 18, 22, 27],
    [20, 22, 23, 24, 17, 31],
    [14, 19, 13, 30, 23, 22],
    [21, 14, 17, 25, 15, 23],
    [17, 23, 18, 20, 16, 24],
]

# Número de trabalhadores e tarefas
n_trabalhadores = len(custos)
n_tarefas = len(custos[0])

# Criar o modelo de programação linear
modelo = pulp.LpProblem("Exercicio-6", pulp.LpMinimize)

# Variáveis de decisão: x[i][j] indica se o trabalhador i faz a tarefa j
x = [[pulp.LpVariable(f"x_{i}_{j}", cat="Binary") for j in range(n_tarefas)] for i in range(n_trabalhadores)]

# Função objetivo: minimizar o tempo total
modelo += pulp.lpSum(custos[i][j] * x[i][j] for i in range(n_trabalhadores) for j in range(n_tarefas))

# Restrição 1: Cada trabalhador realiza exatamente uma tarefa
for i in range(n_trabalhadores):
    modelo += pulp.lpSum(x[i][j] for j in range(n_tarefas)) == 1

# Restrição 2: Cada tarefa é realizada por exatamente um trabalhador
for j in range(n_tarefas):
    modelo += pulp.lpSum(x[i][j] for i in range(n_trabalhadores)) == 1

# Resolver o problema
modelo.solve()

# Exibir os resultados
print(f"Status da solução: {pulp.LpStatus[modelo.status]}")
print(f"Tempo total minimizado: {pulp.value(modelo.objective)}")

# Exibir a alocação
for i in range(n_trabalhadores):
    for j in range(n_tarefas):
        if pulp.value(x[i][j]) == 1:
            print(f"Trabalhador {i+1} -> Tarefa {j+1}")
```

```
Status da solução: Optimal
Tempo total minimizado: 99.0
Trabalhador 1 -> Tarefa 1
Trabalhador 2 -> Tarefa 4
Trabalhador 3 -> Tarefa 5
Trabalhador 4 -> Tarefa 3
Trabalhador 5 -> Tarefa 2
Trabalhador 6 -> Tarefa 6
```

Exercício 7

Variáveis de decisão:

x_i: variável binária

i	projeto

$x_i = 1$: caso o projeto i seja selecionado; $x_i = 0$, caso contrário

Função Objetivo:

$$Maximizar \quad Z = 25000x_1 + 40000x_2 + 100000x_3 + 80000x_4 + 60000x_5 + 130000x_6 + 160000x_7 + 100000x_8 + 130000x_9 + 150000x_{10}$$

Restrições:

1. Orçamento

$$20000x_1 + 35000x_2 + 70000x_3 + 90000x_4 + 60000x_5 + 150000x_6 + 170000x_7 + 80000x_8 + 90000x_9 + 100000x_{10}$$

2. Restrições entre projetos

a) $x_2 \leq x_1$

b) $x_3 + x_4 \leq 1$

c) $x_5 \leq x_4$

d) $x_6 + x_7 \leq 1$

e) $x_8 + x_9 + x_{10} \leq 1$

3. Quantidade de gerentes

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} \leq 5$$

4. Integridade

$$x_i \in \{0,1\}$$

$$i \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$$

```python
from pulp import LpMaximize, LpProblem, LpVariable, lpSum

# Definição do problema
prob = LpProblem("Exercicio-7", LpMaximize)

# Definição das variáveis de decisão
x = {i: LpVariable(f"x{i}", cat="Binary") for i in range(1, 11)}

# Coeficientes de Valor Presente (VP) e Investimento Inicial (INV)
vp = [25, 40, 100, 80, 60, 130, 160, 100, 130, 150]
inv = [20, 35, 70, 90, 60, 150, 170, 80, 90, 100]

# Função objetivo: Maximizar o VP
prob += lpSum(vp[i - 1] * x[i] for i in range(1, 11)), "Valor_Presente_Total"

# Restrição de orçamento
prob += lpSum(inv[i - 1] * x[i] for i in range(1, 11)) <= 400, "Orcamento_Disponivel"

# Restrições de relacionamento entre projetos
prob += x[2] <= x[1], "Projeto_2_Complementar_1"
prob += x[3] + x[4] <= 1, "Projetos_3_e_4_Mutualmente_Exclusivos"
prob += x[5] <= x[4], "Projeto_5_Complementar_4"
prob += x[6] + x[7] <= 1, "Projetos_6_e_7_Mutualmente_Exclusivos"
prob += x[8] + x[9] + x[10] <= 1, "Projetos_8_9_10_Mutualmente_Exclusivos"

# Dependências entre projetos 6, 7, 8, 9, 10 e 3, 4
prob += x[8] + x[9] + x[10] <= x[6] + x[7], "Dependencia_8_9_10_com_6_7"
prob += x[8] + x[9] + x[10] <= x[3] + x[4], "Dependencia_8_9_10_com_3_4"

# Restrição do número de gerentes disponíveis
prob += lpSum(x[i] for i in range(1, 11)) <= 5, "Limite_de_Gerentes"

# Resolver o problema
prob.solve()

# Exibir os resultados
print("Status:", prob.status)
print("\nPlano de Investimentos:")
for i in range(1, 11):
    print(f"Projeto {i}: {'Selecionado' if x[i].varValue == 1 else 'Não selecionado'}")

print("\nValor Presente Total (VP):", prob.objective.value())
print("Investimento Total:", sum(inv[i - 1] * x[i].varValue for i in range(1, 11)))
```

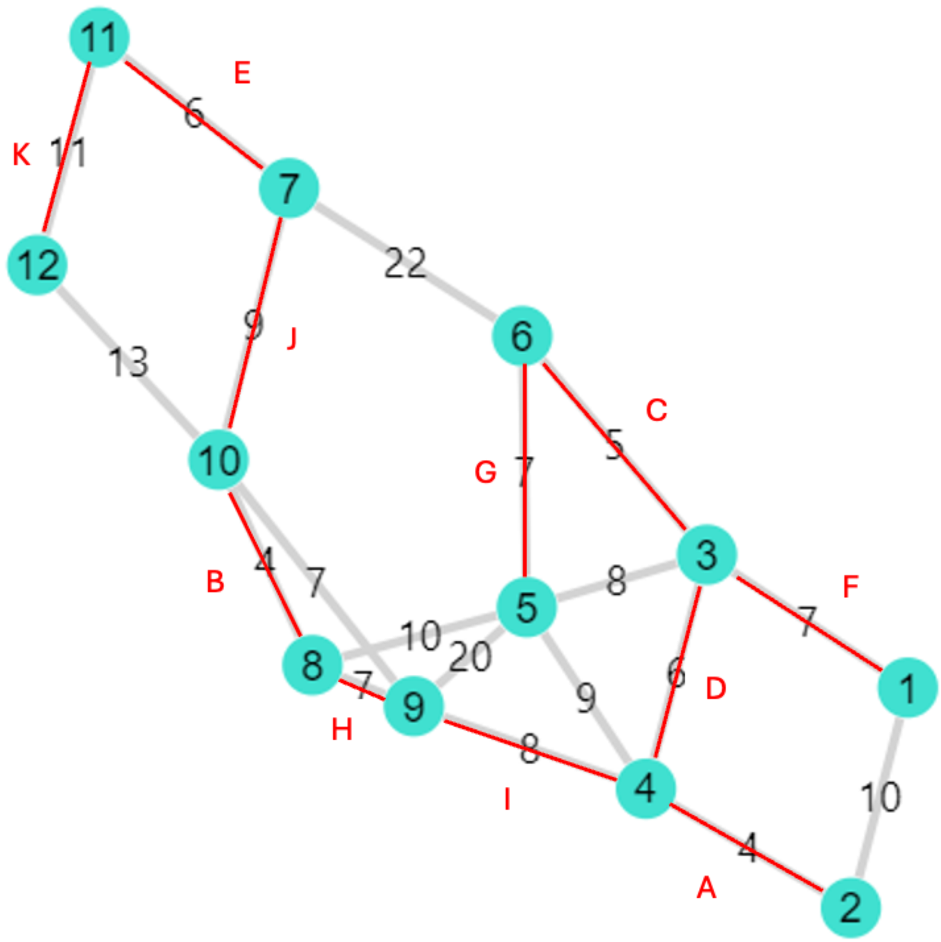
```
Status: 1

Plano de Investimentos:
Projeto 1: Selecionado
Projeto 2: Selecionado
Projeto 3: Selecionado
Projeto 4: Não selecionado
Projeto 5: Não selecionado
Projeto 6: Não selecionado
Projeto 7: Selecionado
Projeto 8: Não selecionado
Projeto 9: Não selecionado
Projeto 10: Selecionado

Valor Presente Total (VP): 475.0
Investimento Total: 395.0
```

Parte 4 - Problemas de Fluxo máximo e mínima arborescência

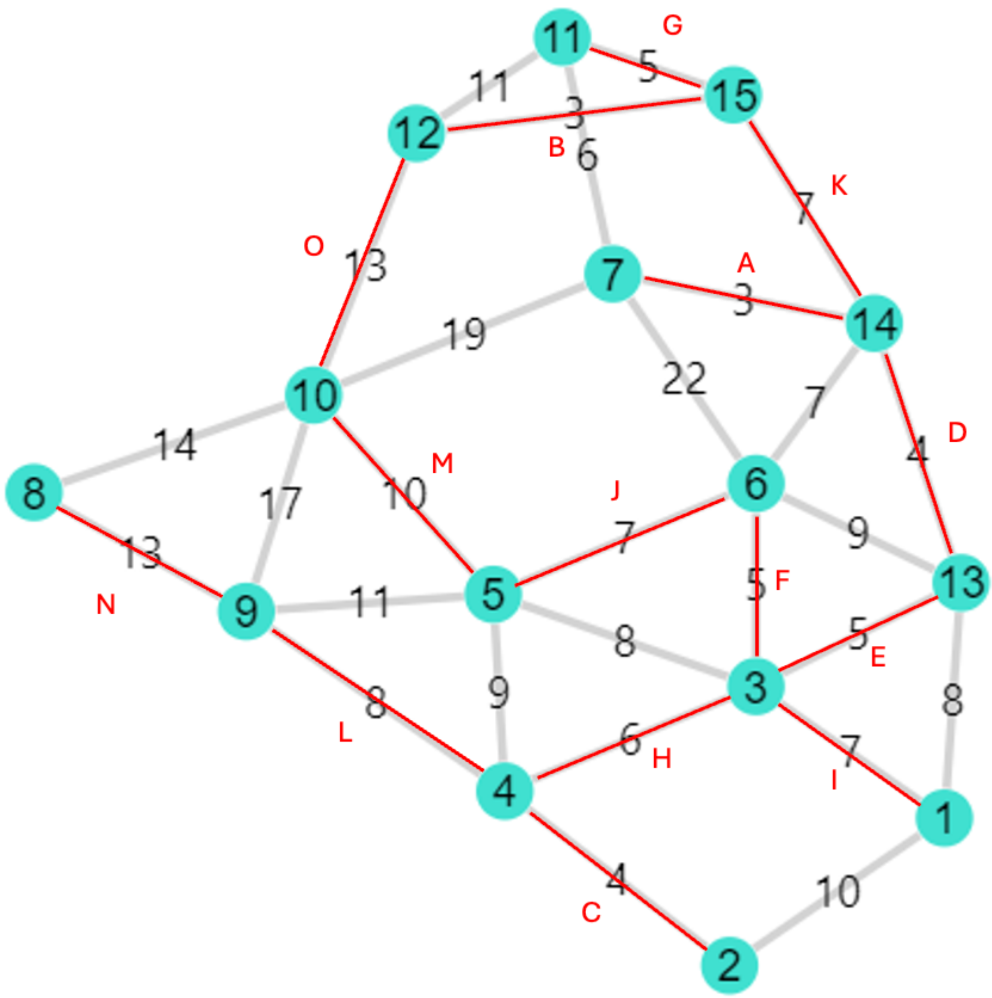
Exercício 1



Sequência de Decisões	Origem	Destino	Peso
A	2	4	4
B	8	10	4
C	3	6	5
D	4	3	6
E	7	11	6
F	1	3	7
G	5	6	7
H	9	8	7
descartada	9	10	7
I	4	9	8

descartada	3	5	8
descartada	4	5	9
J	10	7	9
descartada	1	2	10
descartada	5	8	10
K	12	11	11
descartada	10	12	13
descartada	5	9	20
descartada	6	7	22

Exercício 2



Sequência de Decisões	Origem	Destino	Peso
A	14	7	3
B	12	15	3
C	2	4	4
D	13	14	4
E	3	13	5
F	6	3	5
G	15	11	5
H	4	3	6
descartada	7	11	6
I	1	3	7
descartada	14	6	7
J	6	5	7
K	14	15	7
descartada	1	13	8
descartada	3	5	8
L	4	9	8
descartada	4	5	9
descartada	13	6	9
descartada	2	1	10
M	5	10	10
descartada	5	9	11
descartada	12	11	11
N	9	8	13
O	10	12	13
descartada	8	10	14
descartada	9	10	17
descartada	10	7	19
descartada	6	7	22

Exercício 3

	Fonte	Destino	Capacidade
	1	2	10
	1	3	7
	2	4	4
	3	6	5
	3	5	8
	3	4	6
	4	5	9
	4	9	8
	5	6	7
	5	8	10
	5	9	20
	6	7	22
	7	11	6
	7	10	9
	8	9	7
	8	10	4
	9	10	7
	10	12	13
	11	12	11

Exercício 4

Exercício 5