[MATHEUS PEIXOTO RIBEIRO VIEIRA](mailto:matheus.peixoto@aluno.ufop.edu.br) - 22.1.4104

**1) Para o problema de seleção de atividades, prove que a atividade que começa por último faz parte de alguma solução ótima do problema.**

Seja S o conjunto de atividades propostas.

Seja Sij o conjunto de atividades que começam após i, e terminam antes de j.

Acrescentamos duas atividades fictícias, a0 e an+1, onde f0=0 e sn+1 = ∞.

Logo, S = S0 n+1

Se uma solução para Sij inclui ak, então ak gera dois subproblemas Sik e Skj ⊂ Sij.

Uma solução ótima para Sij seria: Aij = Aik U {ak} U Akj

Considere qualquer subproblema não vazio Sij e seja am a atividade que começa por último. Então:

am está em algum conjunto de solução ótima

O problema Smj é vazio, então Sim é não vazio e ele deve ser resolvido.

Suponha que exista alguma atividade ak ∈ Smj. Então Sj ≥ Fk > Sk ≥ Fm > Sm → Sk > Sm. O que é uma contradição, já que Sm é o início mais recente.

Portanto a solução que contenha am é uma solução ótima. Logo, a atividade que começa por último faz parte de uma solução ótima do problema.

class Atividade:

counter = 1

def \_\_init\_\_(self, s, f) -> None:

self.i = self.counter

Atividade.counter = Atividade.counter + 1

self.s = s

self.f = f

def \_\_str\_\_(self):

return f"{self.i} - Si: {self.s} - Fi: {self.f}"

atividades = [

Atividade(1, 4),

Atividade(3, 5),

Atividade(0, 6),

Atividade(5, 7),

Atividade(3, 8),

Atividade(5, 9),

Atividade(6, 10),

Atividade(8, 11),

Atividade(8, 12),

Atividade(2, 13),

Atividade(12, 14)

]

def algoritmo\_guloso(atividades: Atividade):

# Ordenando de forma decrescente do tempo de inicialização

# Ordenação de forma estável

atividades.sort(key=lambda x: x.s, reverse=True)

# Obter a quantidade de atividades

n = len(atividades)

# Colocar a atividade com maior tempo inicial no conjunto solução

A = [atividades[0]]

i = 0

# Loop do algoritmo guloso

for m in range(1, n):

# Verificar se a próxima atividade acaba antes da atual

# Caso verdadeiro, adicioná-la no conjunto solução

if atividades[m].f <= atividades[i].s:

A.append(atividades[m])

i = m

return A

solucao\_otima = algoritmo\_guloso(atividades)

for a in solucao\_otima:

print(a)

"""

11 - Si: 12 - Fi: 14

8 - Si: 8 - Fi: 11

4 - Si: 5 - Fi: 7

2 - Si: 3 - Fi: 5

"""

