MATHEUS PEIXOTO RIBEIRO VIEIRA - 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 S_{ij} o conjunto de atividades que começam após i, e terminam antes de j.

Acrescentamos duas atividades fictícias, a_0 e a_{n+1}, onde $f_0=0$ e $s_{n+1} = \infty$.

Logo, $S = S_{0\ n+1}$

Se uma solução para S_{ij} inclui a_k, então a_k gera dois subproblemas S_{ik} e $S_{kj} \subset S_{ij}$.

Uma solução ótima para S_{ij} seria: $A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$

Considere qualquer subproblema não vazio S_{ij} e seja a_m a atividade que começa por último. Então:

a_m está em algum conjunto de solução ótima

O problema S_{mj} é vazio, então S_{im} é não vazio e ele deve ser resolvido.

Suponha que exista alguma atividade $a_k \in S_{mj}$. Então $S_j \geq F_k > S_k \geq F_m > S_m \rightarrow S_k > S_m$. O que é uma contradição, já que S_m é o início mais recente.

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

Python

```python
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
"""
```