Nome: Tiago Faustino de Siqueira **Matrícula**: 22102193

Exercício 1: Componentes Fortemente Conexas (CFC)

Implementei a função `CFC` para identificar componentes fortemente conexas em grafos dirigidos. Utilizei um dicionário para armazenar os tempos de descoberta e finalização dos vértices (`T` e `F`), e um conjunto (`C`) para registrar os vértices visitados. Optei por dicionários devido à eficiência no acesso e manipulação de dados, essencial para operações de DFS. A inversão das arestas do grafo também foi implementada para encontrar as componentes de forma eficaz, utilizando a ordem decrescente dos tempos de finalização (`F_decrescente`).

Exercício 2: Ordenação Topológica

Utilizei um conjunto (`C`) para os vértices visitados, dicionários para armazenar os tempos de descoberta e finalização (`T` e `F`), e uma lista (`O`) para a ordem topológica. A lista é ideal para manter a sequência de visitação, garantindo a ordem correta.

Exercício 3: Algoritmo de Prim

Para encontrar a árvore geradora mínima, escolhi implementar o algoritmo de Prim na função `prim`. Usei um heap para gerenciar os vértices a serem processados, armazenando o vértice com menor peso pendente de visitação. Utilizei dicionários para armazenar a árvore geradora mínima (`A`) e os pesos mínimos (`K`) de cada vértice. A estrutura de heap (`heapq`) foi escolhida pela eficiência na extração do menor elemento, essencial para a operação de arg min. O dicionário `K` permite acesso rápido ao peso atual de cada vértice, enquanto `A` armazena o predecessor de cada vértice na árvore geradora mínima.