# Universidade Federal de Ouro Preto
# PCC104 - Projeto e Análise de Algoritmos
# Prova - Algoritmos Gulosos

Prof. Rodrigo Silva

June 19, 2023

## Orientações

- Esta prova tem 4 questões.
- É obrigatória a entrega do código fonte dos algoritmos gulosos. Provas sem os códigos fonte não serão corrigidas e terão nota 0.
- A avaliação do código apresentado entre na avaliação das questões relacionadas.

## Questões

1. Quando devemos utilizar algoritmos gulosos?
2. Considere o pseudo-código do algoritmo de Prim abaixo, a sua própria implementação e responda:

**ALGORITHM** *Prim(G)*
    //Prim's algorithm for constructing a minimum spanning tree
    //Input: A weighted connected graph $G = \langle V, E \rangle$
    //Output: $E_T$, the set of edges composing a minimum spanning tree of $G$
    $V_T \leftarrow \{v_0\}$    //the set of tree vertices can be initialized with any vertex
    $E_T \leftarrow \varnothing$
    **for** $i \leftarrow 1$ **to** $|V| - 1$ **do**
        find a minimum-weight edge $e^* = (v^*, u^*)$ among all the edges $(v, u)$
        such that $v$ is in $V_T$ and $u$ is in $V - V_T$
        $V_T \leftarrow V_T \cup \{u^*\}$
        $E_T \leftarrow E_T \cup \{e^*\}$
    **return** $E_T$

Figure 1: Algoritmo de Prim

   (a) Como você implementou a operação que encontra a aresta de menor peso? Indique no seu código e apresente a análise de complexidade completa desta operação.

   (b) Apresente a análise de complexidade da sua implementação do algoritmo de Prim.

   (c) Discuta a possibilidade de melhorar o seu código. Caso não seja possível melhora, explique o por quê.

3. Considere o pseudo-código do algoritmo de Kruskall abaixo, a sua própria implementação e responda:

**ALGORITHM** *Kruskal(G)*

//Kruskal's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph $G = \langle V, E \rangle$
//Output: $E_T$, the set of edges composing a minimum spanning tree of $G$
sort $E$ in nondecreasing order of the edge weights $w(e_{i_1}) \leq \cdots \leq w(e_{i_{|E|}})$
$E_T \leftarrow \varnothing$; $\quad ecounter \leftarrow 0$ $\qquad$ //initialize the set of tree edges and its size
$k \leftarrow 0$ $\qquad$ //initialize the number of processed edges
**while** $ecounter < |V| - 1$ **do**
$\quad k \leftarrow k + 1$
$\quad$ **if** $E_T \cup \{e_{i_k}\}$ is acyclic
$\qquad E_T \leftarrow E_T \cup \{e_{i_k}\}$; $\quad ecounter \leftarrow ecounter + 1$
**return** $E_T$

Figure 2: Algoritmo de Kruskall

(a) Como você implementou a operação que verifica se uma aresta faz um ciclo? Indique no seu código e apresente a análise de complexidade completa desta operação.

(b) Apresente a análise de complexidade da sua implementação do algoritmo de Kruskall.

(c) Discuta a possibilidade de melhorar o seu código. Caso não seja possível melhora, explique o por quê.

4. Considere o problema da árvore geradora mínima definido como um problema de decisão. Ou seja, ´´Existe uma árvore geradora com custo menor ou igual a X?”. Responda:

(a) A qual classe de problemas este problema pertence, P, NP ou NP-completo? Justifique.

(b) O que precisaríamos fazer para provar que este problema pertence à classe NP?

(c) O que precisaríamos fazer para provar que este problema pertence à classe NP-completo?