# Resolução de Problemas com Grafos

#### Prof. Vinícius M. A. Souza

Atividade 4

**PARTE 1**

Considerando as duas estruturas de dados para a representação de grafos implementadas nos mini-projetos (matriz de adjacências e lista de adjacências), discuta e compare os custos de cada uma das estruturas para as seguintes tarefas:

1. Armazenamento da estrutura
2. Retornar os vizinhos de um vértice
3. Verificar se dois vértices estão conectados
4. Adicionar um vértice
5. Adicionar uma aresta
6. Remover um vértice
7. Remover uma aresta

**PARTE 2**

Retome a sua implementação de grafo utilizando **matriz de adjacências** e adapte o código para que o usuário possa escolher se o grafo será direcionado ou não direcionado, além de ponderado ou não ponderado, no momento da instanciação.

Por exemplo:

G = Grafo(5, direcionado=True, ponderado=True) # irá gerar um grafo com 5 vértices, considerando arestas **direcionadas** e **ponderadas**

G = Grafo(5, direcionado=True, ponderado=False) # irá gerar um grafo com 5 vértices, considerando arestas **direcionadas** e **não ponderadas**

G = Grafo(5, direcionado=False, ponderado=True) # irá gerar um grafo com 5 vértices, considerando arestas **não direcionadas** e **ponderadas**

G = Grafo(5, direcionado=False, ponderado=False) # irá gerar um grafo com 5 vértices, considerando arestas **não direcionadas** e **não ponderadas**

Realize as modificações necessárias para que os seguintes métodos\* funcionem corretamente:

* 1. def adiciona\_aresta(u, v, peso) # cria uma aresta com peso entre os vértices u e v do grafo G. No caso de um grafo não ponderado, deve ser atribuído um peso = 1 para as arestas.
  2. def remove\_aresta(u, v) # remove a aresta entre os vértices u e v do grafo G.
  3. def tem\_aresta(u, v) # verifica se existe uma aresta entre os vértices u e v do grafo G e retorna True ou False.
  4. def grau(u) # retorna a quantidade de arestas conectadas ao vértice u do grafo G.
  5. def grau\_entrada(u) # retorna a quantidade de arestas que chegam até o vértice u do grafo direcionado G.
  6. def grau\_saida(u) # retorna a quantidade de arestas que saem do vértice u do grafo direcionado G.
  7. def retorna\_adjacentes(u) # retorna uma lista com os vértices adjacentes ao vértice u. Lembre que no caso de grafos não direcionados, todos os vértices conectados ao vértice u são considerados adjacentes. No caso de grafos direcionados, os adjacentes de u são somente os vértices em que é possível sair de u e chegar até outro vértice a partir de uma aresta.

\* Caso algum destes métodos ainda não esteja implementado, implemente-o.