Neste trabalho, você deve implementar uma classe que represente Grafos como uma classe em C++ e um dos algoritmos vistos em sala de aula, de forma completa e bem estruturada. A classe deve permitir operações de criação, inserção e remoção de vértices e arestas execução de algum dos algoritmos vistos em aula, ou seja, o código desenvolvido deve fornecer, no mínimo, as seguintes funcionalidades:

a) Imprimir o grafo completo, mostrando todos os vértices e suas arestas (com pesos, se existirem);
b) Buscar um vértice ou aresta, informando se existe e exibindo seus dados;
c) Inserir um novo vértice e/ou aresta no grafo;
d) Remover um vértice e/ou aresta do grafo;
e) Executar um algoritmo de grafos estudado em aula, à sua escolha.

Você pode escolher a representação mais adequada ao seu contexto (por exemplo, direcionado, não direcionado, ponderado ou não ponderado) e também a estrutura de representação (lista de adjacência, matriz de adjacência, ou outra abordagem justificada). É permitido (e recomendado) criar classes auxiliares, como vértice, aresta, ou estruturas de suporte para listas, filas e pilhas.

O grafo deve representar uma aplicação concreta, escolhida por você — por exemplo: redes sociais, conexões de transporte, redes de computadores, circuitos, proteínas, cidades e estradas, entre outros.

Informações Importantes:

a) O trabalho pode ser desenvolvido em grupo de até três alunos (sem exceções); um único trabalho deve ser entregue pelo grupo.
b) Os nomes dos integrantes do grupo com seus respectivos números USP devem estar presentes em todos os arquivos-fonte, na forma de comentário.
c) Todos os arquivos desenvolvidos devem ser entregues em um **único** arquivo de formato ZIP no Tidia-AE
d) Data de entrega: verifique no Tidia-Ae a data limite de entrega.

Na nota do trabalho também serão considerados os seguintes critérios:

1. Qualidade técnica: O código faz sentido e está bem estruturado? Os métodos estão corretamente implementados? Há clareza e coerência na representação do grafo e no algoritmo escolhido?
2. Eficiência e otimização: A implementação utiliza estruturas adequadas e evita redundâncias? Há preocupação com o custo computacional de operações sobre o grafo?
3. Interface de uso e documentação: O programa é fácil de usar? As mensagens são claras? O código está comentado adequadamente e possui explicações suficientes para facilitar a compreensão?
4. Novidade e criatividade: Foram incorporadas ideias novas, representações alternativas ou comparações entre algoritmos ou tipos de grafo? Houve análise de desempenho, visualização gráfica ou recursos adicionais?