Universidade Federal do Maranhão
Departamento de Informática
Disciplina: ***Estrutura de Dados II***
Prof.: ***João Dallyson Sousa de Almeida***

Atividade Prática (20%) - Individual

Unidade III

Nesta atividade você desenvolverá implementar um sistema de recomendação de conexões de amizades para uma rede social fictícia chamada "SocialNet". O sistema permitirá gerenciar usuários (vértices) e suas conexões (arestas), com funcionalidades de recomendação baseadas em análise de grafos.

1) Implemente duas representações: **MatrizAdjacencia** (usará matriz de adjacência) e **ListaAdjacencia** (usará lista de adjancência)
2) Implemente a classe **RedeSocial** que será responsável pela a gestão básica, incluindo o **cadastro de usuários, adicionar/remover conexões (amizades) e visualizar rede de amizades**
3) Implemente os algoritmos de busca e análise presentes na **Classe AnalisadorRedeSocial.**
4) Implemente um menu interativo tomando como base a classe **SocialGraphMenu**
5) Crie um dataset com no mínimo 30 usuários e conexões variadas.

Segue estrutura básica da implementação a ser seguida:

```
public interface Grafo<T> {
    void adicionarVertice(T vertice);
    void adicionarAresta(T origem, T destino);
    void adicionarAresta(T origem, T destino, double peso);
    void removerVertice(T vertice);
    void removerAresta(T origem, T destino);
    List<T> getVertices();
    List<T> getAdjacentes(T vertice);
    int getGrau(T vertice);
    boolean existeAresta(T origem, T destino);
    double getPeso(T origem, T destino);
    int getNumeroVertices();
    int getNumeroArestas();
}
```

Implemente a Classe usuário que inclui os seguintes dados: id, nome, email, dataNascimento e lista de interesses.

```
public class Usuario {
   private String id;
   private String nome;
   private String email;
   private LocalDate dataNascimento;
   private byte[] fotoPerfil;
   private List<String> interesses;

   // Construtor, getters, setters, equals, hashCode, toString
}
```

Implemente a Classe Conexao

```
public class Conexao {
   private Usuario origem;
   private Usuario destino;
   private double forcaConexao; // peso da aresta (0.0 a 1.0)
   private LocalDate dataConexao;

   // Construtor, getters, setters
}
```

```
public class SocialGraphMenu{
   public void executar() {
      while (true) {
         exibirMenu();
         int opcao = lerOpcao();

         switch (opcao) {
            case 1: // Cadastrar usuário
            case 2: // Adicionar amizade
            case 3: // Recomendar amizades
            case 4: // Visualizar rede
            case 5: // Encontrar caminho entre usuários
            case 6: // Identificar comunidades
            case 7: // Mostrar influenciadores
            case 0: // Sair
         }
      }
   }
}
```

```
public class AnalisadorRedeSocial {
   // Busca em Largura (BFS) - Amigos em comum
   public List<Usuario> encontrarCaminhoConexao(Usuario origem, Usuario destino);

   // Busca em Profundidade (DFS) - Exploração completa
   public Set<Usuario> explorarRedeCompleta(Usuario origem);

   // Recomendação por amigos em comum (2º grau)
   public List<Usuario> recomendarAmizades(Usuario usuario);

   // Grafo de Cointeresse (grafos bipartidos)
   public Map<String, List<Usuario>> agruparPorInteresse();

   // Componentes conexas da rede
   public List<Set<Usuario>> identificarComunidades();

   // Usuários mais influentes (maior grau)
   public List<Usuario> encontrarInfluenciadores(int topN);

   // Dijkstra - Força da conexão
   public Map<Usuario, Double> calcularDistanciasSocial(Usuario origem);

   // Verificar se há ciclos na rede
   public boolean possuiCiclos();

  // Detecção de pontes (arestas críticas)
   public List<Conexao> encontrarPontes();

  Árvore geradora mínima (Prim/Kruskal)
   public Grafo<Usuario> encontrarRedeEssencial();
}
```

Entrega:

- Código fonte do programas em JAVA (bem identado e utilizando os recurso da Orientação a Objetos).
- Grave um vídeo de 5 a 10 minutos explicando e demonstrando a execução.
- Relatório dos testes realizados.
- Upload do relatório dos testes no SIGAA, incluindo o link para o vídeo.

Referências:

SEDGEWICK, Robert; WAYNE, Kevin. Algorithms (4th edn). **Google Scholar Google Scholar Digital Library Digital Library**, 2011.

SKIENA, Steven S. The Algorithm Design Manual. 2008.

LOUDON, Kyle. **Mastering algorithms with C**. " O'Reilly Media, Inc.", 1999.

CORMEN, Thomas; LEISERSON, Charles; RIVEST, Ronald. **Algoritmos**. Elsevier Brasil, 2017.