



Instituto Superior de Engenharia do Porto

# Licenciatura em Engenharia Informática Relatório de Estruturas de Informação Game Project

Turma: 2DG Grupo 1

Raúl Piloto da Silva Nunes Correia – 1090657

Rui Miguel Teixeira Ribeiro – 1150344

Turma: 2DG Grupo 1

Raúl Piloto da Silva Nunes Correia - 1090657

Rui Miguel Teixeira Ribeiro - 1150344

### Introdução



No âmbito da unidade curricular de Estruturas de Informação foi-nos proposto o desenvolvimento de uma aplicação que permite gerir o funcionamento de um jogo de computador de estratégia. A aplicação desenvolvida baseia-se num jogo de conquistas através de um sistema de locais, estradas, forças e alianças entre jogadores.

Posto isto, um Local é caracterizado pelo nome, uma personagem (dono), se tiver, e a sua dificuldade.

As estradas são ligações entre locais, caracterizadas também por uma dificuldade.

Uma personagem é caracterizada pelo seu nome e pela sua força.

Uma aliança é uma parceria entre personagens, caracterizada pela sua força que é o resultado da soma das forças dos 2 aliados multiplicado pelo fator de compatibilidade.

Uma conquista acontece quando uma personagem tem força necessária para percorrer a estrada, aceder ao local e, caso exista, derrotar o dono atual do local a conquistar. Este processo repete-se para todos os locais, estradas e personagens intermédios entre o Local A e Local B.

Para a primeira parte do trabalho, foi criada a classe ControloDoJogo, que funciona como um Controller, onde são guardadas a rede de locais e estradas e a rede de personagens e alianças e onde se encontram as respostas ás perguntas presentes no enunciado.

Em termos de algoritmos, utilizou-se as classes realizadas nas aulas PL e a complementar, a classe AlgoritmosJogo que contém uma adaptação do algoritmo de dijkstra para considerar a dificuldade de estradas, locais e personagens donos dos locais.

As restantes classes são objetos adicionais que complementam a execução do jogo, tanto na criação dos elementos do jogo como na execução do mesmo.

Foi criada uma classe Ficheiro, que contêm os métodos para ler ficheiros genéricos de texto e métodos de leitura para Personagens, Alianças, Locais e Estradas



# Conteúdo

Diagrama de Classes	4
Complexidade	q
Resumo	
Completo	10



## Diagrama de Classes

Visual Paradigm Standard (Raúl Correia (Instituto Superior de Engenharia do Porto)) ControlaDoJagoTisst CantralaDaJago Hest Adido ner Local \_3ergs() : void Hest Adido ner Local \_Local() : void LOCAS S: Sring ="Tools Stat" LOCAS M: String ="Tools M tof testAdidonarEstrada() : void LOCAS L: String - Tocals L.bdf testNurEstradas(): void LOCAS XL : String = 1cox is XL bit\* testLerAlanca() : void LOCAS TESTE: String = Totals TEST Ix\* PERSONAGEM S: String = "pers S tot" testLerLoggia'): void PERSONAGEM M: String = "para Mtxf" testCarrinh oConWenorDificuldade(): void PERSONAGEM L: String = "para\_Ltxf" HeatDevolverTodos/Alados(): void HeatDeterminarAlanca(MaisForte(): void PERSONAGEM\_XL: String = "para XL tel" PERSONAGEM\_TESTE: String = "para TEST tel" biov : (Noons (AsyoMisse FICH S: String ="S" FICH\_M: String ="M" estronquistarCom/liados(): void RCH L: String = "L" testGenarGnafsAliangesPublicas(): void HestaGenarGnatcSemLocalsAllados(): void HestDeterminarFatorCompatibilidade(): void #FICH\_TESTE: String = TEST\*
grafo\_locals\_estractes : Atjacs.royMatrixGraph\*Local\_Doublegrafo\_personagens\_ellences: Graph-Personagem Boolean-Hert.coaks(norme\*icheiro: Staing): void roaminho ComWenorDiffoul dede(pouce : Local, tag et : Local): LinkedList-Localvet ficerCompublicipers: Pesconopers, target : Local, grafo\_locals\_aternativo : AdjacencyMatrixCooph-Local, Double-) : Computes for Alancal/normFicheiro : String): void rdevolverTodosAlados(source : Personagem) : HerablerPessonagemi-rdeterminarAlamos(Alabiforte() : Alamos(NelsForte -LOCAS: String - "LOCALS" mova-Himosip, source : Personagem, p\_tag et : Personagem, épositience : bodeen, fetor\_competibil dade: double): bodeen-positivis Noves-Aliences(): Graph-Personagem, Bodeen--CAMINHOS: String = "CAMINHOS" -FERSONAGENS: Siring - "PERSONAGENS" no registerCom/Hisdos(pCrig : Resonagem, terget : Local ) : CompleteCom/Hisdo-redocumerLocal/normeLocal : String difficultiade : Int. p : Personagemi : booleen -ALIANCAS: Bring = "ALIANA§AS" Herficheiro(normificheiro : Shing) : List+Shing> Herfis normgensAlemos(normificheiro : Shing, jg : ContostsDaulogo) : void HerLacois(normificheiro : Shing, jg : ContostsDaulogo) : void radicionarLocal/local : Local) : biodean adicionarEstrada()\_a : Local, (\_b : Local, cueto\_estra da: double): boolean tadicionarPersona gamip : Personagam) : bodiein tadicionarAlancaig, a : Personagam, p. b : Personagam, tipoAlanca : bodiein, fator\_comp : double) : bodiein obterEstradara : Local b : Local ) : double obterTipoAlanca(a: Personagem, b: Personagem): int. oberPersonagenPorNome(come : String) : Personagem obterLocalPorNome(nome : String) : Local munEstradas(): int Conquista ConquistaComAliado munt.coxist) : int onsegue\_conquister: bodeen ePopatyre-aliado : Birsonagem mumPersonagens(): int mumMianos(): int forca necessaria : double CompustoComAliedo(coresegue\_compuistor: boole en, force\_necesseri e: doubl e, localinterm: LinkedList-Local>, el ledo : Personag eProperyee (opaisintermedias: Local ConquistaComAlado(conquista: Conquista, aliado: Personagem) numAliadoDiretos(source : Personagem) : int +Conquista(consegue\_conquistar : bod ean, forca\_necessaria : double, locaisin termedics : LinkedList+Locai+) 📢 obterFatorCompAlianca/a: Personagem b: Personagemi : double doneAlancas): Graph-Personagem, BooleanraproegueConquistar(): boldean rforcaNecesseria(): double bterLocalAssociadoAPersonagem(p : Resonagem) : Local terDados(nomeFicheiro: String): void rse o Ni adosija : Person agem, b : Personagem) : boolean rdevolverTodasPersonagens(): Itemble-Personagent-rdevolverTodasLocalas(): benable-Local > Local sinternedica rgenerGnafsAlian cosPublicas(): Gnaph<Plersonagem, Boolean> genarGnafsAllan ossApenseRublicasGamPess(): Gnaph-Penson agem, Boolean > rgene Christollen ass Tarde Publi ass Sen Piesa"): Geiph «Personagem, Badeen» «gene Christolen Loos is All ados (prig: Piena nagem; el lado: Personagem): Adjecency Méris Chiph-Loos (Double» ePropertyre name: String eliado oProperyor dificuldade: int. determinarFatorCompathilidade(pers\_a: Personagem, pers\_b: Personagem apenas\_publicas: boolean): double personagem: Personagem determinarForcaAllanca(a : Personagem, b: Personagem): int. Property enome: String \*Local/nome : String, diffoul dade: int, personagem: Personagem +Local(nome : String, difficul dade : int) +Plersonagem(nome : String, forca : int +Local/local : Local) Birsonagem(p: Personagem) tgeDono () : Personagem se Dono(p : Personagem): void toString) : String Alianc aMaisForts rdone(): Object Propertyre force: double requals(obj : Object) : boolean thashCode(): int tequals(obj : Object): boolean «Property» - pes\_a : Pesonagem rdone(): Object Popetyo-pes\_b : Pesonagem +AlancaMisForte(force : double, pers\_e : Personagem, pers\_b : Personagem)



EdgeAsDoubleGraphAlgorithms

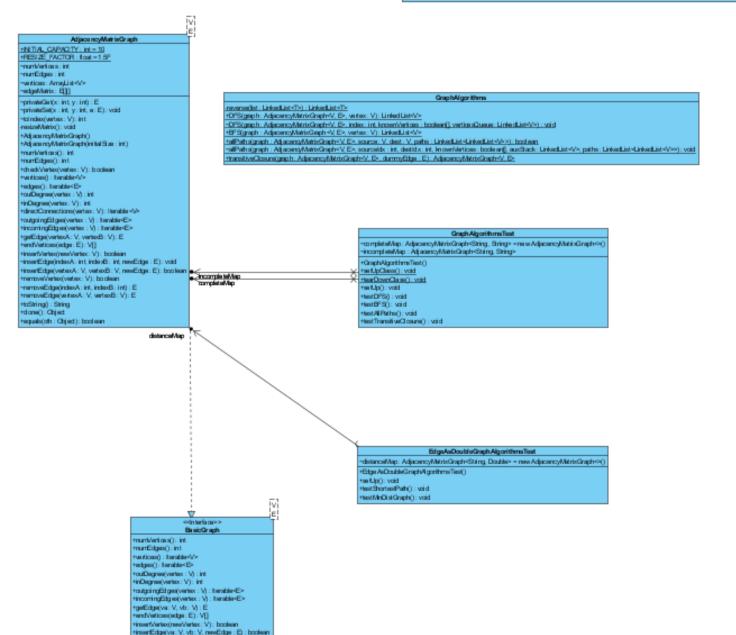
shortestPathigraph : Atlana myMatrixGraph="V Doubles, sourcelds: int. known/erices : boolean() vertices int() minDist : double() : void shotesPathgraph : AdjacencyNatrixGraph+V, Double+, source: V, dest : V, path: LinkedList+V+): double ecress Bithigraph: AdjacencyMitrixGraph-V. Double-, sourcelids: int, deatids: int, vertices index; intfl. path: UnledList-V-V: vaid

Visual Paradigm Standard (Paul Correia (Instituto Superior de Engenharia do Porto))
Agorbnos Jogo

shortestPathConquista(graph: Adjacency/AttrixCraph-Local, Chubler, sourceids: int, Incven/vertices: boolean(), vertices(index: int() minDist; double()), void

remove/Vertex(vertex: V): boidean removeEdge(va : V, vb : V) : E

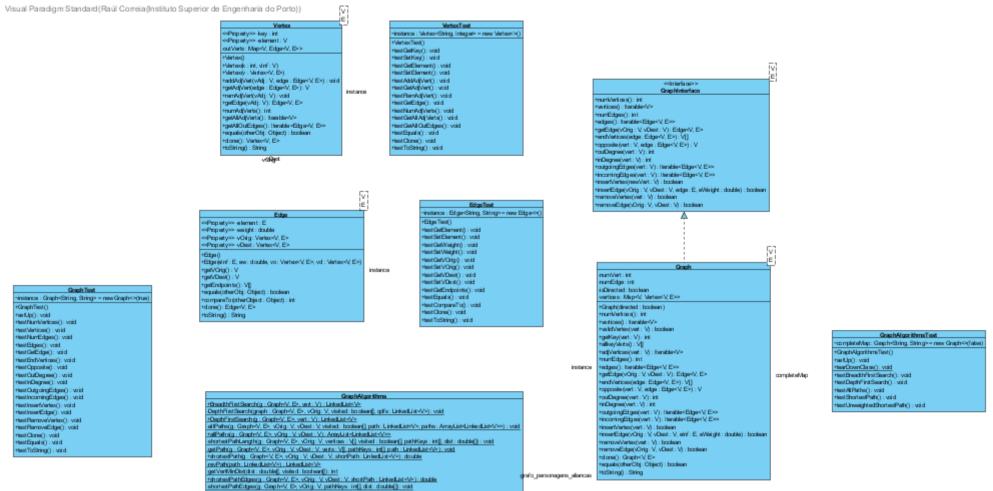
tshortesiPathConquistergriph : Adjaceng/Werk/Criph\*Local, Double\*, source : Local, dest : Local, path : Linke (List\*Local\*) : double recreate Pathigraph. Adjacomy/MarixGraphsLocal. Doubles, sourceids, int. destids, int. vertices intelles, int[] path. LinkedLiet-Local\*), void minDistCraph(graph : AdjacomyMatrixCraph<\/ Double>): AdjacomcyMatrixCraph</ Double>



Adjacency Matrix Graph Test rAdjacencyMatrixGraphTest() tsetUpClass(): void HearDownClass(): void HestNurf/erfices(): void HestNurEdges(): void Hest/Artices(): void HestEitges(): void HestOuDagma(): void HestinDegree(): void HestDirectConnections(): void Hest Cutg ringEdges(): void HestincomingEdges() : void HestGeEdge(): void HeatEnd/lentices(): void HestinserEdge(): void Heatinserf/ertexi'): void HeatRemoveVertex(): void rtestRemoveEdge(): void HestClone(): void HeatEqualsObject(): void restToString() : void

engenharia do

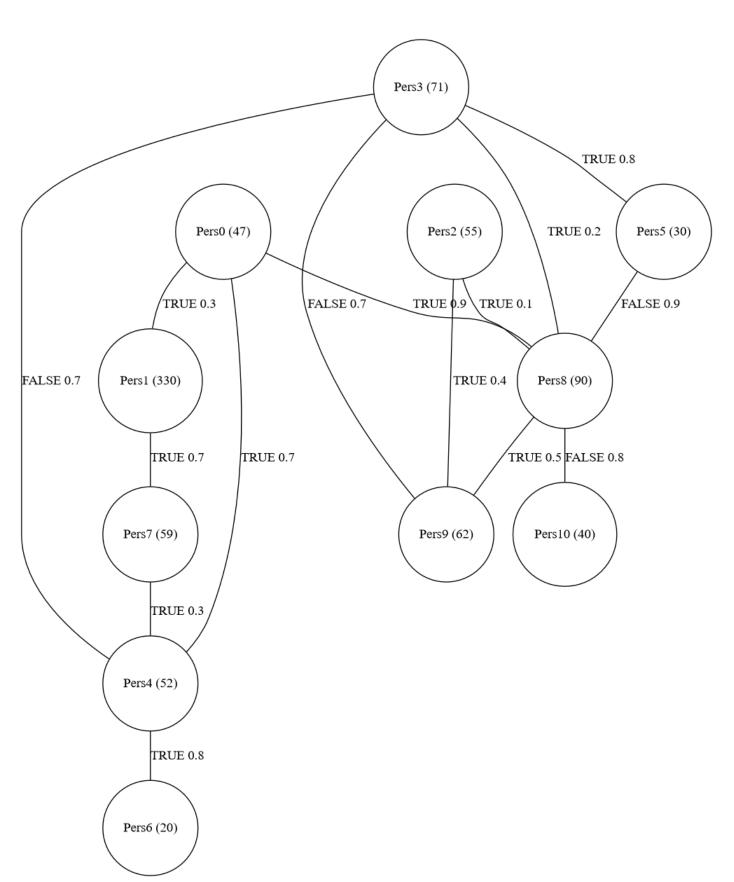






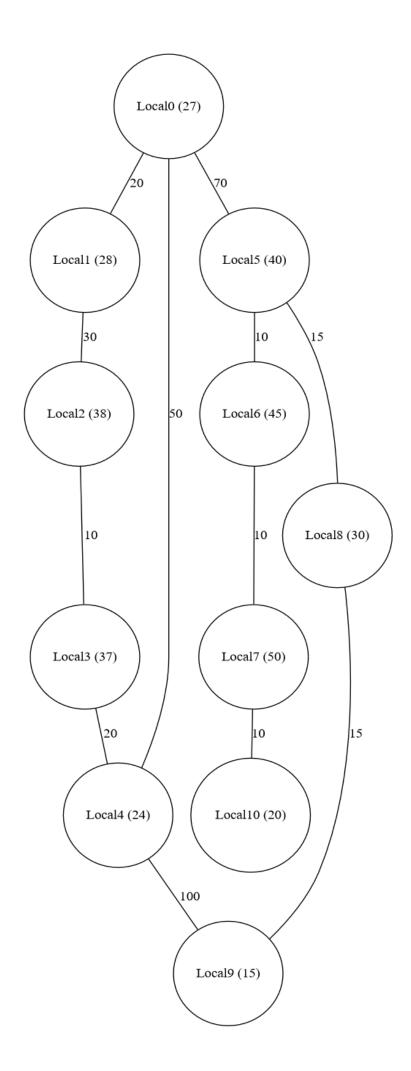
# Grafos de Teste

Grafo Personagens





### **Grafo Locais**





# Complexidade

## Resumo:

**	1		
		A) Ler ficheiro de dados	
		<ul> <li>ControloDoJogo.lerLocais(String nomeFicheiro)</li> </ul>	$O(n*V^2_{local})$
		B) Caminho com menor dificuldade	
		<ul> <li>ControloDoJogo.caminhoComMenorDificuldade(Local source, Local target)</li> </ul>	$O(V^2_{local})$
		C) Verificar se a conquista é possível	
		<ul><li>ControloDoJogo.verificarConquista(Personagem pers, Local source</li></ul>	, Local target,
		AdjacencyMatrixGraph <local, double=""> grafo_locais_alternativo)</local,>	$O(V^2_{locais})$
**	2		
		A) Ler ficheiro de Alianças	
		<ul><li>ControloDoJogo.lerAlianca(String nomeFicheiro)</li></ul>	$O(n*V_{pers})$
		B) Listar os aliados de uma personagem	
		<ul> <li>ControloDoJogo.devolverTodosAliados(Personagem source)</li> </ul>	O(1)
		C) Determinar a aliança mais forte	
		<ul><li>ControloDoJogo.determinarAliancaMaisForte()</li></ul>	O(V <sup>2</sup> <sub>personagem</sub> )
		D) Realizar uma nova aliança	
		<ul> <li>ControloDoJogo.novaAlianca(Personagem p_source, Personagem p_target,</li> </ul>	
		boolean tipoalianca, double fator_compatibilidade)	$O(V^2_{personagem})$
		E) Criar grafo de alianças	
		<ul><li>ControloDoJogo.possiveisNovasAliancas()</li></ul>	$O(V^3_{personagem})$
**	3		
		F) Verificar se a conquista é possível com um dos aliados	
		<ul> <li>ControloDoJogo.conquistaComAliados(Personagem a, Local target)</li> </ul>	$O(V_{personagem} * V^2_{locais})$



### Completo:

1.

```
O(n*V^2_{local})
              lerLocais(String nomeFicheiro)
public List<String> lerFicheiro(String nomeFicheiro) {
    Scanner scn = null;
                                                                     //0(1)
    List<String> lista = new ArrayList<>();
                                                                     //0(1)
try {
      scn = new Scanner(new FileReader(nomeFicheiro));
                                                                     //0(1)
      while (scn.hasNext()) {
                                                                     //0(n)
                                                                     //0(1)
        lista.add(scn.next());
    } catch (FileNotFoundException ex) {
      System.out.printf("Foi imposs vel ler o ficheiro %s\n", nomeFicheiro);//0(1)
    } finally {
      if (scn != null) {
                                                                     //0(1)
        scn.close();
                                                                     //0(1)
    return lista;
                                                                     //0(1)
                                                                     //Total O(n)
public void lerLocais(String nomeFicheiro) {
  Ficheiro f = new Ficheiro();
                                                                              //Total: O(n*Vlocal^2)
  f.lerLocais(nomeFicheiro, this);
}
public void lerLocais(String nomeFicheiro, ControloDoJogo jg) {
    List<String> conteudoFich = lerFicheiro(nomeFicheiro);
                                                                              //0(n)
    boolean lerLocais = false;
                                                                              //0(1)
    boolean lerCaminhos = false;
                                                                              //0(1)
    String linhaSplit[] = null;
                                                                              //0(1)
    for (String linha : conteudoFich) {
                                                                              //0(n)
      if (linha.equals(LOCAIS)) {
                                                                              //0(1)
        lerLocais = true;
                                                                              //0(1)
        continue;
      if (linha.equals(CAMINHOS)) {
                                                                              //0(1)
                                                                              //0(1)
        lerLocais = false;
        lerCaminhos = true;
                                                                              //0(1)
        continue:
      if (lerLocais == true) {
                                                                              //0(1)
        linhaSplit = linha.split(",");
                                                                              //0(z)
        Local l = new Local(linhaSplit[0], Integer.parseInt(linhaSplit[1]));
                                                                              //0(1)
        if (linhaSplit.length >= 3) {
                                                                              //0(1)
          Personagem p = jg.obterPersonagemPorNome(linhaSplit[2]);
                                                                             //0(V)
          if (p!= null) {
                                                                              //0(1)
            l.setDono(p);
                                                                              //0(1)
        jg.adicionarLocal(l);
                                                                              //0(v^2)
        continue:
      if (lerCaminhos == true) {
                                                                              //0(1)
        final int CAMPO_LOCAL_A = 0;
                                                                              //0(1)
        final int CAMPO_LOCAL_B = 1;
                                                                              //0(1)
        final int CAMPO_DIF_ESTRADA = 2;
                                                                              //0(1)
        linhaSplit = linha.split(",");
                                                                              //0(m)
        String string_local_a = linhaSplit[CAMPO_LOCAL_A];
                                                                              //0(1)
        String string_local_b = linhaSplit[CAMPO_LOCAL_B];
                                                                              //0(1)
        Local locala = null, localb = null;
                                                                              //0(1)
        for (Local l : jg.devolverTodosLocais()) {
                                                                              //0(v)
          if (string_local_a.equals(l.getNome())) {
                                                                              //0(1)
```



```
locala = l;
          if (string_local_b.equals(l.getNome())) {
                                                                              //0(1)
            localb = l;
                                                                              //0(1)
          if (locala != null && localb != null) {
                                                                              //0(1)
            double e = Double.parseDouble(linhaSplit[CAMPO_DIF_ESTRADA]);//0(1)
            ig.adicionarEstrada(locala, localb, e);
                                                                              //0(V^2local)
            break; //break ao for
          }
        }
    }
                                          //Total:O(n)*(O(z)+O(Vlocal^2)+O(m)+O(e)+(Vlocal^2)
  }
}
                                          //Total: O(n*Vlocal^2)
```

```
b. caminhoComMenorDificuldade(Local source, Local target)

public LinkedList<Local> caminhoComMenorDificuldade(Local source, Local target) {
    LinkedList<Local> path = new LinkedList<>();
    EdgeAsDoubleGraphAlgorithms.shortestPath(grafo_locais_estradas, source, target, path);
    return path;
}

//O(1)

//O(1)

//Total:O(V^2)
```

verificarConquista(Personagem pers, Local source, Local target, AdjacencyMatrixGraph<Local,</li>
 Double> grafo locais alternativo)

O(V²<sub>locais</sub>)

```
public Conquista verificarConquista(Personagem pers, Local source, Local target, AdjacencyMatrixGraph<Local, Double>
grafo locais alternativo) {
  AdjacencyMatrixGraph<Local, Double> grafo_locais_a_utilizar = grafo_locais_estradas;
  if (grafo_locais_alternativo != null) {
                                                                                               //0(1)
    grafo_locais_a_utilizar = grafo_locais_alternativo;
                                                                                               //0(1)
 if (!grafo_personagens_aliancas.validVertex(pers) || !grafo_locais_a_utilizar.checkVertex(target)) {//0(1)
    return new Conquista(false, -1, null);
                                                                                               //0(1)
  if (target.getDono() != null && target.getDono().equals(pers)) {
                                                                                               //0(1)
    return new Conquista(false, -1, null);
                                                                                               //0(1)
 if (source.getDono() != null && !source.getDono().equals(pers)) {
    return new Conquista(false, -1, null);
  final double SEM_CAMINHO = -1;
                                                                                               //0(1)
  LinkedList<Local> path = new LinkedList<>();
                                                                                               //0(1)
  double dificuldade = graph.AlgoritmosJogo.shortestPathConquista(grafo_locais_a_utilizar, source, target, path,pers);
                                                                                               //0(V^2)
  if (dificuldade != SEM_CAMINHO) {
                                                                                               //0(1)
    if (path.peekFirst() == source) {
                                                                                                //0(1)
      path.removeFirst();
                                                                                               //0(1)
    if (path.peekLast() == target) {
                                                                                               //0(1)
      path.removeLast();
                                                                                               //0(1)
    boolean consegue_conquistar = false;
                                                                                               //0(1)
    if (pers.getForca() > dificuldade) {
                                                                                               //0(1)
      consegue_conquistar = true;
                                                                                               //0(1)
    Conquista cq = new Conquista(consegue_conquistar, dificuldade, path);
                                                                                               //0(1)
    return cq;
                                                                                               //0(1)
                                                                                               //0(1)
  return new Conquista(false, -1, path);
                                                                                               //Total: 0(V^2)
```

2.



### lerAlianca(String nomeFicheiro)

O(n\*V<sub>pers</sub>)

```
public void lerAlianca(String nomeFicheiro) {
  Ficheiro f = new Ficheiro():
  f.lerPersonagensAliancas(nomeFicheiro, this);
                                                                                                      //Total O(n*Vpers)
public void lerPersonagensAliancas(String nomeFicheiro, ControloDoJogo jg) {
   List<String> conteudoFich = lerFicheiro(nomeFicheiro);
                                                                                                      //0(n)
   boolean lerPersonagens = false;
                                                                                                      //0(1)
   boolean lerAliancas = false;
                                                                                                      //0(1)
   String linhaSplit[] = null;
                                                                                                      //0(1)
   for (String linha : conteudoFich) {
                                                                                                       //0(n)
                                                                                                      //0(1)
     if (linha.equals(PERSONAGENS)) {
       lerPersonagens = true;
                                                                                                      //0(1)
       continue;
     if (linha.equals(ALIANCAS)) {
                                                                                                       //0(1)
                                                                                                       //0(1)
       lerPersonagens = false;
       lerAliancas = true;
                                                                                                       //0(1)
       continue:
     if (lerPersonagens == true) {
                                                                                                       //0(1)
       linhaSplit = linha.split(",");
                                                                                                      //0(1)
       Personagem p = new Personagem(linhaSplit[0], Integer.parseInt(linhaSplit[1]));
                                                                                                      //0(1)
       ig.adicionarPersonagem(p);
                                                                                                      //0(1)
       continue
                                                                                                      //0(1)
     if (lerAliancas == true) {
       final int CAMPO_PERS_A = 0;
                                                                                                      //0(1)
       final int CAMPO_PERS_B = 1;
                                                                                                       //0(1)
       final int CAMPO_TIPO_ALIANCA = 2;
                                                                                                      //0(1)
       final int CAMPO_ALIANCA_FATOR_COMPATIBILIDADE = 3;
                                                                                                      //0(1)
       linhaSplit = linha.split(",");
                                                                                                       //0(1)
       String pers_a = linhaSplit[CAMPO_PERS_A];
                                                                                                      //0(1)
       String pers_b = linhaSplit[CAMPO_PERS_B];
                                                                                                      //0(1)
       Boolean tipoAlianca = Boolean.parseBoolean(linhaSplit[CAMPO_TIPO_ALIANCA]);
                                                                                                       //0(1)
       Double fator_comp = Double.parseDouble(linhaSplit[CAMPO_ALIANCA_FATOR_COMPATIBILIDADE]);//O(1)
       Personagem persA = null, persB = null;
                                                                                                       //0(1)
       for (Personagem p : jg.devolverTodasPersonagens()) {
                                                                                                      //0(V)
         if (pers_a.equals(p.getNome())) {
                                                                                                      //0(1)
           persA = p;
                                                                                                      //0(1)
           continue:
         if (pers_b.equals(p.getNome())) {
                                                                                                      //0(1)
                                                                                                      //0(1)
           persB = p;
         if (persA != null && persB != null) {
                                                                                                      //0(1)
           jg.adicionarAlianca(persA, persB, tipoAlianca, fator_comp);
                                                                                                      //0(1)
         }
                                                                                                      //Total O(n*Vpers)
```



O(1)

#### b. devolverTodosAliados(Personagem source)

```
O(V<sup>2</sup>personagens)
               determinarAliancaMaisForte()
         C.
public AliancaMaisForte determinarAliancaMaisForte() {
                                                                                          //0(1)
  Personagem p_a = null;
  Personagem p_b = null;
                                                                                          //0(1)
                                                                                          //0(1)
//0(V)
  double forca alianca = -1;
  for (Personagem p : grafo_personagens_aliancas.vertices()) {
    for (Personagem pAdj : grafo_personagens_aliancas.adjVertices(p)) {
                                                                                          //0(V)
      if (forca_alianca == -1) {
                                                                                          //0(1)
                                                                                          //0(1)
        p_a = p;
        p_b = pAdj;
                                                                                          //0(1)
        forca_alianca = (p.getForca() + pAdj.getForca()) * grafo_personagens_aliancas.getEdge(p, pAdj).getWeight();//0(1)
        double outra_forca = (p.getForca() + pAdj.getForca()) * grafo_personagens_aliancas.getEdge(p, pAdj).getWeight(); //O(1)
        if (outra_forca > forca_alianca) {
                                                                                          //0(1)
          p_a = p;
                                                                                          //0(1)
          p_b = pAdj;
                                                                                          //0(1)
          forca_alianca = outra_forca;
                                                                                           //0(1)
                                                                                           //Total bloco : O(V^2)
  if (p_a != null && p_b != null) {
    AliancaMaisForte amf = \frac{1}{1} amf = \frac{1}{1} AliancaMaisForte(forca_alianca, p_a, p_b);//0(1)
    return amf;
                                                                                          //0(1)
  return null;
                                                                                          //0(1)
                                                                                          //Total 0(V^2)
}
```

Sep instituto superior de encenharia do

d. novaAlianca(Personagem p\_source, Personagem p\_target, boolean tipoalianca, double fator compatibilidade)

```
O(V<sup>2</sup>personagens)
public boolean novaAlianca(Personagem p_source, Personagem p_target, boolean tipoalianca, double fator_compatibilidade) {
  final double SEM_CAMINHO = -1;
  if (!grafo_personagens_aliancas.validVertex(p_source) || !grafo_personagens_aliancas.validVertex(p_target)) {
                                                                                                                 //0(1)
    return false;
                                                                                                                 //0(1)
  }
  LinkedList<Personagem> path = new LinkedList<>();
                                                                                                                 //0(1)
  double dist = 0;
                                                                                                                 //0(1)
  //Vamos ver o caminho de p_source para p_target
  Graph < Personagem, Boolean > grafo_aliancas_publicas_sem_peso = gerarGrafoAliancasApenasPublicasSemPeso(); <math>//O(V^2)
  dist = graphbase.GraphAlgorithms.shortestPath(grafo_aliancas_publicas_sem_peso, p_source, p_target, path); //0(V^2)
if (dist == SEM_CAMINHO) {
                                                                                                                 //0(1)
    return adicionarAlianca(p_source, p_target, tipoalianca, fator_compatibilidade);
                                                                                                                 //0(1)
  } else {
    int numPers = path.size();
                                                                                                                 //0(1)
     //Se o número de Personagens é maior do que 1, então é n-1 ramos
    if (numPers > 1) {
                                                                                                                 //0(1)
      numPers--;
                                                                                                                 //0(1)
    double fator_comp = 0;
                                                                                                                 //0(1)
    Personagem a = null;
                                                                                                                 //0(1)
                                                                                                                 //0(1)
    Personagem b = null;
    while (!path.isEmpty()) {
                                                                                                                 //0(V)
      if (a == null && b == null) {
                                                                                                                 //0(1)
        a = path.pop();
                                                                                                                 //0(1)
        b = path.pop();
                                                                                                                 //0(1)
        fator_comp = grafo_personagens_aliancas.getEdge(a, b).getWeight();
                                                                                                                 //0(1)
      } else {
        a = b;
                                                                                                                 //0(1)
        b = path.pop();
                                                                                                                 //0(1)
        fator_comp += grafo_personagens_aliancas.getEdge(a, b).getWeight();
                                                                                                                 //0(1)
    double mediaComp = fator_comp / numPers;
                                                                                                                 //0(1)
    grafo_personagens_aliancas.insertEdge(p_source, p_target, tipoalianca, mediaComp);
                                                                                                                 //0(1)
    return true;
                                                                                                                 //0(1)
  }
}
                                                                                                        //Total:0(V^2)
public Graph<Personagem, Boolean> gerarGrafoAliancasApenasPublicasSemPeso() {
  Graph<Personagem, Boolean> grafo_aliancas_publicas = grafo_personagens_aliancas.clone();
  for (Personagem pOrig : grafo_personagens_aliancas.vertices()) {
                                                                                                        //0(V)
    for (Personagem pAdj : grafo_personagens_aliancas.adjVertices(pOrig)) {
                                                                                                        //0(V)
      if (grafo_personagens_aliancas.getEdge(pOrig, pAdj) != null && grafo_personagens_aliancas.getEdge(pOrig, pAdj).getElement(
) == false) {
                                                                                                        //0(1)
        grafo_aliancas_publicas.removeEdge(pOrig, pAdj);
                                                                                                        //0(1)
      } else {
        if (grafo_personagens_aliancas.getEdge(pOrig, pAdj)!= null && grafo_personagens_aliancas.getEdge(pOrig, pAdj).getEleme
nt() == true) {
                                                                                                        //0(1)
          grafo_aliancas_publicas.getEdge(pOrig, pAdj).setWeight(1);
                                                                                                        //0(1)
  return grafo_aliancas_publicas;
                                                                                                        //0(V^2)
}
```



#### e. possiveisNovasAliancas()

```
public Graph<Personagem, Boolean> possiveisNovasAliancas() {
  Graph<Personagem, Boolean> ng = (Graph<Personagem, Boolean>) grafo_personagens_aliancas.clone();
  for (Personagem p_k : grafo_personagens_aliancas.vertices()) {
    for (Personagem p_i : grafo_personagens_aliancas.vertices()) {
                                                                                                   //0(v)
       \textbf{if} (!p\_i.equals(p\_k) \&\& ng.getEdge(p\_i, p\_k) != \textbf{null}) \{ \\
        for (Personagem p_j : grafo_personagens_aliancas.vertices()) {
                                                                                                   //0(v)
          if (!p_i.equals(p_j) && !p_j.equals(p_k) && ng.getEdge(p_k, p_j) == null) {
            if (ng.getEdge(p_i, p_j) != null) {
                                                                                                   //0(1)
//0(1)
              double ed1 = ng.getEdge(p_k, p_i).getWeight();
              double ed2 = ng.getEdge(p_i, p_j).getWeight();
                                                                                                   //0(1)
              double media = (ed1 + ed2) / 2;
                                                                                                   //0(1)
              ng.insertEdge(p_k, p_j, Boolean.TRUE, media);
                                                                                                   //0(1)
       }
  return ng;
                                                                                //O(V) * O(V) * O(V)
```





f. conquistarComAliados(Personagem a, Local source, Local target)  $O(V_{personagens} * V^2_{locais})$ 

```
public ConquistaComAliado conquistarComAliados(Personagem pOrig, Local source, Local target) {
  final double INVALIDO = -1;
  if (!grafo_personagens_aliancas.validVertex(pOrig) || !grafo_locais_estradas.checkVertex(target)) {
                                                                                                                 //0(v)
    return null;
  if (target.getDono() != null && target.getDono().equals(pOrig)) {
                                                                                                                 //0(1)
    return new ConquistaComAliado(false, -1, null,null);
                                                                                                                 //0(1)
  if (source.getDono() != null && !source.getDono().equals(pOrig)) {
                                                                                                                 //0(1)
    return new ConquistaComAliado(false, -1, null,null);
                                                                                                                 //0(1)
  int forca_antiga = pOrig.getForca();
                                                                                                                 //0(1)
 for (Personagem aliado : grafo_personagens_aliancas.adjVertices(pOrig)) {
                                                                                                                 //0(V)
    Adjacency Matrix Graph < Local, Double > grafo_sem\_locais\_aliados = gerar Grafo SemLocais Aliados (pOrig, aliado); //O(V^2)
    pOrig.setForca(forca_antiga);
                                                                                                                 //0(1)
                                                                                                                 //0(1)
    pOrig.setForca(determinarForcaAlianca(pOrig, aliado));
    Conquista cq = verificarConquista(pOrig,source,target, grafo_sem_locais_aliados);
                                                                                                         //0(V_locais_^2)
    if (cq.consegueConquistar()) {
                                                                                                                 //0(1)
                                                                                                                 //0(1)
      pOrig.setForca(forca_antiga);
      ConquistaComAliado cqal = new ConquistaComAliado(cq, aliado);
                                                                                                                 //0(1)
      return cgal;
                                                                                                                 //0(1)
  pOrig.setForca(forca_antiga);
                                                                                                                 //0(1)
  return new ConquistaComAliado(false, INVALIDO, new LinkedList<Local>(), null);
                                                                                                                 //0(1)
}
                                                                    //O(Vpersonagens) * (O(V^2 Locais) + O(V^2 Locais))
public AdjacencyMatrixGraph<Local, Double> gerarGrafoSemLocaisAliados(Personagem orig, Personagem aliado) {
    AdjacencyMatrixGraph<Local, Double> grafo_locais_estra (AdjacencyMatrixGraph<Local, Double>) grafo_locais_estra
das.clone();
                                                                                               //0(V^2)
    for (Local loc : grafo_locais_estradas.vertices()) {
                                                                                               //0(n)
      if (loc.getDono() != null) {
                                                                                               //0(1)
        if (loc.getDono().equals(aliado)) {
                                                                                               //0(1)
          grafo_locais_sem_o_aliado.removeVertex(loc);
                                                                                               //0(V)
    return grafo_locais_sem_o_aliado;
                                                                                      //0(V^2)
```