RELATÓRIO INTERMÉDIO DO PROJETO LAPR1

EQUIPA 1

1181242_ISAAC_MATEUS

1171887_DUARTE_PAULO

1181257_LEONOR_CARVALHAIS

1180567_JOÃO_PEDRO

ÍNDICE

RELATÓRIO INTERMÉDIO DO PROJETO LAPR1

1.Introdução

No âmbito da disciplina de Laboratório/Projeto 1, foi nos proposto a elaboração de um programa na linguagem Java, utilizando conhecimentos adquiridos em Algoritmia e programação e em seguida a realização de um relatório em relação ao projeto em causa.

Este projeto tem como objetivos o desenvolvimento do trabalho em equipa e , adicionalmente, colocar em prática os nossos conhecimentos da linguagem Java.

2. Metodologia de trabalho

2.1 EduScrum

A definição de EduScrum é a colaboração intensa, a partilha de conhecimento por parte de todos os elementos do grupo/empresa e elevada autonomia dos mesmos. Esta metodologia foi nos muito útil e fundamental durante a realização do nosso projeto, porque ajudou a promover a colaboração entre todos os membros do grupo, fomentar o desenvolvimento de múltiplas competências e a coordenação do nosso trabalho foi estudada de forma rigorosa, permitindo assim uma melhor colaboração entre todos.

Ao longo do nosso projeto, todos os elementos tiveram dúvidas e questionaram aos restantes membros, discutindo ideias e trocando informação crucial para superar as mesmas e, assim, continuar o desenvolvimento do projeto.

2.2 Planeamento e distribuição de tarefas (Iteração 1 e 2)

Durante as três primeiras duas semana de trabalho, até dia 8 de Janeiro, realizamos as seguintes tarefas:

- Implementação dos módulos necessários à leitura de dados que representam uma rede social e o cálculo de um conjunto de medidas que caracterizam a rede.
- Estudo de análise de redes sociais, em especial as medidas ao nível dos nós e ao nível da rede definidas nas subsecções 3.2 e 3.3.
- Estudar também o cálculo de valores e vetores próprios.
- Foram criados métodos para se ler: as potências da matriz de adjacências, o grau de um nó, os valores próprios, ficheiros de redes sociais, a informação carregada de redes sociais numa matriz, a densidade, o grau médio de uma matriz.
- Criação da classe de testes.
- Criação de um main com um menu com opções para cada um dos métodos
- Desenvolvimento da nossa aplicação.
- Estudar a forma de como vamos carregar os redes socias (grafos), armazenadas e descritas em dois ficheiros de texto (txt), sendo que um ficheiro descreve as entidades da rede (nós) e o outro os relacionamentos entre nós (ramos).
- Criar métodos para calcular os graus de entrada e saída dos nós – ISAAC e LEONOR

- Criar compatibilidade do programa para grafos orientados e modificar o programa original para mostrar os resultados para todos os nós instantemente.

- Fazer e criar a matriz para o método PageRank

- Realização do projeto final proposto.

As tarefas foram distribuídas da seguinte forma:

- Criar método para ler ficheiros de informação das várias instituições – ISAAC
- Fazer o main para funcionar no unix – ISAAC
- Criar método para calcular as potências da matriz de adjacências – JOÃO
- Criar método para calcular o grau de um nó – DUARTE
- Criar classe de testes – ISAAC
- Criar método para calcular os valores próprios – DUARTE e LEONOR
- Criar métodos de ler ficheiros de redes sociais – JOÃO
- Criar método para carregar informação de redes sociais numa matriz – LEONOR
- Criar método para calcular a densidade – LEONOR
- Criar main com um menu com opções para cada um dos métodos – ISAAC
- Criar método para calcular o grau médio de uma matriz – ISAAC
- Criar métodos para calcular os graus de entrada e saída dos nós – ISAAC e LEONOR
- Criar compatibilidade do programa para grafos orientados – ISAAC
- Modificar o programa original para mostrar os resultados para todos os nós instantemente – LEONOR
- Fazer e criar a matriz para o método PageRank – ISAAC
- Relatório final – DUARTE e JOÃO
    - Introdução – JOÃO
    - Metodologia de trabalho – DUARTE
    - Análise das redes sociais – DUARTE
    - Desenvolvimento e implementação da aplicação – JOÃO
    - CONCLUSÃO – DUARTE e JOÃO
- Alteração do relatório final para a iteração 2 – DUARTE
    - Introdução – JOÃO e DUARTE
    - Metodologia de trabalho – DUARTE
    - Análise das redes sociais – DUARTE
    - Desenvolvimento e implementação da aplicação – JOÃO e DUARTE
    - CONCLUSÃO – DUARTE e JOÃO

2.3 Autoavaliação

A equipa 1 realizou todas as tarefas que foram propostas a tempo e de forma organizada sem problemas. Estivemos sempre prontos para ajudarmo-nos uns aos outros, sempre que surgia uma dúvida. A autoavaliação de grupo é positiva, pois não foram colocados aspetos negativos a nenhum membro da equipa, visto que realizaram todos as tarefas que lhes foram propostas.

3. Análise de Redes Sociais

Uma rede social consiste num conjunto finito de atores com as suas relações, ou laços, definidos entre eles. Estas podem ser de natureza pessoal ou profissional e podem variar de um relacionamento casual a um vínculo familiar próximo. Além das relações sociais, os links podem também representar fluxo de informações / bens / dinheiro, interações, semelhanças, entre outros. A estrutura dessas redes é geralmente representada por grafos matemáticos. Um grafo é composto por duas unidades fundamentais: vértices e arestas. Cada borda é definida por um par de vértices. Os vértices são capazes de representar uma ampla variedade de entidades individuais (por exemplo, pessoas, organizações, países, documentos, produtos, plantas e animais) de acordo com o campo de aplicação. Por sua vez, uma borda é a linha que liga dois vértices e pode representar vários tipos de relações entre entidades individuais (por exemplo, comunicação, cooperação, amizade, parentesco, conhecidos e comércio). As bordas podem ser orientadas ou não, dependendo se a natureza da relação é assimétrica ou simétrica.

Os dois tipos principais de estruturas de dados teóricos de grafos são representados da seguinte forma: o primeiro contém as estruturas de lista e o segundo as estruturas matriciais. Estas estruturas são apropriadas para armazenar grafos em computadores, a fim de analisá-los ainda mais usando ferramentas automáticas. Listar estruturas, tais como listas de incidência e lista de adjacências, são adequados para armazenar grafos esparsos, devido ao reduzido espaço de armazenamento. Por outro lado, as estruturas matriciais, como as matrizes de incidência, matrizes de adjacência, matrizes Laplacianas e distância (idênticas às matrizes de adjacência, com a diferença de que entradas da matriz são os comprimentos dos caminhos mais curtos entre pares de vértices) são apropriados para representar matrizes completas.

Vários tipos de grafos podem ser usados para modelar diferentes tipos de redes sociais. Por exemplo, há grafos que podem ser classificados de acordo com a direção de seus links. Isso nos leva à diferenciação entre grafos não orientados e orientados. Grafos não orientados são grafos cujas arestas ligam pares não ordenados
de vértices. Grafos orientados podem ser definidos como grafos cujas bordas têm uma orientação atribuída. Os grafos orientados ligam os vértices através de setas. Já os grafos não orientados ligam os vértices através de uma linha, pois esta pode ser bidirecional. Um exemplo típico de um grafo não orientado
O gráfico é o Facebook™, uma vez que, nessa rede social, o laço de amizade estabelecido é mútuo, pois ao aceitar o pedido, teoricamente ambas as pessoas são amigas. Da mesma forma, o Twitter™ é um exemplo de um grafo orientado já que uma pessoa pode ser seguida por outros sem necessariamente segui-los. Neste caso uma relação unidirecional é estabelecida.
Em relação aos valores atribuídos às arestas, podemos fazer uma distinção entre não ponderado e gráficos ponderados. A menos que seja explicitamente dito, sempre assumimos que os grafos não são valorados.
Grafos não valorados são binários, pois as arestas estão presentes ou ausentes. Por outro lado, os grafos valorados, são grafos mais ricos, pois cada ligação tem um peso

associado. De acordo com Mark Granovetter nas redes sociais, o peso de uma ligação é geralmente em função de duração, intensidade emocional, frequência de interação, intimidade e troca de Serviços. Portanto, laços fortes geralmente representam amigos íntimos, e laços fracos representam conhecidos. Em outros tipos de redes, o peso de uma ligação pode representar uma variedade de coisas,
dependendo do contexto; por exemplo, uma ligação pode representar o número de lugares sentados entre aeroportos, o
número de produtos trocados, etc.

Para grafos não orientados e não valorados, as matrizes de adjacência são binárias e simétrica. Para grafos orientados e valorados, as entradas das matrizes têm valores de intervalo [] e são não simétricas. Nos dois casos, lidamos com matrizes não negativas.

Análise de redes sociais (SNA) é um processo de análise quantitativa e qualitativa de uma rede social. O principal objetivo desta técnica é examinar tanto os conteúdos quanto os padrões de relacionamento nas redes sociais, a fim de compreender as relações entre os atores e as implicações dessas relações.

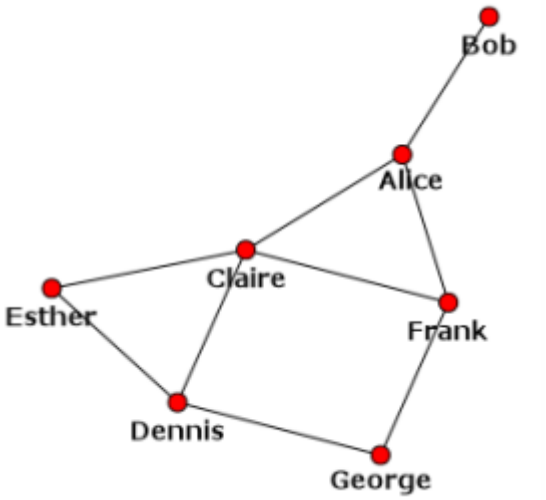


Figura 1 – Retirada do enunciado, é um exemplo de grafo que representa uma rede social de pessoas.

Vamos apresentar o grafo da figura 1 com a sua respetiva matriz. É um grafo não orientado e não valorado.

$$
\mathrm{A} = \begin{array}{c} \\ \textit{Alice} \\ \textit{Bob} \\ \textit{Claire} \\ \textit{Dennis} \\ \textit{Esther} \\ \textit{Frank} \\ \textit{George} \end{array}
\begin{array}{c}
\begin{array}{ccccccc} \textit{Alice} & \textit{Bob} & \textit{Claire} & \textit{Dennis} & \textit{Esther} & \textit{Frank} & \textit{George} \end{array} \\
\begin{pmatrix}
0 & 1 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 0
\end{pmatrix}
\end{array}
$$

Como calcular as medidas ao nível dos nós?

Por uma lado, começa-se por calcular o grau de um nó. Esta medida representa o número de arestas incidentes em um determinado nó. Dito de outra forma, esta métrica representa o número de vizinhos do nó. Pode ser calculado da seguinte forma:

$$k_v = \sum_{j=1}^{n} a_{vj}, \qquad 0 < k_v < n,$$

em que a_{vj} representa um coeficiente da matriz de adjacências A e n é o número de nós da rede.

Por outro lado, para calcular a centralidade do vetor próprio (Esta medida generaliza a medida grau do nó, incorporando a importância dos nós vizinhos) pode ser calculado da seguinte forma:

$$x_i = \sum_{j=1}^{n} \frac{1}{\lambda} a_{ij} x_j,$$

em que x_i representa a centralidade do no i, a_{ij} representa um coeficiente da matriz de adjacências A e λ é o maior valor próprio da matriz de adjacências A.

Relativamente às medidas ao nível de rede, começamos por calcular o grau médio. Esta medida representa a média dos graus de todos os nós de uma rede e permite medir a conectividade global desta rede. É calculado da seguinte forma:

$$\bar{k} = \frac{1}{n}\sum_{i=1}^{n} k_i,$$

em que k_i é o grau do nó *i* e *n* é o número de nós da rede

De seguida, calculamos a densidade. Esta medida é importante para explicar o nível geral de conectividade de uma rede. Este valor representa a proporção de ramos na rede em relação ao número máximo possível de ramos. A densidade, representada aqui por a letra ρ, é uma quantidade que varia entre um valor mínimo de 0, no caso da rede não ter ramos, até um valor máximo de 1, caso estarmos perante a presença de um grafo completo. Desta definição podemos concluir que valores elevados estão associados a redes densas e valores baixos de densidade estão associados a redes difundidas. Pode ser calculada da seguinte forma:

$$\rho = \frac{m}{m_{max}}, \qquad 0 < \rho < 1,$$

em que *m* é o número de ramos da rede e m_{max} é o número de ramos se considerarmos que existe um ramo entre cada um dos pares de nós da rede em análise.

Por fim, calculamos a potência da matriz de adjacência. A matriz de adjacência diz-nos quantos caminhos de comprimento um existem entre cada par de nós. A matriz de adjacência ao quadrado diz-nos quantos caminhos de comprimento dois existem entre dois nós. A matriz de adjacência A^k permite obter o número de caminhos de comprimento k entre qualquer par de nós. Pode ser calculada da seguinte forma:

$$A^k = \prod_{i=1}^{k} A,$$

em que A^k é a *k*-enésima potencia da matriz de adjacências A.

4.1 Redes/Grafos Orientados

Na Figura 2 é apresentado um exemplo deste tipo de rede. Neste caso particular a matriz de adjacências associada poderia ser a matriz apresentada na Tabela 2 ou a sua transposta.

$$\mathrm{A} = \begin{array}{c} \\ \textit{Alice} \\ \textit{Bob} \\ \textit{Claire} \\ \textit{Dennis} \end{array} \begin{array}{c} \begin{array}{cccc} \textit{Alice} & \textit{Bob} & \textit{Claire} & \textit{Dennis} \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \end{array}$$

Em primeiro lugar, calcula-se o grau de entrada de um nó. O grau de entrada de um nó *v*, geralmente identificado como k_v^+, é uma medida da adjacência imediata e do envolvimento de um nó na rede, que representa o número de ramos orientados que terminam (ou entram) em v. Esta medida pode ser escrita e calculada da seguinte forma:

$$k_v^+ = \sum_{j=1}^{N} a_{vj}, \qquad 0 < k_v^+ < N.$$

O grau de saída de um nó o *v*, geralmente identificado como k_v^-, é uma medida da adjacência imediata e do envolvimento do nó na rede, que representa o número de ramos orientados que iniciam (ou saem) em v. Esta medida pode ser representada com a seguinte expressão:

$$k_v^- = \sum_{j=1}^{N} a_{jv}, \qquad 0 < k_v^- < N.$$

O algoritmo Page Rank é utilizado atualmente, com algumas modificações, no motor de pesquisa Google para apresentar o resultado de uma pesquisa. Este algoritmo permite estimar a popularidade/ranking de uma página web utilizado as ligações (links) entre páginas e a sua importância. Mais tarde, e porque a internet (conjunto de páginas web) pode ser vista como uma rede social / grafo de páginas web, em que as páginas são nós e os links são ramos orientados, este algoritmo passou a ser utilizado para analisar redes sociais. Neste trabalho vamos explorar uma versão muito simplificada do algoritmo que inclui

estratégias para resolver apenas dois problemas existentes em redes sociais que afetam o desempenho do algoritmo Page Rank. O primeiro problema é a existência de nós que não têm qualquer ramo de saída ($K_v^- = 0$). O segundo problema é a existência de redes em que não é possível fazer um percurso (seguindo os ramos orientados) entre quaisquer dois nós da rede/grafo. A rede apresentada na Figura 2, para implementar o algoritmo Page Rank temos que transformar a rede numa matriz que permita encontrar os nós mais populares, com melhor ranking. Em primeiro lugar começamos por definir uma matriz quadrada M, de dimensão N, que representa a rede social e que reduz o impacto da existência de "dangling nodes" e de estarmos na presença de uma reducible network:

$$M = d\bar{A} + \frac{1-d}{N}\mathbb{1},$$

em que A é uma matriz quadrada de dimensão N, cujos coeficientes são definidos da seguinte forma:

$$\bar{a}_{ij} = \begin{cases} \frac{1}{L(j)}, & \text{se existir um ramo orientado do nó j para o nó i} \\ \frac{1}{N}, & \text{se o nó j for um dandling node} \\ 0, & \text{outros casos} \end{cases}$$

em que L(p) representa o número de ramos de saída do nó; 1 representa uma matriz cujos coeficientes têm todos o valor 1 e d é uma constante, com valor compreendido entre zero e um. Definida a matriz M, o algoritmo Page Rank pode ser implementado de forma simples recorrendo a um algoritmo iterativo, em que K é o número de iterações definido pelo utilizador:

1. Inicializar um vetor x0, de dimensão N × 1, com componentes não negativos.

2. Para l=1 até K fazer: $x_l = Mx_{l-1}$.

3. Fim Para

4. Apresentar x_K

O vetor apresentado no passo 4 é também conhecido por vetor Page Rank. Normalmente o vetor x0 é um vetor cujos coeficientes tomam todos o valor 1. Quanto maior o número de iterações deste algoritmo mais relevante (no sentido de encontrar os nós mais importantes) é o vetor Page Rank e mais este se aproxima do vetor próprio associado ao maior valor próprio da matriz M. Dito de outra forma, este algoritmo iterativo converge para o vetor próprio associado ao maior valor próprio da matriz M. Com isto podemos concluir que uma outra forma de calcular o vetor Page Rank seria calcular o vetor próprio associado ao maior valor próprio da matriz M.

4.Desenvolvimento e Implementação da aplicação

4.1 Método Main

Durante o desenvolvimento do nosso programa fomos criando vários métodos para simplificar o código e facilitar a sua leitura. Criamos também algumas classes com o intuito de reduzir a quantidade de código.

```
public static void main(String[] args) throws FileNotFoundException {

   if (args.length >= 3) {

      String[][] infoNos = new String[Config.N_MAX_NOS][Config.N_CAMPOS_INFO];
      int nosLength = LeitorFicheiros.lerMatriz(args[args.length - 2], Config.N_CAMPOS_INFO, infoNos, Config.N_MAX_NOS);
      String[][] ramos = new String[Config.N_MAX_NOS][Config.N_CAMPOS_RELACOES];
      int ramosLength = LeitorFicheiros.lerMatriz(args[args.length - 1], Config.N_CAMPOS_RELACOES, ramos, Config.N_MAX_RAMOS);
      if (nosLength > 0 && ramosLength > 0) {
         double[][] Matriz = criarMatriz(infoNos, ramos, ramosLength, nosLength);
```

Criação das matrizes

No método main são criadas várias matrizes com a finalidade de carregar em memória todas informações necessárias para a análise da rede. Estas matrizes carregam informações de dois ficheiros de texto, um com as entidades da rede (nós) e outro com o relacionamento entre nós (ramos).

```
if (args.length == 5 && args[0].equals("-t") && args[1].equals("-k")) {

   Formatter output = criarEAbrirFicheiroTexto(args);

   CalculosMatrizes.calcCentralidadeVetorProprio(Matriz);
   outputInfoGrausNos(output, Matriz, infoNos);
   output.format("Grau médio= %f.2%n%n", CalculosMatrizes.calcularGrauMedio(Matriz));
   output.format("Densidade= %f.2%n%n", CalculosMatrizes.calcularDensidade(Matriz, calcNumRamos(ramos, ramosLength, infoNos, nosLength)));
   output.format("Centralidade do vetor próprio: %f.2%n%n", CalculosMatrizes.calcCentralidadeVetorProprio(Matriz));

   if (Integer.parseInt(args[2]) > 0) {
      outputMatrizPotencia(output, Integer.parseInt(args[2]), Matriz);
   }else{
      output.format("A matriz de potências não pode ser criada pois o valor inserido não é válido.%n");
   }
   outputValoresProprios(output, Matriz);
   outputVetoresProprios(output, Matriz);

   String data = new SimpleDateFormat("MMdd").format(Calendar.getInstance().getTime());
   if (data.equals("1225")) {
      output.format("Feliz Natal!");
   }
   if (data.equals("1231")) {
      output.format("Boas entradas!");
   }
   if (data.equals("0610")) {
      output.format("Feliz dia de Portugal!");
   }

   output.close();
   System.out.println("Criação do ficheiro bem sucedida.");
```

Criação do ficheiro de texto e escrita do mesmo.

É também no método main que o utilizador faz a escolha do output. Caso seja introduzido o comando -t-k, o output do programa será um ficheiro de texto, que é criado no momento com a data no seu nome. Caso o utilizador introduza o comando -n, o output pretendido passa a ser o ecrã, então o menu é inicializado e o utilizador pode escolher uma das várias funções disponíveis.

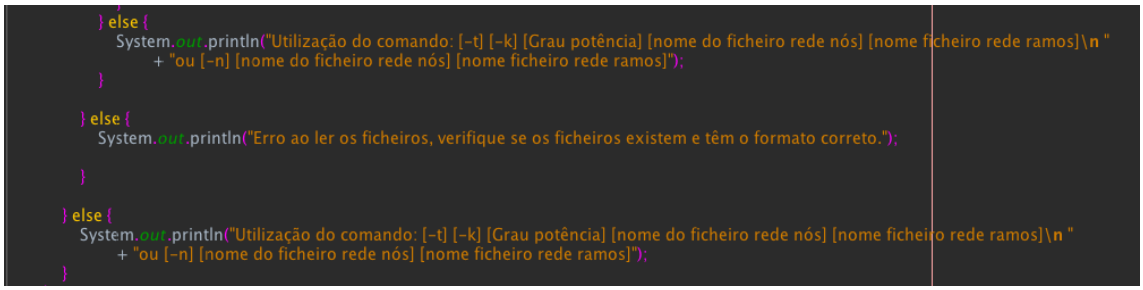
```
public static int Menu() {
    System.out.println("Escolha uma opção");
    System.out.println("1- Grau de um nó ");
    System.out.println("2- Centralidade do vetor próprio");
    System.out.println("3- Grau médio da rede");
    System.out.println("4- Densidade da rede");
    System.out.println("5- Potências da Matriz de Adjacências");
    System.out.println("0-Cancelar");

    int opcao;
    do {
        System.out.println("Escolha a opção:");

        opcao = sc.nextInt();
    } while (opcao > 5 || opcao < 0);

    sc.nextLine();
    return opcao;
}
```

Método Menu

Após o utilizador inserir a opção pretendida, esta é devolvida ao método main que começa a respectiva ação.

```
} else if (args[0].equals("-n")) {
    Formatter output = new Formatter(System.out);
    int opcao = Menu();
    switch (opcao) {
        case 1:
            output.format("Insira o id do nó qual quer calcular o grau%n");
            String id = sc.nextLine();
            int posId = Utilitarios.procurarString(infoNos, id, nosLength);
            if (posId >= 0) {
                output.format("O grau do nó %s é: %d%n", id, CalculosMatrizes.calcularGrauNo(Matriz, posId));
            }else{
                output.format("O id inserido não existe%n");
            }
            break;
        case 2:
            output.format("A centralidade do vetor próprio é: %.2f%n", CalculosMatrizes.calcCentralidadeVetorProprio(Matriz));
            break;
        case 3:
            output.format("O grau médio da rede é %.2f%n", CalculosMatrizes.calcularGrauMedio(Matriz));
            break;
        case 4:
            output.format("A densidade da rede é: %.2f%n", CalculosMatrizes.calcularDensidade(Matriz, calcNumRamos(ramos, ramosLength, infoNos, nosLength)));
            break;
        case 5:
            output.format("Insira o grau da potência da matriz que quer calcular%n");
            int grauPotencia = sc.nextInt();
            if (grauPotencia > 0) {
                outputMatrizPotencia(output, grauPotencia, Matriz);
            } else {
                System.out.println("O grau da potência tem de ser 1 ou maior.");
            }
            break;
    }
```

Switch case que inicia cada uma das funcionalidades do programa



Caso o programa não consiga realizar as suas funções, alerta o utilizador de como o usar corretamente ou indica que ocorreu um erro.

```
public static Scanner sc = new Scanner(System.in);
```

Método scanner

Um método público scanner foi criado para não ser necessário criar um novo objeto scanner sempre que é preciso ler informação.

4.2 Classe CalculosMatrizes

Na classe CalculosMatrizes estão presentes vários métodos relativos a operações com matrizes para facilitar o código, tornando o programa mais rápido e para ajudar no tratamento de dados.

Importamos também a biblioteca la4j (Linear Álgebra for Java) para não serem necessários tantos cálculos matemáticos reduzindo assim a probabilidade de aparecimento de erros.

```
public static float calcularDensidade(double[][] Matriz, int nrRamos) {
    int nrMaxRamos = (int) Math.pow(Matriz.length, 2) - Matriz.length;

    return (float) nrRamos / nrMaxRamos;
}
```

Este método permite medir o número de ligações diretas existentes mediante um número total de ligações possíveis.

```
public static float calcularGrauMedio(double[][] ramos) {
    int somaNos = 0;
    for (int i = 0; i < ramos.length; i++) {
        somaNos += calcularGrauNo(ramos, i);
    }

    return (float) somaNos / ramos.length;
}
```

Este método calcula a média dos graus de todos os nós de uma rede e permite medir a conectividade global desta rede.

```
public static double[] calcularValoresProprios(double[][] ramos) {

    Matrix adjacencias = new Basic2DMatrix(ramos);

    EigenDecompositor eigenD = new EigenDecompositor(adjacencias);

    Matrix[] matrizPropria = eigenD.decompose();

    double matValProprio[][] = matrizPropria[1].toDenseMatrix().toArray();

    double[] valoresProprios = new double[ramos.length];
    for (int i = 0; i < ramos.length; i++) {
        valoresProprios[i] = matValProprio[i][i];
    }
    return valoresProprios;
}
```

Este método calcula os valores próprios, com o auxílio da biblioteca la4j.

```
public static int calcularGrauNo(double[][] ramos, int posicaoID) {
    int grauNo = 0;

    if (posicaoID != -1) {
        for (int coluna = 0; coluna < ramos.length; coluna++) {

            if (ramos[posicaoID][coluna] != 0) {
                grauNo++;
            }
        }
    } else {
        System.out.println("O ID inserido não é válido");

    }
    return grauNo;
}
```

Método que permite calcular o número de ligações de um determinado nó.

```
public static double[][] calcularVetoresProprios(double[][] ramos) {

    Matrix adjacencias = new Basic2DMatrix(ramos);

    EigenDecompositor eigenD = new EigenDecompositor(adjacencias);

    Matrix[] matrizPropria = eigenD.decompose();

    return matrizPropria[0].toDenseMatrix().toArray();
}
```

Calcula o vetor próprio com o auxílio de uma matriz que é depois transformada em vetor.

```
public static double[][] potenciaMatrizAdjacencias(double[][] ramos, int grau) {
    double[][] resultado = ramos;
    for (int nGrau = 1; nGrau < grau; nGrau++) {
        resultado = multiplicarMatrizes(resultado, ramos);
    }

    return resultado;
}
```

Calcula o número de caminho de comprimento k entre quaisquer dois nós.

```
public static double calcCentralidadeVetorProprio(double[][] matAdjacencias) {

    Matrix adjacencias = new Basic2DMatrix(matAdjacencias);

    EigenDecompositor eigenD = new EigenDecompositor(adjacencias);

    Matrix[] matrizPropria = eigenD.decompose();

    double matVecProprio[][] = matrizPropria[0].toDenseMatrix().toArray();
    double matValProprio[][] = matrizPropria[1].toDenseMatrix().toArray();


    int posMaiorValProprio = 0;
    double maiorValProprio = matValProprio[0][0];

    for (int i = 1; i < matValProprio.length; i++) {
        if (matValProprio[i][i] > maiorValProprio) {
            maiorValProprio = matValProprio[i][i];
            posMaiorValProprio = i;
        }
    }
    double centralidade = matVecProprio[0][posMaiorValProprio];
    for (int i = 1; i < matVecProprio.length; i++) {
        if (matVecProprio[i][posMaiorValProprio] > centralidade) {
            centralidade = matVecProprio[i][posMaiorValProprio];
        }
    }

    return centralidade;
}
```

Este método calcula a influência de um nó numa rede.

4.3 Classe LeitorFicheiros

A classe LeitorFicheiros contêm o método lerMatriz, que carrega os dados de um ficheiro, cujo nome está guardado na variável nomeFich e coloca-o numa matriz, onde cada linha da matriz corresponde a uma linha do ficheiro de texto e cada elemento de uma coluna esta separado por um certo caractere Config.Separador_Dados (caractere que pode ser trocado nas configurações do programa).

```
public class LeitorFicheiros {

    public static int lerMatriz(String nomeFich, int nCamposInfo, String[][] matriz, int maxValue) throws FileNotFoundException {

        if (new File(nomeFich).exists()) {
            int pos=0;
            try (Scanner fInput = new Scanner(new File(nomeFich))) {
                int nLinha = 0;
                while (fInput.hasNextLine()) {
                    String linha = fInput.nextLine();
                    nLinha++;
                    if ((linha.trim()).length() > 0 && nLinha > 1 && pos < maxValue) {
                        String[] temp = linha.split(Config.SEPARADOR_DADOS);
                        for (int i = 0; i < nCamposInfo; i++) {
                            matriz[pos][i] = temp[i];
                        }
                        pos++;
                    }
                }
            }
            return pos;
        }
        return 0;

    }
}
```

4.4 Package Testes

Criamos um package de testes, para testarmos as funcionalidades da classe CalculosMatrizes. Para cada ação inserimos determinados dados e, sabendo o resultado esperado, comparamos os valores obtidos pelo programa.

```
System.out.println("Teste do método dos vetores próprios");

double[][] vetoresProprios = CalculosMatrizes.calcularVetoresProprios(matComparacao);
System.out.println(vetoresProprios[0].length);
for (int i = 0; i < vetoresProprios[0].length; i++) {
    for (int j = 0; j < vetoresProprios.length; j++) {
        System.out.print(vetoresProprios[i][j] + "  ");
    }
    System.out.println();
}

System.out.println("Teste cálculo do valor próprio máximo");
double maxValProprio = CalculosMatrizes.calcMaxValProprio(matComparacao);
System.out.println("Valor esperado: 1");
System.out.println("Resultado: " + maxValProprio + "\n");
```

Exemplo de um teste de um método da classe CalculosMatrizes

4.5 Segunda Iteração (Upgrade do Main – Novas funcionalidades)

Na 2ªa iteração, foi necessário fazer uma adaptação ao código para que este suportasse as novas funcionalidades.

```
boolean orientedNetwork = false;
String[][] infoNos = new String[Config.N_MAX_NOS][Config.N_CAMPOS_INFO];
int nosLength = LeitorFicheiros.lerMatriz(args[args.length - 2], Config.N_CAMPOS_INFO, infoNos, Config.N_MAX_NOS);
infoNos = adaptarMatrizNos(infoNos, nosLength);
nosLength--;
String[][] ramos = new String[Config.N_MAX_NOS][Config.N_CAMPOS_RELACOES];
int ramosLength = LeitorFicheiros.lerMatriz(args[args.length - 1], Config.N_CAMPOS_RELACOES, ramos, Config.N_MAX_RAMOS);
System.out.println(ramos[0][0].trim());
if (ramos[0][0].trim().equals("networkType:oriented")) {
   orientedNetwork = true;
} else if (ramos[0][0].trim().equals("networkType:nonoriented")) {
   orientedNetwork = false;
} else {
   System.out.println("O ficheiro não indica o tipo de network do grafo");
   ramosLength = 0;
}
```

```
} else if (orientedNetwork) {
   outputGrausEntrada(output, Matriz, infoNos);
   outputGrausSaida(output, Matriz, infoNos);
   if (args.length >= 7) {
      if ((args[1].equals("-k") && args[3].equals("-d")) || (args[1].equals("-d") && args[3].equals("-k"))) {
         int nIteracoes;
         double dampingFactor;
         if ((args[1].equals("-k") && args[3].equals("-d"))) {
            nIteracoes = Integer.parseInt(args[2]);
            dampingFactor = Double.parseDouble(args[4]);
            outputPageRank (output, Matriz, dampingFactor, nIteracoes, infoNos);
         } else if (args[1].equals("-d") && args[3].equals("-k")) {
            nIteracoes = Integer.parseInt(args[4]);
            dampingFactor = Double.parseDouble(args[2]);
            outputPageRank (output, Matriz, dampingFactor, nIteracoes, infoNos);
         }else{
            System.out.println("Não é possível apresentar o page rank. Verifique os argumentos introduzidos");
         }
```

O metodo main teve de ser reestruturado, para identificar se o ficheiro de input contem uma matriz orientada ou não orientada e, consequentemente, apresentar ao utilizador o menu com as funcionalidades disponíveis para cada tipo de matriz.

```
public static int MenuOriented() {
   System.out.println("Escolha uma opção");
   System.out.println("1- Graus de entrada dos nós ");
   System.out.println("2- Graus de saída dos nós");
   System.out.println("3- PageRank");
   System.out.println("0-Cancelar");
```

4.6 - PageRank

Também foi acrescentado ao metodo main a função de mandar o output dos resultados obtidos caso se trate de uma matriz ordenada para um ficheiro de texto. Caso seja essa a intenção do utilizador, deverá escrever "-k" seguido do numero de iterações e "-d" seguido do valor do camping factor. A ordem das duas letras pode ser trocada sem o aparecimento de um erro.

```
public static double[][] calcPageRank(double[][] matriz, double dampingFactor, int nIteracoes) {
    double[][] matrizA = criarMatrizAPageRank(matriz);
    double[][] matrizM = criarMatrizMPageRank(matrizA, dampingFactor);
    double[][] pageRank = new double[matriz.length][Config.LENGTH_VETOR_PAGERANK];
    for (int i = 0; i < pageRank.length; i++) {
        pageRank[i][Config.LENGTH_VETOR_PAGERANK] = 1;
    }
    for (int i = 0; i < nIteracoes; i++) {
        pageRank = multiplicarMatrizes(matrizM, pageRank);
    }
    return pageRank;
}
```

```
private static void outputPageRank(Formatter output, double[][] Matriz, double dampingFactor, int nIteracoes, String[][] infoNos) {
    double[][] pageRank = CalculosMatrizes.calcPageRank(Matriz, dampingFactor, nIteracoes);
    output.format("%16s%n", "Graus de entrada:");
    output.format("%10s%10s%n", "ID do nó", "Grau");
    for (int i = 0; i < pageRank.length; i++) {
        output.format("%10s%10d%n", infoNos[i][Config.POSICAO_ID_NOS], pageRank[i][Config.LENGTH_VETOR_PAGERANK]);
    }
}
```

```
public static double[][] criarMatrizMPageRank(double[][] matrizA, double dampingFactor) {
    double[][] matrizM = new double[matrizA.length][matrizA[0].length];
    for (int i = 0; i < matrizA.length; i++) {
        for (int j = 0; j < matrizA[i].length; j++) {
            matrizM[i][j] = (dampingFactor * matrizA[i][j]) + (1 - dampingFactor) / matrizA.length;
        }
    }
    return matrizM;
}
```

4.7 –Metodo para calcular o Grau de entrada e de saída

Adicionamos os métodos CalcularGrauSaida e CalcularGrauEntrada com o objetivo de obter o valor tanto do grau de saída como do grau de entrada de um determinado nó, calculando o numero de saídas/entradas e retornando uma variável com esse valor.

```
public static int calcularGrauSaida(double[][] matriz, int posicaoID) {
   int grauSaida = 0;

   for (int coluna = 0; coluna < matriz.length; coluna++) {

      if (matriz[posicaoID][coluna] != 0) {
         grauSaida++;
      }
   }

   return grauSaida;
}

public static int calcularGrauEntrada(double[][] ramos, int posicaoID) {
   int grauEntrada = 0;

   for (int linha = 0; linha < ramos[0].length; linha++) {

      if (ramos[linha][posicaoID] != 0) {
         grauEntrada++;
      }
   }

   return grauEntrada;
}
```

```
public static int calcularGrauEntrada(double[][] ramos, int posicaoID) {
    int grauEntrada = 0;

    for (int linha = 0; linha < ramos[0].length; linha++) {

        if (ramos[linha][posicaoID] != 0) {
            grauEntrada++;
        }
    }

    return grauEntrada;
}
```

6. Conclusão

Por fim, a realização deste trabalho foi bastante positiva visto que nos proporcionou uma nova dinâmica de trabalho de equipa e ajudou nos a melhorar certos aspetos e habilidades do trabalho em grupo. Aprendemos também o que é e como é analisar uma rede social, já que isto é algo novo para nós. Achamos que conseguimos atingir os objetivos propostos, visto que conseguimos solucionar o problema que nos foi apresentado. Penso que não ficamos aquém das expectativas do cliente.

7. Bibliografia

- Moodle ISEP - https://moodle.isep.ipp.pt/course/view.php?id=6432
- Google - https://www.google.com/
- Youtube (Compreender partes de código) - https://www.youtube.com/?gl=PT&hl=pt-PT
- Livro "Social Media Mining: An Introduction" Ficheiro - https://moodle.isep.ipp.pt/pluginfile.php/248728/mod_resource/content/2/Book_SNA.pdf
- Artigo "An Overview of Social Network Analysis" Ficheiro - https://moodle.isep.ipp.pt/pluginfile.php/248729/mod_resource/content/1/artigo_JoaoGama.pdf
- Descrição do algoritmo Page Rank a implementar (Iteração 2) - https://moodle.isep.ipp.pt/pluginfile.php/250048/mod_resource/content/1/pagerank.pdf