

Licenciatura em Engenharia Informática
ESINF 2020/2021

Trabalho Prático 2

Autores:

[1190402](#) António Fernandes

[1191045](#) Rui Soares

Turma: 2DK

Data: 30/11/20

Docente: Ana Madureira [AMD](#)

Índice:

Diagrama de Classes	3
Análise de Complexidade das Funcionalidades Implementadas	4
Melhoramentos Possíveis	5

[illegible]

Análise de Complexidade das Funcionalidades Implementadas

```
public List<String> getMutualFriendsInNPopularUsers(int n)
```

Devolve os amigos em comum entre os n utilizadores mais populares da rede social.

```
Private List<UsersQuantityOfFriendships> getNElements(List<UsersQuantityOfFriendships> usersTotalFriendships, int n)
```

Devolve os n elementos mais populares da rede social.

```
private List<String> getMutualFriends(List<UsersQuantityOfFriendships> popularFriends)
```

Devolve os amigos em comum entre os n utilizadores mais populares.

```
public int verifyFriendshipNet() throws CloneNotSupportedException
```

Verifica se o grafo da rede de amizades é conexo.

```
public int getShortPath()
```

Devolve o menor caminho.

```
public ArrayList<String> returnFriends(String user, int nBorders)
```

Devolver para um utilizador os **amigos que se encontrem nas proximidades**, isto é, amigos que habitem em cidades que distam um dado **número de fronteiras** da cidade desse utilizador. Devolver para cada cidade os respetivos amigos.

```
public static ArrayList<String> iterableToArray(Iterable<String> it)
```

Transforma um Iterable num Array.

```
public String userToCountry(String user)
```

Retorna o cidade do utilizador.

```
public ArrayList<String> returnNCities(int n, int p)
```

Devolver as **n cidades com maior centralidade** ou seja, as cidades que em média estão mais próximas de todas as outras cidades e onde habitem pelo menos **$p\%$** dos utilizadores da rede de amizades, onde $p\%$ é a percentagem relativa de utilizadores em cada cidade.

```
private void getAverageDistance(double[] averageDistance, List<Country> allRegisteredCities)
```

Devolve a distância media para todos os países.

```
public double averageDistance(Country country)
```

Calcula a distância media a outras cidades de um dado país.

```
private void getPercentageCity(double[] percentageUsersPerCity, List<Country> allRegisteredCities)
```

Calcula a percentagem de utilizadores por cidade.

```
private void insertInClass(List<Country> allRegisteredCities, double[] averageDistance, double[] percentageUsersPerCity, ArrayList<CountryDistancePercentage> cdp)
```

Cria objetos da classe CountryDistancePercentage e adiciona-os a uma lista.

```
public double returnLandPath(String s1, String s2, int n)
```

Devolver o **caminho terrestre mais curto** entre dois utilizadores, passando obrigatoriamente pelas **n cidade(s) intermédias** onde cada utilizador tenha o **maior número de amigos**. Note que as cidades origem, destino e intermédias devem ser todas distintas. O caminho encontrado deve indicar as cidades incluídas e a respetiva distância em km.

```
private void returnCitiesMostFriends(String user, String city1, String city2, int n, List<String> citiesBetweenFriends)
```

Retorna a cidade onde o utilizador tem mais amigos.

```
private Map<String, List<String>> groupFriendsPerBorder(List<String> friendsCities, List<String> listFriends)
```

Agrupar amigos por fronteira.

```
private List<String> friendsOfCity(List<String> listFriends, String friendsCity)
```

Retorna os amigos na cidade.

Análise de complexidade:

getMutualFriendsInNPopularUsers-> $O(n^3)$

getMutualFriends-> $O(n^3)$

verifyFriendshipNet->Não determinístico: $O(n^2)$ se grafo nulo e $O(n^3)$ se grafo não nulo

returnFriends-> $O(n^3)$

returnNCities-> $O(n^2)$

returnLandPath-> $O(n^4)$

Melhoramentos Possíveis

Alguns melhoramentos que seriam talvez possíveis seriam melhorar a complexidade dos algoritmos que produzimos.

O relatório deverá servir de ferramenta de avaliação posterior à apresentação. Nele devem apresentar as classes definidas, **análise de complexidade** de todas as funcionalidades implementadas e melhoramentos possíveis.