DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

1º Semestre 2021/2022

Rede de Transporte de Mercadorias

Estruturas de Informação

O presente projeto envolve a criação de uma biblioteca de classes respetivos métodos e testes relativos a uma **rede de transporte de mercadorias** entre vários países.

Esta rede de transporte inclui ligações terrestres entre as capitais dos países e ligações marítimas entre os portos dos países[1].

```
countries.txt: continent, alpha.2.code, alpha.3.code, country, population, capital,
               latitude, longitude
ports.csv: continent, country, port_id, port_name, latitude, longitude
borders.txt: country1, country2
seadists.txt: from_country, from_port_id, from_port, to_country, to_port_id, to_port,
              sea_distance
```

O transporte marítimo é assegurado através de diferentes navios que enviam mensagens durante as viagens à guarda costeira dos U.S. O ficheiro (ships.csv) contém informação estática sobre os detalhes do navio e dados dinâmicos relativos à viagem. Os dados relativos à viagem são mensagens AIS (Automatic Identification System) que contêm a informação posicional e de deslocamento de cada navio num determinado instante[2].

Campos de **dados estáticos** de detalhe de um navio:

`MMSI`: código único de 9 dígitos de identificação do navio

`VesselName`: nome do navio

`IMO`: número de identificação internacional único de 7 dígitos, que permanece inalterado após a transferência do registo do navio para outro país

`CallSign`: indicativo único do navio

`VesselType`: tipo de navio, codificado numericamente, ver aqui detalhes

`Length`: comprimento do navio, em metros

`Width`: largura do navio, em metros

`Draft`: distância vertical entre a linha d'água e o fundo do casco do navio, em metros. Varia com a carga do navio e a densidade da água

Campos de **dados dinâmicos** relativos aos dados de posicionamento de um navio:

`BaseDateTime`: data / hora da mensagem AIS

`LAT`: latitude do navio (em graus: [-90; 90], valor negativo representa Sul, 91 indica 'não disponível')

[1] https://sea-distances.org

[2] https://www.youtube.com/watch?v=3M7mxBimT70

`LON`: longitude do navio (em graus: [-180; 180], valor negativo representa Oeste, 181 indica 'não disponível')

`SOG`: velocidade sobre o solo

`COG`: curso sobre o solo, direção relativa ao Norte absoluto (em graus: [0; 359])

`Heading`: rumo do navio (em graus: [0; 359], 511 indica 'não disponível')

`Cargo`: código do navio em reboque

`TranscieverClass`: classe to *transciever* utilizado no envio dos dados

Com recurso à **classe BST** crie um conjunto de classes que permitam da forma **mais eficiente**:

1. Guardar os dados estáticos e dinâmicos dos navios. As classes devem permitir pesquisa dos detalhes de um navio através de qualquer um dos seus códigos: MMSI, IMO ou CallSign. Os dados de viagem devem estar organizados temporalmente e associados a cada um dos navios de modo a ser possível aceder eficientemente a qualquer um do(s) valore(s) de posicionamento de um navio numa determinada data ou período de tempo.

2. Fazer um sumário dos movimentos de um navio. Para um determinado navio devolver numa estrutura adequada um dos seus códigos (MMSI, IMO ou CallSign), VesselName, VesselType, BaseDateTime inicial, BaseDateTime final, tempo total dos movimentos, número total de movimentos, MaxSOG, MeanSOG, MaxCOG, MeanCOG, DepartureLatitude, DepartureLongitude, ArrivalLatitude, ArrivalLongitude, TraveledDistance (soma incremental da distância entre cada mensagem de posicionamento) e DeltaDistance (distância linear entre as coordenadas do primeiro e último movimento)[3].

3. Devolver para todos os navios o MMSI, o número total de movimentos, TraveledDistance e DeltaDistance ordenado por TraveledDistance (ordem decrescente) e número total de movimentos (ordem crescente).

4. Obter os top-N navios com mais quilómetros percorridos (TraveledDistance) e respetiva velocidade média (MeanSOG) num período tempo (BaseDateTime inicial/final) agrupado por tipo de navio (VesselType).

[3] http://www.movable-type.co.uk/scripts/latlong.html

5. Devolver os pares de navios com rotas com coordenadas de partida/chegada próximas (ambas não distem mais do que 5 Kms) e com diferentes TraveledDistance. Ordenado pelo código MMSI do 1º navio e por ordem decrescente da diferença de TraveledDistance.

   Nota: Não considerar navios com TraveledDistance inferior a 10 kms.

   Exemplo:

Ship1 MMSI	Ship2 MMSI	distOrig	distDest	Movs	TravelDist	Movs	TravelDist
366759530	366772990	1.595	1.595	1217	78.724	1251	360.596
366759530	366772760	1.595	1.595	1217	78.724	1237	344.448
366759530	366866930	0.041	0.041	1217	78.724	1213	164.209
366759530	367439390	1.962	1.962	1217	78.724	1232	138.238
366759530	368045710	3.788	3.788	1217	78.724	1205	24.244
563076200	636015975	2.431	2.431	808	220.471	836	153.906
366866930	368045710	3.829	3.829	1213	164.209	1205	24.244
366866930	367439390	2.0	2.0	1213	164.209	1232	138.238
…..							

Sprint 2

1. Criar uma **BST 2D-tree** com as localizações dos portos (`ports.csv`).
2. Encontrar o porto mais próximo de um navio dado o seu CallSign, numa determinada data

Normas

- O trabalho deverá ser realizado em grupo (mesmo grupo de LAPR3).
- O projeto tem de ser desenvolvido em Java e todas as funcionalidades testadas através de testes unitários usando os ficheiros de teste disponibilizados, versão small e big. É obrigatório o uso de uma ferramenta de controle de versões.
- A avaliação do trabalho será feita principalmente em função das classes propostas, nomeadamente em termos da sua **conformidade com o Paradigma POO** e a **eficiência do código implementado** na realização das funcionalidades solicitadas.
- No relatório devem justificar o diagrama de classes definido, fazer a análise de complexidade das funcionalidades implementadas, e indicar a contribuição de cada membro do grupo nas funcionalidades (código/teste/análise de complexidade).