Licenciatura em Engenharia Informática
ESINF – 2020/2021

Relatório Estruturas de Informação

Projeto 1 – Países/Covid-19

Autores:
[1191097] [Tiago Machado]
[1191111] [Tomas Flores]

Turma: 2DL

Data: [30/11/20]

Introdução

Foi-nos proposto, para primeiro projeto da disciplina de Estruturas de Informação, o desenvolvimento de uma aplicação que consiga gerir os dados de vários países em relação á pandemia que vivemos nos tempos de hoje. Para tal, aplicamos os conhecimentos adquiridos nesta disciplina sobre a Java Collections Framework para dar uma resposta mais eficiente aos requisitos do projeto.

1º Requisito - Carregar e guardar a informação relativa aos países e respetivos dados da pandemia COVID-19

Para este primeiro requisito decidimos armazenar a informação recolhida do ficheiro Excel que nos foi disponibilizado em 2 classes: Pais e Casos. Para atingir uma organização eficiente recorremos a *Maps* que, através da indexação de objetos a *keys*, conseguimos obter uma pesquisa mais eficiente. Usamos então o Pais como *key* para manipular os vários casos de determinado país. Dessa maneira, uma vez escolhido um país conseguiríamos facilmente obter todos os dados relativos á pandemia sobre o mesmo.

Para o desenvolvimento deste requisito tivemos que escolher também a melhor implementação do *Map* para este tipo de projeto. Por um lado, tínhamos o *TreeMap*, que mantém as entradas ordenadas de acordo com a ordem natural das suas *keys*, por outro tínhamos o *HashMap*, onde cada objeto tem um *hash code*, que ajuda os programas a correr mais rapidamente, sendo as entradas ordenadas de acordo com este. Verificamos também que o *HashMap* é normalmente mais rápido e fornece desempenho de tempo constante médio O(1) para operações básicas como *get()* e *put(),* sendo que o *TreeMap* para estas mesmas operações proporciona um desempenho de tempo O(log (n)). Visto que neste projeto trabalhamos com uma lista de dados extensa optamos por utilizar o HashMap. Para uma melhor organização entre continentes, alteramos o *hash code* dos países para usar o nome dos continentes em que se encontram, assim a pesquisa de dados entre continentes será muito mais rápida e eficiente.

Por fim, uma vez concluído o método *lerFicheiro(),* os dados serão todos organizados pelos continentes dos Paises, onde será possível, através do pais como key, aceder aos dados dos vários casos desse pais.

2º Requisito - Apresentar uma lista de países ordenados por ordem crescente do número mínimo de dias que foi necessário para atingir os 50.000 casos positivos.

Para este requisito, temos o método *numMinDias()*, que vai criar um *HashMap*, em que a chave é um País e o *value* é um *Arraylist* de *Integers*, com uma posição e um dia. Neste requisito, a lista a ser apresenta tem de estar ordenada, o que tornava o uso de um *TreeMap* a opção mais evidente, contudo, como a ordenação é feita pelo *value* e não pela *key* é utilizado um *HashMap,* com a ordenação a ser feita por outro método.
Para cada país, vamos percorrer a lista de casos que se encontra dentro do mapa criado no requisito 1, e vamos localizar a posição na lista em que os casos passam os 50 mil. O método que faz esta pesquisa é o *binarySearch()*. Foi usado o método de pesquisa de *binarySearch* devido ao seu pior tempo logarítmico de O(log n).
Com a posição, o método vai buscar a data à lista de casos, e calcula o número de dias necessários para atingir 50 mil casos. Depois, a posição e o número de dias são adicionados a um *Arraylist* de *Integers* (a posição é guardada pois é necessária para a

impressão de certos valores que são necessários apresentar), e este *array* é por sua vez adicionado ao *HashMap* criado no início do método, sendo o *array* o valor e a chave o pais associado a estes valores.
Depois do *HashMap* estar preenchido, é feita uma ordenação através do método *sortByValue,* que vai ordenar por valor e guardar os dados guardados num *LinkedHasHMap*, de modo a preservar a ordenação. Este novo *LinkedHashMap* é utilizado para apresentar os valores.

3. Requisito - Devolver o total de novos_casos/novas_mortes por continente/mês, ordenado por continente/mês.

Para este requisito, percorremos a lista dos vários países onde, para cada país, através de um *ArrayList,* armazenamos todos os novos casos e novas mortes para todos os meses de dados do continente do pais. Á medida que se percorrem os dados, através de um TreeMap criado anteriormente, conseguimos adicionar ao total de novos casos e novas mortes de um continente por mês o total de novos casos e novas mortes por mês dos vários países.

Assim, percorremos a lista dos países de forma eficiente, necessitando apenas de os percorrer uma vez para conseguir devolver os dados pretendidos.

4. Requisito - Devolver para cada dia de um determinado mês e para um dado continente, os países ordenados por ordem decrescente do número de novos casos positivos.

Neste requisito, temos o método *casosPositivosNumContinente()*, que vai criar um *HashMap*, em que a *key* é um *Integer* que representa um dia, e o *value* é um outro *HashMap,* que por sua vez tem um país como *key* e um *Integer* que representa o novo número de casos nesse dia. Mais uma vez não é utilizado um TreeMap pela necessidade de ordenar o *Map* pelo *value* e não pela *key*.

O método obtém o primeiro dia com novos casos registados, do mês pedido, para cada país.

Para cada dia do mês vai ser guardado um *HashMap*, com pais como *key*, e novo número de casos como *value*. Este *HashMap* é guardado como *value* no *map* inicial, com a *key* associada sendo o dia em questão.
Por fim, o *HashMap* contido no *map* inicial é ordenado e convertido para *LinkedHashMap* de modo a preservar a ordenação.

5. Requisito - Devolver numa estrutura adequada, todos os países com mais de 70% de fumadores, ordenados por ordem decrescente do número de novas mortes.

Por fim, para o requisito 5, percorremos a lista dos vários países, onde para cada pais obtemos a percentagem do seu total de fumadores e se esta for mais que 70%, dados como o nome do país, a sua percentagem total de fumadores e o seu total de novas mortes serão copiados para um ArrayList.

Mais tarde, recorrendo ao algoritmo não determinístico *Quicksort*, que no melhor caso terá uma complexidade O(n log(n)) e no pior $O(n^2)$, será feito uma ordenação decrescente deste ArrayList em função do número de novas mortes.

Uma vez concluída a ordenação, o ArrayList devolverá os paises com mais de 70% e os respetivos dados.

Diagrama de Classes do projeto