ATAD 2022/23

Algoritmos e Tipos Abstratos de Dados

RELATÓRIO DE PROJETO

Análise de Terramotos

(Processamento Estatístico e Visualização de Dados)

GRUPO		
Número	**Nome**	**Leader (X)?**
202002037	David Saroco (EI-03 – Miguel Bugalho)	
201701984	Inês Palet (EI-05 - Miguel Bugalho)	
202002358	Marcel Becheanu (EI-03 – Miguel Bugalho)	X
202000905	Mauro Amaro (EI-01 – Bruno Silva)	

Data: junho/2023

ÍNDICE

1. INTRODUÇÃO

Durante o segundo semestre do segundo ano da Licenciatura em Engenharia Informática, foi-nos proposto o desenvolvimento de um projeto em linguagem C com o objetivo de extrair/apresentar informações úteis a partir de arquivos que contêm dados sobre sismos naturais e artificiais ocorridos em vários países.

O projeto consiste, essencialmente, num interpretador de comandos que permite ao utilizador obter diversos tipos de informações, principalmente informação estatística sobre terramotos em diferentes territórios/países.

O programa faz uso de ficheiros .csv que armazenam os dados referentes aos respetivos terramotos, estatísticas e localizações, e a partir daqui, tendo em conta determinadas estruturas de dados e ADT's definidos pelos docentes da unidade curricular, foi-nos pedido para desenvolver uma solução que permita manipular e visualizar a informação fornecida.

2. DIVISÃO DE TAREFAS

Distribuição tarefas (final)

Tarefa/ Funcionalidade	**Estudante**
Comando LOADCL	Mauro Amaro
Comando LOADEA	Marcel Becheanu + Inês Palet
Comando LOADST	Mauro Amaro
Comando CLEAR	Mauro Amaro
Comando QUIT	Mauro Amaro
Comando SHOWALL	Mauro Amaro
Comando SHOW_Y	Mauro Amaro
Comando SHOW_T	Mauro Amaro + Inês Palet
Comando SHOW_YT	David Saroco
Comando LIST_T	Marcel Becheanu
Comando COUNT	Marcel Becheanu
Comando HISTOGRAM	Marcel Becheanu
Comando COUNTRY_S	Marcel Becheanu
Comando REGION_AVG	Mauro Amaro + Inês Palet
Comando TOPN	Marcel Becheanu
Complexidades Algorítmicas	David Saroco
Algoritmos (Pseudo-Código)	Mauro Amaro

Auto-avaliação de participação (%) no projeto

Estudante	**Participação (%)**
David Saroco	10%
Inês Palet	20%
Marcel Becheanu	35%
Mauro Amaro	35%
TOTAL	**100**

3. DESCRIÇÃO DE ADTs UTILIZADOS

ADT LIST:

Política de acesso.

A estrutura de dados List permite o acesso aos elementos armazenados por meio de ranks. Os elementos podem ser acedidos, inseridos ou removidos em qualquer posição da lista. Essa característica torna a List adequada para cenários em que são esperadas modificações frequentes.

Especificação (.h).

A especificação para o ADT List inclui operações como criação da lista, inserção de elementos, remoção de elementos, obtenção do tamanho da lista, acesso aos elementos por ranks, entre outros.

Implementação utilizada.

A escolha dessa implementação se baseou na necessidade de suportar acesso eficiente aos elementos por rank, bem como inserções e remoções em qualquer posição da lista com complexidade temporal média de O(1).
A ArrayList é capaz de alocar um espaço contíguo na memória para armazenar os elementos, o que permite acesso rápido aos índices. Além disso, ela pode crescer dinamicamente à medida que novos elementos são adicionados.

ADT MAP:

Política de acesso.

O ADT Map permite o acesso aos elementos armazenados através de chaves únicas. Cada elemento é mapeado para uma chave específica, permitindo a recuperação eficiente do valor associado a essa chave.

Especificação (.h).

A especificação para o ADT Map inclui operações como inserção de elementos com chave-valor, remoção de elementos, obtenção do valor associado a uma chave, verificação da existência de uma chave, entre outros.

Implementação utilizada.

Na compilação do programa, utilizamos uma implementação de ADT MAP baseada em ArrayList para o ADT Map. Essa escolha foi feita com base

na necessidade de suportar acesso rápido e eficiente aos elementos por meio das chaves únicas, bem como inserções e remoções eficientes. A implementação baseada em ArrayList utiliza uma estrutura de array para armazenar os pares chave-valor, permitindo um acesso rápido e direto aos elementos por índices.

4. COMPLEXIDADES ALGORÍTMICAS

Neste tópico é apresentada a complexidade algorítmica das implementações de cada comando implementado.

LOADEA

Implementação	Complexidade Algorítmica
countryLocationsArraySize()	O (1), pois todas as operações realizadas têm uma complexidade constante.
loadEarthquakes()	O(n), onde "n" representa o número de linhas no arquivo "earthquakes.csv".
listSize()	O (1), pois todas as operações realizadas têm uma complexidade constante.

LOADCL

Implementação	Complexidade Algorítmica
loadCountryLocations()	O(n), onde "n" representa o número de linhas no arquivo "world_country_locations.csv".
countryLocationsArraySize()	O (1), pois todas as operações realizadas têm uma complexidade constante, independentemente do tamanho do array, a função 'countryLocationsArraySize' verifica se o ponteiro array é nulo, atribui o tamanho do array à variável apontada por 'ptSize' e retorna o código de status 'ARRAY_OK'.

LOADST

Implementação	Complexidade Algorítmica
loadCountryStatistics()	O(n), onde "n" representa o número de linhas no arquivo "world_country_statistics.csv".
mapSize()	O (1), pois todas as operações realizadas têm uma

	complexidade constante, a função 'mapSize' verifica se o ponteiro 'map' é nulo, atribui o tamanho do mapa à variável apontada por 'ptSize' e retorna o código de status MAP_OK.

SHOWALL

Implementação	Complexidade Algorítmica
printEarthquakes()	O(n), onde "n" é o número de registos na lista de terremotos (list).

SHOW_Y

Implementação	Complexidade Algorítmica
showEarthquakesbyYear()	$O(n^2)$, devido a possíveis iterações alinhadas na função 'filterEarthquakesByYear', que podem levar a uma complexidade quadrática.

SHOW_T

Implementação	Complexidade Algorítmica
showEarthquakesByCountry()	O(n), onde n é o tamanho da lista original de terremotos 'earthquakes'.

LIST_T

Implementação	Complexidade Algorítmica
displayCountriesWithEarthquake()	O(n), onde "n" é o tamanho da lista de terremotos.

COUNT

Implementação	Complexidade Algorítmica
countEarthquake()	O(n), onde "n" é o tamanho da lista de terremotos.

HISTOGRAM

Implementação	Complexidade Algorítmica
showHistogram()	O(n^2), onde "n" é o número de países, o código solicita ao utilizador o número de países e seus códigos, calcula o número de terremotos em diferentes faixas de magnitude para cada país selecionado e exibe os resultados.

COUNTRY_S

showCountryStatisticsByOrder()	O(n), onde "n" é o número de países no mapa.

REGION_AVG

calculateStatistics()	O(n), a função e suas subfunções apenas depende de "n".

TOPN

showTopEarthquakeData()	O(n^2), as suas funções encadeiam ciclos tendo como complexidade máxima (n^2);

5. ALGORITMOS

Neste ponto do relatório são apresentados os respetivos pseudo-códigos de 3 funcionalidades (1 de cada categoria entre **B, C e D**), que podem ser visualizados de seguida:

Pseudocódigo (B - ShowAll)

```
Algorithm printEarthquakes
    input: list - a list of earthquake records
    output: (prints earthquake records with pagination)
BEGIN
    size <- 0
    listSize(list, &size)

    currentPage <- 0
    pageSize <- 50
    WHILE currentPage * pageSize < size
    DO
        PRINT "Number of records found: $size"

        PRINT "%-4s %-15s %-11s %-10s %-9s %-12s %-16s %-6s %-10s %s",
            "\nID", "Country Code", "Date", "Time", "Latitude", "Longitude", "Type", "Depth",
"Magnitude", "Magnitude Type"
        PRINT "-------------------------------------------------------------------------------------------------"

        // Print earthquake records for the current page
        FOR i <- currentPage * pageSize TO ((currentPage + 1) * pageSize) - 1 DO
            IF i >= size THEN
                BREAK
            END IF

            Earthquake earthquake
            listGet(list, i, &earthquake)

            // Print earthquake data
            listElemPrint(earthquake)
        END FOR

        PRINT "\n1. Next 50\n2. Return\nOPTION> "

        choice <- 0
        IF scanf("%d", &choice) != 1 THEN
            BREAK
        END IF

        IF choice == 2 THEN
            BREAK
        END IF

        currentPage++

    END WHILE
    END
```

Pseudocódigo (C - Histogram)

```
Algorithm showHistogram
    input: list - a list of earthquakes
    output: (prints histogram)
BEGIN
    numOfCountries <- 0
    PRINT " Por favor, insira quantos países deseja utilizar no histograma (máximo 3):"
    READ numOfCountries
    listOfCodesOfCountries <- allocate memory for an array of strings with size
numOfCountries

FOR i <- 0 TO numOfCountries-1 DO
  listOfCodesOfCountries[i] <- allocate memory for a string of size 4
  PRINT " "Por favor, insira o código do $i + 1 país: ": "
  READ listOfCodesOfCountries[i]
END FOR

countMagnitude <- allocate memory for a 2D array of integers with size [numOfCountries][6]

// Initialize the countMagnitude array with zeros
SET all elements of countMagnitude to 0

FOR i <- 0 TO numOfCountries-1 DO
  calculateNumberEarthquakeMagnitude(list, listOfCodesOfCountries[i], countMagnitude[i])
END FOR

ranges <- array of strings containing the magnitude ranges

PRINT "Código | Magnitude | Número de Terramotos (escala logarítmica)"
PRINT "-------------------------------------------------------------"

FOR i <- 0 TO 5 DO
  FOR j <- 0 TO numOfCountries-1 DO
    symbols <- getMagnitudeSymbol(countMagnitude[j][i])
    PRINT listOfCodesOfCountries[$j], "|", ranges[$i], "|", symbols, countMagnitude[$j][$i]
    DEALLOCATE symbols
  FOR END

  PRINT "-------------------------------------------------------------"

FOR END

// Cleanup
FOR i <- 0 TO numOfCountries-1 DO
  DEALLOCATE listOfCodesOfCountries[i]
END FOR

DEALLOCATE listOfCodesOfCountries

    END
```

Pseudocódigo (D - TOPN)

```
Algorithm showTopEarthquakeData
      input: locations - an array of country locations
      earthquakes - a list of earthquake data
      output: (prints top earthquake data by country)
BEGIN

sizeOfCountries <- 0
sizeOfEarthquake <- 0

IF countryLocationsArraySize(locations, &sizeOfCountries) != ARRAY_OK OR
sizeOfCountries == 0 OR listSize(earthquakes, &sizeOfEarthquake) != LIST_OK OR
sizeOfEarthquake == 0 THEN
   PRINT "Please load 'country location data' and 'earthquake data' first."
   RETURN
END IF

numCountriesToShow <- 0

WHILE numCountriesToShow <= 0 OR numCountriesToShow > sizeOfCountries DO
   PRINT "Insira o valor de N (máximo", sizeOfCountries, "):"
   readInteger(&numCountriesToShow)

   IF numCountriesToShow > 0 AND numCountriesToShow <= sizeOfCountries THEN
      BREAK
   END IF

   PRINT "N tem que ser maior que zero e menor ou igual a", sizeOfCountries, "."
END WHILE

earthquakeData <- allocate memory for an array of strings with size

FOR I <- 0 TO sizeOfCountries –1 DO
strcpy(earthquakeData[i].code, locations->locations[i].code)
      strcpy(earthquakeData[i].territoryName, locations->locations[i].territoryName)
 END FOR

fillListWithEarthquakeByCountries(earthquakeData, earthquakes, sizeOfCountries)
sortEarthquakeDataByNumberOfEarthquakes(earthquakeData, sizeOfCountries)
printListWithEarthquakes(earthquakeData, numCountriesToShow)

DEALLOCATE earthquakeData
END
```

6. LIMITAÇÕES DO PROJETO

Como em qualquer projeto, fomos encontrando dificuldades pelo caminho, à medida que íamos desenvolvendo os comandos pedidos ou funções auxiliares que suportam esses comandos. Salientamos o desafio na implementação do comando **REGION_AVG, TOPN e COUNTRY_S,** comandos estes que embora nos tenham dado um pouco mais trabalho a desenvolver, acabámos por conseguir finalizá-los e torná-los funcionais.

Pensamos que o projeto não tenha nenhuma limitação em relação aos comandos, visto que conseguimos desenvolver todos estes e fizemos os respetivos testes (valgrind, algumas validações, etc) de modo a validar o seu bom funcionamento.

Verificamos também os resultados obtidos com os resultados disponibilizados pelos docentes, de forma a validar.

7. CONCLUSÕES

O desenvolvimento deste projeto foi uma experiência enriquecedora e desafiante que permitiu colocar em prática conceitos cruciais da unidade curricular de Algoritmos e Tipos Abstratos de Dados, de modo que os comandos pedidos fossem desenvolvidos e executados com sucesso.

Pensamos ter cumprido com grande parte dos objetivos propostos, seja na implementação dos comandos, como das funções auxiliares a estes, o que nos permitiu aprofundar as nossas competências baseadas na programação em linguagem C, bem como na definição e implementação das estruturas e ADT's relevantes, cruciais na nossa formação, não só académica, mas também profissional.