**ATAD 2022/23**

Algoritmos e Tipos Abstratos de Dados

LEI

RELATÓRIO DE PROJETO

Análise de Terramotos

(Processamento Estatístico e Visualização de Dados)



Turma: 8

Professor: Aníbal Ponte

|  |  |  |
| --- | --- | --- |
| **GRUPO** | | |
| **Número** | **Nome** | **Leader (X)?** |
| 202200278 | Bernardo Vaz |  |
| 202200354 | Tiago Ramada |  |
| 202200286 | Daniel Pais | X |
| 202200285 | Tomás Alves |  |

**Data:** 6 / 2023

ÍNDICE

[1. INTRODUÇÃO 3](#_Toc138066174)

[2. DIVISÃO DE TAREFAS 4](#_Toc138066175)

[Distribuição de tarefas (planeamento previsto) 4](#_Toc138066176)

[Distribuição tarefas (final) 4](#_Toc138066177)

[Auto-avaliação de participação (%) no projeto 4](#_Toc138066178)

[3. DESCRIÇÃO DE ADTs UTILIZADOS 5](#_Toc138066179)

[ADT List 5](#_Toc138066180)

[ADT Map 5](#_Toc138066181)

[4. COMPLEXIDADES ALGORÍTMICAS 6](#_Toc138066182)

[Comando LOADCL 6](#_Toc138066183)

[Comando LOADEA 6](#_Toc138066184)

[Comando LOADST 6](#_Toc138066185)

[Comando SHOWALL 6](#_Toc138066186)

[Comando SHOW\_Y 6](#_Toc138066187)

[Comando SHOW\_T 6](#_Toc138066188)

[Comando SHOW\_YT 6](#_Toc138066189)

[Comando LIST\_T 6](#_Toc138066190)

[Comando COUNT 6](#_Toc138066191)

[Comando COUNTRY\_S 6](#_Toc138066192)

[Comando TOPN 7](#_Toc138066193)

[5. ALGORITMOS 7](#_Toc138066194)

[SHOWALL 7](#_Toc138066195)

[Algoritmo earthquakeListShowAll 7](#_Toc138066196)

[SHOW\_Y 8](#_Toc138066197)

[Algoritmo earthquakeListGetYearCount 8](#_Toc138066198)

[Algoritmo earthquakeListShowYear 8](#_Toc138066199)

[SHOW\_T 9](#_Toc138066200)

[Algoritmo earthquakeListGetCountryCount 9](#_Toc138066201)

[Algoritmo earthquakeListShowCountry 9](#_Toc138066202)

[6. LIMITAÇÕES DO PROJETO 10](#_Toc138066203)

[7. CONCLUSÕES 11](#_Toc138066204)

# INTRODUÇÃO

Neste projeto foi-nos proposto a criação de um pequeno interpretador de comandos, tornando-se assim o nosso objetivo fazer a interpretação da escrita do utilizador e assim disponibilizar a informação pretendida acerca dos terramotos que foi o tema que nos foi fornecido.

Para o cumprimento dos requisitos foi necessário colocar em prática todos os conhecimentos aprendidos no decorrer desta UC e implementarmos o conjunto de ADTs aprendidos em aula, e são eles o ADT Map e ADT List que vão ser descritos no seguinte tópico.

Esperámo-nos deparar com algumas dificuldades o que foi o sucedido (mais desenvolvido na conclusão), mas mesmo assim com resiliência e muito esforço pretendemos ser bem-sucedidos.

# DIVISÃO DE TAREFAS

## Distribuição de tarefas (planeamento previsto)

(a validar pelo docente na primeira aula de apoio – vale 10% da Documentação)

|  |  |
| --- | --- |
| **Tarefa/**  **Funcionalidade** | **Estudante** |
| Histogram, ShowAll, Show\_Y, LoadEA, LoadSt, Region\_Avg | Bernardo Vaz |
| Histogram, Show\_T, Show\_YT, LoadCL, List\_T ,Region\_Avg | Tiago Ramada |
| Histogram, Clear, Country\_S, Count, Region\_Avg | Daniel Pais |
| Histogram, Quit, TopN, Region\_Avg | Tomás Alves |
| … |  |

## Distribuição tarefas (final)

|  |  |
| --- | --- |
| **Tarefa/**  **Funcionalidade** | **Estudante** |
| ShowAll, Show\_Y, Show\_T, Show\_YT, LoadEA, LoadCL Clear, Relatório | Bernardo Vaz |
| LoadEA, LoadSt, LoadCL, List\_T, Count, Show\_T, Show\_YT ,Quit | Tiago Ramada |
| Country\_S, LoadSt | Daniel Pais |
| TopN, LoadEA | Tomás Alves |
| … |  |

## Auto-avaliação de participação (%) no projeto

|  |  |
| --- | --- |
| **Estudante** | **Participação (%)** |
| Bernardo Vaz | 29 |
| Tiago Ramada | 31 |
| Daniel Pais | 20 |
| Tomás Alves | 20 |
| **TOTAL** | **100** |

# DESCRIÇÃO DE ADTs UTILIZADOS

## ADT List

Começamos pelo ADT List que tem uma política de acesso por ranks (índices), queremos dizer com isto que podemos aceder sendo assim a qualquer elemento no rank pretendido quando necessitarmos.

No nosso contexto utilizamos como pedido no enunciado para armazenar uma lista de elementos Earthquakes importados, onde cada elemento Earthquake vai conter a informação presente na estrutura Earthquake e são elas countryCode, date, time, latitude, longitude, type, depth, magnitude e magnitudeType.

Optamos por utilizar a implementação por base em ArrayList por nos sentirmos mais confortáveis na utilização desta forma de utilização ao invés da LinkedList.

Algumas das principais operações são a de adicionar/remover elementos e de retorno ou alteração do próprio elemento como dito anteriormente sempre por base em rank. Sendo que as de retorno e alteração (listGet (L, i) e listSet (L, i, e)) tem complexidades algorítmicas de O (1) e as de adicionar/remover (listAdd (L, i, e) e listRemove (L, i)) tem complexidade algorítmica de O(n).

## ADT Map

Já o ADT Map tem uma forma bastante singular de acesso e completamente diferente da lista, sendo que invés de aceder por índice esta, irá utilizar tuplos de chave-valor, isto é, vai conter chaves que estão associados a um valor sendo que não pode haver chaves repetidas. Assim podemos invés de aceder a um índice aceder a uma chave que posteriormente vai ter um valor.

No nosso contexto utilizamos como pedido no enunciado para armazenar neste caso valores do tipo CountryStatistic importados onde cada um destes valores irá ter a informação presente na estrutura CountryStatistic e são elas name, region, population, area, gdp\_capita, literacy, birthrate, deathrate, sendo que para chaves como não pode ser repetida utilizamos a StringWrap de modo a criar uma chave único que é o código do país. Tal como no ADT List preferimos utilizar por base em ArrayList devido as mesmas razões mencionadas.

Algumas das principais operações são a mapPut (M, key, value), mapRemove (M, key), mapGet (M, key) e mapContains (M, key), sendo que cada um destes métodos tem complexidade algorítmica de O (n).

# COMPLEXIDADES ALGORÍTMICAS

## Comando LOADCL

Este comando vai ter uma complexidade algorítmica de O (1) pois apenas tem um ciclo while, que não está dependente de nenhuma variável.

## Comando LOADEA

Este comando vai ter uma complexidade algorítmica de O (1) pois apenas tem um ciclo while, que não está dependente de nenhuma variável.

## Comando LOADST

Este comando vai ter uma complexidade algorítmica de O (1) pois apenas tem um ciclo while, que não está dependente de nenhuma variável.

## Comando SHOWALL

Como este comando tem uma função “earthquakeListShowAll” de complexidade algorítmica de O (n) e não está dependente nem de mais nenhuma função nem de variáveis então a própria função vai assumir a complexidade O (n).

## Comando SHOW\_Y

Neste comando temos 2 funções auxiliares ambas de complexidade O (n) mas como não estão encadeadas então ela irá assumir a complexidade O (n) também.

## Comando SHOW\_T

Neste comando temos 2 funções auxiliares ambas de complexidade O (n) mas como não estão encadeadas então ela irá assumir a complexidade O (n) também.

## Comando SHOW\_YT

Neste comando temos 2 funções auxiliares ambas de complexidade O (n) mas como não estão encadeadas então ela irá assumir a complexidade O (n) também.

## Comando LIST\_T

Neste comando usamos uma função auxiliar, que tem dois for encadeados, sendo que uma depende do tamanho da lista e outra depende da contagem atual, desta maneira significa que ambas tem complexidades similares de O (n), estando encadeadas no for, o que irá suceder é que a função vai assumir uma complexidade de O (n2).

## Comando COUNT

Neste comando temos 2 funções auxiliares ambas com complexidade algorítmica de O (n), não estando de alguma forma encadeadas, o comando irá assumir a complexidade algorítmica de O (n) também.

## Comando COUNTRY\_S

Neste comando usamos 4 funções auxiliares, sendo cada uma para os diferentes cenários possíveis (ser literacia ou gdp a ordem, e ser ascendente ou decrescente o outro critério) e como a implementação é similar em todas (O (n2)), só sendo possível a função utilizar uma das funções para o caso que se adeque melhor, assume a função a complexidade algorítmica de qualquer uma destas sendo todas iguais, tornando-se por si também (O (n2)).

## Comando TOPN

Neste comando usamos 2 funções auxiliares, tendo uma delas 2 ciclos for encadeados que dependem de variáveis, e mais um ciclo for também este dependente de uma variável, ou seja irá assumir a complexidade maior que acaba por ser O (n2), já na outra função acaba por ser parecida tendo também ela 2 ciclos for encadeados, ou seja no comando sendo assim assume novamente a maior que acaba por ser O (n2).

# ALGORITMOS

## SHOWALL

Para realizar o método showall só necessitamos de uma função auxiliar sendo ela a seguinte:

### Algoritmo earthquakeListShowAll

Inputs: earthquakeList – lista que contém os earthquake

ptEarthquakeListCount – ponteiro para guardar o valor da contagem de terramotos já mostrados

ptPageCount – ponteiro para guardar a contagem de páginas já mostradas

Begin

ListElem elem

int eCount <- ptEarthquakeListCount

int pCount <- ptPageCount

FOR eCount TO eCount+50 DO

lisGet(earthquakeList, i, &elem)

print Elem

END FOR

ptPageCount <- pCount+1

ptEarthquakeListCount <- eCount+20

END

Este algoritmo apresenta uma complexidade algorítmica de O (1) pois o ciclo for tem sempre o mesmo comprimento eCount além de que como listGet também tem uma complexidade de O (1) não afetará o ciclo.

## SHOW\_Y

Para este comando, necessitamos de 2 funções auxiliares, uma para retornar o número de terramotos acontecerem naquele ano introduzido para mostrar no ecrã, e outra retornar e mostrar esses mesmos terramotos.

### Algoritmo earthquakeListGetYearCount

Inputs: earthquakeList – lista que contém os earthquakes

Year – introduzido pelo utilizador para saber que ano pretende na pesquisa

Count – ponteiro para guardar o número de terramotos existentes no ano fornecido

Begin

List Elem

Int size

Int c = 0

listSize(earthquakeList, &size)

FOR i<-0 TO Size DO

listGet(earthquakeList, i, &elem)

test <- elem.date.year

IF YEAR = test DO

C++

END IF

END FOR

\*count <- c

END

Neste algoritmo a complexidade algorítmica é de O (n) pois o ciclo for está dependente do tamanho da lista.

### Algoritmo earthquakeListShowYear

Inputs: earthquakeList – lista que contém os earthquakes

Year – introduzido pelo utilizador para saber que ano pretende na pesquisa

Begin

List Elem

Int size

listSize(earthquakeList, &size)

FOR i<-0 TO Size DO

listGet(earthquakeList, i, &elem)

IF YEAR = elem.date.year DO

Print elem

END IF

END FOR

END

Neste algoritmo a complexidade algorítmica é de O (n) também já que o ciclo for volta a depender apenas do tamanho da lista.

## SHOW\_T

Para este comando o processo é praticamente idêntico ao do comando SHOW\_Y sendo que também necessitamos de 2 funções auxiliares realizando uma delas também a contagem dos países que tivessem aquele código e outra novamente para posteriormente retornar esses mesmos países.

### Algoritmo earthquakeListGetCountryCount

Inputs: earthquakeList – lista que contém os earthquakes

CountryCode – introduzido pelo utilizador para saber o código do país que pretende pesquisar

Count – ponteiro para guardar o número de países existentes com esse código

Begin

List Elem

Int size

Int c = 0

listSize(earthquakeList, &size)

FOR i<-0 TO Size DO

listGet(earthquakeList, i, &elem)

IF elem.countrycode[0] = countrycode DO

c++

END IF

END FOR

\*count <- c

END

Neste algoritmo a complexidade algorítmica é de O (n) pois o ciclo for está dependente apenas do tamanho da lista.

### Algoritmo earthquakeListShowCountry

Inputs: earthquakeList – lista que contém os earthquakes

Year – introduzido pelo utilizador para saber que ano pretende na pesquisa

Begin

List Elem

Int size

listSize(earthquakeList, &size)

FOR i<-0 TO Size DO

listGet(earthquakeList, i, &elem)

IF elem.countrycode[0] = countrycode DO

Print elem

END IF

END FOR

END

Neste algoritmo a complexidade algorítmica é de O (n) pois o ciclo for está dependente apenas do tamanho da lista.

# LIMITAÇÕES DO PROJETO

Devido ao contratempo descrito na introdução e finalizado na conclusão, achamos que nos afetou e causou algumas limitações ao nosso trabalho desenvolvido, o que passamos a descrever abaixo.

**Não foi implementado o comando Histogram e Region\_AVG**

-Devido além do infortuno que tivemos, mas também com o curto intervalo entre momentos de avaliação das diferentes UCs foi-nos impossível sequer tentar a implementação do comando Histogram que admitimos que seria bastante exigente e implicaria bastante domínio do conhecimento ensinado nas aulas.

-Já o Region\_AVG, tivemos complicações ao adicionar os códigos do país ao map pois esse mesmo código que estaria presente na chave, também estaria no valor correspondente á chave, o que fez com que tivéssemos dificuldade á adição ao map dos códigos do país o que nós afetou na implementação deste comando, que nos levou a não conseguirmos implementar de todo este mesmo comando

**Nas funcionalidades Country\_S, TopN**

-Devido a um possível problema na adição de elementos ao mapa como dito anteriormente este comando não mostra o pretendido apesar de acreditarmos que até seria uma boa proposta de solução ao problema proposto.

-Neste segundo comando, sucedesse praticamente o mesmo com exceção de que o problema reside no cálculo de country\_code, que achamos que essa função possa possivelmente, não estar tão bem desenvolvida como gostaríamos. Sendo que a mesma impactou negativamente o desempenho também dos comandos List\_T e Count pois estes também necessitam do country\_code, o que nos leva a querer que talvez com a devida implementação desta função de cálculo estás funções não apresentassem falhas.

# CONCLUSÕES

Retomando o ultimo ponto da introdução, como era de esperar sentimos dificuldades, de tal modo que tivemos que começar do projeto de novo no fim de semana de entrega, devido a erro que não conseguimos perceber a sua origem em que não permitia nem sequer testar o programa, o que nos atrasou e provocou bastantes constrangimentos, mas com isto dito, apesar de este contratempo massivo que nós levou a exaustão deixa-mos aqui um sentimento de resiliência enorme e de tentativa de concretização após este infortuno acontecimento.

Para realizar este projeto foi preciso aplicar todo o conhecimento aprendido em aula e desta forma conseguiu-nos ajudar a consolidar o mesmo sendo que para a programação é necessário “meter a mão na massa”.

Como dito anteriormente apesar de todo este contratempo conseguimos implementar praticamente todos os comandos (talvez apresentem algumas incoerências ou falhas). Ainda assim acreditamos que tais falhas/incoerências sejam altamente diferenciadoras para nos desvalorizarem muto no trabalho realizado.