

Índice

[1. TAD Utilizados 3](#_Toc11546409)

[1.1. Estrutura ClinicalData 3](#_Toc11546410)

[1.1.1. Estrutura 3](#_Toc11546411)

[1.2. Estrutura Patient 3](#_Toc11546412)

[1.1.1. Funções 3](#_Toc11546413)

[1.1.2. Funções 3](#_Toc11546414)

[1.1. Estrutura ClinicalDataStats 4](#_Toc11546415)

[1.1.1. Estrutura 4](#_Toc11546416)

[1.1.1. Funções 4](#_Toc11546417)

[1.2. List - ArrayList 4](#_Toc11546418)

[1.3. Map – ArrayList 4](#_Toc11546419)

[1.4. Queue – ArrayList 5](#_Toc11546420)

[2. Complexidades Algorítmicas 5](#_Toc11546421)

[2.1. Comandos/Funções 5](#_Toc11546422)

[LOAD 5](#_Toc11546423)

[Sort 6](#_Toc11546424)

[AVG 6](#_Toc11546425)

[NORM 7](#_Toc11546426)

[QUEUE 8](#_Toc11546427)

[CHECKDISTRICT 9](#_Toc11546428)

[3. Limitações 10](#_Toc11546429)

[4. Conclusão 10](#_Toc11546430)

# TAD Utilizados

Os TAD utilizados para o este projeto foi o ***List***, ***Map*** e ***Queue*** onde cada um teve uma implementação específica na estrutura do programa.

## Estrutura ClinicalData

Estrutura que guarda os dados clínicos de cada consulta.

## Estrutura

typedef struct clinicalData {

float age;

float bmi;

float glucose;

float insulin;

float mcp1;

int disease\_type;

int clinicalDataCount;

} ClinicalData;

/\* Estrutura dos dados Clínicos\*/

## Estrutura Patient

Guarda os dados de um determinado paciente, incluindo os seus dados clínicos.

## Funções

typedef struct patient {

int id;

Date birthdate;

char gender;

String hospital;

String district;

ClinicalData clinicalData;

} Patient;

/\* Estrutura dos dados do Paciente\*/

## Funções

Patient patientCreate(int id, Date birthdate, char gender, char \*hospital, char \*district);

void patientPrint(Patient patient);

void patientNormPrint(Patient patient);

## Estrutura ClinicalDataStats

Estrutura que guarda os valores médio de cada dado clínico.

## Estrutura

typedef struct clinicalDataStats{

float age;

float bmi;

float glucose;

float insulin;

float mcp1;

int disease\_type;

int clinicalDataCount;

/\* Usado apenas na opcao NEURALNET \*/

float c1;

float c2;

float c3;

float c4;

} ClinicalDataStats;

## Funções

ClinicalDataStats clinicalDataStatsCreate();

void clinicalDataStatsPrint(ClinicalDataStats \*clinicalDataStats);

## List - ArrayList

Usado maioritariamente em toda a estrutura do programa onde a estrutura de acesso é baseada por ***ranks*** permitindo assim o acesso a aleatório a qualquer instante a um elemento presente na lista. Esta estrutura foi usada para fazer carregar a lista de pacientes e dados clínicos.

## Map – ArrayList

Um mapa representa um contentor de elementos que implementa uma memória associativa (dicionário) onde são armazenados tuplos, **{chave: valor}.** A chave está apenas associada a um valor para além de não existir chaves duplicadas, há valores duplicados, nuca chaves duplicadas. Os acessos são feitos indicando a **chave**.

Esta estrutura foi escolhida para fazer a média de dados clínicos de cada paciente. A estrutura ficou com a seguinte configuração:

/\* definicao do tipo do chave\*/

typedef String MapKey;

/\* definicao do tipo do valor\*/

typedef ClinicalDataStats MapValue;

## Queue – ArrayList

A fila é um contentor de elementos que são inseridos e removidos de acordo com o princípio FIFO.

* Os elementos podem ser inseridos na fila a qualquer altura, mas apenas o que se encontra na fila há mais tempo pode ser removido.
* Os elementos são inseridos no final da fila e removidos do início da fila.

Este TAD foi utilizado na implementação para ver todos pacientes que se enquadram num determinado tipo de critério, os dados clínicos médios num intervalo mínimo e máximo e de uma idade mínima e máxima, onde a cada comando **NEXT** dentro comando **QUEUE** mostra os dados do paciente um após o outro sucessivamente.

# Complexidades Algorítmicas

## Comandos/Funções

## LOAD

Esta função lê dois ficheiros. O primeiro são os dados dos pacientes onde é inicializado os dados clínicos do paciente a zero, a seguir lê-se os dados do ficheiro que contém os dados clínicos associando assim a cada paciente onde é feito uma pesquisa na lista de pacientes para atualizar os dados clínicos de cada um paciente. A complexidade é **O (n2)**.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

## Sort

Mostra os pacientes importados de forma ordenada crescentemente. O critério de ordenação deve ser solicitado ao utilizador e pode ser um dos seguintes:

* - Data de nascimento
* - Hospital (desempate pela data de nascimento)
* - Distrito (desempate pelo hospital);

A complexidade deste comando é **O(n2)**.

Uma imagem com captura de ecrã, texto

Descrição gerada automaticamente

## AVG

Mostra a média dos dados clínicos de cada paciente (age, bmi, glucose, insulina e mcp1) em cada distrito, e ordenados por distrito. A complexidade deste algoritmo é **O(n3)**.

Uma imagem com interior

Descrição gerada automaticamente

## NORM

Mostra, para cada paciente, os seus dados clínicos normalizados entre -k e k, segundo a normalização *min-max*.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Figura 1 - Função normalizeClinicalData

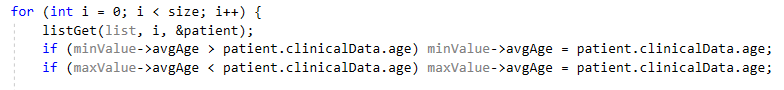


Figura 2 - Função findMinMaxAndAVG

No **NORM**, depois de ter recebido a lista de pacientes e o valor inserido de **k** pelo utilizador, executamos a função ***normalizeClincalData()*** que vai percorrer toda a lista normalizando os dados clínicos de cada paciente. A função ***findMinAndMaxAVG()***devolve os valores médios e mínimos de todos os dados clínicos.

A complexidade do comando **NORM** é quadrática **O(n2)**.

## QUEUE

Este comando copiar para fila todos os pacientes que se enquadrem nos seguintes critérios:

* - A sua idade seja inferior ao valor médio do intervalo [min(age), max(age)] das idades de todos os pacientes.

ou

* - A sua idade seja superior ao valor médio do intervalo [min(age), max(age)] das idades de todos os pacientes.

- O valor dos atributos bmi, glucose, insulina e mcp1 sejam inferiores ao valor médio do intervalo entre o [min(atr),max(atr)] para cada um destes 4 atributros. A complexidade deste comando é **O(n2)**.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

Uma imagem com captura de ecrã

Descrição gerada automaticamente

## CHECKDISTRICT

Este comando segmenta os pacientes com as suas médias de dados clínicos de acordo com o seu distrito de residência. A complexidade deste comando é **O(n2)**.

Uma imagem com captura de ecrã

Descrição gerada automaticamente

# Limitações

Não conseguimos implementar o comando **NEURALNET**.

# Conclusão

Após a conclusão da implementação do projeto e dado que este cumpre quase tudo o que é pedido. O que resulta da tentativa de simplificar ao máximo os processos através da implementação de Tipos Abstratos de Dados é bastante satisfatório e permite uma melhor interpretação e organização do código e, por isso, maior eficiência.

O objetivo principal do projeto foi cumprido sendo que foram usados os TADs ***Queue****,* ***List*** *e* ***Map***com a variante do tipo ***ArraList****.*

Dito isto prevemos que o projeto foi uma enorme ajuda na captação e utilização dos conceitos utilizadas nas aulas teóricas e práticas que este foi concluído com sucesso.