

1º Projecto Prático

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Algoritmos e Estruturas de Dados**

2016/2017

João Afonso Póvoa Marques – XXXXXXXXXX

José Miguel Saraiva Monteiro – XXXXXXXXXX

Leonardo Machado Alves Vieira – 2015236155

**Índice**

[**Índice** 2](#_Toc478235854)

[**1 – Introdução** 2](#_Toc478235855)

[**2 – Estruturas** 3](#_Toc478235856)

[Estrutura 1: 3](#_Toc478235857)

[Estrutura 2: 4](#_Toc478235858)

[Estrutura 3: 4](#_Toc478235859)

[**3 - Operações Implementadas** 5](#_Toc478235860)

[Estrutura 1: 5](#_Toc478235861)

[Estrutura 2: 5](#_Toc478235862)

[Estrutura 3: 6](#_Toc478235863)

[**4 – Casos de Teste** 6](#_Toc478235864)

[Estrutura 1: 6](#_Toc478235865)

[Estrutura 2: 6](#_Toc478235866)

[Estrutura 3: 6](#_Toc478235867)

[**5 – Comparações** 6](#_Toc478235868)

[**6 – Conclusões** 6](#_Toc478235869)

# **1 – Introdução**

Neste projeto foi nos proposta a tarefa de criar 3 estruturas de dados suportadas em abstrações diferentes que possam armazenar e manipular a informação disponibilizada. No contexto deste projeto, informação era fornecida através do ficheiro dados*.csv* sobre a evolução do acesso a redes elétricas pela população mundial desde 1960 até 2016.

Sobre todas estas estruturas foram definidas e implementadas funções de pesquisa, inserção, edição e remoção para ambos anos e países, que foram adaptadas e melhoradas de acordo com a estrutura e que podem ser facilmente utilizadas através de um interface funcional (igual para todas as estruturas) que é apresentado ao utilizador ao iniciar o programa.

Também foram criadas em todas as estruturas, com uma vertente de teste, funções auxiliares para realização de *benchmarks* em que são feitos testes extremos e contabilizado o tempo de resposta para cada um.

Todo o trabalho foi realizado em python

As estruturas implementadas foram as seguintes:

* Listas Duplamente Ligadas de Listas Duplamente Ligadas
* Stacks de Stacks
* Árvores AVL de Árvores AVL

# **2 – Estruturas**

## Estrutura 1:

Stacks de Stacks

[Implementação base retirada das estruturas apresentadas nas aulas de AED]

## Estrutura 2:

Listas Duplamente Ligadas de Listas Duplamente Ligadas

[Implementação base retirada das estruturas apresentadas nas aulas de AED]

Uma das estruturas implementadas foi uma Lista Duplamente Ligada de Listas Duplamente Ligadas. Esta estrutura foi escolhida por diversas razões, sendo uma delas a relativa facilidade de implementação e o facto de já termos tido contacto com este tipo de estruturas previamente. Para além disso foi também tido em conta a complexidade *BigO* das várias funções a implementar sobre esta estrutura. Assim sendo, o facto de esta ter uma complexidade de Θ(1) para as operações (isoladas) de inserção e remoção foi bastante apelativo. Isso faz com que as operações referidas sejam uma vantagem desta estrutura.

No entanto, como no contexto do problema não faz sentido haver duplicação de dados e faz sentido alguns valores estarem ordenados algumas desvantagens são criadas. Isto é, para cada operação de adição é necessário uma primeira pesquisa para garantir que o valor a adicionar não se encontra previamente na lista. Se o valor estiver de alguma forma ordenado, será escusado a pesquisa completa da lista, no entanto com valores desordenados terá sempre que iterar por todos os nós da lista à procura de um com o valor igual. Com isto, facilmente se entende que uma das desvantagens desta estrutura será a pesquisa, pois está subentendida em todas as outras operações e tem uma complexidade de Θ(n) (sendo o *n* minimizado quando operando sobre valores ordenados).

## Estrutura 3:

Árvore AVL de Árvores AVL

[Implementação base retirada....]

# **3 - Operações Implementadas**

## Estrutura 1:

Stacks de Stacks

* Pesquisa:
* Inserção:
* Edição:
* Remoção:

## Estrutura 2:

Listas Duplamente Ligadas de Listas Duplamente Ligadas

* Pesquisa:

Como referido anteriormente a pesquisa é um dos

pontos fracos desta estrutura.

Num caso base, a pesquisa teria sempre uma complexidade de Θ(n), no entanto, na lista de anos, os anos encontram-se ordenados, o que faz com que para esta lista, o *n* seja reduzido. Para este caso temos num limite inferior uma complexidade de pesquisa de Θ(1) e num limite superior a complexidade base de Θ(n).

Foi também implementada uma função de pesquisa na lista dos anos, que permite a pesquisa de anos dado um limite inferior e superior de percentagens. Esta função tem uma complexidade de Θ(n) pois esta vai correr toda a lista imprimindo apenas os anos cujo valor esteja dentro dos limites.

Para a lista de países foi apenas implementada uma pesquisa simples através de uma função Θ(n) que corre a lista toda à procura do nó correto que depois é retornado para ser usado noutras funções.

* Inserção:

Também já foi referida a vantagem da inserção neste tipo de estruturas.

Foram implementadas dois tipos de inserção. Para a inserção na lista (desordenada) de países, recorre-se primeiramente a uma pesquisa pelo nome e tag do país a inserir, para garantir que o país não está já na lista evitando assim duplicação de informação. Depois desta pesquisa e em caso negativo de duplicação, procede-se a uma inserção no final da lista do novo nó. Esta parte da operação é realizada com uma complexidade de Θ(1), visto que a pesquisa inicial retorna o ultimo nó da lista, de maneira a evitar precorrer novamente esta.

Será importante notar que quando é adicionado um novo país, a lista de anos encontra-se a *None.*

A outra implementação foi feita sobre a lista (ordenada) de anos de um país. À semelhança da anterior, também é necessario uma pesquisa prévia para garantir que não existe duplicação de informação que também retorna um nó. Porém, esta é uma inserção ordenada, o que implica que o nó retornado nem sempre é o ultimo, mas sim o primeiro, caso exista, com valor maior do que o nó a adicionar. No melhor caso esta inserção é optima com uma complexidade de Θ(1). Nos restantes dos casos será de uma complexidade de Θ(n), aumentado o *n* até ao pior caso, que é ter de precorrer toda a lista.

* Edição:
* Remoção:

## Estrutura 3:

Árvore AVL de Árvores AVL

* Pesquisa:
* Inserção:
* Edição:
* Remoção:

# **4 – Casos de Teste**

## Estrutura 1:

Stacks de Stacks

## Estrutura 2:

Listas Duplamente Ligadas de Listas Duplamente Ligadas

## Estrutura 3:

Árvore AVL de Árvores AVL

# **5 – Comparações**

# **6 – Conclusões**