2017

RELATÓRIO DO PROJETO 1

ALGORITMOS E ESTRUTURAS DE DADOS - PL3

PAULO FERNANDES, 2015238525
TERESA SALAZAR, 2015234237
VÍTOR FERREIRA, 2015228102

INTRODUÇÃO

Foi-nos proposto um trabalho que que consistia na criação e estudo de 3 estruturas de dados diferentes que permitissem trabalhar sobre um determinado ficheiro csv fornecido. O objetivo é analisar e comparar a performance destas estruturas em funções de pesquisa, inserção, edição e remoção sobre os dados fornecidos, da forma mais otimizada possível. Foi também pedido a elaboração de uma interface funcional para interagir com as estruturas.

O documento csv disponibilizado contém dados acerca da evolução do acesso a redes elétricas pela população mundial com valores desde o ano de 1960 até 2016, que indicam a percentagem de população com acesso a redes elétricas em certos países (p. e. Portugal) ou regiões do mundo (p. e. União Europeia). O ficheiro está organizado alfabeticamente de acordo com o código de cada país/região na forma: «"Country Name";"Country Code";"1960";..;"2016"». Tínhamos também de ter em conta o facto de haver anos em que não para os quais não havia informação disponível, para um ou mais países.

PROGRAMA

```
CHOOSE:
1 - Search
2 - Insert
3 - Edit
4 - Remove
5 - Exit

Option:
```

FIGURA 1 – MENU INICIAL

Apesar das diferentes estruturas, os programas das três são semelhantes. As mesmas funções são realizadas, os menus são idênticos, os passos que cada um realiza são similares.

Cada programa começa por ler os dados do ficheiro csv e coloca as informações na devida estrutura. A partir daí o utilizador escolhe as operações que pretende realizar (o menu da figura 1 aparece).

Nas opções 2, 3 e 4, começa por escolher se quer escolher o país por nome ou por código e escreve de seguida o país. Neste momento, o programa procura o país inserido na estrutura e informa caso não tenha encontrado. Depois escolhe o ano e no caso do *insert* e *edit*, escolhe o valor. Nos 3 casos é recebido um alerta caso não seja possível realizar ação (se já existir um valor para esse ano no *insert*, ou se não existe um valor para editar/remover nos outros dois casos. Estas operações apenas podem ser realizadas sobre valores de países nos anos compreendidos entre 1960 e 2016, inclusive. Após cada uma destas operações, em caso de sucesso, o ficheiro é atualizado, sendo apenas reescrita a linha do país que foi alterado. Salvaguarda-se, desta forma, a perda de informação caso o programa seja desligado involuntariamente.

No caso da operação *search,* aparece um novo menu com várias opções de pesquisa.

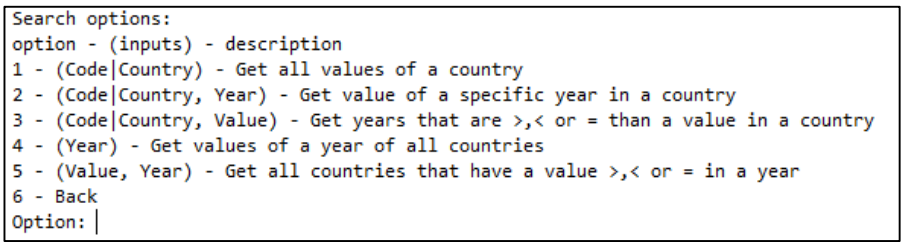


FIGURA 2 – MENU SEARCH

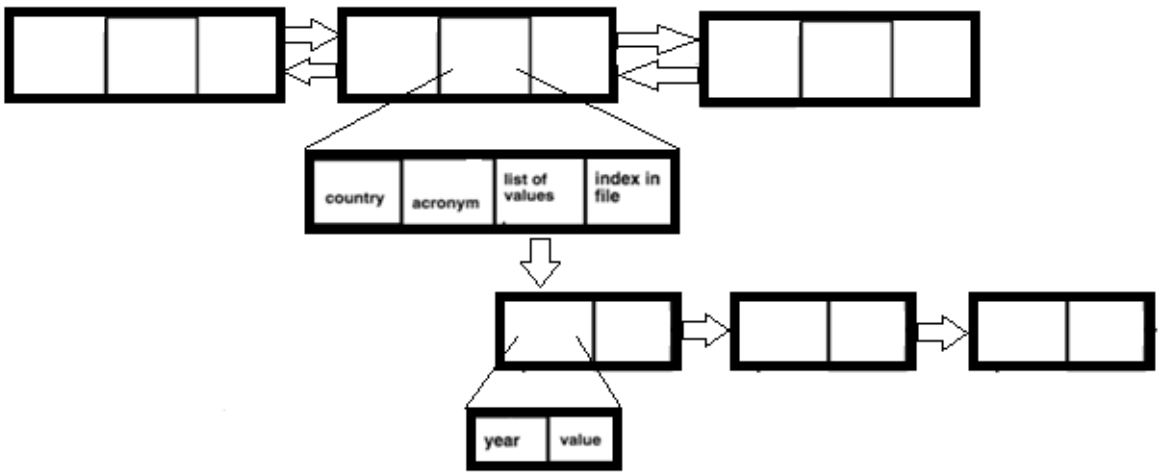
Nas 3 primeiras opções, à semelhança das funções do menu principal, é perguntado ao utilizador se procurar por nome ou código do país, recebendo a mesma notificação caso o país não seja encontrado. Na opção 2 é ainda pedido o ano, e na 3 é pedido o valor de comparação e de seguida o utilizador escolhe se pretende anos com valores maiores, menores ou iguais ao valor que introduziu. Nas opções 4 e 5 é pedido um ano ao utilizador, e, além disso, na 5 é também pedido um valor de comparação e se pretende procurar valores maiores menores ou iguais àquele que inseriu, à semelhança da opção 3.

ESTRUTURA DE DADOS 1

Descrição da Estrutura

A primeira estrutura de dados que implementámos consiste numa lista duplamente ligada, em que cada nó contém um array. Este array contém o nome do país no índice 0, a sigla no índice 1, uma referência para uma lista ligada de valores no índice 2 e o iD do país (de 1 até 265 – corresponde à linha no ficheiro) no último índice. A lista duplamente ligada encontra-se ordenada alfabeticamente pelo nome do código.

A lista ligada de valores, em cada nó, tem um array de tamanho 2, contendo o ano e a percentagem de população com acesso a redes elétricas relativamente a esse ano. Se não houver informação relativamente a um ano, então existe um nó correspondente a esse ano na lista. Desta forma, o tempo de execução das diversas operações diminui substancialmente, uma vez que a quantidade de dados é menor. Esta lista encontra-se ordenada por ordem crescente de ano, para que a pesquisa seja o mais rápida possível.

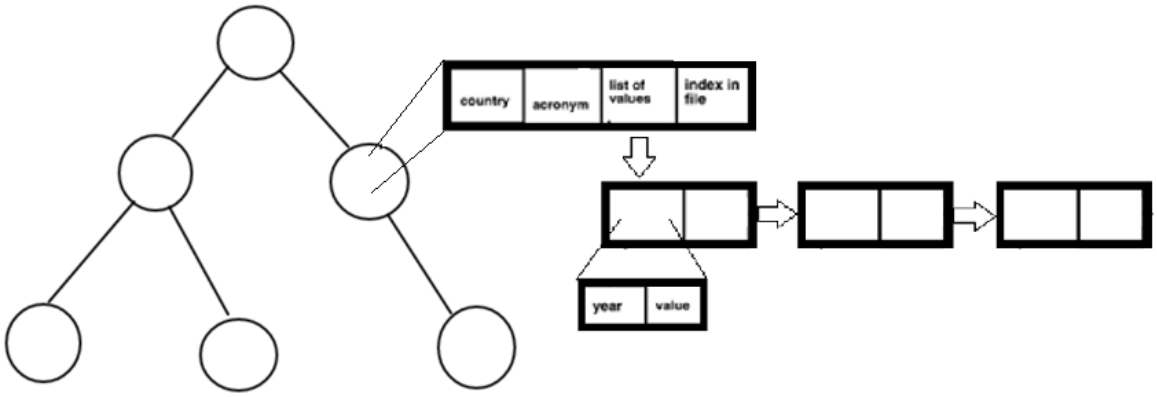


ESTRUTURA DE DADOS 2

Descrição da Estrutura

A segunda estrutura de dados que implementámos consiste numa árvore AVL, em que cada nó contém um array. Este array contém o nome do país no índice 0, a sigla no índice 1, uma referência para uma lista ligada de valores no índice 2 e o iD do país (de 1 até 265 – corresponde à linha no ficheiro) no último índice (semelhante à estrutura de dados 1). A árvore AVL encontra-se balanceada alfabeticamente pelo nome do país.

A lista ligada de valores, em cada nó, tem um array de tamanho 2, contendo o ano e a percentagem de população com acesso a redes elétricas relativamente a esse ano. Se não houver informação relativamente a um ano, esta não é inserida. Desta forma, o tempo de execução das diversas operações diminui substancialmente, uma vez que a quantidade de dados é menor. Esta lista encontra-se ordenada por ordem crescente de ano, para que a pesquisa seja o mais rápida possível.

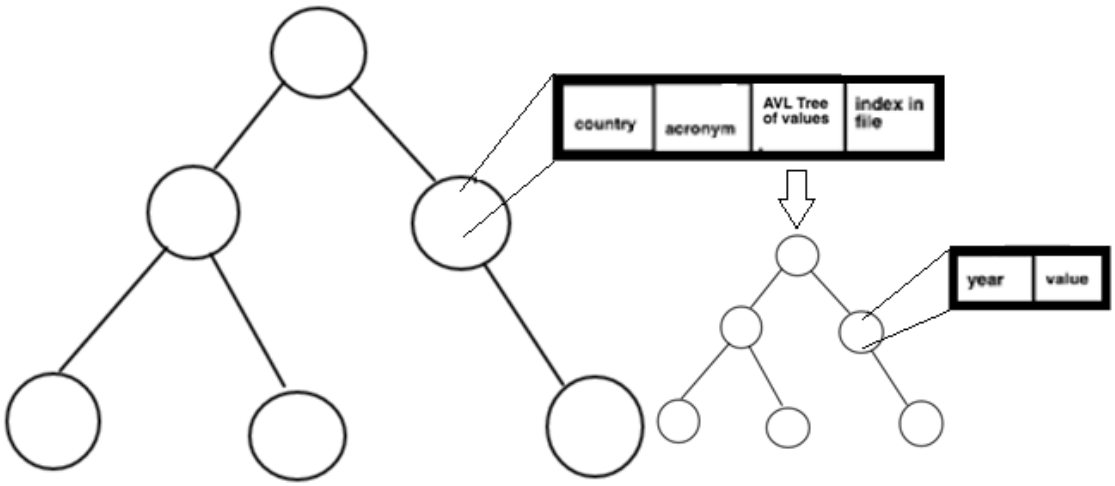


ESTRUTURA DE DADOS 3

Descrição da Estrutura

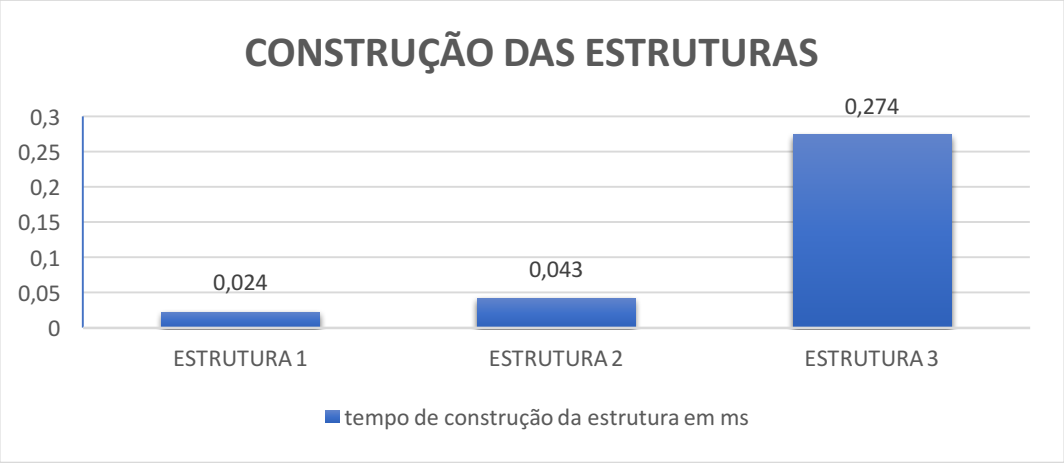
A terceira estrutura de dados que implementámos consiste numa árvore AVL, em que cada nó contém um array. Este array contém o nome do país no índice 0, a sigla no índice 1, uma referência para uma árvore AVL de valores no índice 2 e o iD do país (de 1 até 265 – corresponde à linha no ficheiro) no último índice (semelhante à estrutura de dados 1). A árvore AVL de países encontra-se balanceada alfabeticamente pelo nome do país.

A árvore AVL de valores, em cada nó, tem um array de tamanho 2, contendo o ano e a percentagem de população com acesso a redes elétricas relativamente a esse ano. Se não houver informação relativamente a um ano, o nó relativamente a esse ano não existe. Desta forma, o tempo de execução das diversas operações diminui substancialmente, uma vez que a quantidade de dados é menor. Esta árvore AVL de valores encontra-se balanceada por ano, para que a pesquisa seja o mais rápida possível.



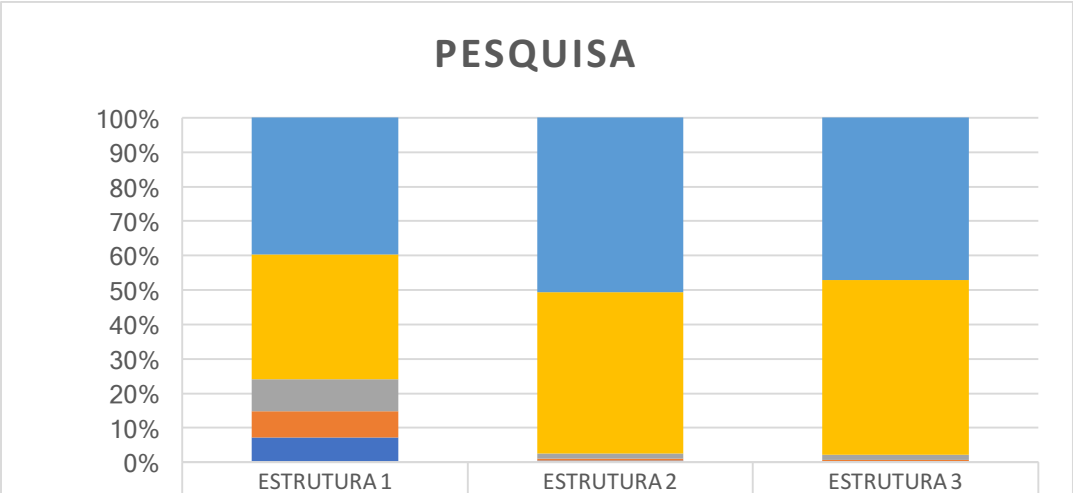
OPERAÇÃO/ ESTRUTURA			ESTRURTURA 1 LISTAS DE LISTAS	ESTRUTURA 2 ÁRVORE DE LISTAS	ESTRUTURA3 ÁRVORE DE ÁRVORES
CRIAÇÃO			*get_double_linked_list():*	get_tree():	get_trees():
			Inserção dos dados na estrutura e criação um dicionário para correspondência entre nomes e códigos dos países/regiões		
PESQUISA	1 País		get_country_values(estrutura, dicionário): Pede um país ao utilizador e devolve, caso exista, o array do nó correspondente ao mesmo ([nome, código, referência para os valores, nº da linha no ficheiro])		
			Dado o país o programa decide se começa do início ou do fim da lista, tendo em conta a ordem alfabética, diminuindo, assim, o tempo máximo de procura para cerca de metade. O(n)	É feita a procura na árvore pelo país tendo em conta o facto de esta estar balanceada de acordo com a ordem alfabética dos nomes dos países O(log n)	
		1 Valor	lista_ligada.get_value_of_year(ano):		tree.search_tree_of_values(ano):
			É procurado o valor de um ano, escolhido pelo utilizador, para o país, tirando partido da ordenação por ordem crescente da lista ligada de valores. O(n)		É procurado o valor de um ano, escolhido pelo utilizador, para o país, tirando partido do balanceamento da árvore por ano. O(log n)
		Vários valores >,<,=	estrutura_de_valores.get_years_with_filter(valor, opção):		
			Após ter sido perguntado ao utilizador o valor de comparação e se pretende receber valores maiores, menores ou iguais a esse, é feita uma pesquisa em todos os anos do país, por valores com as características pretendidas. O(n)		
	1 Ano	Todos os países	double_list.get_values_of_a_year_of_all_countries(ano):	tree.values_by_year(ano):	tree_of_trees.get_values_by_year(ano, [])
			Dado um ano introduzido pelo utilizador, são procurados em todos os países o valor que lhes corresponde nesse ano. O(n^2)		
		Todos os países >,<,=	double_ list.get_countries_with_filters(valor, ano, filtro):	tree.values_by_year (ano, filtro, valor):	tree_of_trees.get_countries_with_filter(year, option, value, []):
			Dado um ano e um filtro (>,<,=) introduzidos pelo utilizador, são procurados em todos os países o valor que lhes corresponde nesse ano e que passam por esse filtro. O(n^2)		
INSERÇÃO			linked_list.insert(ano, valor):		tree_of_values.insert_tree([ano, valor]):
			Procura na lista a posição onde colocar o novo valor, tirando partido ordenação por ordem crescente de anos. Se já existir esse ano na lista o novo valor não é inserido. O(n)		Pesquisa o local onde colocar o novo ano e valor na árvore, tirando partido do balanceamento por ano. Se já existir esse ano na lista o novo valor não é inserido. A árvore é rebalanceada se necessário O(log n)
EDIÇÃO			linked_list.edit(ano, to_insert):		tree_of_values.edit_tree([ano, valor]):
			Procura o ano que deve ser editado, como no método anterior, e edita se encontrar. O(n)		Procura o ano pretendido como no método *insert* e, se existir, faz a sua edição. O(log n)
REMOÇÃO			linked_list.remove (ano):		tree_of_values.delete(ano):
			Procura o ano que deve ser removido, à semelhança dos métodos acima, e procede à sua remoção caso exista. O(n)		Procura o ano a ser removido como nos métodos acima, e, caso exista procede à sua remoção. A árvore é rebalanceada se necessário. O(log n)

ESTUDO COMPARATIVO



Analisando, este gráfico, podemos concluir que a estrutura 3 é sem dúvida a estrutura que demora mais tempo a construir. Isto deve-se ao facto de se terem de construir bastantes árvores e de inserir valores nos seus nós, o que é processo bastante demorado. Pode-se concluir assim que a construção de listas é bastante mais rápida.

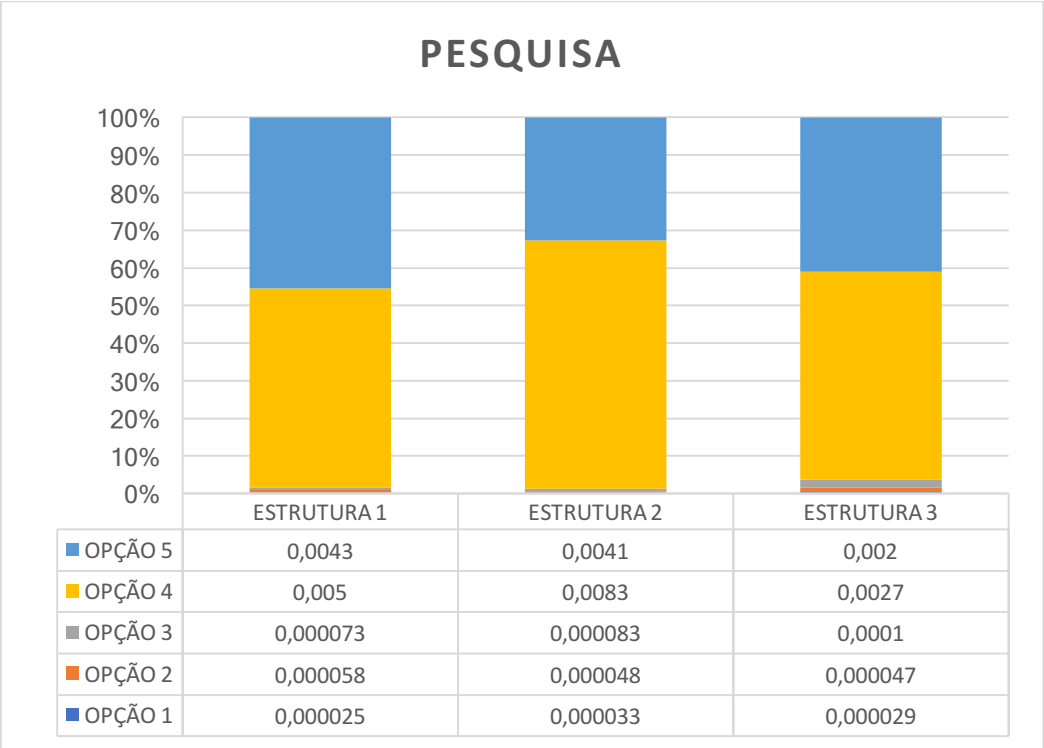
CODE: PRT | YEAR: 1972 | VALUE = 50 (GREATER)



	ESTRUTURA 1	ESTRUTURA 2	ESTRUTURA 3
OPÇÃO 5	0,0011	0,0027	0,0037
OPÇÃO 4	0,001	0,0025	0,004
OPÇÃO 3	0,00026	0,00008	0,00011
OPÇÃO 2	0,00021	0,000034	0,00003
OPÇÃO 1	0,0002	0,00002408	0,000027

Analisando este gráfico, pode-se concluir que, neste caso, a estrutura 2 é a estrutura que permite uma pesquisa mais rápida. Isto acontece, por um lado, porque a procura do país na árvore é bastante rápida, pois tem complexidade O(log n). Na estrutura 1 a complexidade de procura do país é O(n), apesar de verificarmos na lista duplamente ligada se o valor a procurar está mais perto do início ou do fim.

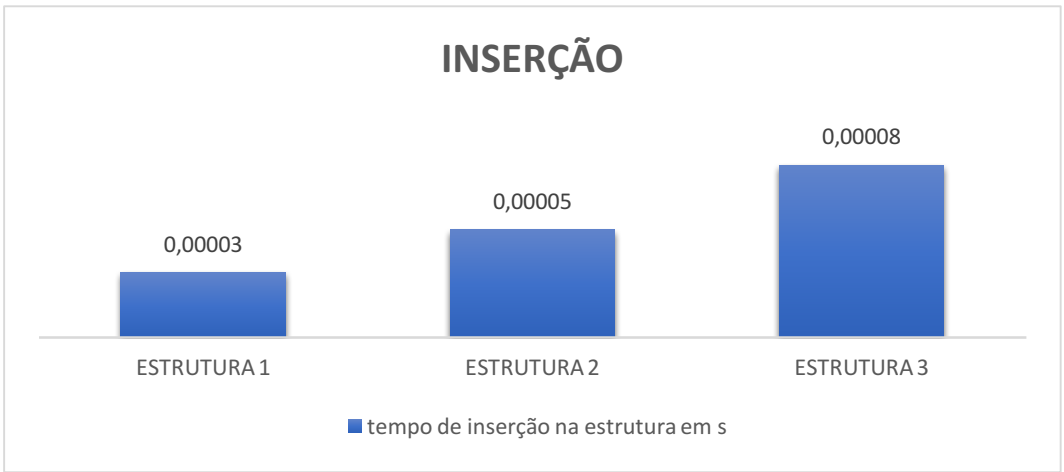
CODE: ARM | YEAR: 2000 | VALUE: 94.28774 (EQUAL)



	ESTRUTURA 1	ESTRUTURA 2	ESTRUTURA 3
OPÇÃO 5	0,0043	0,0041	0,002
OPÇÃO 4	0,005	0,0083	0,0027
OPÇÃO 3	0,000073	0,000083	0,0001
OPÇÃO 2	0,000058	0,000048	0,000047
OPÇÃO 1	0,000025	0,000033	0,000029

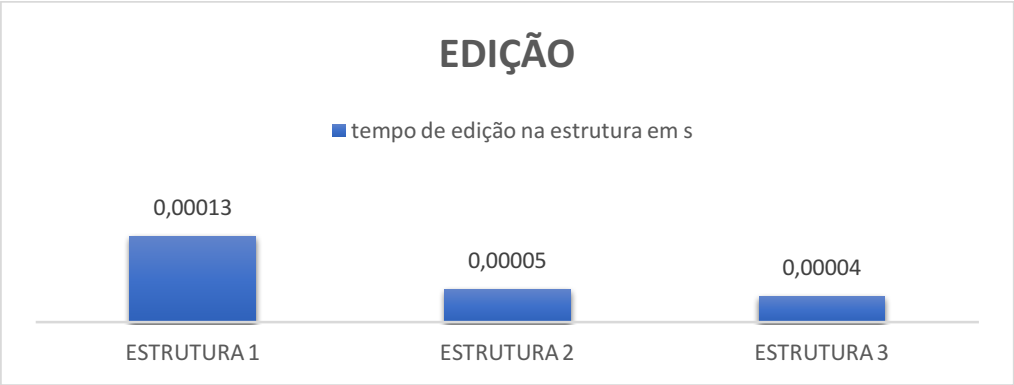
Neste input, uma vez que o país a pesquisar está no início da lista ligada na estrutura 1, o tempo de pesquisa do país diminuiu substancialmente. No entanto, a procura do ano na estrutura 3 na árvore, continua a prevalecer sobre a procura do ano em listas ligadas. Mais uma vez, verifica-se que a opção 4, é a opção que consome mais tempo de execução, uma vez que é necessário percorrer todos os países.

CODE: ZMB | YEAR: 1962



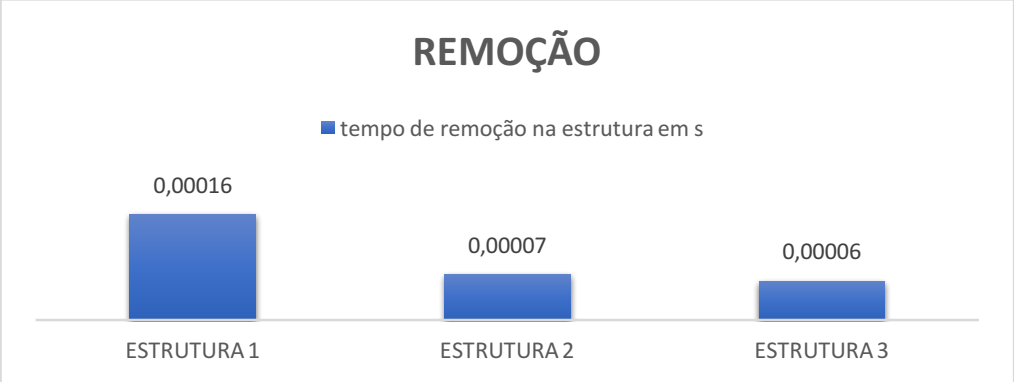
Analisando este gráfico, podemos verificar que a inserção na estrutura 1 foi bastante mais rápida do que nas outras estruturas. Isto deve-se ao facto de a procura do país na lista duplamente ligada começou no último nó, e uma vez que o país de input se encontra muito próximo do último nó, a procura foi quase imediata. Quanto à inserção, na estrutura 3, ao inserir um novo nó na árvore, provavelmente teve que se balancear a estrutura, levando a um maior tempo de execução.

CODE: LBY | YEAR: 2000



Neste caso, uma vez que o país a procurar se encontra no meio na lista duplamente ligada, a estrutura 1 verifica-se a mais lenta na procura do país. Quanto à edição, a estrutura 3 revela-se mais rápida, uma vez que a procura do ano tem complexidade O(log n), enquanto que a procura do ano estrutura 2 tem complexidade O(n).

CODE: LBY | YEAR: 2000



Analisando este gráfico, podemos verificar que os valores obtidos são semelhantes aos valores do gráfico anterior. Isto provavelmente deve-se ao facto de não ter sido necessário balancear a árvore após a remoção do nó.

CONCLUSÃO

Através dos resultados obtidos no estudo comparativo, consideramos que a Estrutura 2 (árvore de listas ligadas) é a melhor estrutura no geral. Como referimos anteriormente, esta revela-se rápida tanto na sua construção, como nas outras operações.

No entanto, se houvesse mais dados no programa, era bastante provável que a estrutura 3 se revelasse mais eficiente.

Concluindo, acreditamos que fizemos um bom trabalho em conjunto (33 % cada um, aproximadamente). Com três estruturas de dados conseguimos fazer todas as operações pedidas através de funções que demonstram ser rápidas na sua execução.

Usando recursividade e diversos algoritmos, conseguimos tornar o programa bastante eficiente e eficaz. O código está também protegido de todos os inputs do utilizador, impedindo o programa deixar de funcionar. O programa está bastante intuitivo e a disposição gráfica para o utilizador também é bastante *user friendly*.