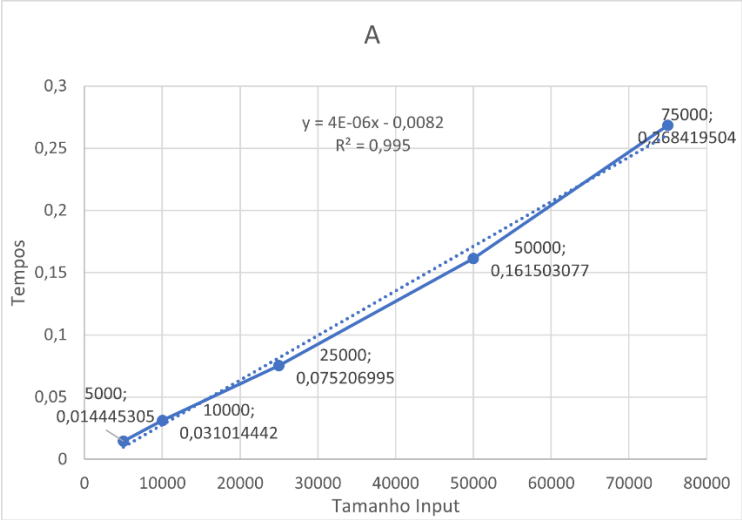
Relatório Projeto 3.1 AED 2021/2022

Nome: João Miguel Fernandes Moura
Nº Estudante:2020235800
PL (inscrição): 7 *Login* no *Mooshak:2020235800*

Tabela

Tamanho Input(nós)	Tempo de execução(s)
5000	0.0144453048706054
10000	0.0310144424438476
25000	0.0752069950103759
50000	0.161503076553344
75000	0.268419504165649

Gráfico



A expressão f(N) está de acordo com o esperado? Justifique.

Sim, dado que a estrutura de dados utilizada é uma árvore não equilibrada a complexidade teórica esperada seria linear que se comprova ao se analisar a regressão dos resultados obtidos. Esta complexidade linear deve-se à operação de impressão ter de percorrer n nós. Numa árvore perfeitamente equilibrada a complexidade seria log(n) dado que a altura da árvore seria reduzida uma vez que cada nó difere no máximo 1 de altura em cada sub-árvore. Isto não acontece nas árvores não equilibradas, onde as grandes divergências de altura tornam o algoritmo menos eficiente.

O projeto 3.1 pode ser implementado seguindo uma abordagem iterativa e uma recursiva.

Explique sucintamente o essencial das duas implementações em termos de estruturas de dados utilizadas e do cálculo da valorização das categorias e impressão da árvore.

A abordagem iterativa passaria pela utilização de uma estrutura de pilha que tem complexidade linear (no melhor dos casos) dado que quando é necessário pré-alocar memória para colocar o elemento mais recente, esta operação tem complexidade linear logo o processo de inserção na pilha tem a mesma complexidade. O processo de remoção e impressão dos elementos da pilha também será linear dado que por cada nível apenas existe um elemento. A abordagem recursiva, a utilizada na resolução da proposta, é a árvore não equilibrada. Como se verificou anteriormente esta abordagem tem complexidade linear no pior dos casos e log(n) no melhor dos casos (caso os nós sejam originalmente equilibrados), ao contrário da pilha por cada nível da árvore podem existir vários nós logo imprime vários elementos por nível, sendo ainda mais eficiente no caso de ser uma estrutura equilibrada, dado ter menos níveis de altura.

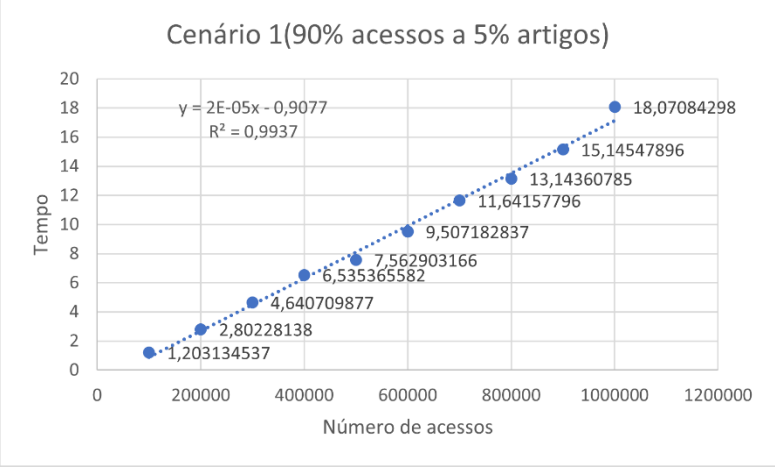
Assim podemos verificar que a estrutura mais eficiente para a resolução desta proposta é a abordagem recursiva, árvore não equilibrada, dado que na melhor das hipóteses tem complexidade log(n) enquanto a abordagem iterativa terá sempre complexidade N em qualquer circunstância.

Relatório Projeto 3.2 AED 2021/2022

Nome: João Miguel Fernandes Moura
Nº Estudante:2020235800
PL (inscrição): 7 *Login* no *Mooshak:2020235800*

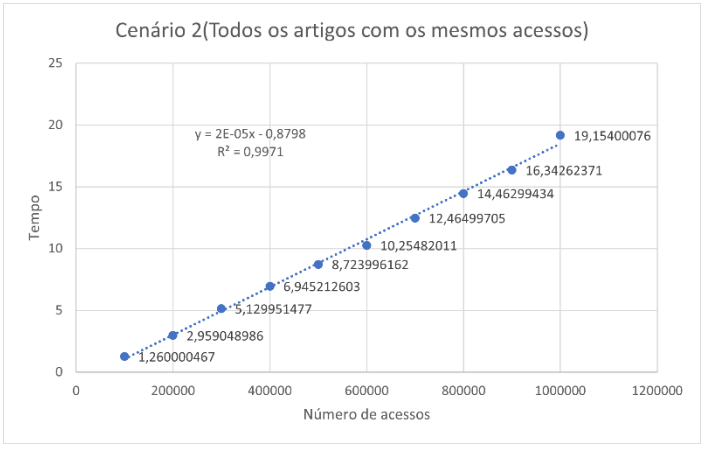
Cenário 1

Número de acessos	Tempo(s)
100000	1,20313453674316
200000	2,8022813796997
300000	4,64070987701416
400000	6,53536558151245
500000	7,56290316581726
600000	9,50718283653259
700000	11,6415779590606
800000	13,1436078548431
900000	15,1454789638519
1000000	18,0708429813385



Cenário 2

Número de acessos	Tempo(s)
100000	1,26000046730041
200000	2,95904898643493
300000	5,12995147705078
400000	6,94521260261535
500000	8,72399616241455
600000	10,2548201084136
700000	12,4649970531463
800000	14,4629943370819
900000	16,3426237106323
1000000	19,1540007591247



A evolução dos tempos de execução está de acordo com o esperado? Justifique.

Observando os resultados obtidos podemos concluir que a complexidade deste algoritmo em ambos os cenários é linear(O(n)). Esta complexidade corresponde a uma aproximação relativamente fiável da prevista teoricamente dado que o tempo de execução em média será de O(nlog2(n))(dado que temos n consultas/inserções) podendo, no pior caso, ser O(n) como mostram os resultados. A aproximação a uma complexidade linear deve-se ao facto da estrutura utilizada (*Splay tree*) não ser estritamente equilibrada, ou seja, pode haver alguns desequilíbrios na construção da árvore.

Embora em ambos os cenários a regressão obtida seja semelhante, denota-se maior eficiência no cenário 1, tendo este demorado menos tempo em média a executar. Esta otimização deve-se à capacidade da *splay tree* de tornar o último nó consultado na raiz da árvore, ou seja, é mais eficiente quando existem nós que são acedidos múltiplas vezes, como no cenário 1 em que 90% das consultas se concentram em 5% dos nós.

Relatório Projeto 3.3 AED 2021/2022

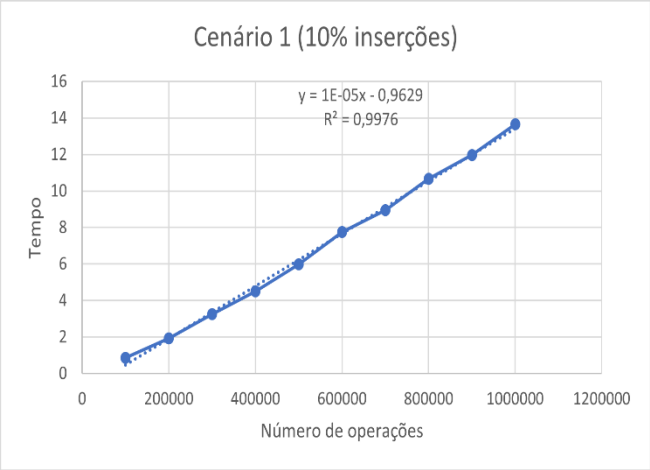
Nome: João Miguel Fernandes Moura
Nº Estudante:2020235800
PL (inscrição): 7 *Login* no *Mooshak:2020235800*

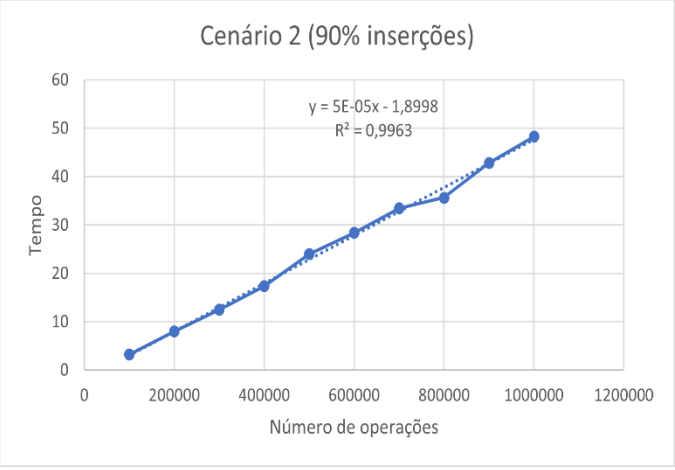
Cenário 1

Número de operações	Tempo
100000	0,872012853622436
200000	1,92800188064575
300000	3,25594902038574
400000	4,50300002098083
500000	5,9800329208374
600000	7,7580120563507
700000	8,95281767845153
800000	10,6549673080444
900000	11,9719710350036
1000000	13,655963897705

Cenário 2

Número de operações	Tempo
100000	3,22610688209533
200000	8,01797890663146
300000	12,4495854377746
400000	17,38392496109
500000	24,0111036300659
600000	28,3980662822723
700000	33,4340233802795
800000	35,6560299396514
900000	42,8231344223022
1000000	48,3370099067687





Os tempos de execução estão de acordo com o esperado? Justifique.

Os resultados estão de acordo com o esperado dado que a estrutura utilizada é uma árvore binária de pesquisa (AVL). A complexidade das operações para inserção e de procura de um determinado nó numa árvore equilibrada levam a que a complexidade esperada seja de O(nlog2(n)) (dado que envolve percorrer o número de nós, N, e é uma árvore binária, log2(N)) em ambos o cenário a regressão é linear o que corresponde a uma aproximação bastante fiável do teoricamente previsto.

A estrutura de árvore AVL é otimizada para fazer pesquisas e consultas rapidamente sem a necessidade de percorrer todos os nós para encontrar o pretendido logo, e como demonstram os resultados, o cenário 1 (90% de consultas) tem tempos de execução muito inferiores ao do cenário 2 (10% de consultas), sendo os resultados no cenário 1 na ordem dos segundos enquanto para o cenário 2 são da ordem das dezenas de segundos, demonstrando o que foi previsto teoricamente.

Relatório Projeto 3.4 AED 2021/2022

Nome: João Miguel Fernandes Moura
Nº Estudante:2020235800
PL (inscrição):7 *Login* no *Mooshak:2020235800*

Estrutura de Dados Principal usada em cada sub-projeto:

PROJ 3.1 : General Tree

PROJ 3.2 : Splay Tree
PROJ 3.3 : AVL Tree

Estruturas de dados usadas	General Tree	Splay Tree	AVL Tree
VANTAGENS GERAIS (max 3)	• Implementação simples • Estrutura fixa do nó, não necessita da criação de estruturas auxiliares • Não apresenta limite no número de nós filhos que cada nó pode ter	• Autoajustável (nós frequentemente acedidos ficam próximos da raiz sendo acedidos mais rapidamente) • Performance média semelhante às restantes árvores(O(log(n)) • Pouco impacto na memória	• Autoequilibrada • Pesquisa eficiente • Complexidade O(log(n))
DESVANTAGENS GERAIS (max 3)	• Em geral, pior performance que as restantes árvores • Irregularidade na organização dos nós. • Dado não ter uma estrutura de nós organizada apresenta maior complexidade na tomada de decisões,	• No pior caso pode ter complexidade O(n) • Pode-se reorganizar sozinha após cada pesquisa logo inútil caso haja pesquisas paralelas • Obsoleta caso os acessos aos diferentes nós tenham frequência semelhante	• Lenta inserção e remoção • No pior dos casos tem uma complexidade O(log〗^2 (n)) • Implementação de operações de inserção/remoção é complexa devido às rotações que têm de executar

	dado que consoante o nó a operação pode durar mais ou menos tempo em relação a outros		
Justificação para a escolha no PROJ 3.1	No contexto deste problema a escolha óbvia seria uma *General Tree* dado que cada categoria corresponderia a um nó e cada um destes poderia ter n subcategorias, então a estrutura a utilizar não poderia ser binária. A organização também não é problemática dado que o *input* é inserido pela ordem que é suposto colocar na árvore logo não é necessário estruturas melhor otimizadas na inserção. Há outras estruturas possíveis de utilizar, no entanto, árvores são mais rápidas que as restantes estruturas.		
Justificação para a escolha no PROJ 3.2	Dado que o cenário a ser resolvido envolve uma grande quantidade de acessos de uma maneira bastante assimétrica, ou seja, maior frequência de acessos a uma quantidade relativamente pequena de nós então a estrutura deve ser eficiente a fazer consulta de dados partindo do princípio que dados recentemente e frequentemente consultados têm maior chance de serem consultados novamente, logo a escolha óbvia é uma *Splay Tree*. Esta estrutura coloca o último nó acedido como raiz, auto ajustando-se, logo quando se dão múltiplos acessos repetidos aos mesmos nós esta estrutura é peculiarmente eficiente a encontra-los, dado que o caminho até eles é reduzido.		
Justificação para a escolha no PROJ 3.3	Neste cenário temos uma grande quantidade de acessos de uma maneira igualmente distribuída logo interessa-nos que o acesso a qualquer nó leve um tempo semelhante, ou seja, não haja desequilíbrios no acesso, sendo a melhor escolha para esse cenário uma AVL Tree. Esta estrutura autoajusta-se de forma manter a distância entre todos nós semelhante para que o acesso a qualquer um demore o mesmo tempo. Esta estrutura serve-se de uma série de rotações para manter a árvore sempre equilibrada. Outra estrutura possível de utilizar seria uma árvore vermelha e preta, que embora seja mais eficiente no que toca a inserções e remoções, organiza de forma menos eficiente tornando-se mais lenta nas consultas.		