Relatório Projeto 3.1 AED 2021/2022

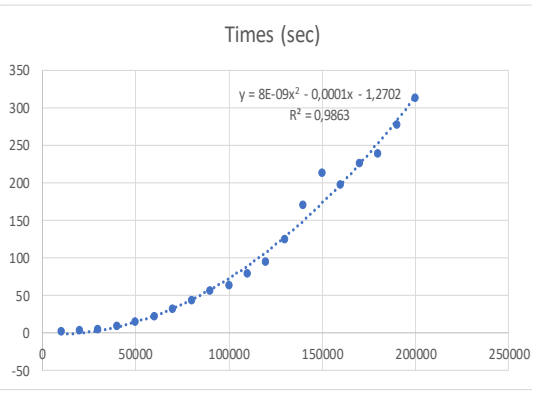
Nome: João Emanuel Sousa Moreira
Nº Estudante: 2020230563
PL (inscrição): PL2
Login no *Mooshak:* 2020230563

Correr a implementação do projeto 3.1 para um número crescente de categorias e obter os tempos de execução (excluindo tempo de leitura). Produzir tabela, gráfico e regressão relevantes.

Tabela

N	Times (sec)
10000	1
20000	3
30000	5
40000	9
50000	14
60000	21
70000	31
80000	43
90000	56
100000	63
110000	79
120000	94
130000	125
140000	171
150000	213
160000	198
170000	226
180000	239
190000	277
200000	314

Gráfico



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

Eu esperava um $O(N^2)$, (isto vem de O(M*N), com M nós filho), pois percorremos N vezes a árvore para inserir N nós. Como foi obtido um R-value muito próximo de 1, podemos concluir que a expressão está de acordo com o esperado.

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.

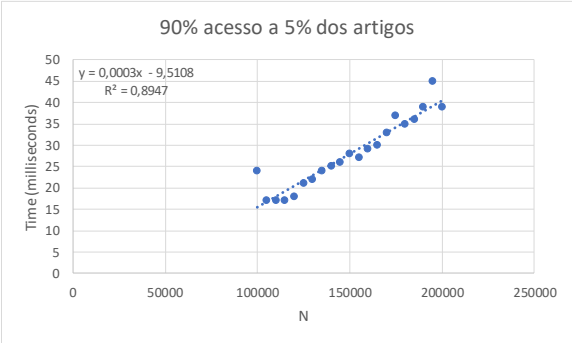
Numa solução recursiva, a função que insere/imprime é chamada por ela própria. Já na iterativa, adicionamos os nós a uma pilha. Ambas têm a mesma complexidade, pois na chamada de funções é criada uma pilha. Para calcular as categorias usei a recursividade, navegava ao longo da arvore e adicionava um nó se tivesse espaço ou voltava para cima com a recursão. Imprimi iterativamente. Adicionava os filhos dos nós do nível atual no fim de uma *queue* e imprimia o nó do inicio. Isto torna-se mais eficiente, pois acedo diretamente ao inicio e ao final da *queue.*

Relatório Projeto 3.2 AED 2021/2022

Nome: João Emanuel Sousa Moreira
Nº Estudante: 2020230563
PL (inscrição): PL2
Login no *Mooshak:* 2020230563

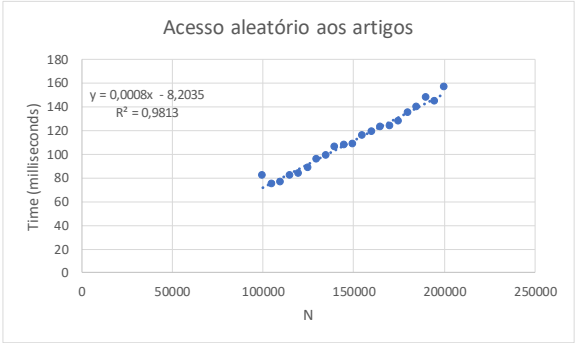
Correr a implementação do projeto 3.2 para um número crescente de acessos com dois cenários: (1) 90% dos acessos são feitos a 5% dos artigos (2) todos os artigos têm sensivelmente o mesmo número de acessos. Obter os tempos de execução (excluindo tempo de leitura e impressão de resultados). Produzir respetivas tabelas, gráficos e regressões relevantes.

Cenário 1



N	Times (millis)
100000	24
105000	17
110000	17
115000	17
120000	18
125000	21
130000	22
135000	24
140000	25
145000	26
150000	28
155000	27
160000	29
165000	30
170000	33
175000	37
180000	35
185000	36
190000	39
195000	45
200000	39

Cenário 2



N	Times (millis)
100000	82
105000	75
110000	77
115000	82
120000	84
125000	89
130000	96
135000	99
140000	106
145000	108
150000	109
155000	116
160000	119
165000	123
170000	124
175000	128
180000	135
185000	140
190000	148
195000	145
200000	157

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

Sim, os tempos de execução estão de acordo com o esperado.

Para consultar um nó numa árvore temos complexidade O(log n), para m nós O(m log n). Conseguimos obter uma complexidade O(n), algo melhor do que esperado.

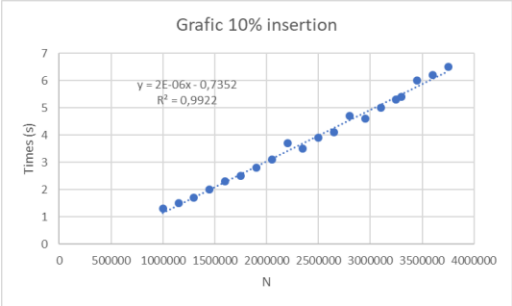
Os tempos do cenário 1 foram muito mais baixos em relação ao cenário 2, isto porque os nós mais acedidos ficam no topo da árvore e o acesso é muito mais rápido, o que é o oposto do cenário 2.

Relatório Projeto 3.3 AED 2021/2022

Nome: João Emanuel Sousa Moreira
Nº Estudante: 2020230563
PL (inscrição): PL2
Login no *Mooshak:* 2020230563

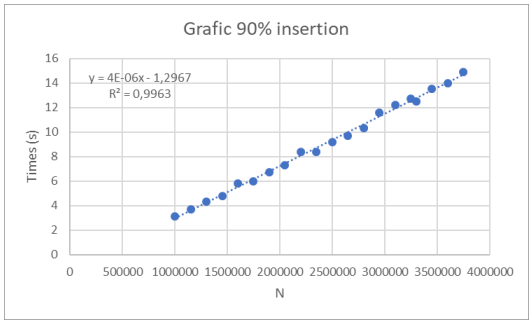
Correr a implementação do projeto 3.3 para um número crescente de registos/acessos com dois cenários: (1) 10% de inserções (2) 90% de inserções. Obter os tempos de execução (excluindo tempo de leitura e impressão de resultados). Produzir respetivas tabelas, gráficos e regressões relevantes.

Cenário 1



N	Times (s)
1000000	1,3
1150000	1,5
1300000	1,7
1450000	2
1600000	2,3
1750000	2,5
1900000	2,8
2050000	3,1
2200000	3,7
2350000	3,5
2500000	3,9
2650000	4,1
2800000	4,7
2950000	4,6
3100000	5
3250000	5,3
3300000	5,4
3450000	6
3600000	6,2
3750000	6,5

Cenário 2



N	Times (s)
1000000	3,1
1150000	3,7
1300000	4,3
1450000	4,8
1600000	5,8
1750000	6
1900000	6,7
2050000	7,3
2200000	8,4
2350000	8,4
2500000	9,2
2650000	9,7
2800000	10,3
2950000	11,6
3100000	12,2
3250000	12,7
3300000	12,5
3450000	13,5
3600000	14
3750000	14,9

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

Inserir um nó tem complexidade O(log n), inserir m nós tem complexidade O(m log n). O mesmo se passa para a pesquisa. Obtive uma complexidade linear, algo que não é pior do que aquilo que estava à espera.

Os tempos de "10% de inserção" são menores do que os de "90% de inserção", algo que já era esperado. Ao haver mais inserções, a árvore fica cada vez maior e na inserção necessita de percorrer até ao nível mais baixo e ainda precisa de ser balanceada. Enquanto que com mais consultas, poderei não ter de ir ao pior caso e a árvore esta mais pequena.

Relatório Projeto 3.4 AED 2021/2022

Nome: João Emanuel Sousa Moreira Nº Estudante: 2020230563
PL (inscrição): PL2 *Login* no *Mooshak: 2020230563*

Estrutura de Dados Principal usada em cada sub-projeto:

PROJ 3.1 M-ary tree

PROJ 3.2 Splay tree

PROJ 3.3 Avl tree

Estruturas de dados usadas	M-ary tree	Splay tree	Avl tree
VANTAGENS GERAIS (max 3)	• Ordenação por ordem de entrada • Poucos níveis de nós • Rápido acesso aos nós de níveis mais baixo	• Raiz com o último nó acessado • Rápido acesso aos últimos nós inseridos/consultados	• Rápida consulta • Árvore balanciada • Nós ordenados
DESVANTAGENS GERAIS (max 3)	• Complexidade temporal alta • Inserção não ordenada	• Desbalanciamento • Fazer splay a cada inserção/consulta	• Inserção lenta • Balanciamento lento para árvores grandes • Gasto de memória para guardar o fator de equilibrio
Justificação para a escolha no PROJ 3.1	Cada categoria tem m filhos, onde cada filho pode ter m filhos. Podemos usar uma m-ary tree, onde cada nó contem a categoria. Para obter o resultado esperado é só imprimir a árvore por níveis.		
Justificação para a escolha no PROJ 3.2	Vamos querer fazer multiplas consultas a determinados nós. Como a splay leva o nó consultado para a raiz, aqueles que são mais consultados estão no topo da árvore.		
Justificação para a escolha no PROJ 3.3	Visto que queremos acessar a multiplos nós de forma aleatória e fazer poucas inserções, usamos uma AVL onde ela tem os nós balanciados, ou seja, tem quase o mesmo tempo de acesso para todos os nós.		