PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

INSTITUTO DE CIÊNCIAS EXATAS E INFORMÁTICA

UNIDADE EDUCACIONAL PRAÇA DA LIBERDADE

Bacharelado em Engenharia de Software

Ravi Antônio Gonçalves de Assis

TRABALHO PRÁTICO II DE LABORATÓRIO DE COMPUTAÇÃO II

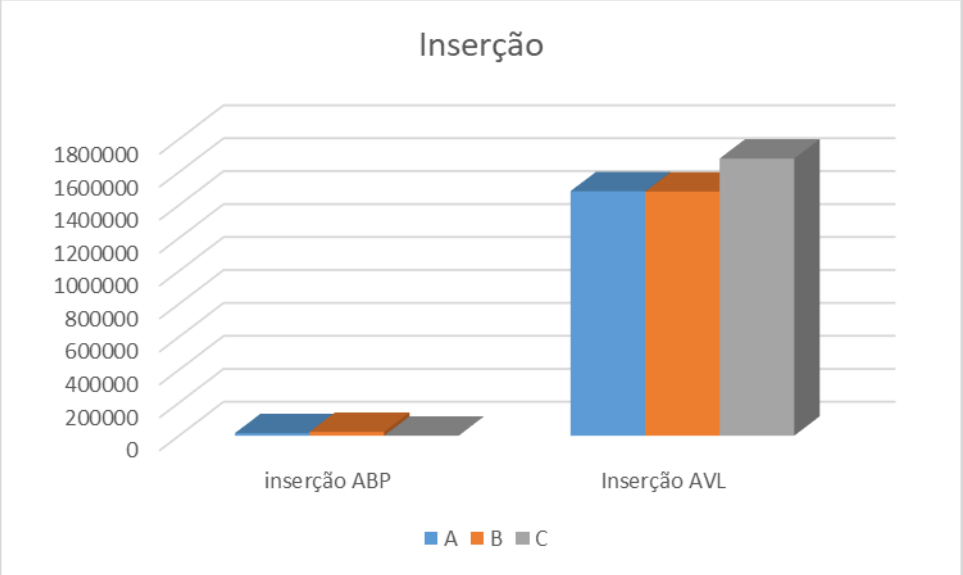
Belo Horizonte

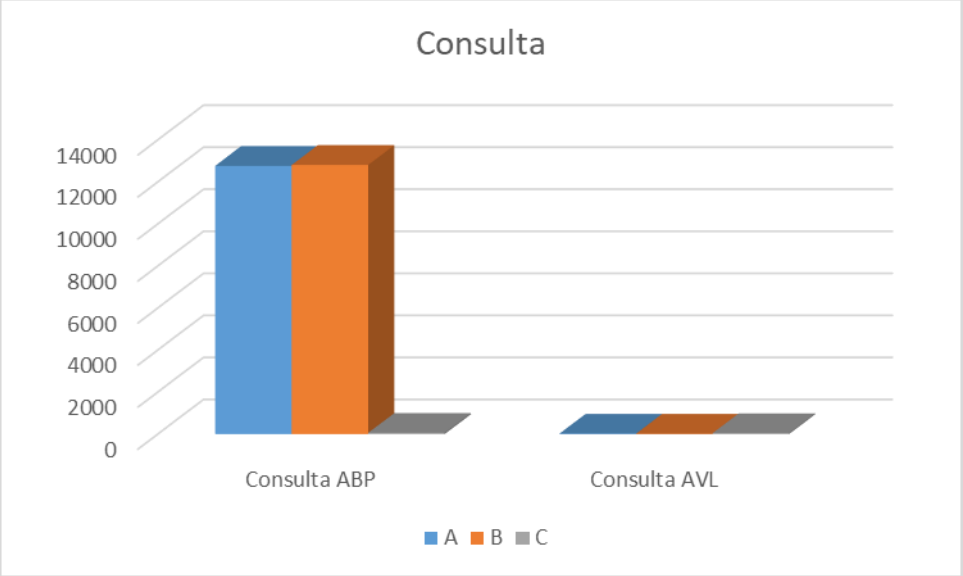
2018

O seguinte documento é um relatório evidência as análises e comparações realizadas em cima da performance e desempenho dos métodos de inserção e consulta das estruturas de dados baseadas em árvore binária (ABP) e árvore AVL.

Segue abaixo a tabela com os dados obtidos das análises de desempenho realizado nos algoritmos inserção e consulta das árvores e o gráfico de comparação.

	inserção ABP	Inserção AVL	Consulta ABP	Consulta AVL
A	15666	1484185	12764	21
B	23216	1482134	12807	18
C	58	1682436	59	44





Abaixo segue código com os testes de desempenho realizados.

```
public class Application {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		int tam = 100000;
		int p = 1000;
		int [] vetA = new int[tam];
		int [] vetB = new int[tam];
		int [] vetC = new int[tam];
		CArvBin bTreeA = new CArvBin();
		CArvBin bTreeB = new CArvBin();
		CArvBin bTreeC = new CArvBin();
		TreeAVL avlTreeA = new TreeAVL();
		TreeAVL avlTreeB = new TreeAVL();
		TreeAVL avlTreeC = new TreeAVL();
		long start;
		long end;
		vetA = fillVetA(vetA);
		vetB = fillVetB(vetB);
		vetC = fillVetC(vetC);


		System.out.print("Inserir vetor A na Árvore Binária");
		start = System.currentTimeMillis();
		for (int i = 0; i < vetA.length; i++) {
			bTreeA.insereFor(vetA[i]);
			if ( i != 0 && i % p == 0) System.out.print(".");
		}
		end = System.currentTimeMillis();
		System.out.println("\nTempo: " + (end - start) + "
ms");

		System.out.print("Consultar vetor A na Árvore
Binária");
		start = System.currentTimeMillis();
		for (int i = 0; i < vetA.length; i++) {
			bTreeA.pesquisaFor(vetA[i]);
			if ( i != 0 && i % p == 0) System.out.print(".");
		}
		end = System.currentTimeMillis();
		System.out.println("\nTempo: " + (end - start) + "
ms");
		System.out.println();

		System.out.print("Inserir vetor B na Árvore Binária");
		start = System.currentTimeMillis();
		for (int i = 0; i < vetB.length; i++) {
			bTreeB.insereFor(vetB[i]);
```

```
            if ( i != 0 && i % p == 0) System.out.print(".");
        }
        end = System.currentTimeMillis();
        System.out.println("\nTempo: " + (end - start) + "
ms");

        System.out.print("Consultar vetor B na Árvore
Binária");
        start = System.currentTimeMillis();
        for (int i = 0; i < vetB.length; i++) {
            bTreeB.pesquisaFor(vetB[i]);
            if ( i != 0 && i % p == 0) System.out.print(".");
        }
        end = System.currentTimeMillis();
        System.out.println("\nTempo: " + (end - start) + "
ms");
        System.out.println();

        System.out.print("Inserir vetor C na Árvore Binária");
        start = System.currentTimeMillis();
        for (int i = 0; i < vetC.length; i++) {
            bTreeC.insereFor(vetC[i]);
            if ( i != 0 && i % p == 0) System.out.print(".");
        }
        end = System.currentTimeMillis();
        System.out.println("\nTempo: " + (end - start) + "
ms");


        System.out.print("Consultar vetor C na Árvore
Binária");
        start = System.currentTimeMillis();
        for (int i = 0; i < vetC.length; i++) {
            bTreeC.pesquisaFor(vetC[i]);
            if ( i != 0 && i % p == 0) System.out.print(".");
        }
        end = System.currentTimeMillis();
        System.out.println("\nTempo: " + (end - start) + "
ms");
        System.out.println();

        System.out.print("Inserir vetor A na Árvore AVL");
        start = System.currentTimeMillis();
        for (int i = 0; i < vetA.length; i++) {
            avlTreeA.insert(vetA[i], vetA[i]);
            if ( i != 0 && i % p == 0) System.out.print(".");
        }
        end = System.currentTimeMillis();
        System.out.println("\nTempo: " + (end - start) + "
ms");

        System.out.print("Consultar vetor A na Árvore AVL");
```

```
            start = System.currentTimeMillis();
            for (int i = 0; i < vetA.length; i++) {
                 avlTreeA.search(vetA[i]);
                 if ( i != 0 && i % p == 0) System.out.print(".");
            }
            end = System.currentTimeMillis();
            System.out.println("\nTempo: " + (end - start) + "
ms");
            System.out.println();

            System.out.print("Inserir vetor B na Árvore AVL");
            start = System.currentTimeMillis();
            for (int i = 0; i < vetB.length; i++) {
                 avlTreeB.insert(vetB[i], vetB[i]);
                 if ( i != 0 && i % p == 0) System.out.print(".");
            }
            end = System.currentTimeMillis();
            System.out.println("\nTempo: " + (end - start) + "
ms");

            System.out.print("Consultar vetor B na Árvore AVL");
            start = System.currentTimeMillis();
            for (int i = 0; i < vetB.length; i++) {
                 avlTreeB.search(vetB[i]);
                 if ( i != 0 && i % p == 0) System.out.print(".");
            }
            end = System.currentTimeMillis();
            System.out.println("\nTempo: " + (end - start) + "
ms");
            System.out.println();

            System.out.print("Inserir vetor C na Árvore AVL");
            start = System.currentTimeMillis();
            for (int i = 0; i < vetC.length; i++) {
                 avlTreeC.insert(vetC[i], vetC[i]);
                 if ( i != 0 && i % p == 0) System.out.print(".");
            }
            end = System.currentTimeMillis();
            System.out.println("\nTempo: " + (end - start) + "
ms");

            System.out.print("Consultar vetor C na Árvore AVL");
            start = System.currentTimeMillis();
            for (int i = 0; i < vetC.length; i++) {
                 avlTreeC.search(vetC[i]);
                 if ( i != 0 && i % p == 0) System.out.print(".");
            }
            end = System.currentTimeMillis();
            System.out.println("\nTempo: " + (end - start) + "
ms");

      }
```

```
	private static int[] fillVetC(int[] vet) {
		// TODO Auto-generated method stub
		Random r = new Random();
		for (int i = 0; i < vet.length; i++) {
			vet[i] = r.nextInt(vet.length);
		}
		return vet;
	}


	private static int[] fillVetB(int[] vet) {
		// TODO Auto-generated method stub
		int tam = vet.length;
		for (int i = 0; i < tam; i++) {
			vet[i] = tam -i;
		}
		return vet;
	}

	private static int[] fillVetA(int [] vet) {
		// TODO Auto-generated method stub
		for (int i = 0; i < vet.length; i++) {
			vet[i] = i+1;
		}
		return vet;
	}

}
```