

FACULDADE ANHANGUERA DE BELO HORIZONTE

Sistemas de Informação

CLASSIFICAÇÃO E PESQUISA

Unidade Andradas

2014

FACULDADE ANHANGUERA DE BELO HORIZONTE

UNIDADE ANDRADAS

ATPS – ETAPA 1,3 e 4

Professor – Juliana Santiago

**AFONSO RODRIGUES RA 3226019666**

**BRUNO TIAGO DUARTE PIMENTA RA 3218523299**

**DIEGO RICHELLY RA 3226040723**

**FERNANDA LEONCIO RA 3226019669**

**GLEIZIANE FERNANDA RA 2504108831**

**MACILENE MARIA DA SILVA RA 3245550171**

BELO HORIZONTE, 05 DE JUNHO DE 2014

Sumário

[Introdução 4](#_Toc389696054)

[Sobre o SQLAlchemy 4](#_Toc389696055)

[Etapa 1 4](#_Toc389696056)

[Analise dos Algoritmos de busca 4](#_Toc389696057)

[Considerações Etapa 1 6](#_Toc389696058)

[Etapa 3 7](#_Toc389696059)

[Analise dos Algoritmos de Arvore 7](#_Toc389696060)

[Analise dos Resultados 7](#_Toc389696061)

[Arvore Binária 8](#_Toc389696062)

[Arvore AVL 8](#_Toc389696063)

[Arvore B 8](#_Toc389696064)

[Etapa 4 9](#_Toc389696065)

[Contextualização das Etapas 1 e 2 9](#_Toc389696066)

[Melhores práticas 9](#_Toc389696067)

[Considerações finais 9](#_Toc389696068)

[Bibliografia 9](#_Toc389696069)

# Introdução

Este relatório visa apresentar as analises requerida no ATPS da matéria de classificação e pesquisa.

O trabalho foi implementado na linguagem Python associado ao framework SQLAlchemy.

Os testes foram realizados em uma máquina com processador Intel Core i5, 6 GB de RAM e Windows 7.

## Sobre o SQLAlchemy

Conforme descrito no site do SQLAlchemy[[1]](#footnote-1), é uma biblioteca Open Source de mapeamento objeto-relacional desenvolvida em Python. Ela está na versão 0.9.4 (versão utilizada no trabalho) e está sob a licença MIT.

Ela fornece uma gama de modelos e métodos para persistência em Banco de Dados. Em suma ela é uma interface de comunicação com o banco, simplificando e otimizando a sua utilização.

Neste trabalho este framework foi utilizado para persistir as informações da coleta de dados como tempo de execução, quantidade de trocas e quantidade de comparações.

# Etapa 1

## Analise dos Algoritmos de busca

Os algoritmos de busca implementados (Busca Linear, Busca Linear com Sentinela e Busca Binária) apresentaram-se extremamente eficientes quando utilizados na máquina utilizada para testes. Demonstraram valores diferentes para o numero de comparações, porém com tempo de execução igual à 0 com os valores pré-definidos no ATPS.

Na tabela execução etapa 1 é demonstrado os valores abordados:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tipo | Dados em ordem Aleatória | | | | | | | | |
| Instância | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | |
| Dados | 10.000 | | | 30.000 | | | 50.000 | | |
| Algoritimo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo |
| Busca Binaria | 390 | 0 | 0 | 430 | 0 | 0 | 440 | 0 | 0 |
| Busca Linear | 100000 | 0 | 0 | 300000 | 0 | 0 | 304550 | 0 | 0 |
| Busca Linear Sentinela | 100020 | 0 | 0 | 300020 | 0 | 0 | 304560 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |
| Tipo | Dados em ordem Decrescente | | | | | | | | |
| Instância | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | |
| Dados | 10.000 | | | 30.000 | | | 50.000 | | |
| Algoritimo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo |
| Busca Binaria | 370 | 0 | 0 | 400 | 0 | 0 | 410 | 0 | 0 |
| Busca Linear | 99130 | 0 | 0 | 299130 | 0 | 0 | 499130 | 0 | 0 |
| Busca Linear Sentinela | 99140 | 0 | 0 | 299140 | 0 | 0 | 499140 | 0 | 0 |
|  |  |  |  |  |  |  |  |  |  |
| Tipo | Dados em ordem Crescente | | | | | | | | |
| Instância | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | |
| Dados | 10.000 | | | 30.000 | | | 50.000 | | |
| Algoritimo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo |
| Busca Binaria | 370 | 0 | 0 | 400 | 0 | 0 | 370 | 0 | 0 |
| Busca Linear | 880 | 0 | 0 | 880 | 0 | 0 | 880 | 0 | 0 |
| Busca Linear Sentinela | 890 | 0 | 0 | 890 | 0 | 0 | 890 | 0 | 0 |

Tabela 1 - Execução Etapa 1

Para poder indicar qual algoritmo é mais eficiente quanto ao desempenho será analisado a complexidade por número de comparações e execução.

|  |  |  |  |
| --- | --- | --- | --- |
| Algoritimo | Complexidade melhor caso | Complexidade médio caso | Complexidade pior caso |
| Busca Binaria | O(1) | O (n/2) | O(n) |
| Busca Linear | O(1) | O (n/2) | O(n) |
| Busca Linear Sentinela | O(1) | O (n/2) | O(log n) |

Tabela 2 - Complexidade Etapa 2

É possível verificar que quem apresenta a melhor complexidade é o algoritmo de busca binária onde o próprio é demonstrado na Tabela 1 com as menores quantidades de comparações.

## Considerações Etapa 1

Com a realização desta etapa foi possível verificar a eficiência do algoritmo de busca binária quanto ao numero de comparações e aprimorar a implementação do ATPS.

# Etapa 3

## Analise dos Algoritmos de Arvore

### Analise dos Resultados

A Tabela Execução Etapa 3 apresenta os valores coletados nesta Etapa:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tipo | Dados em ordem Aleatória | | | | | | | | |
| Instância | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | |
| Dados | 10.000 | | | 30.000 | | | 50.000 | | |
| Algoritimo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo |
| Arvore AVL | 4098250 | 1486870 | 10 | 12251510 | 4335460 | 30 | 19388470 | 6783110 | 50 |
| Arvore B-Tree | 400000 | 300000 | 0 | 1200000 | 900000 | 0 | 2000000 | 1500000 | 10 |
| Arvore Binaria | 3882880 | 1609220 | 0 | 13596680 | 5575450 | 10 | 23137940 | 9508120 | 20 |
|  |  |  |  |  |  |  |  |  |  |
| Tipo | Dados em ordem Decrescente | | | | | | | | |
| Instância | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | |
| Dados | 10.000 | | | 30.000 | | | 50.000 | | |
| Algoritimo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo |
| Arvore AVL | 4170550 | 1049040 | 10 | 13442700 | 3148940 | 30 | 23187240 | 5248900 | 40 |
| Arvore B-Tree | 400000 | 300000 | 0 | 1200000 | 900000 | 0 | 2000000 | 1500000 | 10 |
| Arvore Binaria | 1000000000 | 500050000 | 840 | 9000000000 | 4500150000 | 7600 | 25000000000 | 12500250000 | 21280 |
|  |  |  |  |  |  |  |  |  |  |
| Tipo | Dados em ordem Crescente | | | | | | | | |
| Instância | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | | Média entre instâncias 1,2 e 3 | | |
| Dados | 10.000 | | | 30.000 | | | 50.000 | | |
| Algoritimo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo | Nº de Comparações | Nº de Trocas | Tempo |
| Arvore AVL | 4070690 | 1248760 | 10 | 13142850 | 3748640 | 30 | 4070690 | 1248760 | 10 |
| Arvore B-Tree | 400000 | 300000 | 0 | 1200000 | 900000 | 0 | 400000 | 300000 | 0 |
| Arvore Binaria | 1499950000 | 500050000 | 930 | 13499850000 | 4500150000 | 8710 | 1499950000 | 500050000 | 930 |

Tabela 3 - Execução Etapa 3

### Arvore Binária

Entre os três tipos de arvores abordados neste ATPS, a arvore binária é a que possui a implementação mais simples e que se mostra com o maior custo computacional. Na Tabela Execução Etapa 3 demonstra os valores realizados por ela.

Tem se como aplicação a utilização de busca por de conteúdo.

### Arvore AVL

Esta arvore apresentou-se com uma execução e performance boa, possui uma implementação um pouco mais complexa. Na Tabela Execução Etapa 3 demonstra os valores realizados por ela.

Tem se como aplicação a utilização de busca por de conteúdo.

### Arvore B

Entre os três tipos de arvores abordados neste ATPS, a arvore binária é a que possui a implementação mais complexo e que mostra-se com o melhor desempenho. Na Tabela Execução Etapa 3 demonstra os valores realizados por ela.

Tem se como aplicação em diversos bancos de dados devido a sua eficiência.

# Etapa 4

## Contextualização das Etapas 1 e 2

As etapas 1 e 2 apresentam as analises das performance de algoritmos de busca e ordenação. Podemos aplica-los no cenários mais comumente em analises de bases de dados e desempenho computacional. Com o aumento da quantidade de dados disponível é possível verificar a necessidade meio eficientes de busca e ordenação.

## Melhores práticas

Para exemplificar a escolha de uma solução para o negócio abordado por um possível cliente, vamos considerar os pontos abordados neste trabalho.

Podemos considerar a complexidade do algoritmo usado aliado ao desempenho na máquina e da linguagem escolhida.

Conforme descrito na documentação do Python que é implementado um limite na pilha de recursão, para evitar que o programa entre em recursão infinita.

É possível verificar que na própria implementação do Python é utilizado um algoritmo de ordenação chamado Tim Sort onde é uma mescla do Merge Sort e Insertion Sort tendo excelente performance (O (n log n)), sendo ele o escolhido para implementação padrão da linguagem.

## Considerações finais

Com a realização deste trabalho, foi possível verificar as formas de analise de complexidade e desempenho de um algoritmo relacionado ao seu contexto.

A implementação dos códigos propostos na linguagem Python associado a um framework de persistência.

A busca pelo melhor caminho e considerações de ambiente e execução, como a escolha de um limite para a pilha de recursão.

# Bibliografia

Python, Limite de Recursão <<https://docs.python.org/2/library/sys.html>>, acesso em 04 de Junho de 2014.

TimSort, Algortimo de Ordenação <http://svn.python.org/projects/python/trunk/Objects/listsort.txt>, acesso em 04 de Junho de 2014.

1. Site SQLAlchemy: http://www.sqlalchemy.org/ [↑](#footnote-ref-1)