APLICAÇÃO DA AVALIAÇÃO EMPÍRICA E ASSINTÓTICA ENTRE MÉTODOS DE ORDENAÇÃO DE DADOS

NASCIMENTO, Jonathan da Silva[1]
CAS, Bruno.D[2]
LOPES, Luiz Henrique[3]
SOUZA, Kael de[4]
TEIXEIRA, Jader[5]
CHICON, Patricia Mariotto Mozzaquatro[6]
ANTONIAZZI, Rodrigo Luiz[7]

Resumo: Este artigo foi desenvolvido na disciplina de Estrutura de Dados ministrada no curso de Ciência da Computação, o mesmo tem por finalidade demonstrar testes de comparação entre algoritmos de ordenação de dados avaliando-os de forma empírica e assintótica, ou seja, foram validados vetores dispostos em ordem crescente, decrescente e aleatória, analisando a performance sobre cada uma deles. Por meio de tabelas foram demonstradas a quantidade de comparações e trocas efetuadas, além de exibir o tempo de execução de cada método. Os algoritmos utilizados foram o bubblesort, inserção direta, combsort e shellsort, implementados na linguagem de programação Java e C.

Palavras-chave: Avaliação Empírica. Avaliação Assintótica. Métodos de Ordenação.

Abstract: *This scientific paper was developed in the data structure class, ministered in the computing science course ,the same which has the goal of showing tests comparing data sorting algorithms , evaluating them empirically and asymptotically , which means, that vectors in ascending ,descending and random order were evaluated, analyzing the performance of each one of them .The amount of comparisons and switches were shown through tables, besides showing the runtime of each method. The algorithms used were bubblesort ,direct insertion, combosort and shellsort, implemented through java and C.*

Keywords: *Empirical evaluation. Asymptotic evaluation. Sort methods.*

1. INTRODUÇÃO

É importante que profissionais da área da computação tenham conhecimento sobre os métodos para ordenação de dados, pois, os profissionais podem se debater com alguma situação que eles terão que usar. Pode-se citar como exemplo a lista de contatos do chat do Facebook, que possui os nomes ordenados em ordem alfabética. Toda vez que o usuário

[1] Acadêmico do Curso de Ciência da Computação.
[2] Acadêmico do Curso de Ciência da Computação.
[3] Acadêmico do Curso de Ciência da Computação
[4] Acadêmico do Curso de Ciência da Computação
[5] Acadêmico do Curso de Ciência da Computação
[6] Professora do Curso de Ciência da Computação
[7] Professor do Curso de Ciência da Computação

adiciona novos contatos a lista de nomes é atualizada. A ideia é a mesma da lista de contatos de uma agenda telefônica, e "para isso ocorrer é necessário utilizar um bom algoritmo de ordenação" (Ribeiro et al.,2004).

Este artigo tem por objetivo desenvolver um estudo comparativo entre os métodos de ordenação de dados bolha, inserção direta, sehelsort e combsort. Ambos serão implementados na linguagem de programação C e em Java. Compõe esta pesquisa os seguintes tópicos: Revisão de literatura, isto é, estudo sobre ordenação de dados e os métodos citados, estudo sobre as formas de avaliação empírica e assintótica, metodologia de pesquisa, implementação dos métodos, considerações finais e referências.

2. REVISÃO DA LITERATURA

As subseções a seguir irão abordar os métodos de ordenação de dados bolha, inserção direta, shelsort e combosort. Ainda irá abordar a avaliação empírica e assintótica.

2.1 Métodos de Ordenação de Dados

Os métodos de ordenação foram desenvolvidos com o objetivo de tornar mais simples viável e rápido recuperar determinada informação inserida em um grande conjunto de dados, os métodos de ordenação buscam rearranjar um conjunto de informações em uma certa ordem como ascendente ou descendente de forma a facilitar a recuperação posterior ou analise dos dados (MAGALHÃES et al., 2008). Neste artigo serão abordados os seguintes métodos de ordenação: Bolha, Inserção Direta, Shelsort e Combosort.

Bubble sort é o método com a implementação mais simples porem é o menos eficiente dentre os métodos estudados, basicamente ele faz varreduras no vetor comparando os dados em pares, sempre que o elemento posterior for menor que o anterior o método troca as posições e compara com o próximo, se não, ele continua a varredura a partir do elemento maior, assim quando terminar a varredura aquele elemento vai estar na posição certa não necessitando mais ser verificado. Uma desvantagem encontrada é que mesmo que o vetor já esteja ordenado o algoritmo vai fazer todas as varreduras (JUNIOR, 2008).

O método de inserção direta tem a complexidade de implementação parecida com o bubblesort, e é eficiente em vetores pequenos (com até 20 posições), seu modo de funcionamento é muito simples, ele percorre o vetor procurando o menor valor e o transfere para a posição certa, além disso, é extremamente eficiente em vetores que já estão

parcialmente ordenados. Tem como desvantagem o grande numero de trocas que realiza e é ineficiente em vetores ordenados inversamente (RIBERIO et al., 2004)

Shell sort é uma extensão do método inserção direta, este algoritmo cria um gap dividindo o numero de elementos do vetor por 2, e fazendo a verificação neste gap, após o fim da varredura ele divide o numero de elementos pelo gap anterior e prosseguindo com as varreduras até encontrar o gap igual a um, assim que chega ao gap 1, passa-se a aplicar o método de inserção direta no vetor até terminar a ordenação, assim como o inserção direta este método é mais eficiente em vetores parcialmente ordenados.

Os principais problemas deste algoritmo são a instabilidade do código e o tempo de execução sensível à ordem inicial do arquivo (OLIVEIRA, 2002).

O método Combsort reordena o vetor de maneira semelhante ao bubblesort, porém, utilizando uma técnica parecida com o shellsort, neste algoritmo divide-se o numero de elementos por 1,3 na primeira varredura, e após divide o gap anterior por 1,3, até que o gap seja 1, assim, quando se encontra o gap igual a um, o processo é feito como um bubble sort normal OLIVEIRA, 2002).

2.2 Avaliação Empírica e Assintótica

A avaliação assintótica considera o numero de operações funcionais que o algoritmo vai realizar em relação ao volume de dados de entrada. Assim, "o comportamento assintótico pode ser definido como o comportamento de um algoritmo para grandes volumes de dados de entrada." (AVELLAR, 2008).

Este método de avaliação também leva em conta a complexidade temporal. segundo Garcia (2012), o tempo de execução de um algoritmo pode ser representado por uma função de custo que representa a medida de tempo necessária para a execução de um algoritmo de determinado tamanho. Porem deve-se enfatizar que esta função de tempo não representa o tempo de execução do algoritmo e sim o número de vezes que as operações relevantes são executadas.

Este método também considera três situações de um algoritmo: o melhor caso, onde se necessita menos tempo de execução e operações essenciais, o pior caso que é o maior tempo que o algoritmo necessita para executar determinada instancia, e o caso médio que considera o tempo médio de execução de um algoritmo. Geralmente é considerado o pior caso que avalia o algoritmo em situações complexas e o caso médio que diz como será a execução do algoritmo na maioria dos casos.

Por outro lado a avaliação empírica define regras para os testes de um algoritmo onde pode se definir por observação o melhor, o pior e o caso médio, para tanto são feitos testes em diferentes instancias com diferentes volumes de dados ou ordenações. Onde, geralmente procura-se saber o tempo de execução que é medido em segundos ou milissegundos. Porem, esta forma de avaliação pode variar muito dependendo do hardware e do software que vai ser utilizado (AVELLAR, 2008)..

3. METODOLOGIA

Este artigo é parte integrante de um trabalho desenvolvido na disciplina de estrutura de dados II por acadêmicos do curso de Ciência da Computação. O trabalho foi desenvolvido nas seguintes etapas:

Na etapa 1 foi desenvolvido estudo teórico sobre ordenação de dados, estudo sobre os seguintes métodos: bolha, inserção direta, shelsort e combsort;

Na etapa 2 foi desenvolvido pesquisa sobre a formas de avaliação empírica e assintótica;

Na etapa 3 foram implementados os métodos citados na linguagem C;

Na etapa 4 foram implementados os métodos citados em Java;

Na etapa 5 foi realizado um estudo comparativo entre os métodos de forma empírica e assintótica.

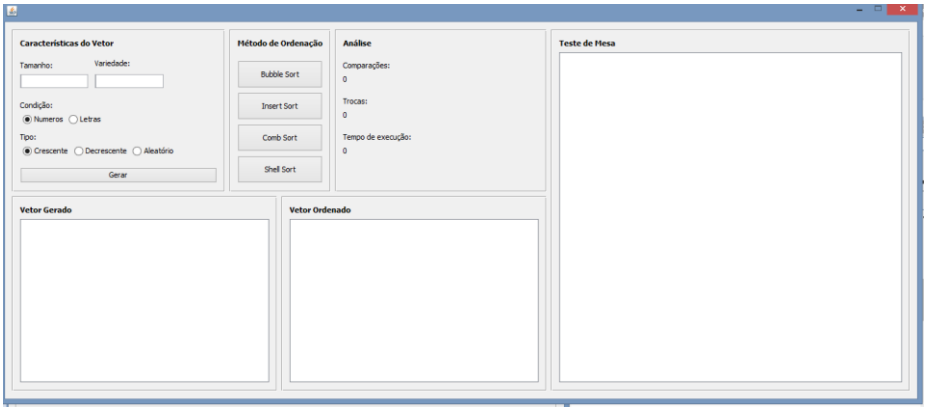
4. RESULTADOS

A seção dos resultados aborda as formas de avaliação empírica e assintótica da implementação dos vetores na linguagem C e em Java. Foram analisados vetores em ordem crescente, decrescente e aleatória. A Figura 1 ilustra a tela inicial do sistema.

Figura 1. Tela Inicial do sistema

A Figura 2 mostra a tela com será apresentado o resultado.

Figura 2. Tela dos resultados



A Tabela 1 apresenta a avaliação empírica na linguagem C e em Java.

Tabela 1. Avaliação Empírica

		FX-8350		i7-4510U		i5-2450M	
		Linguagem C	Java	Linguagem C	Java	Linguagem C	Java
Bubble	1000	0.000	0.015	0.000	0.000	0.000	0.010
	10000	0.187	0.109	0.203	0.078	0.220	0.040
	100000	18.564	9.782	19.063	11.375	22.389	10.218
Inserção	1000	0.015	0.016	0.000	0.015	0.000	0.000
	10000	0.125	0.047	0.156	0.031	0.150	0.040
	100000	13.400	3.058	14.766	2.437	14.315	3.800
Comb	1000	0.000	0.000	0.000	0.000	0.000	0.000
	10000	0.000	0.000	0.000	0.000	0.000	0.000
	100000	0.000	0.015	0.015	0.016	0.010	0.010
Shell	1000	0.000	0.000	0.000	0.000	0.000	0.000
	10000	0.000	0.000	0.000	0.000	0.000	0.000

	100000	0.000	0.000	0.000	0.000	0.000	0.000

A Tabela 2 mostra a avaliação assintótica na linguagem C e em Java.

Tabela 2. Avaliação Assintótica

		FX-8350	
		Linguagem C	Java
Bubble	1000	499500	499500
	10000	49995000	49995000
	100000	704982704	704982704
Inserção	1000	499500	499500
	10000	49995000	49995000
	100000	4999950000	4999950000
Comb	1000	1582	1582
	10000	20078	20078
	100000	244174	244174
Shell	1000	3790	3790
	10000	47302	47302
	100000	517106	517106

Foram avaliados quatro métodos de ordenação implementados em duas linguagens diferentes, no qual chegou-se as seguintes conclusões:

1. BubbleSort: Teve o pior desempenho dos métodos testados em ambas linguagens, um dos motivos é a quantidade de comparações desnecessárias no qual influência o seu tempo de execução. A linguagem Java obteve melhor desempenho, executando a ordenação na metade do tempo que em C.

2. Inserção direta: O desempenho com relação a outros métodos depende da ordenação do vetor em sua fase inicial, sendo que este método é mais utilizado quando o vetor está parcialmente ordenado. Sendo que estamos comparando métodos com o pior caso, este mostrou-se tão ineficaz quanto o bubblesort com relação a trocas, sendo mais efetivo apenas relacionada ao tempo de execução. Como no bubblesort, a linguagem Java teve melhor desempenho.

3. CombSort: Demonstrou ter bom desempenho quando se trata de vetores desordenados em ordem decrescente, pois teve o menor índice de trocas em relação a outros métodos. Quanto ao tempo de execução comparando as linguagens utilizadas, comparou-se ao shellsort. A linguagem C obteve melhor tempo de execução neste caso.

4. ShellSort: Este método mesmo além de realizar mais trocas que o combsort, obteve melhor tempo de execução. Isto deve-se a segunda etapa do método, no qual ordena a lista com o método insertsort, sendo que no combsort é utilizado o bubblesort. Não foi possível comparar as linguagens, pois o tempo de execução do método não foi significativo em ambas.

5. CONSIDERAÇÕES FINAIS

Conclui-se então, que a escolha do algoritmo de ordenação a ser usado depende da lista que ele irá ordenar, do tamanho que essa lista tem e do quão desordenados estão os elementos. Foi notado que o algoritmo bubblesort tem um desempenho baixo em relação aos outros, constatando então que esse possivelmente seria descartado.

Baseado nos testes realizados até então, poderíamos constatar que os algoritmos sofisticados tem maior efetividade na ordenação, e seria aconselhável usá-los para ordenação de listas de tamanho médio e moderadamente grandes. Quanto aos algoritmos simples, no caso o inserção direta, possivelmente seria usado em uma lista de tamanho pequeno onde os elementos do vetor estão quase totalmente ordenados.

Quanto a linguagem de programação seria apenas possível fazer uma análise empírica sobre elas, sendo que nos testes realizados o Java foi a linguagem que obteve melhor desempenho na ordenação de listas decrescente.

Com os conhecimentos obtidos na disciplina de estrutura de dados pode-se aplicar na prática o presente estudo comparativo.

REFERÊNCIAS BIBLIOGRAFICAS

AVELLAR, Alex. APOSTILA I DA DISCIPLINA ANALISE DE ALGORITMOS. 2008. Disponível em: < http://www.sergio.oliveira.nom.br/apostila-complexidade.pdf>. Acesso em: mai. De 2015.

GARCIA, Andre Mendes. **Complexidade de algoritmos**. 2012. Disponível em:< https://prezi.com/4t2nvclqthvo/metodos-de-ordenacao/>. Acesso em: mai de 2015.

JUNIOR, Antonio Carlos de Nazare. **Algoritmos e estrutura de dados:** Métodos de ordenação interna. Ouro preto; 2008. Disponível em: <http://www.decom.ufop.br/menotti/aedI082/tps/tp3-sol1.pdf>. Acesso em: mai de 2015

OLIVEIRA, ALVARO BORGES DE. **Métodos de Ordenação Interna**. Visual Book, São Paulo, 1st edition, 2002.

MAGALHÃES, L. H. de; BATISTA, M. de L. S.; MAGALHÃES, T. M.; BATISTA, T. J. S. **Análise da Complexidade de Algoritmos de Ordenação.** Disponível em: <http://www.viannajr.edu.br/files/uploads/20140313_174444.pdf.> Acesso em mai de 2015.

RIBEIRO, ÚRSULA A. L. F.; QUARTIERI, FERNANDA; ROSSI, RICARDO BRASILIENSE ; NAIMAIER, CARLOS HENRIQUE ; DEPRA, DIEISON ANTONELLO ; KINDEL, CÂNDICE CORRADI ; MENEZES JUNIOR, JORGE, ALBERTO MESSA . SAEMOR - Sistema de apoio ao ensino de Métodos de Ordenação de dados. **In: Workshop de Computação da Região Sul**, 2004, Florianópolis. Anais do I WorkComp-Sul, 2004.