COMPARATIVO ENTRE OS MÉTODOS DE ORDENAÇÃO RADIXSORT E COUNTINGSORT

SILVA, Mauro Rafael R[1].; CAMPOS, Luiz Henrique[2;] SCHRAMMEL, Lucca Alexandre[3]; CHICON, Patricia Mariotto Mozzaquatro[4]; TELOCKEN, Alex Vinícios[5]; SCHUCH, Regis Rodolfo[6]; ANTONIAZZI, Rodrigo Luiz[7]

Palavras-Chave: Ordenação de dados. Radixsort. Countingsort.

INTRODUÇÃO

Visando o grande volume de dados e informações disponíveis na web, torna-se difícil filtrar as mesmas com precisão. Os métodos de ordenação têm um papel fundamental na recuperação e organização desses dados e informações , onde as técnicas computacionais contribuem com a otimização de tempo e agilidade nos processos de ordenação e recuperação. Este resumo é parte integrante de um trabalho de conclusão de curso o qual objetivou a implementação de um método híbrido integrando os métodos Radixsort, Countingsor e Quicksort.

MÉTODOS DE ORDENAÇÃO DE DADOS

A ordenação consiste em arranjar os elementos de um conjunto de modo a facilitar a posterior recuperação ou análise dos dados. Por ser tão importante a ordenação é uma das atividades mais utilizadas na computação (BITENCOURT, 2014).

O método de ordenação *Radixsort* utiliza o conceito de ordenação por contagem, porém ele realiza a contagem com apenas uma parte da representação do elemento, a qual é denominada raiz. O processo de contagem é realizado com esta parte várias vezes até que a representação total do elemento seja analisada. Duas classificações podem ser realizadas com o *radixsort*, LSD (*Least Significant Bits*) e MSD (*Most Significant Bits*), que define a ordem que serão analisadas as partes do elemento, se LSD o processo é iniciado com os bits menos

[1] Discente em Ciência da Computação na Universidade de Cruz Alta - UNICRUZ. E-mail: maurorafael@outlook.com
[2] Discente em Ciência da Computação na Universidade de Cruz Alta - UNICRUZ. E-mail: ziquedc@gmail.com
[3] Discente do Curso de Ciência da Computação, UNICRUZ. E-mail: lucca.a.s@hotmail.com
[4] Professora Ciência da Computação na Universidade de Cruz Alta - UNICRUZ. E-mail: pmozzaquatro@unicruz.edu.br
[5] Professor, UNICRUZ. E-mail: alextelocken@gmail.com
[6] Professor, UNICRUZ. E-mail: regis.schuch@gmail.com
[7] Professor, UNICRUZ. E-mail: rantoniazzi@unicruz.edu.br

significativos e se MSD com os bits mais significativos. Com esta especificação do *radixsort* apresentada acima é possível perceber que este algoritmo pode ser aplicado a ordenação de *strings* como um vetor de entrada [bg, bd, a, f k, f j, e] sendo ordenado como [a, bd, bg, e, f j, f k]. Este algoritmo é muito utilizado para ordenação lexicográfica (MIYAZAWA, 1999). Para ordenação de números inteiros o *radixsort* pode trabalhar uma maneira simples, como a parte do elemento sendo um dígito do elemento, assim a ordenação pode ser feita ordenando os dígitos menos significativos para os mais significativos.

Segundo (OLIVEIRA; SOUZA, 2008) o algoritmo *Radixsort* sequencial tem complexidade O(n) Cormen et al. (2002). Sua versão paralela, quando comparada a sequencial, e bastante superior, pois apresenta, assim como o *Countingsort*, uma complexidade de tempo de O(log n) Grama et al(1994).

Countingsort é um método de ordenação linear, requer o tamanho máximo de um valor dentro da sequência para então utilizar uma sequência virtual de contagem que tem este tamanho máximo, fazendo com que todos os valores da sequência original sejam contados e que seja colocado na posição referente a seu valor nesta sequência virtual a quantidade de valores referentes a cada posição (LIMA JUNIOR; ARAUJO, 2003).

O algoritmo percorre o vetor auxiliar com os valores acumulados gerando um terceiro vetor, vetor resposta, pegando o primeiro valor do vetor principal (zero), verificando a posição do índice que esse valor se encontra no vetor auxiliar, subtraindo 1 do valor encontrado no índice, e logo após inserindo o valor do vetor principal na posição encontrada na subtração realizada, repetindo o processo até o que o vetor se encontre corretamente ordenado.

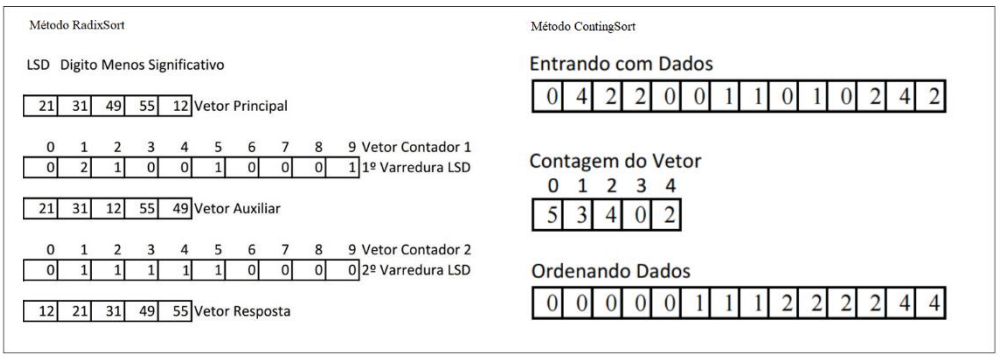
A complexidade assintótica de tempo de computação de um algoritmo minimiza os efeitos de fatores que são dependentes da linguagem de programação e da máquina e concentra-se em determinar a ordem de magnitude da frequência de execução das operações. O comportamento assintótico de um algoritmo é o mais procurado, ou seja para um volume grande de dados é que a complexidade torna-se mais importante. O algoritmo assintoticamente mais eficiente é melhor para todas entradas exceto entradas relativamente pequenas (BARBOSA, 2008).

METODOLOGIA E DESENVOLVIMENTO

A pesquisa foi desenvolvida nas seguintes etapas: estudo teórico sobre os métodos de ordenação *radixsort* e *countingsort*, implementação dos métodos, análise dos

métodos e estudo de caso sobre ambos. A implementação foi desenvolvida na linguagem C. A Figura 1 mostra os métodos *Radixsort* e *Countingsort*.

Figura 1- *Radixsort* e *Countingsort.*



Fonte: Elaborado pelo Autor

RESULTADOS E DISCUSSÕES

Foram realizados testes comparativos com vetores de tamanho 100, 1.000, 10.000 e 100.000 inteiros totalmente desordenados, utilizando um computador *Dell* modelo *laptop Inspiron* 15 5548210-ADNT, processador *Intel Core i7-5500U, 2.4GHz* , memória *RAM* 16 *GB DDR3*, sistema *operacional Linux Fedora Realese 26, GCC*, realizando para cada vetor e método 5 execuções, levando em consideração o pior caso dentre os testes sendo o tempo medido em milissegundos, onde o método *Countingsort* foi superior em todos os testes comparado com o método *Radixsort* , podendo ser observado na Tabela 1.

Tabela 1 – Comparativo entre os métodos de ordenação

Totalmente Desordenados	***S.O. Fedora realese 26, Melhor e Pior Casos***			
Tamanho	*Radix Pior Caso*	*Radix Melhor Caso*	*Counting Pior Caso*	*Counting Melhor Caso*
100	0,006	0,005	0,002	0,001
1000	0,067	0,049	0,008	0,005
10000	0,738	0,512	0,141	0,062
100000	9,48	6,365	0,954	0,625
500000	39,026	36,767	3,991	2,754

Fonte: Elaborado pelo Autor

CONSIDERAÇÕES FINAIS

Os dois métodos são extremante eficientes na ordenação de dados, tendo os tempos de resposta bem próximos, com exceção do vetor de tamanho 500.000 onde o método *Countingsort* foi 9,77 vezes mais rápido que o método *Radixsort.* Com os resultados gerados,

pode-se concluir que o método Countingsort é eficaz para ordenação de dados em vetor com maior capacidade de armazenamento.

REFERÊNCIAS

BARBOSA, Marco Antônio. **Estrutura de Dados II**. Curso de Ciência da Computação. Cruz Alta, 2008.

BITENCOURT, David Couto. **Análise dos métodos de ordenação.** 2014. 13 f. TCC (Graduação) - Curso de Sistemas de Informação, Departamento de Química e Exatas, Universidade Estadual do Sudoeste da Bahia, Jequie -BA, 2014.

CORMEN, T.H.; Leiserson, C. E.; Rivest, R. L.; Sten, C.; Algoritmos **- tradução da 2ª edição americana** – Teoria e Prática, Campus, 2002.

Grama, A.; Karypis,G.; Kumary,V., **Introduction to Parallel Compunting**, Pearson Addison Wesley,1994.

LIMA JUNIOR, Vieira; ARAUJO, Everson Santos. **ALGORITMOS DE ORDENAÇÃO:** estudo comparativo de diversos algoritmos de ordenação. 2003. 15 f. TCC (Graduação) - Curso de Sistemas de Informação, Faculdade de Imperatriz, Imperatriz - MA, 2003.

MIYAZAWA, F. K. **Notas de complexidade de algoritmo**s. Universidade Estadual de Campinas, 1999.

Oliveira, André Luis Faria de; Souza, Uéverton dos Santos. **Algoritmos Paralelos De Ordenação.** 2008. 70 f. TCC (Graduação) - Curso de Tecnologia em Sistemas de Computação, Universidade Federal Fluminense, Niterói - Rj, 2008.