UNIDADE ITUIUTABA | UEMG
Intercursos Revista Científica
Ciências Exatas

ANÁLISE SOBRE ALGUNS MÉTODOS DE ORDENAÇÃO DE LISTAS: SELEÇÃO, INSERÇÃO E SHELLSORT

Analysis on some Sort of Methods of Lists: Select, Insert and Shellsort

Andrew Carlos de Sene Dias, Nayara Almeida Vilela, Walteno Martins Parreira Júnior

RESUMO

Este artigo apresenta análises de complexidade e de tempo de execução de três dos principais algoritmos de ordenação existentes, o Seleção (do inglês, SelectionSort), o Inserção (do inglês, InsertionSort) e o ShellSort. A escolha de um método de ordenação adequado deve ser a primeira ação a se fazer antes de qualquer implementação, pois para cada problema há um método (ou métodos) de ordenação mais indicado, e uma escolha equivocada pode afetar consideravelmente odesempenho do software. As análises permitiram que fosse determinado qual seria o método (ou métodos) mais indicado para vários tipos de aplicações, visando sempre uma melhoria de desempenho.

Palavras-chave: Ordenação. Tempo de execução. Algoritmos de Ordenação.

ABSTRACT

This paper presents analyzes of complexity and execution time of three of the main algorithms of sorting, the Selection (SelectionSort), the Insertion (InsertionSort) and the ShellSort. The choice of an appropriate method of a sorting method should be the first action to be done before any implementation, because for every problem there is a method (or methods) of sorting more indicated, and a wrong choice can significantly affect recommendations to the software. The analysis allowed it to be determined what would be the method (or methods) more suitable for various types of applications, always aiming at an improvement in performance.

Keywords: Sorting. execution Time. Algorithms for Sorting.

INTRODUÇÃO

Na atualidade que a todo instante, milhares de informações são criadas, modificadas ou apagadas, se não houvesse como ordenar estas informações, nem faria sentido armazená-las. Para essa tarefa existem vários métodos de ordenação, com o desenvolvimento de algoritmos baseados nestes métodos, pode-se aplicar a

um computador, que se torna a principal ferramenta quando o assunto é ordenação. Mas entre os vários algoritmos existentes, como determinar qual o melhor?

> Na área de análise de algoritmos, existem dois tipos de problemas bem distintos, conforme apontou Knuth (1971): (i) Análise de um algoritmo particular: Qual é o custo de usar um dado algoritmo para resolver um problema específico? Neste caso, características do algoritmo em questão devem ser investigadas; (ii) Análise de uma classe de algoritmos: Qual é o algoritmo de menor custo possível para resolver um problema particular? Neste caso, toda uma família de algoritmos para resolver um problema específico é investigada (KNUTH apud ZIVIANI, 1999, p.3).

Este trabalho tem como objetivo comparar, através de dados relativos ao Tempo de Execução e Número de Operações coletados previamente, a eficiência de três dos principais métodos de ordenação existentes, o método de Inserção Direta, o de Seleção Direta e o Shellsort. E após as comparações, determinar qual deles é superior, para quais situações cada um deles é indicado, e para quais não são indicados.

Os Métodos de Ordenação

Existe atualmente, com o crescimento cada vez maior da quantidade de dados sendo criada a cada instante, uma grande necessidade de armazenar informações, e tão importante quanto armazenar essas informações é possuí-la da forma mais ordenada possível. E como para tantas outras aplicações, o computador é o recurso mais eficiente tanto para a manipulação, quanto para a ordenação de dados, independente do tipo. Para que um computador ordene uma determinada quantidade de dados, é necessária a aplicação de um algoritmo de ordenação, que ao ser implementado garante uma maior eficiência e confiabilidade a uma ordenação, mas antes de falar sobre algoritmos, deve-se entender o que é, e quais as vantagens da ordenação.

“Ordenar consiste no processo de rearranjar um conjunto de objetos em ordem ascendente ou descendente. O objetivo principal da ordenação é facilitar a recuperação posterior de itens do conjunto ordenado” (PARREIRA JÚNIOR, 2012, p. 29). Ordenações são feitas de forma alfabética ou numérica, podendo variar de ordem crescente à decrescente. Um exemplo prático e de fácil entendimento seria a Lista Telefônica, seria inviável buscar um contato se a lista não seguisse a

ordenação alfabética. Para o problema da Lista Telefônica, e muitos outros, é que existem os algoritmos de ordenação.

Um algoritmo de ordenação, ou método de ordenação, é um algoritmo que coloca os elementos de uma lista em uma determinada ordem escolhida pelo usuário. Os dados de entrada podem estar em forma de matriz, o que permite um acesso aleatório, ou lista, que permite apenas acesso sequencial, e embora existam algoritmos para cada caso específico, existem também aqueles que podem ser aplicados á ambos os casos, com pequenas ou nenhuma modificações.

Quando se fala em algoritmos para implementação para computadores, deve-se sempre pensar em economia de memória e de tempo de execução, é importante escolher um método de ordenação que tenha estas características. Para escolher o algoritmo que utiliza menos memória, deve-se observar se as permutações realizadas por ele é feita no próprio vetor desordenado, os métodos que transportam os elementos de um vetor A para um vetor B são menos indicados.

Um dos conceitos importantes que se pode observar sobre algoritmos de ordenação é a sua estabilidade. Um algoritmo é considerado estável se ele mantém a ordem de registros de chaves iguais, ou seja, se estes registros, depois de ordenados, aparecem na mesma posição que estava antes da ordenação. “Alguns dos métodos de ordenação mais eficientes não são estáveis” (PARREIRA JÚNIOR, 2012, p. 29). É importante notar que essa propriedade só é relevante se as entradas possuírem chaves associadas a elas, e se o arquivo já tiver passado por uma ordenação.

Outro conceito importante é quanto ao arquivo a ser ordenado, neste caso, os métodos de ordenação são classificados em Ordenação Interna e Ordenação Externa.

> Se o arquivo a ser ordenado cabe todo na memória principal, então o método de ordenação é chamado de ordenação interna. Neste caso, o número de registros a ser ordenado é pequeno o bastante para caber em um array do pascal, por exemplo. Se o arquivo a ser ordenado não cabe na memória principal e, por isso, tem que ser armazenado em fita ou disco, então o método de ordenação é chamado de ordenação externa. A principal diferença entre os dois métodos é que, em um método de ordenação interna, qualquer registro pode ser imediatamente acessado, enquanto em um método de ordenação externa, os registros são acessados seqüencialmente ou em grandes blocos (PARREIRA JÚNIOR, 2012, p. 29).

É interessante que se conheça os métodos menos eficientes, antes de trabalhar com métodos de maior eficiência. Segundo Ziviani (1999) estes métodos são chamados de métodos diretos, e existem algumas razões para se iniciar com eles:

- Métodos diretos são mais adequados para entender as principais técnicas de ordenação;
- Os algoritmos são curtos e mais fáceis de serem compreendidos;
- Métodos diretos são mais rápidos quando a lista é pequena.

Neste trabalho, foram analisadas situações em que listas de tamanhos variados são ordenadas utilizando três dos principais métodos de ordenação, utilizando apenas vetores e levando em consideração as configurações do processador utilizado, o nível de complexidade de cada método e o tempo de execução para a ordenação completa de cada lista proposta.

Análise e Comparações dos Algoritmos

Ao analisar a eficiência de um algoritmo, deve-se ficar atento a sua complexidade, não se pode comparar dois ou mais algoritmos sem conhecer suas respectivas complexidades. A complexidade de um algoritmo é representada por uma fórmula, que depende do Número de Operações executadas pelo algoritmo até que o resultado esperado seja obtido. Para entender as fórmulas de complexidade apresentadas neste trabalho, é importante observar que “nos algoritmos que executam operações sobre listas lineares, a complexidade é expressa em função do tamanho da lista. Se n indica o número de registros, temos que a complexidade será uma função de n” (PARREIRA JÚNIOR, 2012, p. 29). A entrada aplicada ao algoritmo também interfere na medida de complexidade, levando isso em consideração, devem ser considerados três casos quanto a entrada:

- Melhor Caso: quando a entrada resulta em um menor crescimento no número de operações;
- Pior Caso: quando a entrada resulta em um maior crescimento no número de operações;

- Caso Médio: quando são consideradas todas as entradas possíveis e suas probabilidades de ocorrência.

Para comparar a complexidade de dois algoritmos, deve-se verificar a notação assintótica de suas respectivas fórmulas de complexidade, a notação assintótica é conhecida também como Grande O, ou Notação O. A Notação O é uma notação matemática, expressa pela letra O, utilizada para analisar o comportamento assintótico de funções, duas funções *f* e *g* possuem comportamento assintótico, se ambas tendem ao infinito com a mesma velocidade, isso significa que em um dado intervalo, se *f* dobrar *g* também dobrará, e vice-versa, essa é a propriedade que interessa à análise algoritmos, visto que "é mais importante saber que o número de operações executadas num algoritmo dobra se dobrarmos o valor de n, do que saber que para n igual a 100 são executadas 300 operações" (PARREIRA JÚNIOR, 2012, p. 29). Para exemplificar, se há dois algoritmos de função de complexidade *f*(n) = O(n²) (seguindo a Notação O, *f*(n) e n² tendem ao infinito com a mesma velocidade, ou seja, têm comportamento assintótico) e *f1*(n) = O(n), então *f1*(n) e *f*(n) podem ser comparadas, e como consequência, comparar a eficiência dos softwares que implementam estes algoritmos, diz-se que, *f*(n) possui uma complexidade quadrática, e *f*1(n) uma complexidade linear.

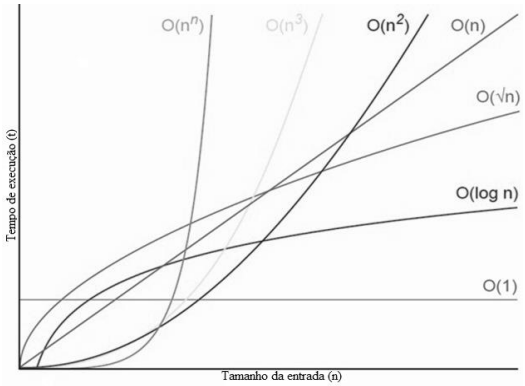

Figura 1: Gráfico do tamanho da entrada em função do tempo para as principais Notações O
Fonte: The Recycle Bin (2008)

Algumas convenções devem ser adotadas quanto à Notação O:

- É comum escrever a Notação O sem os elementos de menor grau, ou seja, a Notação O é representada apenas pelo elemento de maior grau;
- É comum desconsiderar os elementos constantes, no caso de funções constantes, escreve-se *O(1)*.

Para aplicar os conceitos teóricos apresentados até agora, iremos analisar e posteriormente comparar alguns dos principais métodos de ordenação que são os algoritmos: Seleção, Inserção e Shellsort. Os algoritmos Seleção e Inserção são métodos diretos e mais simples, já o Shellsort é um algoritmo mais sofisticado, pois é um refinamento do algoritmo de Inserção.

Método de Inserção. Este é um método muito utilizado pelos jogadores de cartas. Os elementos são conceitualmente divididos em uma sequencia destino $a_1...a_{j-1}$ e uma sequencia fonte $a_i...a_n$. Em cada passo, iniciando-se com *i=2*, e incrementando *i* de uma em uma unidade, o i-ésimo elemento da sequencia vai sendo retirado e transferido para a sequencia destino, e inserido na posição apropriada (WIRTH, 1986). Simplificando, o vetor é percorrido da esquerda para a direita e à medida que avança, vai deixando os elementos mais à esquerda ordenados. Este é um método estável, e se aplicado à uma lista pequena se mostra bem eficiente.

	1	2	3	4	5	6
Chaves iniciais:	**O**	R	D	E	N	A
i = 2	**O**	**R**	D	E	N	A
i = 3	**D**	**O**	**R**	E	N	A
i = 4	**D**	**E**	**O**	**R**	N	A
i = 5	**D**	**E**	**N**	**O**	**R**	A
i = 6	**A**	**D**	**E**	**N**	**O**	**R**

Figura 2: Representação da ordenação do Método de Inserção
Fonte: Ziviani (1999)

A Figura 2 ilustra como é feita a ordenação pelo Método de Inserção, os elementos em negrito representam a sequencia destino.

Quando é considerado o Melhor Caso (ou seja, quando os elementos do vetor já estão ordenados), a complexidade do algoritmo é *O(n)*. Já no Pior Caso (quando os elementos do vetor estão na ordem reversa), a complexidade é $O(n^2)$. Por fim, quando o caso considerado é o Caso Médio, a complexidade é $O(n^2)$.

Para coleta de dados, foram utilizadas listas lineares de 100, 200 e 400 elementos, e foram considerados os três casos, conforme o quadro a seguir.

Lista	Qtidade de Operações	Tempo de execução (segundos)
100 elementos aleatórios	2461	2.486
100 elementos em ordem crescente	99	0.099
100 elementos em ordem decrescente	5048	5.048
200 elementos aleatórios	9974	10.256
200 elementos em ordem crescente	199	0.199
200 elementos em ordem decrescente	20098	20.099
400 elementos aleatórios	39930	40.098
400 elementos em ordem crescente	399	0.399
400 elementos em ordem decrescente	80198	80.204

Quadro1: Dados coletados utilizando o algoritmo de Inserção
Fonte: Próprio Autor (2013)

Foi utilizado um processador Pentium Dual-Core CPU T4400 2.2GHz, e o tempo foi convertido posteriormente para segundo por padrão. Observa-se que o tempo cresce fielmente ao crescimento da lista, com uma proporção de uma operação a cada milissegundo, com pequenas variações.

Método de Seleção. É um dos métodos de ordenação mais simples, ele funciona da seguinte maneira: o menor elemento do vetor é selecionado e trocado com o elemento da primeira posição, estes passos são repetidos para os *n-1* elementos restantes, depois com os *n-2* restantes, até que todos os elementos estejam nas posições desejadas (NAZARE JUNIOR; GOMES, 2008, p.14). Este método não é estável, e não apresenta vantagem quando a lista já está ordenada, embora também não apresente desvantagem quando a lista está desordenada.

	1	2	3	4	5	6
Chaves iniciais:	O	R	D	E	N	A
i=2	**A**	R	D	E	N	**O**
i=3	A	**D**	**R**	E	N	O
i=4	A	D	**E**	**R**	N	O
i=5	A	D	E	**N**	**R**	O
i=6	A	D	E	N	**O**	**R**

Figura 3: Representação da ordenação do Método de Seleção
Fonte: Ziviani (1999)

A Figura 3 ilustra como é feita a ordenação pelo Método de Seleção, os elementos em negrito sofreram uma troca entre si.

Quando é considerado o Melhor Caso (ou seja, quando os elementos do vetor já estão ordenados), a complexidade do algoritmo é $O(n^2)$.Já no Pior Caso (quando os elementos do vetor estão na ordem reversa), a complexidade é $O(n^2)$. Por fim, quando o caso considerado é o Caso Médio, a complexidade é $O(n^2)$.

Para coleta de dados, foram utilizadas listas lineares de 100, 200 e 400 elementos, e foram considerados os três casos, conforme o quadro 2.

Lista	Qtidade de Operações	Tempo de execução (segundos)
100 elementos aleatórios	5049	5.049
100 elementos em ordem crescente	5049	5.049
100 elementos em ordem decrescente	5049	5.050
200 elementos aleatórios	20099	20.119
200 elementos em ordem crescente	20099	20.103
200 elementos em ordem decrescente	20099	20.099
400 elementos aleatórios	80199	81.035
400 elementos em ordem crescente	80199	80.216
400 elementos em ordem decrescente	80199	80.204

Quadro 2: Dados coletados utilizando o algoritmo de Seleção
Fonte: Próprio Autor (2013)

Foi utilizado um processador Pentium Dual-Core CPU T4400 2.2GHz, e o tempo foi convertido posteriormente para segundo por padrão. Observa-se que o tempo cresce fielmente ao crescimento da lista, com uma proporção de uma operação a cada milissegundo, com pequenas variações.

Método Shellsort. Em 1959 D. L. Shell propôs um refinamento do método de ordenação por inserção. Enquanto o método de Inserção troca apenas elementos adjacentes, o Shellsort pode trocar elementos distantes um do outro. Os itens separados por *h* posições são agrupados e ordenados, o valor de *h* é decrementado seguindo uma sequencia qualquer (não há uma sequencia definitiva) até que atinja o valor unitário, onde a ordenação corresponde ao algoritmo de inserção, mas nesse ponto nenhum elemento precisa se mover para uma posição muito distante. (ZIVIANI, 1999). O Shellsort é uma ótima opção para listas de tamanho médio, é um método simples e eficiente.

	1	2	3	4	5	6
Chaves iniciais:	O	R	D	E	N	A
h=4	N	A	D	E	O	R
h=2	D	A	N	E	O	R
h=1	A	D	E	N	O	R

Figura 4: Representação da ordenação do Método Shellsort
Fonte: Ziviani (1999)

Na Figura 4, para *h=4*, O é comparado com N são e trocados, a seguir R é comparado com A e são trocados. Posteriormente, para *h=2*, N, D e O são rearranjados, e a seguir A, E e R também são rearranjados. Por último, para *h=1*, corresponde ao algoritmo de inserção.

Quando é considerado o Melhor Caso (ou seja, quando os elementos do vetor já estão ordenados), a complexidade do algoritmo é $O(n)$. Já no Pior Caso (quando os elementos do vetor estão na ordem reversa), a melhor complexidade conhecida é: $O(n \log^2 n)$. Por fim, quando o caso considerado é o Caso Médio, a complexidade depende de *h*, portanto não pode ser definida.

As fórmulas de complexidade deste algoritmo são bem vagas, pois até hoje ninguém foi capaz de analisar este algoritmo, isso se devea problemas matemáticos muito difíceis que sua análise traz (FARIA; ROCHA; REIS, p.4). O que se sabe é que nenhum *h* pode ser múltiplo do anterior.

Para coleta de dados, foram utilizadas listas lineares de 100, 200 e 400 elementos, e foram considerados os três casos, conforme o quadro 3.

Lista	Operações	Tempo de execução (segundos)
100 elementos aleatórios	1303	1.303
100 elementos em ordem crescente	451	0.451
100 elementos em ordem decrescente	911	0.911
200 elementos aleatórios	3174	3.180
200 elementos em ordem crescente	1032	1.048
200 elementos em ordem decrescente	2496	2.496
400 elementos aleatórios	7710	7.711
400 elementos em ordem crescente	2270	2.277
400 elementos em ordem decrescente	4978	4.979

Quadro 3: Dados coletados utilizando o algoritmo Shellsort
Fonte: Próprio Autor (2013)

Foi utilizado um processador Pentium Dual-Core CPU T4400 2.2GHz, e o tempo foi convertido posteriormente para segundo por padrão. Observa-se que o tempo cresce fielmente ao crescimento da lista, com uma proporção de uma operação a cada milissegundo, com pequenas variações.

COMPARAÇÃO DOS RESULTADOS

A seguir foram comparados os resultados obtidos em cada caso, considerando os algoritmos indicados anteriormente e os vetores propostos e finalmente são realizadas as conclusões.

Para o vetor inicialmente ordenado de forma crescente, que é o Melhor Caso, o algoritmo Inserção apresentou os menores tempos de execução, se mostrando o mais rápido enquanto o Shellsort foi o segundo melhor e por último, nesta comparação, o algoritmo de Seleção teve o pior desempenho, em que apresentou tempos de execução bem maiores que seus concorrentes. Como pode ser observado no quadro 4 com o tempo em segundos.

Melhor Caso (Vetor crescente)						
	Inserção		Seleção		Shellsort	
Vetor	Tempo	Operações	Tempo	Operações	Tempo	Operações
100	0.099	99	5.049	5049	0.451	451
200	0.199	199	20.103	20099	1.048	1032
400	0.399	399	80.216	80199	2.277	2270

Quadro 4: Comparação de resultados no Melhor Caso
Fonte: Próprio Autor (2013)

Para o vetor decrescente (Pior Caso), o Shellsort apresentou os menores tempos de execução, se mostrando o mais rápido desta vez, ambos, Seleção e Inserção empataram, se mostrando bem ineficientes em relação ao Shellsort. Como pode ser observado o quadro 5.

Pior Caso (Vetor decrescente)						
	Inserção		Seleção		Shellsort	
Vetor	Tempo	Operações	Tempo	Operações	Tempo	Operações
100	5.048	5048	5.050	5049	0.911	911
200	20.099	20098	20.099	20099	2.496	2496
400	80.204	80198	80.204	80199	4.979	4978

Quadro 5: Comparação de resultados no Pior Caso
Fonte: Próprio Autor (2013)

Para o vetor aleatório (Caso Médio), o Shellsort novamente apresentou os menores tempos de execução, deixando o segundo lugar para o Inserção, e novamente, o Seleção apresentou o pior desempenho. Como pode ser observado no quadro abaixo.

Caso Médio (Vetor aleatório)						
	Inserção		Seleção		Shellsort	
Vetor	Tempo	Operações	Tempo	Operações	Tempo	Operações
100	2.486	2661	5.049	5049	1.303	1303
200	10.256	9974	20.119	20099	3.180	3174
400	40.098	39930	81.035	80199	7.711	7710

Quadro 6: Comparação de resultados no Caso Médio
Fonte: Próprio Autor (2013)

CONSIDERAÇÕES FINAIS

Levando em conta o que foi observado nos testes realizados, podem-se apresentar algumas comparações em relação à teoria apresentada.

Os testes mostraram que a escolha de um método de ordenação apropriado é de extrema importância, visto que há uma grande diferença de desempenho de cada um dos métodos em cada situação, essa não é uma escolha que possa ser feita de forma arbitrária.

> Depois que um problema é analisado e decisões de projeto são finalizadas, o algoritmo tem que ser implementado em um computador. Neste momento, o projetista tem que estudar as várias opções de algoritmos a serem utilizados, onde os aspectos de tempo de execução e espaço ocupado são considerações importantes (ZIVIANI, 1999, p.3).

Os métodos de ordenação direta, Inserção e Seleção, que são métodos mais simples, mas que na maioria das vezes são menos eficientes do que o Shellsort, e que por sua vez é um método que trabalha de uma forma mais sofisticada, sendo consequentemente mais complicado de se implementar.

Cada algoritmo apresentou comportamento conforme sua complexidade, mostrando que a comparação entre dois ou mais algoritmos pode ser feita a partir dela, não havendo a necessidade de testes muito profundos. Apesar do algoritmo Shellsort não possuir uma definição de valor de complexidade muito precisa, e em alguns casos até mesmo inexistentes, ele se mostrou dentro das expectativas teóricas.

Concluindo, em um aspecto geral, o melhor dos três métodos analisados foi o Shellsort, apesar de ser o mais complexo, é o mais indicado para listas muito grandes, podendo também ser aplicado em listas pequenas com uma performance razoável, mas como não é um método estável, não deve ser aplicado à uma lista que já tenha sido ordenada anteriormente. O Inserção se mostrou a melhor opção para listas praticamente ordenadas, neste caso sua performance superou até mesmo o Shellsort, e o fato de ele ser um método estável apenas reforça esta observação. Já a Seleção teve o pior desempenho dos três, apesar de mostrar estabilidade quanto ao número de operações em todos os casos, seu desempenho foi bem aquém dos demais na maioria dos casos, tendo empatado com o algoritmo Inserção apenas quando considerado o vetor decrescente, este método é indicado apenas quando o objetivo da atividade for implementar um algoritmo simples ou constante.

REFERÊNCIAS

CABRAL, Jacqueline D.; SILVA, Ilma M.; LOPES, Elizabeth A. ***Manual para Elaboração de Artigos Científicos*.** Disponível em: http:// www. unilestemg. Br / portal / biblioteca / downloads / manual – para – elaboracao – de – artigos - cientificos.pdf. Acesso em: 21 ago. 2013.

FARIA, João P.; ROCHA, Ana P.; REIS, Luis P. ***Vectores: Algoritmos de Ordenação*.** Disponível em http:// web. fe. up. Pt / ~ lpreis / prog2_06_07 / AulasTeoricas / 4 . vector Ord . pdf. Acesso em: 20 ago. 2013.

NAZARE JUNIOR, Antonio C.; GOMES, David M. ***Algoritmos e Estrutura de Dados – Métodos de Ordenação Interna***. Disponível em: http://www.decom.ufop.br/menotti/aedI082/tps/tp3-sol1.pdf. Acesso em: 23 ago. 2013.

PARREIRA JÚNIOR, Walteno M. ***Apostila de Análise de Algoritmos***. Ituiutaba: FEIT-UEMG. 2012.

THE RECYCLE BIN. **Big-Oh**. Disponível em:http:// therecyclebin . wordpress . com / 2008 / 03 / 11 / big – oh /. Acesso em: 20 ago. 2013.

WRITH, Niklaus. ***Algoritmos e Estrutura de Dados***. Rio de Janeiro - RJ: LTC Editora. 1986.

ZIVIANI, Nivio. ***Projeto de Algoritmos com Implementações em Pascal e C***. São Paulo - SP: Editora Pioneira, 5. ed. 1999.

AUTORES

Andrew Carlos de Sene Dias, discente do curso de Engenharia de Computação da Universidade do Estado de Minas Gerais (UEMG) – Unidade Ituiutaba-MG.
andrew.carlos@outlook.com

Nayara Almeida Vilela, discente do curso de Engenharia de Computação da Universidade do Estado de Minas Gerais (UEMG) – Unidade Ituiutaba-MG.
nayaraalmeida.nav@gmail.com

Walteno Martins Parreira Júnior, mestre em Educação, especialista em Design Instrucional para EaD e Informática Aplicada à Educação. É professor dos cursos de Engenharia da Computação, Engenharia Elétrica e Sistemas de Informação da Universidade do Estado de Minas Gerais (UEMG) – Unidade Ituiutaba-MG.
waltenomartins@yahoo.com

INTERCURSOS - REVISTA CIENTÍFICA

***Intercursos*, v. 13, n.1, Jan-Jun. 2014 – ISSN 2179-9059**

Universidade do Estado de Minas Gerais (UEMG) - Unidade Ituiutaba.

Periodicidade Semestral.

ISSN Nº 2179-9059
CDD: 011.34