UNIDADE ITUIUTABA | UEMG
Intercursos Revista Científica
Ciências Humanas

METODOS DE ORDENAÇÃO: A IMPORTÂNCIA DA ESCOLHA DO MÉTODO CORRETO

Sorting Methods: the Importance of Choosing the Correct Method

Cristino Divino de Freitas Júnior, Felipe Alencar, Walteno Martins Parreira Júnior

RESUMO

Este trabalho apresenta os resultados de uma pesquisa sobre métodos de ordenação, os quais são algoritmos que possuem técnicas de ordenação diversas para resolver uma mesma tarefa. Dependendo da aplicação, cada algoritmo considerado possui alguma vantagem sobre os outros algoritmos. O objetivo é compreender que existem várias formas de resolver uma tarefa com uma das técnicas de ordenação, sendo que algumas são mais eficazes na aplicação dos métodos, dependendo do tipo de tarefa a ser resolvida.

Palavras-chave: Métodos de ordenação. Técnicas de ordenação. Algoritmos.

ABSTRACT

This paper presents the results of a survey on sorting methods, which are algorithms that have techniques for sorting different to solve the same task. Depending on the application, each algorithm considered has some advantage over the other algorithms. The objective is to understand that there are several ways to solve a task with one of the techniques of ordination, and some are more effective in the application of methods depending on the type of task to be solved.

Keywords: Sorting Methods. Techniques of ordination. Algorithms.

INTRODUÇÃO

Os métodos de ordenação constituem em artifícios para a resolução de tarefas utilizando computadores. As técnicas de ordenação permitem verificar como cada algoritmo se comporta quando executado. Dependendo da aplicação, cada algoritmo considerado possui uma vantagem particular sobre os outros algoritmos.

Ordenar consiste em um método de reorganizar um conjunto de objetos em uma ordem ascendente ou descendente. O objetivo principal da ordenação é simplificar a recuperação posterior de itens do conjunto ordenado.

A finalidade deste trabalho é apresentar qual a importância da escolha de um método de ordenação quando se tem um conjunto de dados a serem ordenados, os resultados da pesquisa desenvolvida dará uma ideia sobre métodos de ordenação e suas técnicas para resolução de uma tarefa.

MATERIAIS E MÉTODOS

Para se testar os métodos de ordenação se faz necessário a utilização do computador que execute os algoritmos para verificar seu desempenho na resolução da tarefa que lhe foi concedida. Segundo Ziviani (2007), os métodos de ordenação são classificados de duas formas: Ordenação interna e Ordenação externa. A Ordenação interna pode ser definida como os métodos que não necessitam de uma memória secundaria para o processo, logo a ordenação é feita na memória principal do computador. A Ordenação externa ocorre quando o arquivo a ser ordenado não cabe na memória principal e, por isso, tem de ser armazenado em fita ou disco para se efetuar o processo de ordenamento.

Segundo Parreira Júnior (2012, p.29), a principal diferença entre os dois grupos ocorre porque no método de ordenação interna qualquer registro pode ser acessado diretamente, enquanto que no método externo é necessário fazer o acesso em blocos de registros por causa da forma de armazenamento dos dados.

Neste trabalho, serão utilizados somente métodos de ordenação interna, para permitir a comparação dos resultados obtidos. Durante a escolha de um algoritmo de ordenação, deve-se observar um aspecto importante, o tempo gasto durante a sua execução. Para algoritmos de ordenação interna, as medidas de complexidade relevantes contam o numero de comparações entre chaves e o número de movimentações de itens do arquivo.

> É possível identificar que o comportamento dos métodos de ordenação varia conforme o tamanho de entrada, onde a eficiência dos métodos esta relacionada ao tempo de execução, número de comparações e trocas a serem efetuadas (SCHWADE et al., 2011, p.2).

Método de Ordenação por Seleção.

É um método simples que consiste em localizar elementos que estão fora de sua posição e aloca-lo em uma posição apropriada.

> A ordenação por seleção consiste, em cada etapa, em selecionar o maior (ou o menor) elemento e alocá-lo em sua posição correta dentro da futura lista ordenada. Durante a execução, a lista com n registros é decomposta em duas sublistas, uma contendo os itens já ordenados e a outra com os elementos ainda não ordenados. No início, a sublista ordenada está vazia e a desordenada contém todos os elementos, no final do processo a sublista ordenada apresentará (n-1) elementos e a desordenada terá um elemento (PARREIRA JÚNIOR, 2012, p.30).

Um dos algoritmos mais simples de ordenação é o de Seleção, de forma resumida pode-se definir o processo conforme apresentado no Quadro 1.

Selecione o menor item do vetor. Troque-o com o item da primeira posição do vetor. Repita essas duas operações com os n-1 itens restantes, depois com os n 2 itens, até que reste apenas um elemento.

Quadro 1 – Algoritmo de Ordenação de Seleção.
Fonte: adaptado de Parreira Júnior (2012, p. 30).

O método é ilustrado no Quadro 2, que apresenta a sequência de ações para ordenar os caracteres da palavra “ordena”.

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

Quadro 2 – Sequência de interações para ordenar uma palavra.
Fonte: adaptado de Ziviani (2007)

As vantagens do método são: a) Custo linear no tamanho da entrada para o número de movimentos de registros; b) É o algoritmo a ser utilizado para arquivos com registros muito grandes; c) É muito interessante para arquivos pequenos.

As desvantagens do método podem ser descrita como:

a) O fato de o arquivo já estar ordenado não ajuda em nada, pois o custo continua quadrático;
b) O algoritmo não é estável.

Método de Ordenação por Inserção

É um método simples que consiste em partir de uma parcela dos elementos a serem ordenados e que a cada interação vai caminhando para organizar uma parcela maior da lista e que somente na última interação é que analisará a lista como um todo.

> O método consiste em cada passo, a partir de i=2, o i-ésimo item da sequência fonte é apanhado e transferido para a sequência destino, sendo colocado na posição correta. A inserção do elemento no lugar de destino é efetuada movendo-se os itens com chave maiores para a direita e então inserindo-o na posição que ficou vazia. Neste processo de alternar comparações e movimentação de registros existem duas situações que podem causar o término do processo, a primeira é quando um item com chave menor que o item em consideração é encontrado e a segunda situação é quando o final da sequência destino é atingido (à esquerda). A melhor solução para a situação de um anel com duas condições de terminação é a utilização de uma sentinela, para isto, na posição zero do vetor colocamos o próprio registro analisado (PARREIRA JÚNIOR, 2012, p.31).

Sobre o método, pode-se escrever que:

a) O número mínimo de comparações e movimentos ocorre quando os itens estão originalmente em ordem;
b) O número máximo ocorre quando os itens estão originalmente na ordem reversa;
c) É o método a ser utilizado quando o arquivo está “quase” ordenado;
d) É um bom método quando se deseja adicionar uns poucos itens a um arquivo ordenado, pois o custo é linear; e) é um método estável.

Método de Ordenação Shellsort

O Método de Ordenação ShellSort pode ser considerado um refinamento do método de inserção, diferindo pelo fato de no lugar de considerar o vetor a ser ordenado como um único segmento, ele considera vários segmentos sendo aplicado o método de inserção direta em cada um deles. Considerando que o algoritmo repassa a lista várias vezes, sempre dividindo o grupo maior em porções menores (OLIVEIRA, 2002).

O Shellsort faz a trocas de registros que estão distantes uns dos outros, de tal forma que os itens que estão separados n posições são rearranjados de forma que todo n-ésimo item leva a uma sequência ordenada (OLIVEIRA, 2002).

> O método shellsort contorna este problema permitindo trocas de registros que estão distantes um do outro. Os itens que estão separados h posições são re-arranjados de tal forma que todo h-ésimo item leva a uma sequência ordenada. Tal sequência é dita estar h-ordenada (PARREIRA JÚNIOR, 2012, p.40).

Método de Ordenação QuickSort

Segundo Ziviani (2007), o método de ordenação QuickSort é considerado como um dos métodos mais rápidos. Seu método consiste em selecionar primeiramente um dos elementos do conjunto a ordenar para ser o elemento pivô, e que é um elemento já ordenado, em seguida faz uma subdivisão do conjunto inicial em dois subconjuntos, onde o primeiro subconjunto conterá todos os elementos menores que o elemento escolhido, o pivô, e o segundo subconjunto com todos os elementos maiores que o pivô.

> A matriz original é dividida em duas submatrizes; a primeira contém os elementos menores ou iguais à chave escolhida chamada de limite ou pivô. A segunda matriz inclui elementos iguais ou maiores [...] o processo de partição é repetido para ambas as submatrizes. Como resultado, dois novos limites são escolhidos, um para cada submatriz. [...] este processo de particionar é realizado até que haja somente matrizes de uma célula que precisem ser ordenadas. [...] [que] resulta em dados já ordenados, no processo de se preparar para ordenar (DROZDEK, 2002, p.421).

Após a definição dos métodos que serão analisados, é necessário identificar a Complexidade destes algoritmos.

> Para o cálculo de complexidade, pode-se medir o número de passos de execução em um modelo matemático denominado maquina de Turing, ou medir o número de segundos gastos em um computador específico. Ao invés de calcular os tempos de execução em máquinas específicas, a maioria das análises conta apenas o número de operações "elementares" executadas. A medida de complexidade é o crescimento assintótico dessa contagem de operações (PARREIRA JÚNIOR, 2012, p.2).

Complexidade	
Inserção	O (n^2)
Seleção	O (n^2)
Shellsort	O (n log n)
Quicksort	O (n log n)

Quadro 3 – Complexidade dos métodos de ordenação.
Fonte: adaptado de Parreira Júnior (2012, p. 46).

RESULTADOS E DISCUSSÃO

Os algoritmos foram implementados e foram executados para uma sequencia de listas de tamanho variado, iniciando com 500 elementos e chegando a 30000 elementos na lista maior. A relação entre os tempos de execução estão apresentados nos quadros a seguir.

Para efeito de facilitar a comparação, o método que levou menos tempo para executar recebeu o valor 1 e os outros receberam valores proporcionais a ele.

Primeiramente, as listas apresentavam registros ordenados aleatoriamente e os resultados estão apresentados no Quadro 4. Pode-se observar que alguns métodos não conseguiram ordenar algumas listas.

Método \ Quantidade	500	5.000	10.000	30.000
Inserção	11,3	87	161	–
Seleção	16,2	124	228	–
Shellsort	1,2	1,6	1,7	2
Quicksort	1	1	1	1

Quadro 4 – Proporção de Tempo para Ordenar uma lista inicialmente aleatória.
Fonte: dos autores.

Neste caso as listas apresentavam registros ordenados inicialmente de forma ascendente e os resultados estão apresentados no Quadro 5. Pode-se observar que o método de Seleção não conseguiu ordenar a maior lista.

Método \ Quantidade	500	5.000	10.000	30.000
Inserção	1	1	1	1
Seleção	128	1.524	3.066	–
Shellsort	3,9	6,8	7,3	8,1
Quicksort	4,1	6,3	6,8	7,1

Quadro 5 – Proporção de Tempo para Ordenar uma lista inicialmente ascendente.
Fonte: dos autores.

Neste caso as listas apresentavam registros ordenados de forma descendente e os resultados estão apresentados no Quadro 6. Pode-se observar que alguns métodos não conseguiram ordenar algumas listas.

Método \ Quantidade	500	5.000	10.000	30.000
Inserção	40,3	305	575	–
Seleção	29,3	221	417	–
Shellsort	1,5	1,5	1,6	1,6
Quicksort	1	1	1	1

Quadro 6 – Proporção de Tempo para Ordenar uma lista inicialmente descendente.
Fonte: dos autores.

Segundo Parreira Júnior (2012, p. 47) pode-se fazer um conjunto de observações sobre os métodos estudados: a) O Shellsort e Quicksort têm a mesma ordem de grandeza; b) O Quicksort é o mais rápido para todos os tamanhos aleatórios experimentados; c) A relação Shellsort/Quicksort aumenta à medida que o número de elementos aumenta; d) Para arquivos pequenos (500 elementos), o Shellsort é mais rápido que o Quicksort; e) Quando o tamanho da entrada cresce, o Quicksort é mais rápido que o Shellsort; f) O Inserção é o mais rápido para qualquer tamanho se os elementos estão ordenados; f) O Inserção é o mais lento para qualquer tamanho se os elementos estão em ordem descendente; g) Entre os algoritmos de custo O (n2), o Inserção é melhor para todos os tamanhos aleatórios experimentados.

Considerando os dados adquiridos ao longo da experiência, pode-se organizar um Quadro com a Influência da ordem inicial dos registros quando se

compara os métodos Shellsort e Quicksort, que está apresentado no Quadro 7, considerando o melhor tempo com o valor unitário e fazendo a proporção para os demais e que é condizente com o que apresenta Parreira Júnior (2012).

Método		Shellsort			Quicksort	
Quantidade	5.000	10.000	30.000	5.000	10.000	30.000
Ascendente	1	1	1	1	1	1
Descendente	1,5	1,6	1,5	1,1	1,1	1,1
Aleatória	2,9	3,1	3,7	1,9	2,0	2,0

Quadro 7 – Comparação entre os métodos Shellsort e Quicksort.
Fonte: dos autores.

Considerando o que apresenta Parreira Júnior (2012, p.47) e também as observações adquiridas na experiência, pode-se tecer algumas considerações entre os dois métodos:

a) O Shellsort é bastante sensível à ordenação ascendente ou descendente da entrada;
b) Em arquivos do mesmo tamanho, o Shellsort executa mais rápido para arquivos ordenados;
c) O Quicksort é sensível à ordenação ascendente ou descendente da entrada;
d) Em arquivos do mesmo tamanho, o Quicksort executa mais rápido para arquivos ordenados;
e) O Quicksort é o mais rápido para qualquer tamanho para arquivos na ordem ascendente.

> Algoritmos mais otimizados como Shellsort, QuickSort e HeadSort têm desempenhos muito superior aos métodos simples tanto no pior caso, como também de forma desordenada, reduzindo consideravelmente seu tempo de execução. Isto se deve ao fato destes algoritmos utilizarem técnicas que reduzem o vetor em análise a cada iteração, com menor número de trocas e algoritmos bem mais otimizados, alguns até utilizando recursividade como o quicksort (SCHWADE et al., 2011, p.3).

CONSIDERAÇÕES FINAIS

É possível identificar que o comportamento dos métodos de ordenação varia conforme o tamanho de entrada, onde a eficiência dos métodos está relacionada ao

tempo de execução, ao número de comparações e também a quantidade de trocas a serem efetuadas. A fim de verificar e analisar o comportamento aplicou-se os métodos em uma escala crescente de valores de entrada, em vetores de 500 a 30000 posições em várias situações iniciais.

Os resultados mostram que na maioria dos casos os métodos da Inserção e da Seleção apresentam desempenho pior que os demais métodos analisados. Os métodos de inserção e seleção também possuem deficiências em seus algoritmos de ordenação, a utilização destes métodos torna-se viável apenas para vetores muito pequenos, mas precisamente na escala de até 500 elementos.

Por fim, algoritmos melhor otimizado como Shellsort e QuickSort têm desempenho muito superior aos métodos simples tanto na situação de pior caso, como também na situação inicial desordenada, reduzindo consideravelmente seu tempo de execução. Isto se deve ao fato destes algoritmos utilizarem técnicas que reduzem o vetor em análise a cada iteração, com menor número de trocas e algoritmos mais otimizado, alguns até utilizando recursividade como é o caso do Quicksort.

A análise sobre os resultados processados permitem avaliar que para vetores de tamanho entre 5000 e 10000 os melhores métodos de ordenação foram Shellsort e Quicksort, enquanto que o pior método foi o de Seleção. Para vetores de tamanho entre 10000 e 30000, o melhor método de ordenação foi o Quicksort enquanto que o pior método foi o de Seleção.

E conclui-se também que o pior método de ordenação aplicado na análise desenvolvida foi, em todos os casos, o método da Seleção. Para a situação de pior caso, o método que apresentou melhor desempenho foi o método Quicksort.

REFERÊNCIAS

DROZDEK, Adam. Estrutura de dados e algoritmos em C++. São Paulo - SP: Pioneira Thomson Learning, 2002.

OLIVEIRA, Álvaro B. Métodos de Ordenação Interna. São Paulo - SP: Visual Book, 1. ed. 2002.

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

SCHWADE, Darlan Eduardo et al. Métodos de ordenação: uma análise quantitativa para o pior caso. **Seminário Interinstitucional de Ensino, Pesquisa e Extensão**, 16. Cruz Alta: Unicruz, 2011.

ZIVIANI, Nivio. Projeto de Algoritmos com implementação em C++ e Java. Thomson Learning, São Paulo - SP, 1. ed., 2007.

AUTORES

Cristino Divino de Freitas Júnior, discente do curso de Engenharia de Computação da Universidade do Estado de Minas Gerais (UEMG) – Unidade Ituiutaba-MG.
cristinocdfj@gmail.com

Felipe Alencar, discente do curso de Engenharia de Computação da Universidade do Estado de Minas Gerais (UEMG) – Unidade Ituiutaba-MG.
felipeleopoldo@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