**CES – Centro de Ensino Superior de Juiz de Fora**

Bacharelado em Sistemas de Informação

4º Período – Projeto e Análise de Algoritmos

Grupo: Bruno Rodrigo, Ernani Guilhon, Isaias Horta Brazil e Rafael Bispo

Professor: Allan Fonseca da Silva

***Métodos de***

***Ordenação Interna***

**INTRODUÇÃO:**

Conforme pesquisado e estudado no livro: - Métodos de Ordenação Interna – Implementação, Análise e Desempenho em Delphi - dos autores Álvaro Borges de Oliveira, Adriana Prada e Reginaldo Rubens da Silva, vimos que:

“... antes de se iniciar a descrição da eficiência dos algoritmos aqui analisados, deixa-se claro que os resultados em tempo de uma implementação específica não são a última palavra, mas sempre fornecem algumas indicações.

Destarte, arrola-se algumas ponderações a respeito da simulação dos resultados:

* Que os algoritmos não são dependentes da máquina, no entanto, estas influenciam na execução do mesmo em função do tempo que levam para processar determinado algoritmo.
* Que a execução de determinado algoritmo em uma máquina pode levar um tempo *m* e em outra *n*, mas sempre proporcional.
* Que a execução de um algoritmo *a*, na máquina *m*, com uma massa de dados *d* resulta um tempo *t*, se rodarmos novamente o algoritmo *a*, na máquina *m*, com uma massa de dados *d* resulta um tempo *t’*, onde *t’* pode ser maior, menor ou igual a *t*, e isto por que:
  + Uma máquina equipada com um único processador não é capaz de executar processamento paralelo propriamente dito;
  + O sistema operacional "Todo o tempo" executa diversas funções as quais algumas têm prioridade maior que o programa aqui analisado;...”.

Usamos para obter o resultado, a seguir demonstrado através de tabelas comparativas, analisando o comportamento do tempo avaliado à medida que as instancias crescem em cada um, bem como comparar o desempenho entre cada um deles em um mesmo computador, com a seguinte configuração: Acer Aspire 5612NWLMI Intel Core Duo processador 1.66 GHZ, memória RAM de 4GB e Hard-Disk de 120GB.

As amostras foram feitas com instâncias de 50.000, 100.000, 250.000, 500.000 e 1.000.000 (um milhão) de elementos numéricos de variáveis do tipo longint, com três diferentes padronizações:

* Aleatórios: dados gerados através da função randon () da unit System; (Caso Médio)
* Decrescente: dados gerados em ordem decrescente; (Pessimista)
* Crescente: dados inteiramente classificados em ordem crescente; (Otimista).

**ESTUDOS DE CASOS:**

**1:** **Caso Otimista:** Conforme podemos observar na tabela 1 (Caso Otimista) verificamos que o BubbleSort é o método de ordenação menos eficaz de todos aqui analisados. Junto a ele podemos enquadrar o SelectSort, que apesar de ser um pouco mais rápido que o anterior, tem um tempo melhor que o mesmo.

Já os métodos InsertSort e ShellSort se revelaram os mais eficazes e mais rápidos na ordenação dos algoritmos. Enquanto os métodos HeapSort e MergeSort se revelaram bastante eficientes porém com o tempo um pouco maior.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO OTIMISTA** | **50000** | **100000** | **250000** | **500000** | **1000000** |
| **BubbleSort** | 00:00:02:281 | 00:00:09:203 | 00:00:57:562 | 00:03:51:375 | 00:16:46:594 |
| **SelectSort** | 00:00:02:062 | 00:00:08:265 | 00:00:51:782 | 00:03:28:313 | 00:15:01:672 |
| **InsertSort** | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 |
| **QuickSort** | Overflow | Overflow | Overflow | Overflow | Overflow |
| **HeapSort** | 00:00:00:000 | 00:00:00:032 | 00:00:00:047 | 00:00:00:110 | 00:00:00:234 |
| **ShellSort** | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 |
| **MergeSort** | 00:00:00:016 | 00:00:00:016 | 00:00:00:047 | 00:00:00:109 | 00:00:00:219 |

*Tabela 1 – Caso Otimista*

**2:** **Caso Médio:** Conforme podemos observar na tabela 2 (Caso Médio) em comparação com a tabela 1 (Caso Otimista) comprovamos que o BubbleSort continua sendo o método de ordenação menos eficaz de todos aqui analisados, porém na instância de entrada de 50.000 ele foi mais lento e nas instâncias maiores que esta foi mais rápido. Junto a ele também comprovamos que o método SelectSort, torna-se mais rápido a medida que cresce o número de instâncias.

Já os métodos InsertSort e ShellSort continuam insensíveis ao aumento de instâncias na ordenação dos algoritmos e também na mudança do tipo de caso.

Enquanto o método HeapSort teve uma alteração em relação as instâncias de 50.000 e 100.000, sendo que na menor instância foi mais lento e na maior foi mais rápido, já nas outras instâncias continuaram insensíveis.

O método MergeSort apresentou mais eficiência na instancia de 50.000 em relação a tabela anterior, enquanto na instância de 250.000 ele se mostrou mais lento do que a tabela 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO MÉDIO** | **50000** | **100000** | **250000** | **500000** | **1000000** |
| **BubbleSort** | 00:00:08:172 | 00:00:09:172 | 00:00:57:562 | 00:03:51:219 | 00:16:46:406 |
| **SelectSort** | 00:00:02:078 | 00:00:08:266 | 00:00:51:765 | 00:03:28:172 | 00:14:52:172 |
| **InsertSort** | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 |
| **QuickSort** | Overflow | Overflow | Overflow | Overflow | Overflow |
| **HeapSort** | 00:00:00:015 | 00:00:00:015 | 00:00:00:047 | 00:00:00:110 | 00:00:00:235 |
| **ShellSort** | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 |
| **MergeSort** | 00:00:00:000 | 00:00:00:015 | 00:00:00:063 | 00:00:00:109 | 00:00:00:219 |

*Tabela 2 – Caso Médio*

**3:** **Caso Pessimista:** Conforme podemos observar na tabela 3 (Caso Pessimista) em comparação com a tabela 1 (Caso Otimista) e tabela 2 (Caso Médio), o método BubbleSort continua sendo o mais lento das três tabelas, e das instâncias destaca-se na de 50.000 com a mesma característica.

No método SelectSort, nos três casos todas as instâncias permaneceram praticamente inalteradas.

Nos métodos InsertSort e ShellSort continuam com um mesmo comportamento mencionado nos casos anteriores.

No método HeapSort só no caso otimista e na instância de 50.000 este caso foi o mais rápido. Nas outras instâncias e nos três tipos de casos, o tempo de ordenação permaneceu relativamente igual.

No método MergeSort foi o mais curioso, pois na instância de 1.000000 o tempo permaneceu o mesmo e na instância de 50.000 só no caso otimista foi o mais lento, nas outras instâncias foram relativamente insensíveis.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CASO PESSIMISTA** | **50000** | **100000** | **250000** | **500000** | **1000000** |
| **BubbleSort** | 00:00:08:437 | 00:00:09:187 | 00:00:57:563 | 00:03:51:078 | 00:16:45:985 |
| **SelectSort** | 00:00:02:063 | 00:00:08:266 | 00:00:51:750 | 00:03:27:938 | 00:14:51:750 |
| **InsertSort** | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 |
| **QuickSort** | Overflow | Overflow | Overflow | Overflow | Overflow |
| **HeapSort** | 00:00:00:016 | 00:00:00:016 | 00:00:00:047 | 00:00:00:109 | 00:00:00:234 |
| **ShellSort** | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 | 00:00:00:000 |
| **MergeSort** | 00:00:00:000 | 00:00:00:016 | 00:00:00:047 | 00:00:00:110 | 00:00:00:219 |

*Tabela 3 – Caso Pessimista*

**Obs.:** O método de ordenação QuickSort só foi possível verificar que o mesmo funciona até a instância de 26.000 elementos, a partir desta instância ocorrem erros de estouro de memória (OVERFLOW), por este motivo não foram realizados as análises em comparação com os demais métodos, pois as instâncias exigidas são superiores ao número máximo suportado pelo método mencionado.

**Conclusão:**

O método mais eficiente: Não foi possível destacar um método, mas sim dois, que são InsertSort e ShellSort. Estes apresentam tempos menores de ordenação do que os demais métodos.

O método menos eficiente: Fazendo uma análise das três tabelas, tanto quanto ao tipo de caso quanto ao tamanho da instância, o método BubbleSort apresenta tempos de ordenação maiores que os demais.

**PORQUE ESCOLHEMOS DELPHI?**

Adotamos a Linguagem Pascal e o Ambiente de Programação Delphi, pois os dois são juntos os que mais a equipe domina. Além disso, o livro adotado como principal fonte de pesquisa: Métodos de Ordenação Interna - Implementação, Análise e Desempenho em Delphi, dos autores Álvaro Borges de oliveira, Adriana Prada e Reginaldo Rubens da Silva dizem que:

"É mister abordar aqui a Linguagem Pascal (IDE - Integrated Development Environment - Delphi),..., pois o sistema programado baseia-se nesta linguagem.

Segundo Corrêa (1995), o Delphi une os recursos visuais à programação orientada a objetos ou a programação estruturada, permitindo a programação em ambiente amigável, oferecendo uma arquitetura reciclável baseado em componentes. Permitindo uma nova abertura na programação, possibilitando o desenvolvimento rápido de aplicativos e a criação de programas de alta velocidade para o ambiente Windows na Linguagem Pascal.

Segundo CANTÙ (1996), há muitos ambientes de programação com os quais se pode trabalhar, mais do Delphi é surpreendente por uma série de motivos. Para CANTÙ (1996), as dez razões principais para usar o Delphi são:

* A linguagem Object Pascal;
* Tecnologia de componentes;
* A estreita integração com a programação do Windows;
* O suporte a banco de dados;
* Um compilador rápido;
* Abordagem baseada em formulários e orientada a objeto;
* A disponibilidade de códigos fonte das bibliotecas;
* O editor, o depurador, o browser e outras ferramentas;
* Os componentes e ferramentas de terceiros;
* Características do Delphi.

Conforme CHIARA (1998), o Delphi é um ambiente de desenvolvimento de aplicações, que permite o desenvolvimento de aplicações baseadas no Windows.

Já para MATCHO (1999), é uma ferramenta de fácil utilização, e assim mesmo, muito poderosa para desenvolvimento de aplicativos Windows. O Delphi usa recursos de compilação modernos para gerar códigos executáveis rápidos.

O Delphi é constituído de vários elementos, ferramentas de designer e de banco de dados para o auxílio ao desenvolvimento e teste de aplicações, de forma rápida e intuitiva. Há várias características no Delphi, como: IDE, arquitetura baseada em componentes, compilador de código nativo de alta performance, capacidade de alterar entre um form e seu código, biblioteca de componentes visuais, arquitetura aberta, suporte à tecnologia Windows, ambiente personalizável, gerenciador de projetos, gerenciador de relatórios, servidor local baseado em SQL, implementação de threads que é um recurso avançado do Delphi, o qual suporta que dois ou mais processos sejam executados ao mesmo tempo de forma que se possa controlar.

**Bibliografia:**

o Métodos de Ordenação Interna - Implementação, Análise e Desempenho em Delphi

o Autores:

 Álvaro Borges de oliveira,

 Adriana Prada e

 Reginaldo Rubens da Silva

o Visual Books Editora

o Estruturas de Dados e seus Algoritmos

o Autores:

 Jaime Luiz Szwarrcfiter e

 Lilian Markenzon

o LTC editora

o Estruturas de Dados com Pascal

o Instituto Brasileiro de Pesquisa em Informática

o IBPI Press

 3ª tiragem

o Pascal Estruturado

o Autores:

 HARRY FARRER,

 CHRISTIANO BECKER,

 EDUARDO FARIA,

 FREDERICO FERREIRA CAMPOS FILHO,

 HELTON FÁBIO DE MATOS,

 MARCOS AUGUSTO DOS SANTOS e

 MIRIAM LOURENÇO MAIA

o 2ª edição

o Editora Guanabara Koogan