COMPARAÇÃO DO TEMPO DE EXECUÇÃO DE ALGORITMOS QUE RETORNAM O MAIOR E MENOR ELEMENTO DE UM VETOR

PRADELA, Izaura Pereira[1], OLIVEIRA, Lucineida Nara de Andrade[1], QUEIROZ, Marcela Dantas[1], PARREIRA JÚNIOR, Walteno Martins[2]

[1]Graduandas do Curso de Engenharia de Computação da UEMG - campus de Ituiutaba (UEMG-FEIT-ISEPI)
email: izaurapradela@bol.com.br, lucineida_nara@hotmail.com ,cellysindy@hotmail.com

[2]Professor dos cursos de Engenharia da Computação, Engenharia Elétrica e Sistema de Informação da UEMG - campus de Ituiutaba (UEMG-FEIT-ISEPI) – email: walteno@ituiutaba.uemg.br.

Área FAPEMIG: Ciência da Computação – 1.03.03.00-6

Resumo: *No desenvolvimento de algoritmos e softwares, é importante avaliar sua correção temporal. Este artigo apresenta as análises de três algoritmos que utilizam a memória principal para a sua execução e que retornam o maior e o menor elemento de uma lista, compara o tempo de execução de cada um e faz uma análise da diferença de tempo entre eles e com os valores esperados.*

Palavras-chave: algoritmos maxmin, maxmin2, maxmin3 e tempo de execução.

1 . Introdução

Um algoritmo pode ser visto como uma seqüência de ações executáveis para a obtenção de uma solução para um determinado tipo de problema. Segundo Ziviani (2002, p.1), Os algoritmos podem ser avaliados por uma variedade de critérios. Na maioria das vezes o interesse é na taxa de crescimento do tempo ou de espaço necessário para a solução de grandes problemas.

Foram feitas comparações de tempos em três algoritmos que retornam o maior e menor elemento de uma lista ,cuja implementação foi realizada em Linguagem C, usando delay's, pois os computadores atuais possuem processamento rápido, dificultando assim estimar o tempo em segundos. Segundo Ziviani (2002, p.3), "*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.*".

Normalmente, o tamanho da entrada influencia o tempo de execução. Nestes algoritmos o tamanho da entrada é um fator direto para o tempo consumido na execução. Como escreve Ziviani (2002, p.6), "*A medida do custo de execução de um algoritmo depende principalmente do tamanho da entrada dos dados.*".

Os algoritmos foram implementados na Linguagem C, compilados no TurboC e executados várias vezes no mesmo computador, permitindo que seus tempos de execução fossem comparados.

2 . Desenvolvimento

Os algoritmos implementados na linguagem C para comparação foram: MaxMin, MaxMin2 e MaxMin3. Estes algoritmos são considerados clássicos e estão disponíveis na literatura. Neste caso, foram utilizados os que estão em Ziviani (2002, p.7-9). Para efeito de comparação, foram trabalhados os dados para resultar no pior caso.

Segundo Parreira Júnior (2006, p.16), o número de comparações depende da posição do maior e do menor elemento para os algoritmos MaxMin e Max Min2, com isso tem-se os casos:

- Melhor caso: ocorre quando os elementos da lista estão em ordem crescente, portanto N-1 comparações são necessárias.
- Pior caso: ocorre quando os elementos da lista estão em ordem decrescente, portanto 2(N-1) comparações são necessárias.
- Caso Médio: elemento[i] (em uma posição qualquer) é maior do que max a metade das vezes, portanto (3N/2 – 3/2) comparações são necessárias.

Número de comparações dos algoritmos pode ser observada na tabela 1.

Algoritmo	Melhor caso	Pior caso	Caso médio
maxmim	2 (n – 1)	2 (n – 1)	2 (n – 1)
maxmin2	n – 1	2 (n – 1)	3n/2 – 3/2
maxmin3	3n/2 – 2	3n/2 – 2	3n/2 – 2

Tabela 1 - Número de comparações

2.1- MaxMin

Algoritmo trivial para calcular o máximo e o mínimo de L seria: considerar M1 como sendo o máximo e o mínimo temporário; se o máximo temporário é menor que do que M2, considerar então M2 como o novo máximo temporário; se o mínimo temporário é maior do que M2, considerar então M2 como sendo o mínimo temporário; repetir o processo para M3, ..., Mn. Após a comparação com Mn, temos que o máximo e o mínimo temporários são os valores desejados. Foram realizadas 2(n-1) comparações do máximo e mínimo temporários com os elementos da lista.

2.2- MaxMin2

É a otimização do algoritmo MaxMin, observando que a comparação A[i] < min só é necessária quando o resultado da comparação A[i] > max é falsa. Nesta caso, diminuindo o número de comparações na execução do programa, segundo Parreira Júnior (2006, p.15).

2.3- MaxMin3

Algoritmo mais eficiente, pois os elementos da lista são comparados de dois em dois e os elementos maiores são comparados com **max** e os menores com **min**. Quando **N** é impar, o elemento que está na posição do vetor elemento[N] é duplicado na posição elemento[N+1] para evitar um tratamento de exceção.

Este algoritmo faz (3N/2 – 2) comparações, independentemente da ordenação inicial da lista.

3. Resultados obtidos

A análise de tempo de execução dos algoritmos é observado em relação com o número de comparações executadas por cada algoritmo. Na tabela 2, tempo está em segundo, para permitir uma maior percepção durante os testes. Na execução dos programas (ver tabela 2) é possível ver a evolução do tempo em função do tamanho da entrada, pois quanto maior a entrada, maior o tempo gasto.

Quadro de Nro de Comparações x tempo						
	MáxMin		**MaxMin2**		**MaxMin3**	
Tamanho do vetor	2 (n – 1)	(Tempo em segundo)	2 (n – 1)	(Tempo em segundo)	3n/2 – 2	(Tempo em segundo)
10	18	12s	18	11s	13	6s
20	38	22s	38	13s	28	11s
30	58	31s	58	14s	43	15s
40	78	40s	78	15s	58	16s

Tabela 2 – Tempo de execução

Como no Maxmim3 não faz diferença a escolha de casos, usamos o pior caso para o MaxMim e o MaxMim2, ou seja, o vetor de entrada estava em ordem decrescente, fazendo assim mais comparações.

4. Conclusão

Concluímos que o tempo de execução do MaxMim2 é menor que o tempo do MaxMim, pois o algoritmo MaxMim2 é um algoritmo melhorado, mesmo tendo o mesmo número de comparações. A complexidade é O(n) em todos os casos, o que altera são os números de comparações para cada algoritmo.

Como o MaxMin é o algoritmo não otimizado, os tempos tendem a ser maiores, e foi comprovado na pesquisa, conforme pode ser observado na tabela 2.

No algoritmo MaxMim3, o tempo de execução é menor que no algoritmo maxMim2 para os vetores de tamanho 10 e 20, pois o número de comparações é menor. Para os vetores de tamanho 30 e 40, o tempo é 1 segundo maior para o MaxMim3, pois à medida que o tamanho do vetor aumenta, o tempo de execução tende a se aproximar, e também deve-se levar em conta que são valores pequenos e podem ocorrer erros de aproximação. Para trabalhos futuros, uma das propostas é alterar a medida do tempo e/ou aumentar o tempo de dalay (espera) do processador para melhorar os valores encontrados.

5. Bibliografia

PARREIRA JÚNIOR, Walteno Martins. **Análise de algoritmos (Apostila).** Ituiutaba: FEIT-UEMG, 2006.

ZIVIANI, Nivio. **Projeto de algoritmos: com implementação em Pascal e C**. São Paulo: Pioneira – Thonson Learning, 2002.

Para Referencia do Artigo:

PRADELA, Izaura Pereira, OLIVEIRA, Lucineida Nara de Andrade, QUEIROZ, Marcela Dantas & PARREIRA JÚNIOR, Walteno M. Comparação do tempo de execução de algoritmos que retornam o maior e menor elemento de um vetor. IN: Seminário de Iniciação Científica e extensão da UEMG, 10, 2008, Divinópolis (MG). **Anais do 10 Seminário da UEMG**. Divinópolis: UEMG e FUNED. 2008. CD-ROM. ISSN: 1983-9693. Disponível em <www.waltenomartins.com.br/artigos>