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

Izaura Pereira Pradela[1], Lucineida Nara de Andrade Oliveira[1], Marcela Dantas Queiroz[1], Walteno Martins Parreira Júnior[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.

10º SEMINÁRIO DE INICIAÇÃO CIENTÍFICA E EXTENSÃO DA UEMG

Palavras-chave: algoritmo, memória cachê, tempo de execução.

1. Introdução

Analisar três algoritmos que utilizam a memória principal para a sua execução e que retornam o maior e o menor elemento de uma lista, comparando o tempo de execução de cada um e fazer uma análise da diferença de tempo entre eles e com os valores esperados.

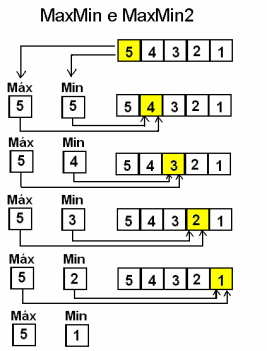
2. Fundamentações Teóricas

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.

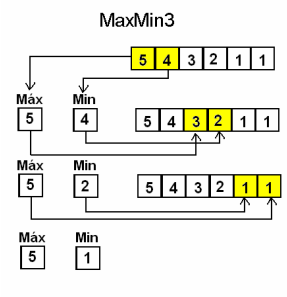
2.2- MaxMin2

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.



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 observado 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

3. desenvolvimento

Os algoritmos foram implementados na Linguagem C, compilados no TurboC e executados várias vezes, usando-se o algoritmo de pior caso.

4. Resultados Obtidos

A análise dos tempos de execução dos algoritmos foi 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

5. 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 maior, 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.

6. 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.