Intercursos Revista Científica
Ciências Exatas

COMPARAÇÃO DO TEMPO DE EXECUÇÃO DE ALGORITMOS MAXMIN EM DIFERENTES PROCESSSADORES

Comparison of the Runtime of Algorithms Maxmin in Different Processsadores

Walteno Martins Parreira Júnior, Marcio Oliveira Costa, Roberto Guimarães Dutra de Oliveira, Renner Augusto A. Lima

RESUMO

Ao desenvolver softwares é importante saber a eficiência do software e como ele executará em computadores com especificações diferentes. Não só o processador, mas alguns periféricos, tais como memória RAM, memória cachê, também influenciam na velocidade de execução de um programa. Este artigo apresenta a execução de três algoritmos executados em três computadores distintos.

Palavras-chave: Tempo de Execução. Algoritmos de Busca. Processadores.

ABASTRACT

When developing software is important to know the efficiency of software and as it will execute in computers with different specifications. Not only the processor, but some peripherals, such as memory RAM, memory cache, also influence in the speed of execution of a program. This article presents the execution of three algorithms executed in three distinct computers.

Keywords: Time of Execution. Algorithms of Search. Processors

INTRODUÇÃO

Este trabalho relata uma das atividades desenvolvidas no projeto de pesquisa de algoritmos, onde os variados algoritmos apresentados na literatura são analisados e são observadas as suas reações aos diversos ambientes e situações que podem ser encontradas no cotidiano da área.

Segundo Cormen e outros (2002, p. 3), “um algoritmo é qualquer procedimento computacional bem definido que toma algum valor ou conjunto de valores como entrada e produz algum valor ou conjunto de valores de saída. Portanto, um algoritmo é uma sequência de passos computacionais que transformam a entrada em saída”. O conceito de algoritmo é frequentemente ilustrado pelo exemplo de uma receita de culinária, embora muitos algoritmos sejam mais complexos. Eles podem repetir passos ou necessitar de decisões, podendo ser comparações ou lógica, até que a tarefa seja completada. Um algoritmo não representa, necessariamente, um programa de computador, e sim os passos necessários para realizar uma tarefa.

Os algoritmos foram implementados na Linguagem C, compilados no Borland C++ versão 5.02 e executados em três computadores com especificações diferentes, permitindo que seus tempos de execução fossem comparados.

> A linguagem C++ é derivada da linguagem C. O conjunto de instruções que fazem parte da linguagem C também é parte de C++. Os elementos principais adicionados à linguagem C para dar origem a C++ representam a orientação a objetos. [...] A linguagem C é procedural; C++ é rica em recursos que atendem às limitações impostas pelas linguagens procedurais (MIZRAHI, 2006).

Segundo Capron e Johnson (2004, p.309), um processador é a unidade central de processamento (CPU) de um computador. Os processadores utilizados foram: AMD Turion 64X2 TL-50, AMD Athlon XP e Intel Core 2 duo T5550, onde cada um possuía, 1Gb, 1.5Gb e 4Gb de memória RAM, respectivamente.

O objetivo do presente trabalho é a análise e a implementação de algoritmos de busca de menor e maior elemento de um vetor, conhecidos como

MaxMin, testando sua eficiência em máquinas distintas. A ideia principal foi realizar a comparação entre os tempos encontrados na execução de algoritmos em processadores diferentes que estão a disposição nos laboratórios da Instituição.

MATERIAIS E MÉTODO

Os algoritmos implementados fazem parte do método divisão e conquista. Segundo Parreira Júnior (2006, p.16), tal método diz que dado um problema, de tamanho n, o método divide-o em duas ou mais partes, criando subproblemas menores. Cada subproblema é resolvido separadamente e então as soluções são combinadas para a obtenção da solução da instância original.

Foram implementados e testados três algoritmos denominados de: Maxmin1, Maxmin2 e Maxmin3.

> [O algoritmo trivial consiste em] considerar M_1 como sendo o máximo e o mínimo temporário; se o máximo temporário é menor que do que M_2, considerar então M_2 como o novo máximo temporário; se o mínimo temporário é maior do que M_2, considerar então M_2 como sendo o mínimo temporário; repetir o processo para M_3, ..., M_n. Após a comparação com M_n, temos que o máximo e o mínimo temporários são os valores desejados (PARREIRA JÚNIOR, 2006, p. 16).

Em cada algoritmo foi acrescentada uma instrução chamada “delay”, que faz com que o programa demore mais um pouco a executar, para que os valores dos tempos pudessem apresentar uma diferença significativa, facilitando as comparações. Cada algoritmo foi executado em três computadores distintos e para cada um deles foram encontrados tempos de execução diferentes. As especificações dos computadores utilizados na pesquisa estão descritas no quadro 1.

Processador	Velocidade (GHz)	Memória Cachê	Memória Principal	Disco rígido
AMD Turion 64X2 TL-50	1,6	2x256 kb	1 Gb DDR2 667 Mhz	80 Gb SATA 5400 RPM
AMD Athlon XP	1,49	256 kb	1.5 Gb DDR 200 Mhz	80 Gb SATA 5400 RPM
Intel Core 2 duo T5550	1,83	2 Mb	4 Gb DDR2 667 Mhz	160 Gb SATA 7200 RPM

Quadro 1 – Dados principais dos computadores utilizados.

O algoritmo Maxmin1 possui o código mais simples e menos eficiente, comparando-o com os outros dois algoritmos, o Maxmin2 e o Maxmin3. Em todos os casos, independente de onde esteja o valor desejado em um vetor, ele realizará 2(n-1) comparações (ver quadro 3). Isso o torna o algoritmo menos sofisticado e lento, fazendo com que seja algoritmo que gasta o maior tempo de execução. O quadro 2 apresenta o algoritmo do Maxmin1:

```
entradas: A (lista), N (inteiro)
saídas: max (inteiro), min (inteiro)
maxmim(A, max, min)
        max = A[1];
        min = A[1];
        Para I de 2 até N passo 1 repita
                Se A[I] > max então max = A[I];
                Se A[I] < min então min = A[I];
        Fim_para;
```

Quadro 2 – Algoritmo Maxmin1.
Fonte: PARREIRA JÚNIOR (2006, p.16).

O quadro 3 apresenta a quantidade de comparações que são desenvolvidas no algoritmo MaxMin1, dependente da posição em que se encontra o maior e também o menor valor. Pode-se observar que não mudança no número de comparações quando a uma variação da posição do elemento encontrado.

Algoritmo	Melhor Caso	Pior caso	Caso médio
Maxmin1	2(n-1)	2(n-1)	2(n-1)

Quadro 3 – Comparações previstas para o algoritmo Maxmin1.
Fonte: PARREIRA JÚNIOR (2006, p.18)

Observando o algoritmo Maxmim2. O algoritmo Maxmin1 pode ser facilmente melhorado, observando que a comparação A[i]< min só é necessária quando o resultado da comparação A[i]>Max é falsa. Analisando esse caso percebemos que o seu pior caso seria um vetor ordenado decrescente, onde ele faria as comparações apenas para encontrar o menor elemento que seria o

último do vetor. Seu pior caso se equivale ao melhor caso do Maxmin1 comparando a quantidade de operações realizadas. No caso médio são realizadas 3n/2-3/2 e no melhor caso, que ocorre quando a lista está em ordem crescente, são realizadas n-1 operações (ver quadro 4). Algoritmo do Maxmin2:

```
Entradas: A (lista), N (inteiro)
Saídas: max (inteiro), min (inteiro)
maxmim2(A, max, min)
        max = A[1];
        min = A[1];
        Para I de 2 até N passo 1 repita
                Se A[I] > max então max = A[I]
                senão Se A[I] < min então min = A[I];
```

Quadro 4 - Algoritmo Maxmin2.
Fonte: PARREIRA JÚNIOR (2006, p.16)

Algoritmo	Melhor Caso	Pior caso	Caso médio
Maxmin2	n-1	2(n-1)	3n/2-3/2

Quadro 5 – Comparações previstas para o algoritmo Maxmin2.
Fonte: PARREIRA JÚNIOR (2006, p.18)

O algoritmo Maxmin3 é o código mais refinado e eficiente comparando-o com os dois primeiros. Algoritmo do Maxmin3 está apresentado no Quadro 6:

```
Entradas: A (lista), N (inteiro)
Saídas: max (inteiro), min (inteiro)
Maxmim3 (A, max, min)
        Se (N mod 2) > 0 então
                A[N+1] = A[N]
                                Fim do Anel = N
            senão Fim do Anel = N-1;
Se A [1] > A [2] então
                max = A[1]
                min = A[2]
              senão
                max = A[2]
                min = A[1];
I = 3;
Enquanto I ≤ Fim do Anel repita
        Se A[I] > A[I+1] então
                Se A[I] > max então max = A[I];
                Se A[I+1] < min então min = A[I+1];
              senão
                Se A[I] < min então min = A[I];
                Se A[I+1] > max então max = A[I+1];
        I = I + 2;
Fim Enquanto
```

Quadro 6 - Algoritmo Maxmin3.
Fonte: PARREIRA JÚNIOR (2006, p.17)

A quantidade de processos e o tempo de execução são reduzidos consideravelmente melhorando sua eficiência e diminuindo a quantidade de operações realizadas. Independente da posição em que se encontrem o menor e o maior elemento, a quantidade de processos executados é a mesma, 3n/2-2 (ver quadro 7). Supera o Maxmin1 em todos os casos (melhor, pior e médio) e é superior ao Maxmin2 com relação ao pior caso e bastante próximo quanto ao caso médio.

Algoritmo	Melhor Caso	Pior caso	Caso médio
Maxmin3	3n/2-2	3n/2-2	3n/2-2

Quadro 7 – Comparações previstas para o algoritmo Maxmin3.
Fonte: PARREIRA JÚNIOR (2006, p.18)

RESULTADOS E DISCUSSÃO

Após a execução dos algoritmos em cada um dos computadores escolhidos e cuja configuração foi apresentada no Quando 1 pode-se observar a diferença nos tempos de execução entre os diversos processadores. Foram encontrados os seguintes tempos médios:

Computadores	Maxmin1	Maxmin2	Maxmin3
AMD Turion 64X2	5,40s	5,00s	2,37s
AMD Athlon XP	4,87s	4,83s	2,36s
Intel Core 2 duo T5550	4,43s	4,48s	2,14s

Quadro 8 – Resultados obtidos em segundos

Pode-se observar no Quadro 8, que comparando os algoritmos iguais, houve uma variação nos valores dos tempos. Isso acontece devido às diferenças nas especificações de cada computador. No computador denominado AMD Athlon, que é equipado com o processador mais lento dos três, foram encontrados tempos menores do que o AMD Turion que é um processador um pouco mais rápido. Isso ocorreu para os três algoritmos, Maxmin1, 2 e 3. Isso acontece por que o processador não é o único fator que interfere diretamente na velocidade de execução de um programa, a memória RAM também tem impacto direto sobre a velocidade de execução. Já que o processador AMD Athlon possui 1.5Gb de memória RAM, 500MB a mais que o AMD Turion, conseguiu executar os três algoritmos gastando uma quantidade de tempo menor.

Já o computador equipado com o processador Intel Core 2 duo, que possui um processador mais rápido e 4Gb de memória RAM, superou os outros dois processadores e obteve tempos de execução menores em todos os três algoritmos. Comparando a velocidade do Core 2 duo com os outros processadores é visto que não houve tanta variação nos tempos quanto à diferença da velocidade do processador e na quantidade de memória RAM

entre os computadores. Isso acontece porque os compiladores mais antigos não foram desenvolvidos para trabalharem com mais de um processador simultaneamente, fazendo com que apenas um dos processadores do Core 2 duo execute o programa.

CONSIDERAÇÕES FINAIS

Conclui-se que para desenvolver programas computacionais não é necessário saber somente desenvolver algoritmos e programá-los, é necessário entender a arquitetura dos computadores e compreender quais os fatores que influenciam na velocidade de execução de programas. Outro fator importante é saber como os programas são desenvolvidos e suas capacidades para trabalharem com computadores que possuem múltiplos processadores.

Por último, é possível perceber que a eficiência do algoritmo é outro fator que interfere na velocidade de execução de um programa. Comparando os resultados encontrados, o algoritmo Maxmin3 superou o Maxmin1 e Maxmin2 em todos os tempos de execução e em todos os processadores. O mesmo aconteceu para o Maxmin2 que superou a velocidade do Maxmin1 em todos os processadores testados.

REFERÊNCIAS

CAPRON, H. L.; JOHNSON, J. A. **Introdução à informática**. São Paulo - SP: Pearson Prentice Hall, 2004.

CORMEN, Thomas H et al. **Algoritmos:** teoria e prática. Rio de Janeiro - RJ: Elsevier, 2002.

MIZRAHI, Victorine Viviane. **Treinamento em linguagem C++**. 2. ed. São Paulo - SP: Pearson Prentice Hall, 2006. v. 1.

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

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

AUTORES

Walteno Martins Parreira Júnior é professor dos cursos de Engenharia da Computação, Engenharia Elétrica e Sistemas de Informação da Fundação Educacional de Ituiutaba, associada à Universidade do Estado de Minas Gerais, Campus de Ituiutaba-MG. Especialista de Design Instrucional para EaD e Informática Aplicada à Educação. Mestrando em Educação no PPGED-UFU.
waltenomartins@yahoo.com

Marcio Oliveira Costa é professor dos cursos de Engenharia da Computação e Sistemas de Informação da Fundação Educacional de Ituiutaba, associada à Universidade do Estado de Minas Gerais, Campus de Ituiutaba-MG. Especialista em História da Filosofia: Tópicos Especiais e Mestrando em Psicanalise, Educação e Sociedade.
marcioyz@yahoo.com.br

Roberto Guimarães Dutra de Oliveira é discente do curso de Engenharia da Computação da Fundação Educacional de Ituiutaba – FEIT, associada à Universidade do Estado de Minas Gerais – UEMG, Campus de Ituiutaba-MG.
robertoguimaraes8@hotmail.com

Renner Augusto A. Lima é discente do curso de Engenharia da Computação da Fundação Educacional de Ituiutaba – FEIT, associada à Universidade do Estado de Minas Gerais – UEMG, Campus de Ituiutaba-MG.
renneramil@gmail.com