Intercursos Revista Científica
Ciências Exatas

COMPARAÇÃO DE TEMPO DE EXECUÇÃO DE ALGORITMOS MAXMIN EM COMPILADORES DIFERENTES

Runtime Comparison of Algorithms Maxmin in Different Compilers

Walteno Martins Parreira Júnior, Marcio Oliveira Costa, Luan Roger S. Santana, Ricardo de Oliveira Muniz Junior

RESUMO

Existem vários meios de escrever o mesmo algoritmo, alguns deles são executados de forma mais eficiente que outros, mas não é só isso que difere a velocidade da execução dos algoritmos, foram testados três algoritmos MaxMin que encontram o maior e o menor elemento de uma lista em um vetor de quarenta posições em compiladores diferentes para observar se apenas o algoritmo seria fundamental para a execução ficar mais rápida, ou se os compiladores também interferem na velocidade da execução do algoritmo.

Palavras-Chave: Algoritmos MaxMin. Tempo de Execução. Compilador DEV C++. Compilador Borland C.

ABSTRACT

Some ways to write algorithm the same, some of them are executed of form more efficient exist than others, but this is not alone that differs the speed from the execution of the algorithms, had been tested three MaxMin algorithms that find the greater and the lesser element of a list in a vector of forty position in different compilers to observe if only the algorithm would be basic it execution to be faster, or if the compilers also intervene with the speed of the execution of the algorithm.

Keywords: MaxMin Algorithms. Time of Execution. Compiler DEV C++. Compiler Borland C.

INTRODUÇÃO

A pesquisa descreve sobre os três tipos de algoritmos MaxMin utilizados para encontrar o maior e o menor elemento de uma lista de um vetor com quarenta posições, segundo Parreira Júnior (2006, p.16), um algoritmo MaxMin1 trivial pode ser melhorado apenas mudando uma comparação, obtendo o MaxMin2.

Segundo Ziviani (2002, p. 1), um algoritmo pode ser considerado como uma sequência de ações executáveis para a obtenção de uma solução para um tipo determinado de problema.

Para a codificação destes algoritmos foram usados compiladores que implementam uma mesma linguagem de programação. E segundo Salvetti e Barbosa (1998, p.7), “o algoritmo pode ser implementado em qualquer linguagem de programação e essa implementação ser trivial [...] ou trabalhosa, dependendo principalmente das características da linguagem escolhida e dos tipos de dados nela definidos”.

Segundo Capron e Johnson (2004, p.293), um compilador é um “tradutor que converte as declarações simbólicas de uma linguagem de alto nível em linguagem de máquina executável por computador”.

Foram utilizados três compiladores diferentes para ver se encontram resultados significativos de velocidade na compilação, o DEV C++, para o Windows, obtido gratuitamente na internet, o Borland C, shareware encontrado no site do fabricante, e o Anjuta para Linux, obtido gratuitamente na internet também.

> 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. Isso consiste nas classes, objetos, herança, polimorfismo e sobre-carga (MIZRAHI, 2006, p. xx).

O problema consiste em que, os algoritmos sendo executados em compiladores diferentes para ver se encontram resultados significativos de diferença de velocidade na compilação em compiladores diferentes, ou se apenas o código é suficiente para deixar um algoritmo mais veloz.

MATERIAL E MÉTODOS

Os compiladores DEV C++ e o Borland C, foram instalado em um computador quad-core, emulando o Windows XP Professional no Virtual Box Machine, com uma emulação de memória de tamanho 512 Mbytes, e uma emulação de processador, e usando o Sistema Operacional Ubuntu para compilar com o Anjunta, para confirmar se os compiladores, mesmo em sistemas operacionais diferentes, dependem mais da memória ou do Sistema Operacional já instalado.

Segundo Parreira Júnior (2006, p.16) sobre o MaxMin1, um 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.

E acrescenta que sobre o MaxMin2, este algoritmo pode ser facilmente melhorado, observando que a comparação A[i] < min só é necessária quando o resultado da comparação A[i] > max é falsa. Ainda segundo Parreira Júnior (2006, p. 17) sobre o MaxMin3, Os elementos de A 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 A[N] é duplicado na posição A[N+1] para evitar um tratamento de exceção.

O Quadro 1 apresenta uma comparação entre os algoritmos considerando o número de comparações como medida de complexidade em função do tamanho da entrada, aqui representada pelo valor de “n”.

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

Quadro 1. Número de comparações em função do tamanho da entrada.
Fonte: Parreira Júnior (2006)

Segundo Parreira Júnior (2006, p.17) “os algoritmos maxmin2 e maxmin3 são superiores ao maxmin de forma geral. O algoritmo maxmin3 é superior ao maxmin2 com relação ao pior caso, e é bastante próximo quanto ao caso médio”.

Resultados Com O Compilador Dev C++

O Quadro 2 mostra que o compilador DEV C++ executa os algoritmos num tempo muito curto, considerando o compilador simples, gratuito oferece tudo que um bom compilador disponibiliza para os usuários, se mostrando eficiente nas execução dos algoritmos MaxMin1, 2 e 3.

	Melhor (Crescente)	Pior (Decrescente)	Médio (Aleatório)	Quantidade de Operações
MaxMin1	0,390 seg	0,390 seg	0,390 seg	78/78/78
MaxMin2	0,230 seg	0,400 seg	0,390 seg	43/79/79
MaxMin3	0,220 seg	0,220 seg	0,220 seg	42/42/42

Quadro 2. Resultados obtidos com o compilador DEV C++

Na quinta coluna do Quadro 2, são apresentados o número de comparações para cada caso respectivamente, como por exemplo, na segunda linha, os dados referentes ao MaxMin1 são 78 comparações para o melhor caso, 78 para o pior caso e 78 também para o caso médio.

Resultados Com O Compilador Borland

O Quadro 3 mostra que o compilador BORLAND executa os algoritmos num tempo consideravelmente grande, mas, o programa se torna intuitivo à medida que se usa, pois cada dia que passa mais simples ele se torna para o uso diário, já que ele é intuitivo, acaba se tornando uma ótima opção para a primeira utilização.

	Melhor (Crescente)	Pior (Decrescente)	Médio (Aleatório)	Quantidade de Operações
MaxMin1	2,14 seg	2,14 seg	2,14 seg	78/78/78
MaxMin2	1,25 seg	2,15 seg	2,14 seg	43/79/79
MaxMin3	1,21 seg	1,21 seg	1,21 seg	42/42/42

Quadro 3. Resultados obtidos com o compilador BORLAND.

Na quinta coluna do Quadro 3, está apresentado o número de comparações para cada caso respectivamente, como por exemplo, na segunda linha, os dados referentes ao MaxMin1 são 78 comparações para o melhor, pior e para o caso médio; já na terceira linha, o MaxMin2 são 79 comparações para o caso médio e para o pior caso; e apenas 43 para o melhor caso, e na quarta linha, o MaxMin3 são 42 comparações para o pior, melhor e para o caso médio.

Resultados com o compilador KDevelop 4.0 Anjuta 2.26.1.0

O Quadro 4 mostra que o compilador Anjuta utilizado no Ubuntu, apresenta-se tão eficiente em velocidade quanto ao DEV C++, apresentando os mesmo resultados que o Anjuta, mas a interface é bem complicada, com tantos recursos que se torna poluído, mas mexendo nele descobre que ele é como o DEV C++, só que para Linux.

	Melhor (Crescente)	Pior (Decrescente)	Médio (Aleatório)	Quantidade de Operações
MaxMin1	0,390 seg	0,390 seg	0,390 seg	78/78/78
MaxMin2	0,220 seg	0,400 seg	0,390 seg	43/79/79
MaxMin3	0,220 seg	0,220 seg	0,220 seg	42/42/42

Quadro 4. Resultados obtidos com o compilador Anjuta.

Na quinta coluna do Quadro 4, são apresentados o número de comparações para cada caso respectivamente, como por exemplo, na segunda linha, os dados referentes ao MaxMin1 são 78 comparações para o melhor, pior e para o caso médio, já na terceira linha, o MaxMin2 são 79 comparações para o caso médio e para o pior caso, e apenas 43 para o melhor caso, e na quarta linha, o MaxMin3 são 42 comparações para o pior, melhor e para o caso médio, se igualando nas comparações com o DEV C++, apresentado no quadro 2.

Comparação dos resultados dos compiladores

De acordo com o Quadro 3, mostra que o Borland é o mais lento em quesito de velocidade de execução comparado com os quadros 2 e o 4 pertencentes ao DEV C++ e do Ajunta respectivamente, que mostra os mesmo tempos de execução com uma diferença significativa de valores. Em comparações todos devem ter o mesmo tanto de operações, pois foram usados os mesmos dados de entrada.

Alguns compiladores, que oferecem mais recursos acabam se tornando inviável já que necessitam de mais processamento para serem tão eficientes quantos outros que podem ser mais leves, porém, alguns acabam oferecendo os recursos básicos de forma mais intuitiva, mostrando os erros encontrados de forma, onde com alguns cliques podem ser corrigidos.

Voltando a velocidade de execução, prova que sim, algoritmos compilados em compiladores diferentes, podem ser afetados pelo carregamento do programa já compilado para a memória do computador, para ser executado, mas nem sempre o que executa mais rápido, seja o programa mais intuitivo a ser utilizado.

CONSIDERAÇÕES FINAIS

Compiladores podem sim, alterar a velocidade da execução dos algoritmos, apenas melhorando o código, pode não ser suficiente para otimizar um algoritmo, quando compiladores mais leves entram em ação eles conseguem executar algoritmos com mais velocidade, dependendo do algoritmo e da finalidade, isso pode até ajudar, como por exemplo, algoritmos feitos educacionalmente, já que talvez precisa-se de algoritmos grandes, que demorem a ser executado, com muitas contas, se o compilador for lento, ele pode levar até cinco vezes mais tempo, que um compilador eficaz.

Deve-se levar em consideração que também, existem compiladores que por mais que sejam leves e rápidos, não possuem todas as funções que um compilador mais lento tem e que pode agilizar a execução de funções especiais para os códigos. Assim como as bibliotecas, que vêem inclusas em alguns compiladores e que nos compiladores mais leves, devem ser adicionada-as manualmente para que possam ser incorporadas ao código executável.

REFERÊNCIAS

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

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.

SALVETTI, Dirceu Douglas. BARBOSA, Lisbete Madsen. **Algoritmos**. São Paulo - SP: Makron Books, 1998.

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

Luan Roger S. Santana é discente do curso de Engenharia da Computação da Fundação Educacional de Ituiutaba, associada à Universidade do Estado de Minas Gerais, Campus de Ituiutaba-MG.
luanengcomputacao@hotmail.com.

Ricardo de Oliveira Muniz Junior é discente do curso de Engenharia da Computação da Fundação Educacional de Ituiutaba, associada à Universidade do Estado de Minas Gerais, Campus de Ituiutaba-MG.
rricardinh@hotmail.com

INTERCURSOS - REVISTA DAS UNIDADES ACADÊMICAS DA FUNDAÇÃO EDUCACIONAL DE ITUIUTABA.

Intercursos, v. 9, n. 1, Jan-Jun 2010

Universidade do Estado de Minas Gerais, Unidade Associada
Campus de Ituiutaba.

Semestral.
ISSN Nº 2179-9059
CDD: 011.34