Centro Universitário da FEI

Programa de Mestrado e Doutorado em Engenharia Elétrica

PEL 201 – Algoritmos Computacionais

**Relatório 1**

**Máximo Divisor Comum (MDC)**

Claudio Aparecido Borges Junior

Matrícula: 120122-7

**MÁXIMO DIVISOR COMUM**

O máximo divisor comum (MDC) entre dois números inteiros é o maior número inteiro que é fator desses dois números. Para encontrar o MDC pode-se fatorar ambos os números e a multiplicação dos fatores comuns é o MDC. Por exemplo

**SOLUÇÃO ITERATIVA**

A solução iterativa inicia com o menor valor *i* entre ambos números *a* e *b*; cada iteração consiste em verificar se *a* e *b* são ambos divisíveis por *i;* se sim, o MDC foi encontrado e o algoritmo termina; se não*, i* é subtraído de 1 unidade.

Inicialmente, deve-se verificar se *a* ou *b* é igual a 0, nesse caso, deve-se retornar o valor máximo entre eles. Isso ocorre porque *.*

O algoritmo implementado em *Python* foi:

def interactive\_gdc(a, b):

1 i = min(a, b)

2 if i == 0:

3 return max(a, b)

4 while ((a % i) != 0) or ((b % 1) != 0):

5 i -= 1

6 return i

A pior condição ocorre quando , nesse caso o algoritmo inicia em e decresce até 1. A melhor condição ocorre quando .

Na pior condição temos:

def interactive\_gcd(a, b):

1 i = min(a, b) c1 1

2 if i == 0: c2 1

3 return max(a, b) c3 0

4 while ((a % i) != 0) or ((b % 1) != 0): c4 min(a, b)

5 i -= 1 c5 min(a, b) - 1

6 return i c6 1

Portando:

**SOLUÇÃO RECURSIVA**

A solução recursiva utiliza o algoritmo de Euclides. O algoritmo de Euclides pode ser descrito como:

Isto é, a própria função de *mdc* é chamada recursivamente até uma condição de parada. A condição de parada ocorre quando *m* é igual a zero, desde que .

def recursive\_gcd(n, m):

1 if m == 0:

2 return n

3 return recursive\_gcd(m, n % m)

O algoritmo euclidiano se baseia em duas observações:

1. Se m divide n, isto é, então
2. Se , então

A função que representa o resto de uma divisão possui um decaimento representado por . Sendo n o número de bits:

Utilizando o método da substituição podemos demostrar que *T(n)* está em *O(lg n)*:

Portando:

**COMPARATIVO DE DESEMPENHO DE TEMPO**

Como esperado, o algoritmo iterativo apresentou um crescimento linear, enquanto o recursivo apresentou uma ordem de crescimento superior limitada a O(lg n).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **a** | **b** | **mdc** | **min(a, b)** | **t\_iterativa [s]** | **t\_recursiva [s]** |
| 99851527 | 23412337 | 1 | 23412337 | 1,3512E+00 | 1,2457E-05 |
| 199703052 | 46824670 | 2 | 46824670 | 2,6761E+00 | 9,7156E-06 |
| 299554580 | 70236997 | 1 | 70236997 | 4,0204E+00 | 1,0729E-05 |
| 399406103 | 93649330 | 1 | 93649330 | 5,3481E+00 | 9,7156E-06 |
| 499257633 | 117061668 | 3 | 117061668 | 6,7484E+00 | 1,4663E-05 |
| 599109153 | 140473994 | 1 | 140473994 | 8,1550E+00 | 9,1791E-06 |
| 698960682 | 163886332 | 2 | 163886332 | 9,3726E+00 | 9,5367E-06 |
| 798812206 | 187298663 | 1 | 187298663 | 1,0727E+01 | 1,0312E-05 |
| 898663735 | 210710995 | 5 | 210710995 | 1,2044E+01 | 1,0669E-05 |
| 998515253 | 234123326 | 1 | 234123326 | 1,3406E+01 | 1,0192E-05 |
| 1098366776 | 257535658 | 2 | 257535658 | 1,4748E+01 | 9,2983E-06 |
| 1198218302 | 280947989 | 1 | 280947989 | 1,6822E+01 | 1,4961E-05 |
| 1298069830 | 304360324 | 2 | 304360324 | 1,7598E+01 | 1,0729E-05 |
| 1397921357 | 327772656 | 1 | 327772656 | 1,8845E+01 | 1,4246E-05 |
| 1497772877 | 351184990 | 1 | 351184990 | 2,0564E+01 | 1,1742E-05 |
| 1597624408 | 374597322 | 2 | 374597322 | 2,1349E+01 | 1,0192E-05 |
| 1697475928 | 398009652 | 4 | 398009652 | 2,2905E+01 | 1,5497E-05 |
| 1797327458 | 421421979 | 1 | 421421979 | 2,4145E+01 | 1,1027E-05 |
| 1897178983 | 444834317 | 1 | 444834317 | 2,5425E+01 | 1,2755E-05 |