



UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

COMPONENTE CURRICULAR: LABORATÓRIO DE ALGORITMOS E ESTRUTURA DE  
DADOS II

DOCENTE: KENNEDY REURISON LOPES

DISCENTE(S): ISABEL DE PAIVA FREIRE - 2024010417

ATIVIDADE SOBRE COMPLEXIDADE DE ALGORITMOS - DEPURAÇÃO DE ALGORITMOS  
RECURSIVOS

PAU DOS FERROS

MAIO DE 2025

## 1. ANÁLISE DE COMPLEXIDADE

Figura 01: exemplo de algoritmo para cálculo de função recursiva

---

### Algorithm 1: Cálculo da função recursiva $x(n)$

---

**Input:** Algoritmo para  $x(n)$  avaliado para  $n = n_0$

**Output:** Tempo médio do algoritmo  $x(n)$  para  $n = n_0, N_{Max}$

$N = 0;$

$T = 0;$

**if**  $N \leq N_{Max}$  **then**

**return**  $T / N_{Max};$

**end**

**else**

$N = N + 1;$

$T = T + \text{tempo\_execucao}(x(n_0));$

    Execute o algoritmo para calcular  $x(n_0);$

**end**

---

Fonte: material do professor.

- Transcrição do código para a linguagem c (código comentado para melhor compreensão):

```
#include <stdio.h>

#include <time.h> // biblioteca para medir o tempo de execução
do algoritmo

double tempo_execucao(int n) { // função que calcula o tempo de
execução da função recursiva x(n)
    clock_t inicio = clock();
    clock_t fim = clock();
    return (double)(fim - inicio) / CLOCKS_PER_SEC;
}

double calcular_tempo_medio(int n0, int NMax) { // função que
calcula o tempo médio da função recursiva x(n)
    int N = 0;
    double T = 0;

    while (N <= NMax) { // laço de repetição
```

```

        N++;

        T += tempo_execucao(n0); // função para medir o tempo de
        execução da função x(n0)
    }

    return T / NMax;
}

int main() { //função principal
    int n0, NMax;

    printf("Digite o valor inicial n0: "); // interação com o
    usuário
    scanf("%d", &n0);
    printf("Digite o valor maximo NMax: ");
    scanf("%d", &NMax);

    double tempoMedio = calcular_tempo_medio(n0, NMax); // cálculo
    do tempo médio
    printf("Tempo medio de execucao: %.6f segundos\n",
    tempoMedio);

    return 0;
}

```

- Saída do código:

```

PROBLEMAS 1 SAÍDA CONSOLE DE DEPURACÃO TERMINAL PORTAS
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> cd 'c:\Users\Isa\Desktop\Isabel\UFERSA\III
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> cd 'c:\Users\Isa\Desktop\Isabel\UFERSA\III
Período\Lab. algoritmos & estrutura de dados II\lista 01\output'
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01\output> & .\questao01.exe'
Digite o valor inicial n0: 2
Digite o valor maximo NMax: 5
Tempo medio de execucao: 0.000000 segundos
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01\output> 

```

## 2. FUNÇÃO RECURSIVA PARA ANÁLISE

Figura 02: fórmula do cálculo de função recursiva

Considere a seguinte função recursiva definida para valores inteiros positivos  $n$ :

$$x(n) = \begin{cases} 1 & \text{se } n \leq 1, \\ x(n-1) + x(n-2) & \text{caso contrário.} \end{cases}$$

Fonte: material do professor.

- A fórmula apresentada acima pode ser encontrada na sequência fibonacci, traduzida em código abaixo:

```
#include <stdio.h>

int fibonacci (int);
int main() {
    int n = 10;
    printf("%d\n", fibonacci(n));
}

int fibonacci (int n) {
    if (n <= 1)
        return 1;
    else {
        return fibonacci ( n -1) + fibonacci ( n -2 );
    }
}
```

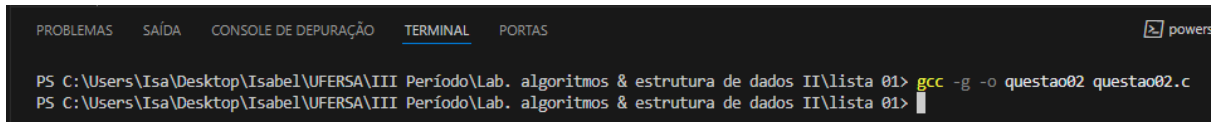
- Saída do código:

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPUÇÃO  TERMINAL  PORTAS
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> cd 'c:\Users\Isa\Desktop\Isabel\UFERSA\III
Período\Lab. algoritmos & estrutura de dados II\aula II\output'
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\aula II\output> & .\exemplo01.exe
89
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\aula II\output> █
```

### 3. DEPURAÇÃO DO CÓDIGO UTILIZANDO O GDB

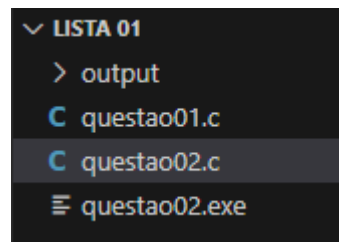
#### 3.1. Compilação do código utilizando a flag -g para incluir as informações de depuração

- A partir do comando `gcc -g -o questao02 questao02.c`:

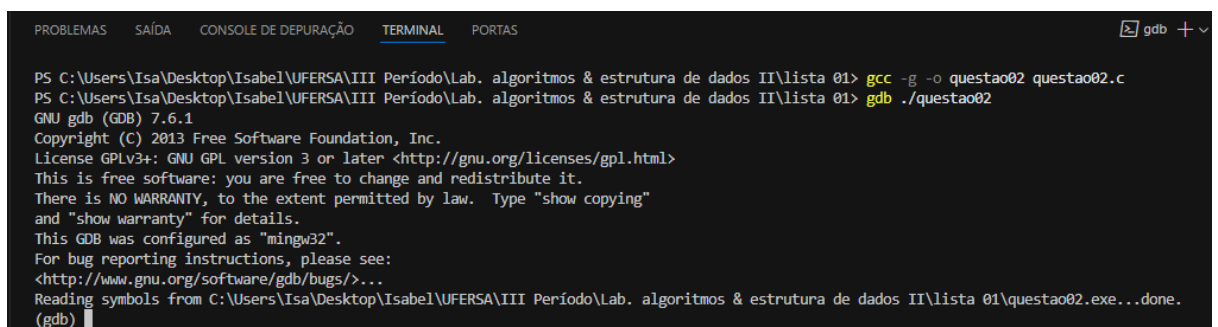


```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> gcc -g -o questao02 questao02.c
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>
```

- Um arquivo executável foi criado:



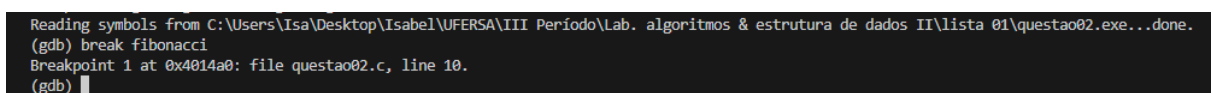
#### 3.2. Iniciar o GDB com o executável gerado:



```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> gcc -g -o questao02 questao02.c
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> gdb ./questao02
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01\questao02.exe...done.
(gdb)
```

#### 3.3. Definir um ponto de interrupção no GDB na função desejada

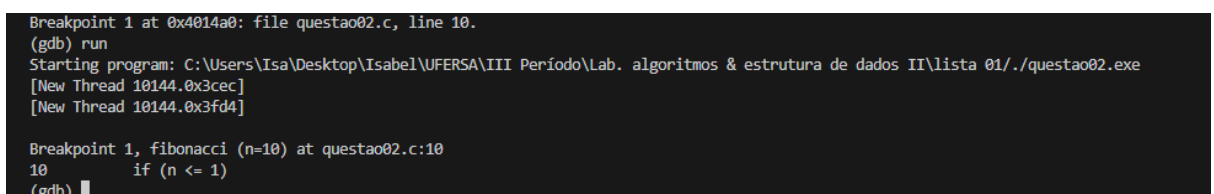
- A função a ser parada foi a função fibonacci, note que no terminal é mostrado a sua linha de comando exata que foi parada a execução:



```
Reading symbols from C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01\questao02.exe...done.
(gdb) break fibonacci
Breakpoint 1 at 0x4014a0: file questao02.c, line 10.
(gdb)
```

#### 3.4. Executar o programa no GDB

- Ao executar o comando `run`, o programa volta a funcionar, podemos perceber isso pelo próprio terminal:



```
Breakpoint 1 at 0x4014a0: file questao02.c, line 10.
(gdb) run
Starting program: C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01\./questao02.exe
[New Thread 10144.0x3cec]
[New Thread 10144.0x3fd4]

Breakpoint 1, fibonacci (n=10) at questao02.c:10
10      if (n <= 1)
(gdb)
```

#### 3.5. Inspeção de variáveis

- Nesse caso, a inspeção ocorreu na variável `n`

```
Breakpoint 1, fibonacci (n=10) at questao02.c:10
10      if (n <= 1)
(gdb) print n
$1 = 10
(gdb) █
```

### 3.6. Avançando linhas de código

- Foi utilizado somente o comando next:

```
$1 = 10
(gdb) next
13      return fibonacci ( n -1) + fibonacci ( n -2 );
(gdb) █
```

Compile Debug

### 3.7. Saindo do gdb

- Basta utilizar o comando quit:

```
(gdb) quit
A debugging session is active.

Inferior 1 [process 10144] will be killed.

Quit anyway? (y or n) y
error return ../../gdb-7.6.1/gdb/windows-nat.c:1275 was 5
error return ../../gdb-7.6.1/gdb/windows-nat.c:1275 was 5
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> █
```

### 3.8. Reiniciando a execução desde o início

- Basta utilizar o comando run novamente, mas antes é necessário entrar novamente no gdb:

```
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> gdb ./questao02
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01\questao02.exe...done.
(gdb) run
Starting program: C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01\./questao02.exe
[New Thread 7216.0x3388]
[New Thread 7216.0x10b4]
89
[Inferior 1 (process 7216) exited normally]
(gdb) █
```

#### 4. CRONOMETRANDO O ALGORITMO

- Montando o algoritmo:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int fibonaccil (int n){
    if (n <= 1){
        return n;
    }
    return fibonaccil (n - 1) + fibonaccil (n - 2);
}

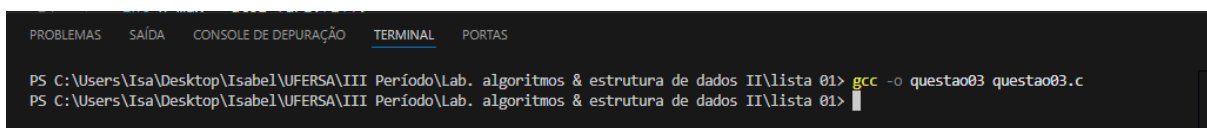
int main(int argc, char *argv[]){
    int n = atoi (argv[1]);
    int n_max = atoi (argv[2]);
    clock_t start, end;
    double tempo = 0.0;
    int resultado;

    for (size_t i = 0; i < n_max; i++){
        start = clock();
        resultado = fibonaccil(n);
        end = clock();
        tempo += ((double) (end - start)) / CLOCKS_PER_SEC;
        printf("Progresso: %.2f%% concluido\n", ((i + 1) / (double)
n_max * 100));
    }

    printf ("Fibonacil na posicao %d is %d\n", n, resultado);
    printf ("Tempo medio: %f s\n", (tempo / n_max) * 1e6);

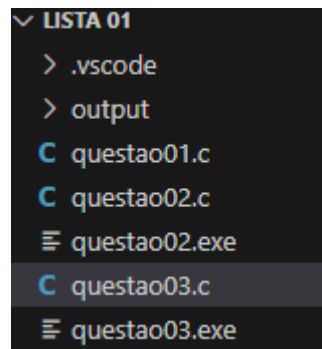
    return 0;
}
```

- Compilando para receber o tempo médio do código:



The screenshot shows a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMAS', 'SAÍDA', 'CONSOLE DE DEPUÇÃO', 'TERMINAL' (which is active), and 'PORTAS'. Below the tabs, the terminal shows the command prompt 'PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>' followed by the command 'gcc -o questao03 questao03.c'. The next line shows the prompt again followed by a cursor.

- Isso gera um arquivo executável:



- Colocando diversos valores:

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS

PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> gcc -o questao03 questao03.c
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> ./questao03 10 5
Progresso: 20.00% concluido
Progresso: 40.00% concluido
Progresso: 60.00% concluido
Progresso: 80.00% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 10 is 55
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>

PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> ./questao03 10 10
Progresso: 10.00% concluido
Progresso: 20.00% concluido
Progresso: 30.00% concluido
Progresso: 40.00% concluido
Progresso: 50.00% concluido
Progresso: 60.00% concluido
Progresso: 70.00% concluido
Progresso: 80.00% concluido
Progresso: 90.00% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 10 is 55
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>

PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> ./questao03 20 10
Progresso: 10.00% concluido
Progresso: 20.00% concluido
Progresso: 30.00% concluido
Progresso: 40.00% concluido
Progresso: 50.00% concluido
Progresso: 60.00% concluido
Progresso: 70.00% concluido
Progresso: 80.00% concluido
Progresso: 90.00% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 20 is 6765
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>
```

- Dessa forma, pode-se concluir que o algoritmo recursivo Fibonacci tem complexidade exponencial, ou seja,  $T(n) = O(2^n)$ . Ao medir o tempo médio de execução para diferentes valores de  $n$ , podemos perceber que o tempo aumenta rapidamente (mesmo que o terminal não acuse, mas o tempo de espera pode ser notado), confirmando o crescimento exponencial. O comportamento observado bate com a análise teórica — o tempo de execução cresce de forma exponencial conforme o valor de  $n$  aumenta.



## 5. MELHORANDO O ALGORITMO - ALGORITMOS OTIMIZADOS

- Melhorias no código:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int fib [1600] = {0};

int fibonacci2(int n){
    if (n <= 1){
        return n;
    }
    if (fib [n] != 0) {
        return fib [n];
    }

    fib[n] = fibonacci2(n - 1) + fibonacci2(n - 2);
    return fib[n];
}

int fibonaccil (int n){
    if (n <= 1){
        return n;
    }
    return fibonaccil (n - 1) + fibonaccil (n - 2);
}

int main(int argc, char *argv[]){
    int n = atoi (argv[1]);
    int n_max = atoi (argv[2]);
    clock_t start, end;
    double tempo = 0.0;
    int resultado;

    for (size_t i = 0; i < n_max; i++){
        start = clock();
        resultado = fibonacci2(n);
        end = clock();
        tempo += ((double) (end - start)) / CLOCKS_PER_SEC;
        printf("Progresso: %.2f%% concluido\n", ((i + 1) / (double)
n_max * 100));
    }
}
```

```

printf ("Fibonacil na posicao %d is %d\n", n, resultado);
printf ("Tempo medio: %f s\n", (tempo / n_max) * 1e6);

return 0;
}

```

- Utilizando o comando:

```

PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL  PORTAS

PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> gcc -o questao04 questao04.c
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>

```

- Cria-se um novo arquivo executável:

```

C questao03.c
≡ questao03.exe
C questao04.c
≡ questao04.exe

```

- Executando algumas vezes com valores diferentes:

```

PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL  PORTAS

PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> gcc -o questao04 questao04.c
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> ./questao04 40 10
Progresso: 10.00% concluido
Progresso: 20.00% concluido
Progresso: 30.00% concluido
Progresso: 40.00% concluido
Progresso: 50.00% concluido
Progresso: 60.00% concluido
Progresso: 70.00% concluido
Progresso: 80.00% concluido
Progresso: 90.00% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 40 is 102334155
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>

```

```

PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO  TERMINAL  PORTAS

Progresso: 87.00% concluido
Progresso: 88.00% concluido
Progresso: 89.00% concluido
Progresso: 90.00% concluido
Progresso: 91.00% concluido
Progresso: 92.00% concluido
Progresso: 93.00% concluido
Progresso: 94.00% concluido
Progresso: 95.00% concluido
Progresso: 96.00% concluido
Progresso: 97.00% concluido
Progresso: 98.00% concluido
Progresso: 99.00% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 400 is 650574555
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>
Ln 35, Col 23  Espaços: 4  UTF-8  CRLF  {}

```

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS
Progresso: 98.70% concluido
Progresso: 98.80% concluido
Progresso: 98.90% concluido
Progresso: 99.00% concluido
Progresso: 99.10% concluido
Progresso: 99.20% concluido
Progresso: 99.30% concluido
Progresso: 99.40% concluido
Progresso: 99.50% concluido
Progresso: 99.60% concluido
Progresso: 99.70% concluido
Progresso: 99.80% concluido
Progresso: 99.90% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 4000 is -1502807888
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>
```

- Testando o algoritmo com valores exorbitantes

Ao executar o mesmo comando com valores diferentes, por exemplo 400 e 100, o algoritmo leva alguns segundos a mais para executar, isso foi perceptível para o usuário da máquina, mas o terminal (marcado em segundos) não acusou o tempo de espera.

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS
Progresso: 97.00% concluido
Progresso: 97.25% concluido
Progresso: 97.50% concluido
Progresso: 97.75% concluido
Progresso: 98.00% concluido
Progresso: 98.25% concluido
Progresso: 98.50% concluido
Progresso: 98.75% concluido
Progresso: 99.00% concluido
Progresso: 99.25% concluido
Progresso: 99.50% concluido
Progresso: 99.75% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 100 is -980107325
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01>
```

Dessa vez utilizando quatro dígitos - 1000 e 4000

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL  PORTAS

Progresso: 99.70% concluido
Progresso: 99.72% concluido
Progresso: 99.75% concluido
Progresso: 99.78% concluido
Progresso: 99.80% concluido
Progresso: 99.83% concluido
Progresso: 99.85% concluido
Progresso: 99.88% concluido
Progresso: 99.90% concluido
Progresso: 99.92% concluido
Progresso: 99.95% concluido
Progresso: 99.97% concluido
Progresso: 100.00% concluido
Fibonacil na posicao 1000 is 1556111435
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> |
```

Ln 35, Col 23 Esp

Já utilizando 5 dígitos (10.000 e 40.000), o algoritmo não foi executado:

```
Fibonacil na posicao 1000 is 1556111435
Tempo medio: 0.000000 s
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> ./questao04 10000 40000
PS C:\Users\Isa\Desktop\Isabel\UFERSA\III Período\Lab. algoritmos & estrutura de dados II\lista 01> |
```

Ln 35, Col 23 Espaços: 4 U

## 6. APRESENTAÇÃO DOS RESULTADOS

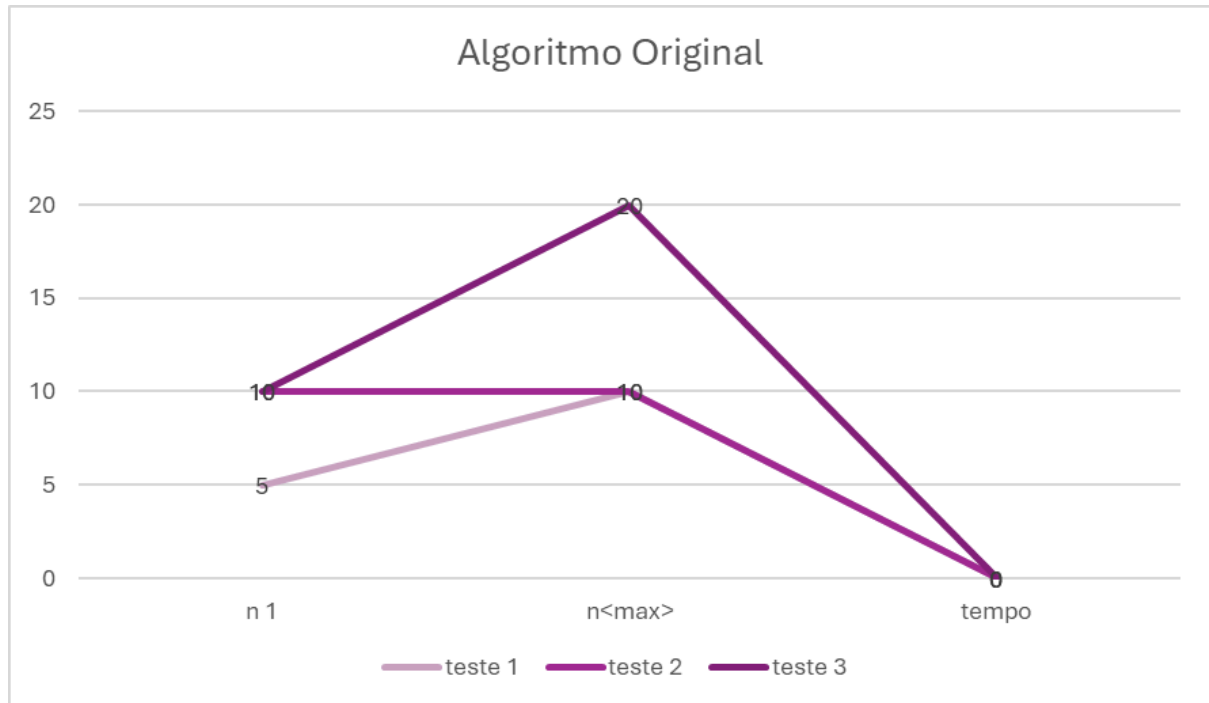


Figura 3: gráfico ilustrativo com o algoritmo original em relação ao tempo medido em segundos. Ao passar de 3 dígitos para o valor de  $n=1$  o programa já não conseguia executar.

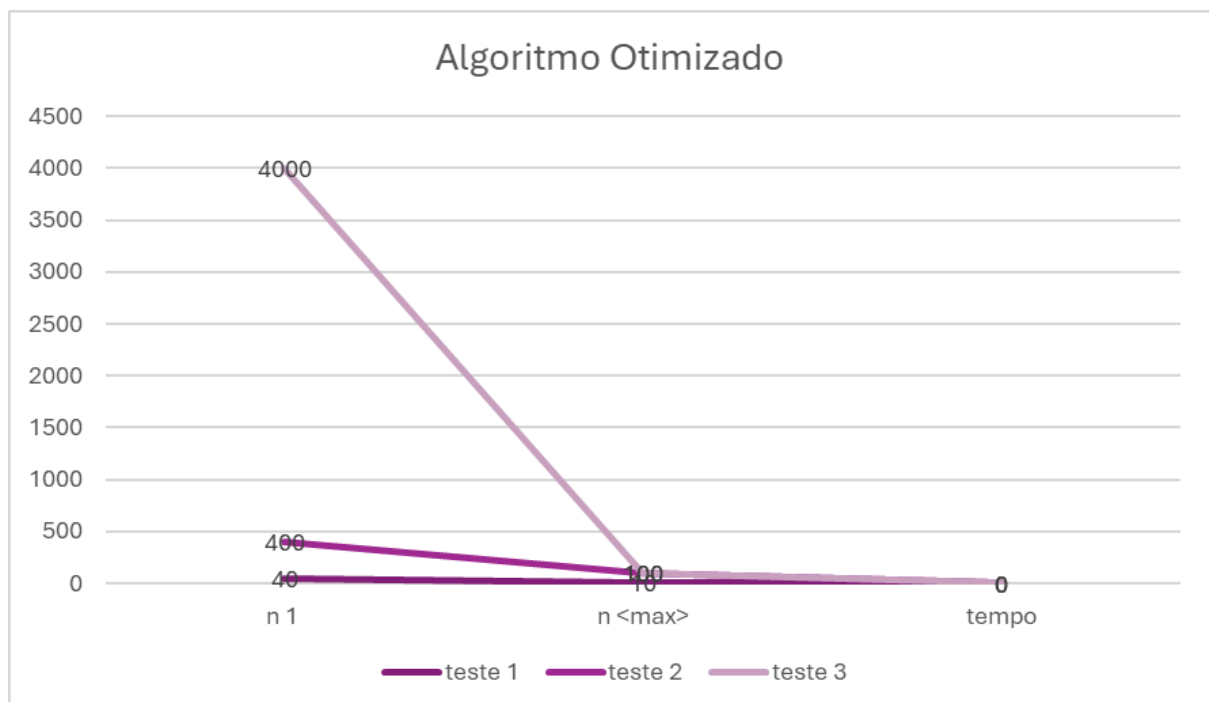


Figura 4: gráfico ilustrativo com o algoritmo otimizado em relação ao tempo medido em segundos. Este conseguia atingir até 4 dígitos para n1 na execução.

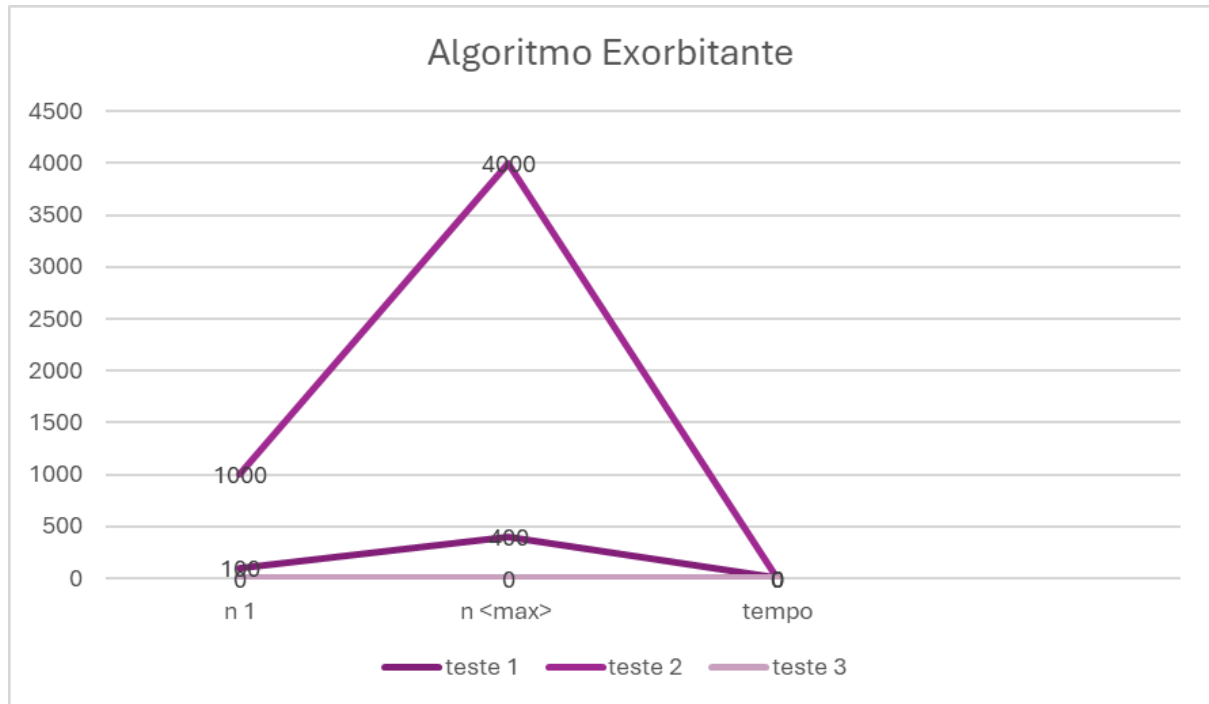


Figura 5: gráfico ilustrativo com o algoritmo com valores exorbitantes em relação ao tempo medido em segundos. Foram realizados 3 testes, o último deles (usando valores de 5 dígitos) não foi executado pela máquina.