Universidade Federal do Rio Grande do Norte
DCT1106 - Programação
Discente: Raquel Lima Fernandes

Avaliação Semana 7

Interatividade versos Recursividade: analisando o algoritmo da Sequência de Fibonacci com melhor desempenho de execução.

1. Usando Recursão

A função recursiva é aquela que “chama a si própria”. Uma possível definição da função de Fibonacci em C seria:

```c
int fiboRec(int n) {
  if ((n == 1) || (n == 2))
    return 1;
  else
    return fiboRec(n-1) + fiboRec(n-2);
}
```

Essa solução é mais legível e possui menos texto, na forma como está escrita, cada chamada da função irá chamar outras duas funções, o que pode vir a se tornar um problema.

2. Usando Interatividade

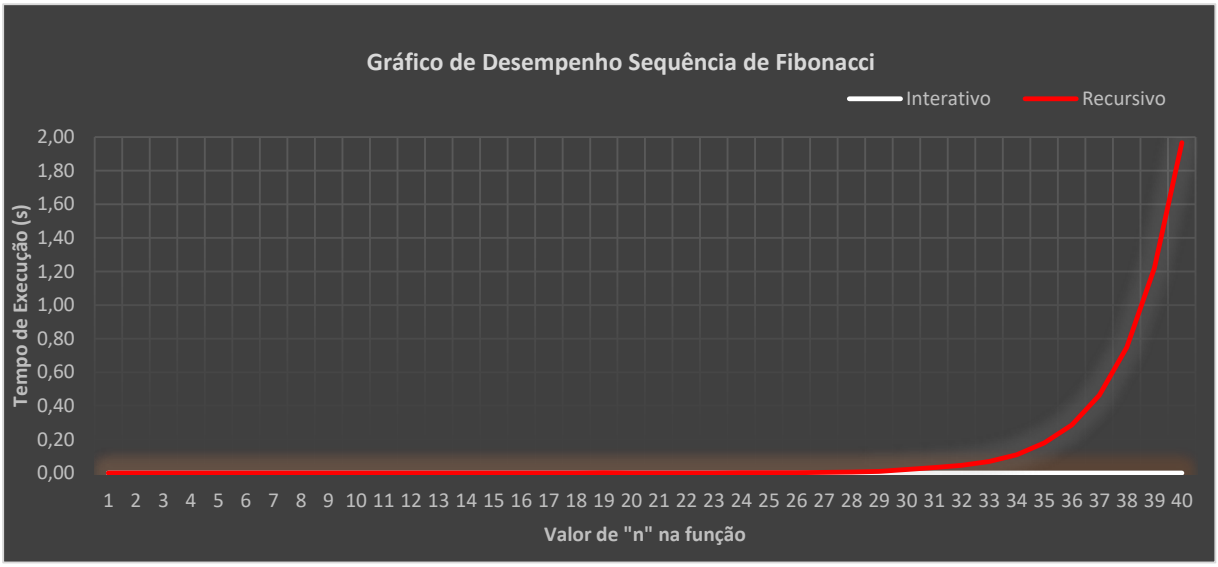
Um algoritmo interativo é aquele que possui uma estrutura de repetição, nesse caso o *for.* Esta seria das formas mais “usuais” de implementar a função da Sequência de Fibonacci em C.

```c
 int fiboItr(int n) {
  if ((n == 1) || (n == 2))
    return 1;
  else {
    int a = 1;
    int b = 1;
    int c;
    for (int k = 3; k <= n; k++) {
      c = a + b;
      a = b;
      b = c;
    }
    return c;
  }
}
```

Á primeira vista, o código parece mais robusto e longo em relação ao recursivo.

Comparando as Execuções Graficamente

Os dois códigos acima foram executados e seus tempos de duração mensurados através do Visual Studio Code utilizando linguagem C. As funções foram testadas para vários *n* diferentes, de 1 até 40. Os desempenhos (em segundos) estão dispostos no gráfico abaixo:



Fonte: Própria autora.

No gráfico, quanto mais a curva subir, mais trabalho o código teve para ser executado, isto é, pior foi o desempenho. Observam-se de início: uma delas fez uma linha reta horizontal constante e a outra até um certo valor de *n* também teve o mesmo comportamento, até que em um determinado ponto cresceu estranhamente muito rápido.

O tempo necessário para executar a forma recursiva virou uma linha exponencial e contrariando as expectativas o código pequeno e legível (o recursivo) apresentou um desempenho de execução bem inferior ao código interativo.

Diferente do código recursivo, a função interativa se mostrou mais otimizada exibindo uma linha de desempenho constante, mesmo com o *n* aumentando, o tempo de execução da função não mudou.