Sequência de Fibonacci

Universidade Federal
de São João del-Rei

Trabalho Pratico

Feito por: Higor Henrique Alves

Introdução

Provavelmente um dos algoritmos mais famosos de todos os tempos, mas ainda sim muitas pessoas tem dificuldade em achar uma implementação eficiente para este problema. Vamos conhecer um pouco mais sobre a sequência de Fibonacci.

Dado um numero N retorne o valor da sequência baseado no N, onde a é dada por:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89...

Após uma rápida olhada, é possível notar facilmente um padrão nesta sequência, a qual todo valor é a soma dos dois valores anteriores.

F(n) = F(n-1) + F(n-2)

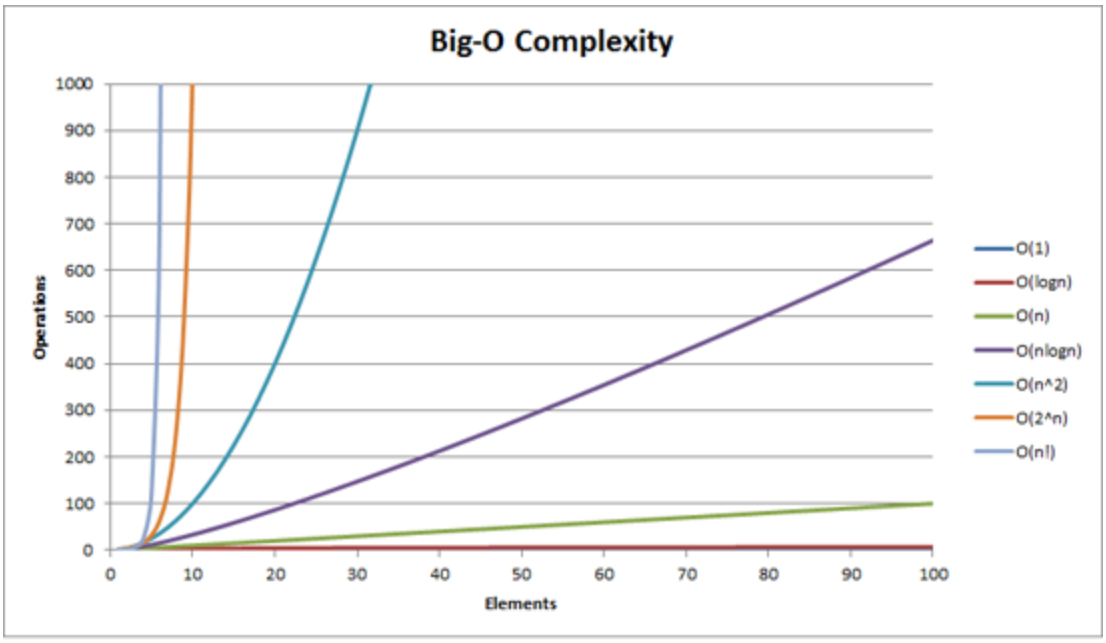
Materiais e Métodos

A implementação foi feita em quatro linguagens diferentes, com paradigmas diferentes, sendo estas linguagens, C, Cobol (por motivos de curiosidade de como funciona a linguagem e seu compilador), Java, JavaScript e Lisp. Todos os códigos tiveram a implementação o mais próxima possível de cada uma.

Nas linguagens mais conhecidas como C, Javascript e Java foram usadas implementações recursivas e imperativas, para fins de comparação, já em Lisp e Cobol foram usadas implementações interativas devido a grande dificuldade em se escrever um código para estas linguagens.

Resultados e Discussão

Para as implementações que não usaram recursões a ordem de complexidade se manteve dentro de O(n), bastante justo para um problema que se resolve bem rápido. Já para as Soluções com recursividade, o tempo de execução foi extremamente maior comparado com os outros, olhando para o gráfico vemos uma linha na cor laranja (2^n) do tempo de complexidade, o que diz que o código foi gerado em uma implementação exponencial.



Nos testes realizados, enquanto o loop gastava 0.000001ms para ser executado com N sendo de tamanho 20 e com 51 chamadas de função, as opções recursivas gastaram em media 176.748ms para serem executadas, lembrando que esta foi uma media realizada entre as linguagens, sendo que o Java ficou com tempo médio de 180.452, Javascript com 175.325ms e C com 170.658ms, tendo em media 20.365.011.074 chamadas de funções o que é extremamente maior que os algoritimos usando loops.

Conclusão

Este foi um trabalho para se analisar como desenhar nossas funções, melhorar e crescer o algoritmo, neste caso a sequência de Fibonacci, a qual é simples o suficiente para entender e surpreendentemente é muito perceptível como nem sempre escrever menos código ou usar técnicas mais avançadas de programação fazem ganhar na performance do algoritmo.

Linguagens mais novas e com paradigmas mais simples, saíram melhor no requisito simplicidade de código já que sua leitura é facilmente realizada em pouco tempo. Paradigmas como o da linguagem Lisp, acabaram atrapalhando a performance do programa por fazer sempre muitas chamadas de função.

Concluímos que para que um projeto tenha qualidade e performance é preciso que estudemos nosso problema e verifiquemos qual melhor paradigma se encaixa ali e qual o a melhor forma de se implementar o mesmo.