Prática Avaliativa - Documentação

Luiz Felipe Gondim Araujo - 2023028188

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

luizfelipegondimaraujo@gmail.com

1 Introdução

Esta documentação apresenta os resultados dos experimentos realizados na Prática Avaliativa 3, na qual deveríamos analisar o tempo de execução dos algoritmos iterativos e recursivos de diferentes problemas. Em suma, foi utilizado a linguagem C++ e o sistema operacional Linux Mint 21 para realizar a prática.

2 Análise do tempo de relógio

Para efetuar os experimentos, foram considerados os números 1, 5, 10, 15, 20 e 25 como valores de entrada para as funções. Com base nisso, para cada valor, houveram 10 testes, sendo que cada resultado - em segundos - foi anotado para fazer uma média de tempo a ser usada nos gráficos abaixo:

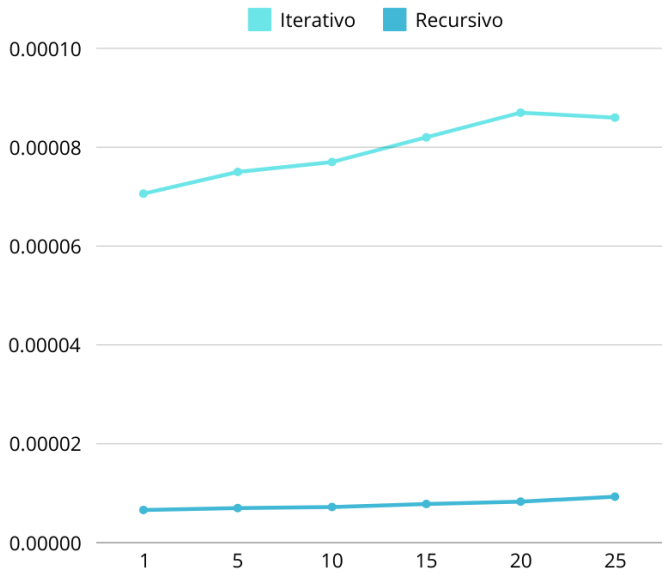

Figura 1: Comparação do código Fatorial

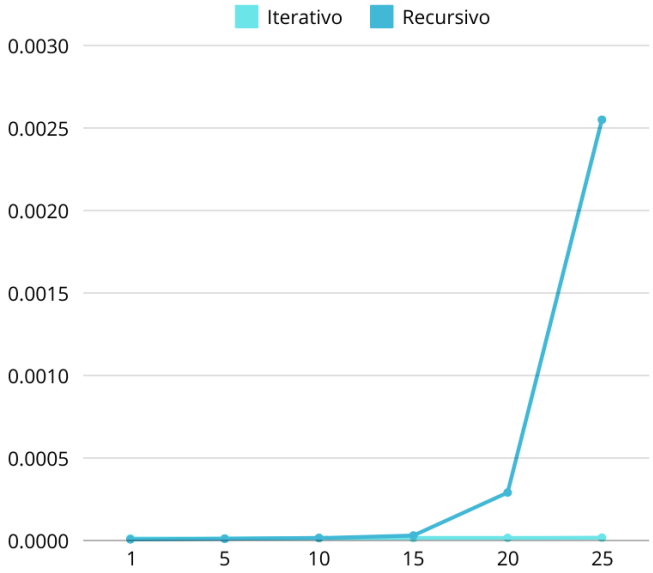

Figura 2: Comparação do código Fibonacci

Em relação às versões do código que calcula o fatorial de n, podemos notar que as duas cresceram em uma taxa linear, assim como suas complexidades de tempo sugerem (O(n)). No entanto, é percepítivel que para a faixa de valores testados, a versão iterativa demora um pouco mais para ser computada.

Para as versões de Fibonacci, conseguimos perceber a diferença na taxa de crescimento das duas funções, dado que a complexidade da função recursiva é $O(2^n)$ e da função iterativa é O(n). Usando valores muito maiores que 25, o tempo recursivo tende a crescer ainda mais, impossibilitando a sua análise.

3 Análise de tempos de usuário e de sistema

A seguinte análise foi realizada por meio do comando "time", uma vez que ele retorna o tempo de usuário, de sistema e o tempo real total gasto pelo software. Para uma melhor exatidão, foram feitas cinco testes com esse comando, retornando os seguintes resultados:

0,00s user 0,00s system 93% cpu 0,002 total
0,00s user 0,00s system 86% cpu 0,003 total
0,00s user 0,00s system 92% cpu 0,003 total
0,00s user 0,00s system 93% cpu 0,003 total
0,00s user 0,00s system 93% cpu 0,002 total

Os resultados mostram tempos de execução quase instantâneos (0,002 a 0,003s de tempo de relógio) com uso mínimo da CPU, o que indica a baixa complexidade prática das funções para os valores pequenos de entrada. Tanto as versões iterativas (de complexidade linear O(n)) quanto as recursivas foram rápidas o suficiente para que o sistema não registrasse tempos distintos de CPU

(user e system time ficaram em 0,00s).

A alta porcentagem de uso da CPU (86% a 93%) sugere um estado ocioso relativamente pequeno para o sistema operacional. Ademais, em valores de entrada maiores, especialmente para Fibonacci recursivo, a complexidade exponencial $O(2^n)$ tornaria o tempo de computação perceptível.

4 Análise de tempo com o gprof

Após a análise realizada com gprof, foi obtido o seguinte resultado:

```
Flat profile:

Each sample counts as 0.01 seconds.
no time accumulated

  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
 0.00   0.00      0.00       1      0.00     0.00     fatorial_recursivo(int)
 0.00   0.00      0.00       1      0.00     0.00     fibonacci_recursivo(int)
 0.00   0.00      0.00       1      0.00     0.00     fatorial(int)
 0.00   0.00      0.00       1      0.00     0.00     fibonacci(int)
```

Ao comparar o tempo de execução de algoritmos recursivos usando o tempo de relógio e o gprof, é possível perceber que cada método tem limitações distintas. O tempo de relógio mede de forma precisa e contínua a duração das chamadas, capturando todos os detalhes de execução. Isso permite uma análise detalhada de cada passo da recursão, especialmente em algoritmos com muitas chamadas rápidas, como Fibonacci. Já o gprof usa uma abordagem periódica para capturar o uso de CPU, acumulando uma visão mais ampla e ideal para funções que executam operações mais prolongadas. No entanto, ele pode não captar bem funções recursivas rápidas, como neste caso, onde os tempos foram todos reportados como zero.

5 Segunda análise de tempo com o gprof

Nessa abordagem, utilizamos uma função que calcula o seno de n 1 milhão de vezes a cada chamada das funções recursivas e obtivemos o seguinte resultado:

```
Flat profile:

Each sample counts as 0.01 seconds.

  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 99.98   156.03   156.03     1       156.03   156.03  fibonacci_recursivo(int)
 0.02    156.06   0.03       1       0.03     0.03    fatorial_recursivo(int)
 0.00    156.06   0.00       1       0.00     0.00    fatorial(int)
 0.00    156.06   0.00       1       0.00     0.00    fibonacci(int)
```

Vemos que a função fibonacci recursivo domina o perfil de execução, consumindo 99,98% do tempo total, com uma execução acumulada de 156 segundos. Isso reflete como gprof se torna mais eficaz em capturar o impacto das operações intensivas dentro de cada chamada, pois o cálculo do seno multiplica o tempo por chamada. Assim, a carga computacional mais alta realça a precisão do gprof para medir o custo de funções recursivas complexas, algo que o tempo de relógio já demonstrava, mas que o gprof agora também representa bem ao focar nas operações prolongadas. Contudo, a função fatorial recursivo ainda mantém um tempo reduzido, evidenciando como o perfil do gprof é diretamente influenciado pela intensidade e frequência de operações dentro das chamadas recursivas.