Trabalho prático 2 – Estrutura de Dados

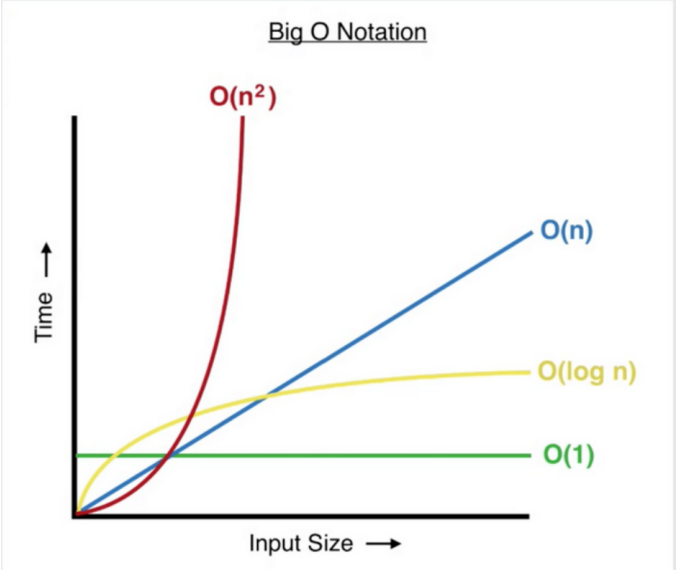
Aluno: Iago da Silva Rodrigues Alves

Relatório de experimentos referentes à avaliação de desempenho de códigos recursivos e não recursivos.

Para este experimento foram utilizadas quatro funções, sendo elas:

- O cálculo de Fibonacci de forma recursiva e iterativa
- O cálculo de Fatorial de forma recursiva e iterativa

Antes de tudo, é válido analisar pelo gráfico disponibilizado logo abaixo, como é a curva de crescimento em relação a diferentes complexidades de algoritmos. Assim, dependendo do tamanho da entrada nas funções temos uma tempo de resposta que pode ser pequeno ou até mesmo muito grande para alguns algoritmos.



Durante os experimentos foram utilizados respectivamente os valores (10, 35, 40, 45) para a análise de desempenhos das funções de Fibonacci e os valores (1, 8 , 12, 20) para as funções Fatorial.

A tabela abaixo traz o tempo de relógio para ambas funções de Fibonacci, as colunas indicam as entradas na função e os valores de cada lacuna indicam o tempo em segundos para resposta. Informações pertinentes que auxiliam na análise desses dados:

- O algoritmo recursivo tem uma complexidade exponencial, O(2^n)
- O algoritmo iterativo tem uma complexidade de tempo linear, O(n)

Agora, juntando essas informações e relacionando-as ao gráfico mostrado anteriormente, fica fácil compreender a diferença entre os dois algoritmos presentes nessa tabela. É visível como o algoritmo recursivo, por ser de complexidade exponencial, à medida que recebe entradas mais altas retorna um tempo cada vez maior para concluir sua função. Já o iterativo, por ser linear, apresenta uma pequena diferença no tempo.

	10	**35**	**40**	**45**
Fibonacci Recursivo	0,000028358	0,068780047	0,737250613	8,141035418
Fibonacci Iterativo	0,000044211	0,000045608	0,000050427	0,000062369

Analisando a mesma tabela mas agora com valores de tempo para as funções Fatorial, percebemos apenas uma pequena variação no tempo de relógio marcado. Estudando um pouco da complexidade desses algoritmos fica claro o motivo pelo qual isso acontece:

- A complexidade dos algoritmos de Fatorial tanto iterativo quanto recursivo é O(n), ou seja, linear

	1	**8**	**12**	**20**
Fatorial Recursivo	0,000035210	0,000036807	0,000043372	0,000055745
Fatorial Iterativo	0,000040648	0,000050456	0,000055245	0,000069632

No entanto, é válido ressaltar que o algoritmo recursivo do fatorial é menos eficiente do que o algoritmo iterativo devido ao overhead causado pela chamada recursiva de função. Além disso, a recursão pode causar problemas de stack overflow para valores muito grandes de n. De qualquer forma, tanto o algoritmo iterativo quanto o recursivo do fatorial têm a mesma ordem de complexidade de tempo e são geralmente adequados para a maioria das situações.

Ademais, agora iremos analisar duas novas tabelas, que trazem valores relacionados ao tempo de usuário (user) e de sistema (sys) para as mesmas funções já citadas.

Primeiramente é válido ressaltar que, o tempo de usuário é o tempo que o processo passa efetivamente executando o seu próprio código, enquanto o tempo de sistema é o tempo que o sistema operacional passa executando as operações do processo em nome dele. Juntos, eles compõem o tempo total de CPU utilizado pelo processo para executar suas operações.

	10	**35**	**40**	**45**
Fibonacci Recursivo	user 0m0,002s sys 0m0,002s	user 0m0,072s sys 0m0,002s	user 0m0,735s sys 0m0,004s	user 0m8,136s sys 0m0,005s
Fibonacci Iterativo	user 0m0,003s sys 0m0,001s	user 0m0,002s sys 0m0,003s	user 0m0,003s sys 0m0,001s	user 0m0,003s sys 0m0,002s

Para os valores de Fibonacci percebemos valores semelhantes ao tempo de relógio na função recursiva com um tempo considerável tomado pelo usuário. Distintivamente, para função iterativa, por se tratar de valores muitos pequenos os dados não são tão conclusivos.

	1	**8**	**12**	**20**
Fatorial Recursivo	user 0m0,003s sys 0m0,001s	user 0m0,002s sys 0m0,003s	user 0m0,002s sys 0m0,003s	user 0m0,002s sys 0m0,002s
Fatorial Iterativo	user 0m0,001s sys 0m0,002s	user 0m0,002s sys 0m0,003s	user 0m0,001s sys 0m0,003s	user 0m0,003s sys 0m0,001s

Nos valores para funções fatoriais temos o mesmo problema, por não possuir valores mais altos fica difícil tirar qualquer conclusão. Mas é nítida uma distribuição entre o tempo de usuário e de sistema.

A partir daqui incluiremos no relatório dados relativos ao desempenho das funções perante a ferramenta GPROF. O GNU profiler (gprof), faz parte do conjunto de ferramentas GNU Binary Utilities (binutils), e tem como principal funcionalidade apontar quanto tempo está sendo gasto em cada parte do seu programa, assim como uma série de estatísticas respectivas a chamadas de função no código.

Fazendo uma análise com dados fornecidos pelo gprof percebe-se a quantidade de chamadas da função recursiva e que outros processos também podem consumir o tempo total do desempenho.

	40	**45**
Fibonacci Recursivo GPROF	100% do total 0.74s para função	99.73% do total 8.18s para função recursiva
Fibonacci Recursivo tempo de relógio	0,737 segundos	8,841 segundos

Nesse caso abaixo, percebe-se que o gprof, pela rápida resposta da função, não computa nenhum dado concreto para que possamos analisar, uma vez que o tempo ocorre na casa dos nanossegundos.

	8	**20**
Fatorial Recursivo GPROF	0.00 segundos	0.00 segundos
Fatorial Recursivo tempo de relógio	0,000036807s	0,000069632s

Caso decidíssemos acrescentar uma outra função que consuma recursos computacionais dentro de nossas funções recursivas já construídas, percebemos um aumento significativo no tempo equivalente à complexidade da função acrescentada. Isso permite que a análise saia da casa dos nanossegundos e traga uma diferença mais visível em desempenho.

%	**cumulative seconds**	**self**	**calls**	**Ts/call**	**Ts/call**	**name**
63.84	7.79	7.79				sin(i)
24.32	10.77	2.97				fibonacci_recursivo(int)
7.90	11.73	0.96				fatorial_recursivo(int)
3.93	12.21	0.48				_init

Na tabela acima eu rodei a função fibonacci_recursivo(20) com um acréscimo de um sin que será contabilizado um milhão de vezes. Fato que acarretou em um tempo de resposta muito diferente do normal.

%	cumulative seconds	self	calls	Ts/call	Ts/call	name
72.12	1.50	1.50				sin(i)
22.12	1.96	0.46				fatorial_recursivo(int)
5.77	2.08	0.12				_init

Já nessa última tabela eu acrescento a função recursiva do fatorial o cálculo do seno cem milhões de vezes que trouxe essa enorme diferença no tempo.