INF 213 - Roteiro da Aula Prática

Arquivos disponibilizados:
https://drive.google.com/file/d/1HCxSlNZLTLq44WMV2FWxgr3yi14eJXUb/view?usp=sharing

Etapa 1

a) Considere o programa “adivinharComplexidade.cpp”. Compile-o (sem usar a flag -O3) e teste-o usando a sintaxe “./a.out N” (onde N é o tamanho da entrada).

Qual a complexidade do programa “adivinharComplexidade.cpp” ? Faça testes com N= 5,6,7,8,....13 e tente adivinhar a complexidade dele (não tente entendê-lo ou olhar na internet o que a função next_permutation faz).
R: A complexidade é O(n!)

b) Considerando o programa “adivinharComplexidade2.cpp”, Compile-o (sem usar a flag -O3) e teste-o usando a sintaxe “./a.out N” (onde N é o tamanho da entrada).

Teste o programa com vários valores de N e analise o código.

Por que as funções “dfjkhbjknbjkcjfhui” e “dfjkhbjknbjkcjfhui2”, apesar de muito parecidas se comportam de forma tão diferente em relação ao tempo de execução?

R: Pode-se dizer que, a partir do momento que a função find executa com complexidade N, e que está dentro de um laço de repetição que também é N, temos que a complexidade de “dfjkhbjknbjkcjfhui2” é $O(N^2)$. Já no caso de “dfjkhbjknbjkcjfhui”, temos um laço de repetição que roda em N, e dentro uma operação de log que roda em 1, logo o resultado é O(N).

Qual complexidade a função “find” (da STL do C++) parece ter? (descubra isso apenas medindo os tempos de execução)

R: $O(N^2)$ - através do teste para n = 100.000, o tempo de execução da funçao2 foi de 25 segundos, já para o caso de n = 200.000, o tempo foi de 106 segundos, aproximadamente 4x maior.

Qual complexidade a função “log” (do C++) parece ter? Descubra isso apenas medindo os tempos de execução -- dica: teste com números muito maiores (exemplo: 500 milhões, 1 bilhão, 2 bilhões) e apague a chamada à segunda função para conseguir fazer essa medição apenas da primeira (caso contrário não dará tempo do programa terminar antes do deadline desta prática, que é ainda neste século).
R: O(N) - para a entrada de n = 500 milhões o tempo foi de 8.7955560. Já para o dobro, n = 1 bilhão, o tempo foi 17.5538510, o dobro, o que mostra o crescimento linear.

Etapa 2

Faça a análise de complexidade das funções presentes no arquivo analise1.cpp (tais funções podem nem compilar -- estamos interessados apenas na complexidade dos algoritmos).

Escreva suas respostas como comentários no topo das respectivas funções (veja o exemplo na primeira função de analise1.cpp). Lembre-se de sempre usar a notação “O” e simplificar ao máximo a resposta final (ou seja, em vez de O(3n^4 + n^3) a resposta deverá ser algo como O(n^4)).

Considere sempre o pior caso de cada função. (a não ser que dito o contrário nos comentários) Preste bastante atenção a todas funções!

Lembrem-se sempre de pedir ajuda ao professor se necessário (não fiquem em dúvida sobre a complexidade de alguma função).

Submissao da aula pratica:

A solucao deve ser submetida ate as 18 horas da proxima Segunda-Feira utilizando o sistema submitty (submitty.dpi.ufv.br). Envie analise1.cpp pelo submitty. Envie também um PDF deste documento após terminar as respostas da Etapa 1 (o nome do arquivo deverá ser roteiro.pdf).