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).

Como o tempo de processamento do codigo aumentou de forma muito rapida, apesar de um aumento pequeno no tamanho de entrada, pode-se perceber a ordem de complexidade do código é O(N!) (Fatorial). Uma vez que, um numero pequeno como 13 o foi gasto cerca de 124 segundos executados.

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?

Elas se comportam de forma tão diferente em relação ao tempo, pois a função 1 possui ordem de complexidade O(N), enquanto a função 2 possui ordem de complexidade O(Nˆ2). Isso ocorre, uma vez que a função 2 possui uma função find, que vai percorrer o array de tamanho N tentando achar o número desejado, senda ela executada N vezes. Já a função 1 executa uma operação log N vezes.

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

A função "find" da STL do C++ parece ter complexidade O(N), já que o tempo de execução de 250 é 4 vezes menor menor que, sendo esse 4 vezes menor que 1000, ou seja, seria uma ordem Nˆ2, porém o operador for possui complexidade O(N).

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).

A complexidade da função "log" do C++ parece ter ordem O(1) , já que mesmo aumentando o tamanho da entrada em bilhões o tempo gasto para executar não é aumentado de forma drástica, ou seja, sua complexidade é constante.

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 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).