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 dele é O(n!)

n	tempo
5	0s
10	0.1420790s
11	1.6119530s
12	19.1545020s
13	263.6584300s

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: Porque o custo das funções log e find são diferentes.

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

R: O(n). Porque quando n = 10000 o tempo de execução é 0.2419300s e quando n = 20000(o dobro de 20000) o tempo de execução é 0.9689620s, ou seja, quando a entrada dobra de tamanho o tempo de execução quadruplica, com isso a função “dfjkhbjknbjkcjfhui2” tem complexidade $O(n^2)$, se é $O(n^2)$ é porque o custo de find é O(n)

visto que a complexidade da função toda é n(número de vezes que o laço repete) * custo da função find.

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(1). Quando n = 500000 o tempo de execução é 0.0020250 e quando n = 1000000(o dobro de 500000) o tempo de execução é 0.0040590, logo quando o n dobra, seu tempo de execução também dobra porque a complexidade da função “dfjkhbjknbjkcjfhui” é O(n), se é O(n) é porque o custo de log é uma constante(1).

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).