A) Através do estudo de complexidade de algoritmos, passamos a compreender melhor como é o funcionamento lógico que devemos ter ao elaborar um algoritmo. O código não passa a ser somente "funcionar" e sim entender como tudo está ligado e quais são as principais consequências que devemos medir ao tomar qualquer decisão, seja ela de adaptação ou até de risco.

Ao perceber a tamanha importância, passei a olhar com mais cuidado o desempenho do meu algoritmo não a meros quantos segundos está levando para fazer uma tarefa simples, mas o quão grande ele pode ficar de acordo com a entrada do usuário. O simples cuidado e escolher melhor quais tipos de variáveis escolher até o mais complexo cálculo de função passou a ser normal diante de qualquer projeto de algoritmo.

B) Com base nos dois textos citados pela questão 3 letra B, podemos chegar a seguinte interpretação:

Ao evoluirmos significativamente tecnologicamente a ponto suficiente de "esbanjarmos" bits a todos os lados, passamos e desconhecer a complexidade dos algoritmos. Como assim? Se traçarmos um paralelo com as máquinas de 20 anos atrás (início da década de 2000) tínhamos processadores com míseros 512k de cache, memória ram de 128Mb de cada slot e um disco rígido de 6.4 GB.

Se compararmos com hoje, chega a ser assustador, como que irei desenvolver em um computador como este? Obviamente o sistema operacional e os programas não eram nem de perto comparável com o que temos hoje, funcionalidades, requisitos de sistema etc. Porém, mesmo que levemos para programas besta que desenvolvemos hoje, na época já eram desenvolvidos. Um simples "Hello World" hoje pode parecer muito trivial, porém para época demandava um esforço significativo para pode compilar. Ao elevarmos um pouco este desafio a um universo de 100 variáveis do tipo Double, 10 Strings de 100 caracteres, por exemplo, em Java podemos chegar a incríveis 800 bytes (100 x 8 bytes) só com variável do tipo Double, e isso fica ainda mais assustador quando colocarmos as strings. Uma string de 100 caracteres chegam a consumir incríveis 16 bits por caractere, logo, 16 x 100 = 1600 bits por String. Multipliquemos por 10 e chegamos a 16000 bits, o que dá 4000 bytes. Ao total, temos 4000 bytes + 800 bytes = 4800 bytes. Pode parecer pouca coisa, atualmente é. Mas levando em conta a quantidade de recursos no início de 2001, teríamos que ter a certeza que precisamos de tudo isto.

Então, podemos chegar a conclusão que atualmente, a complexidade dos algoritmos é ofuscada pelo desperdício de processamento, memória, recursos de forma geral. Não fazendo assim, com que haja uma real necessidade A PRIORI do conhecimento especializado.

fontes:

1 - https://itstillworks.com/specifications-gateway-2000-computer-7699939.html

2 - https://www.devmedia.com.br/tipos-primitivos-e-variaveis-em-java/3149

C) A qualificação de um algoritmo pode variar de acordo com ambiente. Em geral, o tempo de execução não depende somente de um algoritmo, mas de um conjunto de fatores, tais como: (1) compilador; (2) conjunto de instruções do computador; (3) algoritmo desenvolvido pelo programador;

Sendo assim, devemos nos atentar não somente ao tempo de execução mas principalmente ao número de operações primitivas que são executadas. Portanto, as medidas de complexidades são utilizadas. Esforço requerido ou quantidade de trabalho.

A complexidade no pior caso considera a instância que faz o algoritmo funcionar mais lentamente; outra é a complexidade média, onde se considera todas as possíveis instância e é medido o tempo médio.

Com isto, chegamos ao ponto no qual a complexidade é calculada pelo tempo de execução do algoritmo pelas instruções executadas, tendo uma comparação entre tempo x tamanho do problema 'X'. E o espaço de memória utilizado pelo algoritmo, que seria o quanto de espaço é utilizado, seja em disco ou memória, como foi exemplificado na questão B).

D) O primeiro item a se destacar é que os algoritmos são soluções para problemas diferentes. O algoritmo de Gauss é usado para solucionar problemas não lineares, é usado para resolver problemas de mínimos quadrados não lineares. Ele pode ser visto como uma modificação do Método de Newton para achar o mínimo de uma função.

Já o método de Cramer é utilizado para resolver problemas linear, mais especificamente sistemas lineares no qual o número de equações e icônicas forem iguais.

Portanto temos dois tipos de algoritmos que servem para suas determinadas funções.

Enquanto que o método de Cramer é muito eficiente executando sistemas lineares, o método de Gauss é eficiente ao resolver o determinante de uma matriz 40x40m, por exemplo, chegando a levar menos de um segundo enquanto o método de Cramer levaria 5.10^33 séculos.