INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
MINAS GERAIS
Campus Bambuí

Instituto Federal de Minas Gerais
Campus Bambuí
Departamento de Engenharia e Computação
Engenharia de Computação

Atividade: Melhoramento de Tempo
Nome: Gabriel Henrique Silva Duque **R.A:** 0082574

Questão 1) O algoritmo original compara cada número do vetor com todos os outros, usando dois for aninhados. Isso significa que ele faz aproximadamente $n^2/2$ comparações. No pior caso, quando o par nunca é encontrado, ele percorre todos os pares possíveis. Isso dá uma complexidade de $O(n^2)$.

No melhor caso, se o par estiver logo no início, ele resolve em apenas uma comparação. Então temos $\Omega(1)$. Como esse comportamento geral se mantém proporcional a n^2, mesmo com variações pequenas no início ou fim, dizemos que a complexidade assintótica exata é $\Theta(n^2)$.

Ou seja:

$O(n^2)$ → **pior caso**

$\Omega(1)$ → **melhor caso**

$\Theta(n^2)$ → **caso médio (crescimento geral do algoritmo)**

Questão 2) Para deixar o algoritmo mais rápido no pior caso, a ideia é usar uma estratégia com dois ponteiros após ordenar o vetor.

Como funciona:

I. Etapa 1: Ordenar o vetor → isso leva $O(n \log n)$ se usarmos um bom algoritmo como Merge Sort.

II. Etapa 2: Usar dois ponteiros (um no início e outro no fim) e somar os valores apontados. Dependendo da soma, move-se um ponteiro para tentar ajustar o valor. Essa busca leva $O(n)$.

Somando tudo:

Pior caso: $O(n \log n)$ para ordenar + $O(n)$ para buscar → dá $O(n \log n)$

Melhor caso: $O(1)$ se o par for encontrado logo no início

Comportamento geral: O tempo total é dominado pela ordenação, então temos $\Theta(n \log n)$

Melhor caso teórico: $\Omega(1)$, se der sorte e encontrar de primeira

Questão 3) As duas versões foram implementadas em R e testadas com as seguintes condições:

I. 30 execuções cada

II. Vetor com 100.000 elementos positivos aleatórios entre 1 e 1000

III. Par alvo: k = 10

A versão original foi testada com vetores reduzidos para 10.000 elementos, pois o custo era muito alto. Foi medida a média dos tempos em segundos para cada versão

Resultados observados:

O algoritmo original demorou bem mais, mesmo com vetor menor, confirmando que seu tempo cresce rápido (quadraticamente).

O algoritmo melhorado rodou rápido mesmo com vetor grande, provando que sua complexidade cresce mais devagar (quase linear).

A notação O mostra como o tempo cresce com o tamanho da entrada (n), mas não quantos segundos exatos vai levar. A prática (tempo em segundos) confirmou o que a teoria dizia:

I. O algoritmo original ($O(n^2)$) teve crescimento explosivo de tempo.

II. O algoritmo otimizado ($O(n \log n)$) teve desempenho muito melhor.

Simulação no Rstudio: