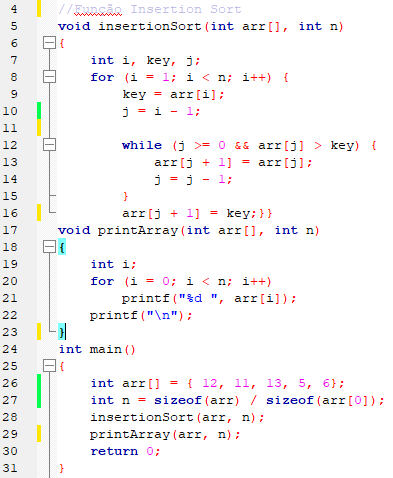
#### Atividade 2 - [Análise da Complexidade do Algoritmo Insertion Sort](https://imt.mrooms.net/pluginfile.php/24291/mod_assign/intro/TAED_Atividade_02.pdf?time=1551299223098)

João Guilherme Martins Jatobá R.A.: 18.01790-8

ECM306 - Tópicos Avançados em Estrutura de Dados

Só irei analisar a função void - Insertion Sort.

|  |  |  |
| --- | --- | --- |
|  | **CÓDIGO** | **TEMPO (OPERAÇÕES)** |
| //1a | int i = 1; | 2 |
| //1b | i < n; | 3\*n |
| //1c | i++ | 4\*(n-1) |
| //1 | for (int i = 1; i< n; i++) | 7\*n - 2 |
| //2 | key = arr[i]; | 5\*(n-1) |
| //3 | j = i-1; | 4\*(n-1) |
| //4 | while ( j >=0 && arr[j] > key) | 10\*(n-1) |
| //5 | arr [j+1] = arr [j]; | 5\*(n-1) |
| //6 | j = j-1; | 4\*(n-1) |
| //7 | arr [j+1] = key; | 2\*(n-1) |
|  | **TOTAL** | 342 n2 - 608n + 266 |

O exercício pede para que se analise a ordem de complexidade do algoritmo na pior das hipóteses, ou seja, todos os números estão em ordem decrescente, o que vai contra o código totalmente, já que este coloca os números da lista em ordem crescente.

O loop *while* está aninhado dentro do loop *for*, logo o processo *while* será executado (se não o processo, a comparação dos parâmetros acontecerá) todas as vezes que o loop *for* for ativado. Isto resulta num elevado número de interações entre os dois loops, por isso os coeficientes da equação final da função *Insertion Sort* ficaram tão elevados. Outra conclusão que se atinge, é que o algoritmo em questão é de complexidade quadrática – **O(n2)**, como se vê na equação **Total** final.