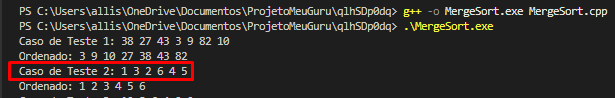
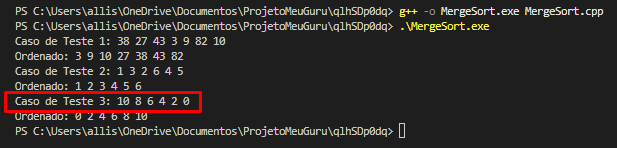
Conjunto 01 - O MergeSort produzido no arquivo MerSort foi implementado em C++ e aqui seus respectivos casos de teste e análise de custo.

**Casos de Teste**

1. Entrada Desordenada
   1. Entrada: {38, 27, 43, 3, 9, 82, 10}
   2. Texto

      Descrição gerada automaticamenteSaída: {3, 9, 10, 27, 38, 43, 82}
2. Entrada com Elementos Próximos e Aleatórios:
   1. Entrada: {1, 3, 2, 6, 4, 5}
   2. Saída: {1, 2, 3, 4, 5, 6}
3. Entrada Totalmente Invertida:
   1. Entrada: {10, 8, 6, 4, 2, 0}
   2. Saída: {0, 2, 4, 6, 8, 10}

**Análise de Custo**

Divisão e Conquista

* Divisão: A cada nível de recursão, o vetor é dividido em dois, resultando em uma complexidade devido ao número de divisões necessárias.
* Conquista: A operação de mesclagem é linear para cada nível da recursão

Complexidade Total

A complexidade de tempo do MergeSort é:

**Operação Básica**

No caso do MergeSort, a operação básica considerada é a comparação entre elementos durante a fase de mesclagem (merge). Essa é a operação mais realizada e diretamente correlacionada ao desempenho do algoritmo.

**Expressão Completa do Custo**

Expandindo a recorrência para observar o comportamento:

1. Primeiro nível (n elementos):

cn

1. Segundo nível (n/2 + n/2):

2c⋅n/2 = cn

1. Terceiro nível (n/4 + n/4 + n/4 + n/4):

4c⋅ n/4 = cn

1. Continua até log2(n) níveis, cada nível custando cn.

Somando todos os níveis:

T(n) = cn + cn + cn+⋯+cn (log2 (n) vezes)

T(n) = cn⋅log2 (n)

**Classe de Problema no Contexto da Complexidade Computacional**

MergeSort resolve um problema da classe P (Polynomial Time), que abrange problemas que podem ser resolvidos por um algoritmo determinístico em tempo polinomial:

* A complexidade de O (n log n) está dentro da classe P, pois n log n cresce mais lentamente que qualquer função exponencial.