**SCC0220 - Laboratório Introdução à Ciência da Computação II**

**Relatório - Aula 7**

**Alunos NUSP**

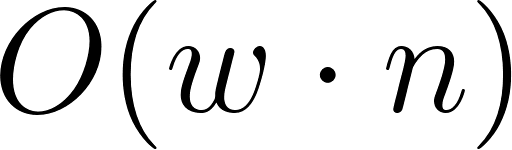
Pedro Henrique de Sousa Prestes 15507819

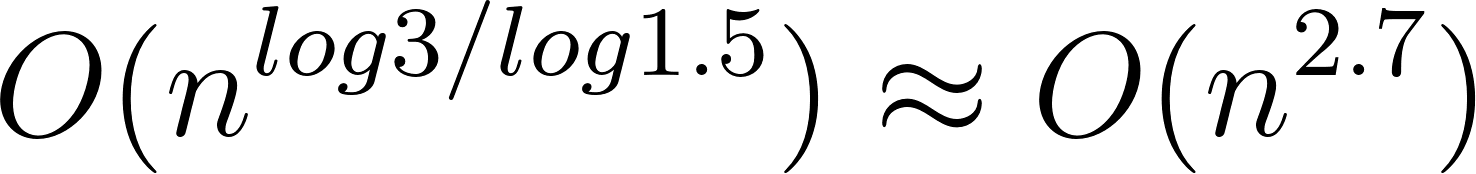
Pedro Lunkes Villela 15484287

**Trabalho 7 - De volta às raízes**

**Radix Sort x Stooge Sort**

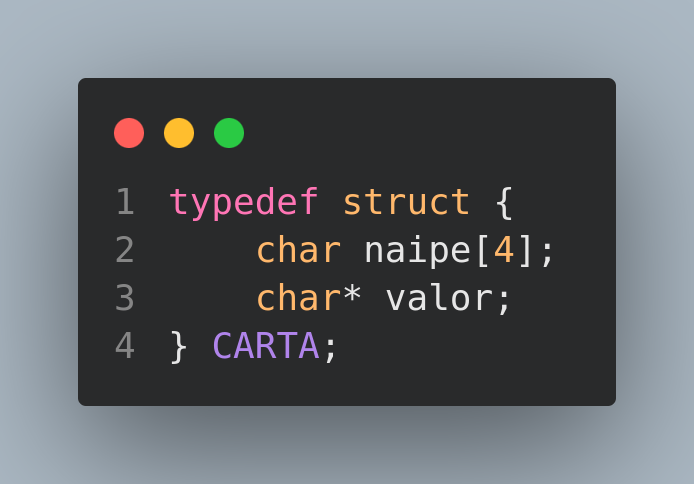
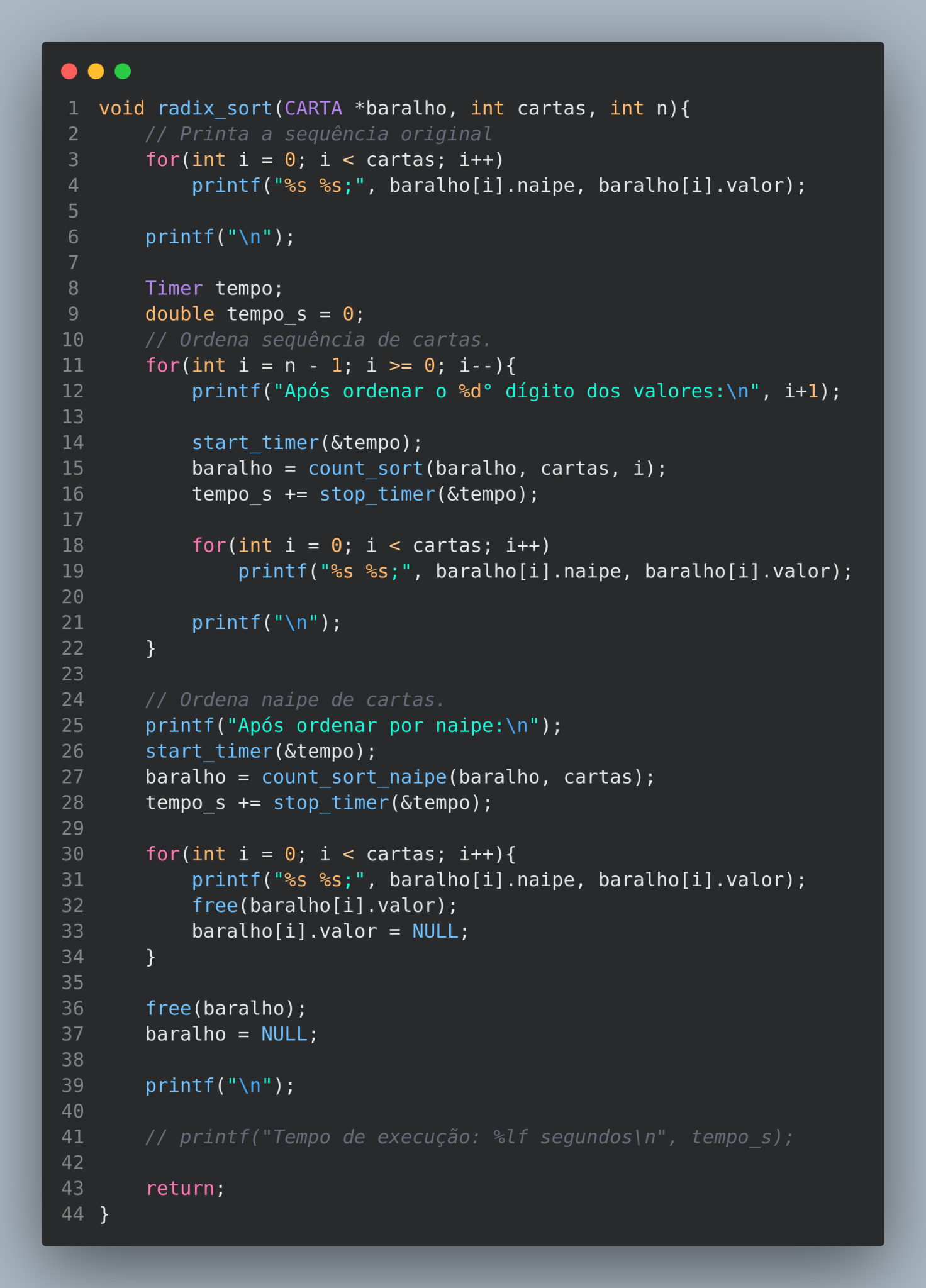
**🡺 Comentário**

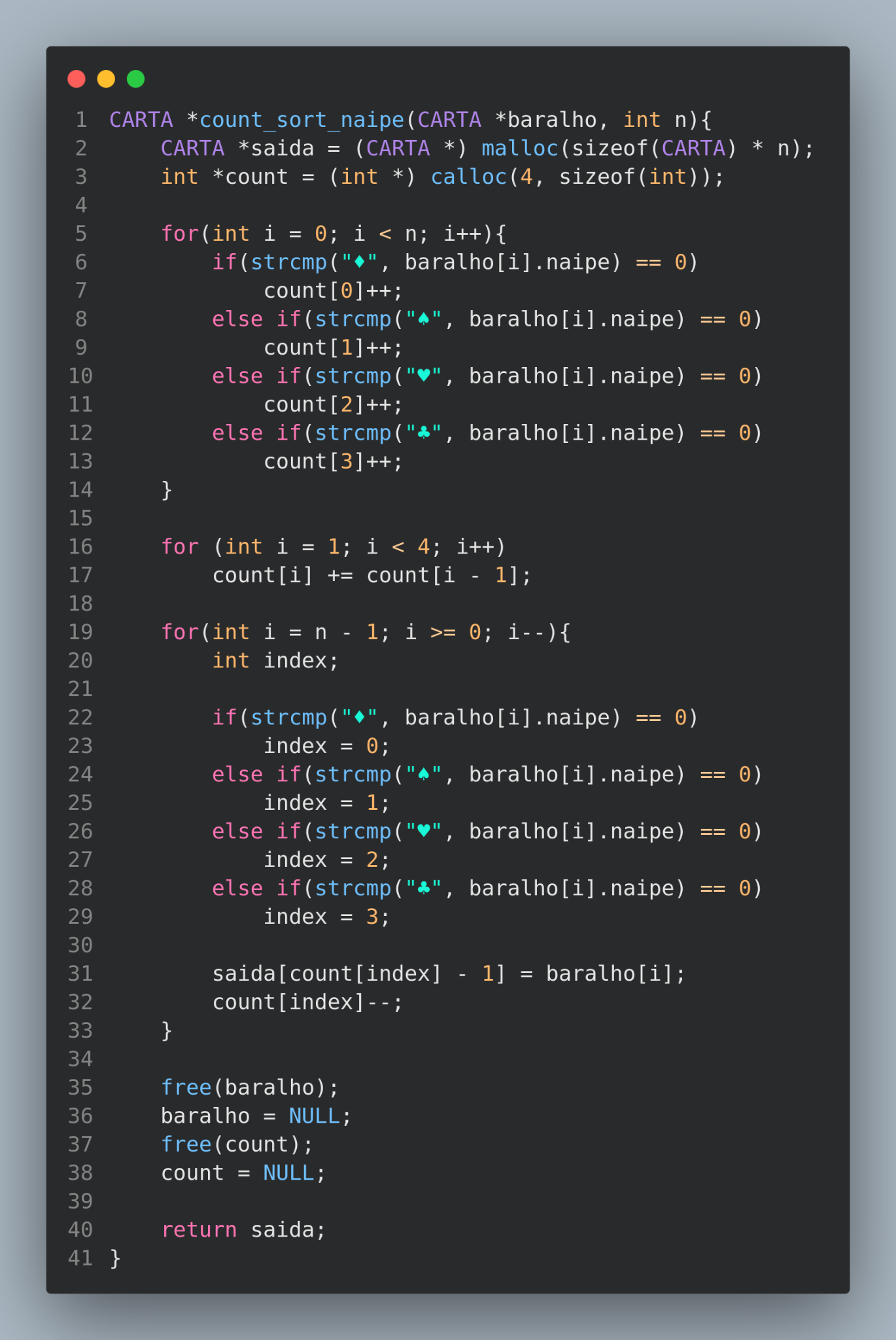
❖ **Radix Sort:** Organiza números inteiros ou strings processando um dígito ou caractere por vez, da posição menos significativa para a mais significativa. Ele usa o Counting Sort para ordenar os elementos com base em cada dígito/caractere, repetindo o processo até que todos os dígitos tenham sido processados. Sua complexidade é de [](https://www.codecogs.com/eqnedit.php?latex=O(w%20%5Ccdot%20n)#0), com [](https://www.codecogs.com/eqnedit.php?latex=w#0) sendo o número de dígitos dos [](https://www.codecogs.com/eqnedit.php?latex=n#0) elementos.

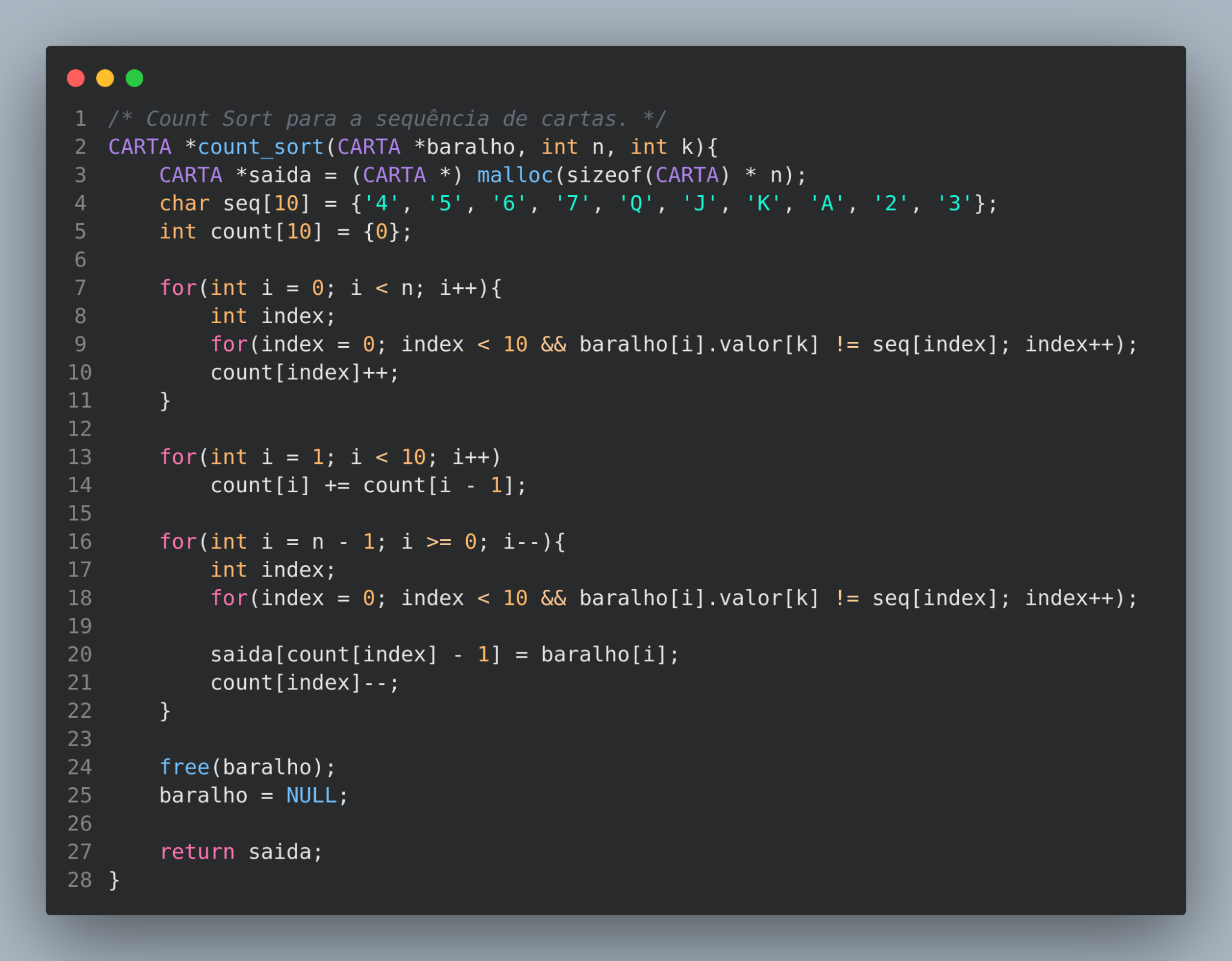
❖ **Stooge Sort:** Divide a sequência original de elementos em terços utilizando o método de divisão e conquista, comparando o início e fim de cada partição e trocando-os caso necessário e depois ordena os dois primeiros terços, os dois últimos terços e novamente os dois primeiros terços da sequência. Sua complexidade é [](https://www.codecogs.com/eqnedit.php?latex=O(n%5E%7Blog%203%20%2F%20log%201.5%7D)%20%5Capprox%20O(n%5E%7B2.7%7D)#0).

Para comparar os dois sorts, foi utilizado um sistema de ordenação de cartas de baralho na ordem do truco, incluindo os naipes únicos para cada sequência de cartas.

* Cada carta representa um dígito e o naipe representa o dígito mais significativo.
* O Radix Sort imprime a sequência para cada dígito ordenado, alterações foram feitas na forma que o tempo de execução é processado para evitar incluir o tempo do printf.

**🡺 Código**









**🡺 Saída**

Seguem alguns casos testados comparando o desempenho de ambos algoritmos.

**3° Caso**

Tempo de execução Radix Sort: 0.000136 segundos

Tempo de execução Stooge Sort: 5.292695 segundos

Tempo de execução Radix Sort: 0.000525 segundos

Tempo de execução Stooge Sort: 5.254897 segundos

Tempo de execução Radix Sort: 0.000487 segundos

Tempo de execução Stooge Sort: 5.374034 segundos

**5° Caso**

Tempo de execução Radix Sort: 0.001694 segundos

Tempo de execução Stooge Sort: 1.748199 segundos

Tempo de execução Radix Sort: 0.001791 segundos

Tempo de execução Stooge Sort: 1.698001 segundos

Tempo de execução Radix Sort: 0.003977 segundos

Tempo de execução Stooge Sort: 1.796248 segundos

**6° Caso**

Tempo de execução Radix Sort: 0.004997 segundos

Tempo de execução Stooge Sort: 5166.701824 segundos

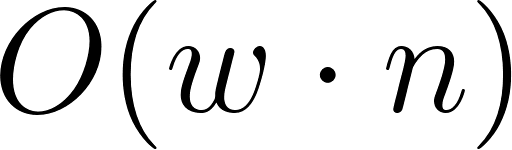
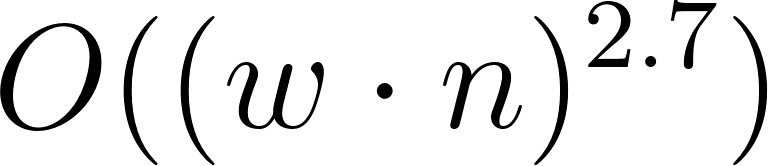
**Obs:** Devido ao tempo elevado do Stooge Sort, foi feito um único teste.

**7° Caso**

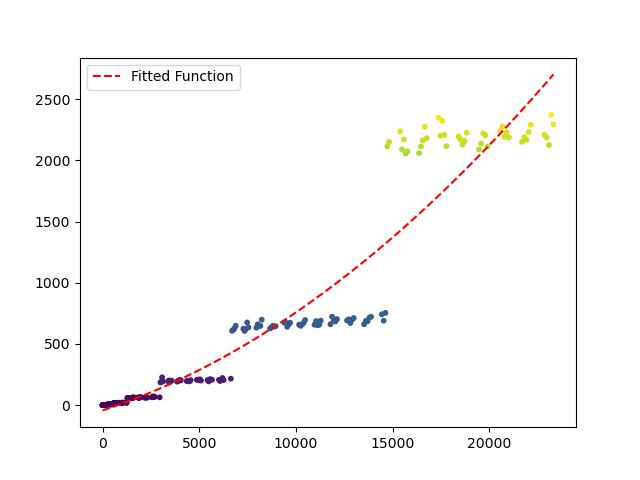
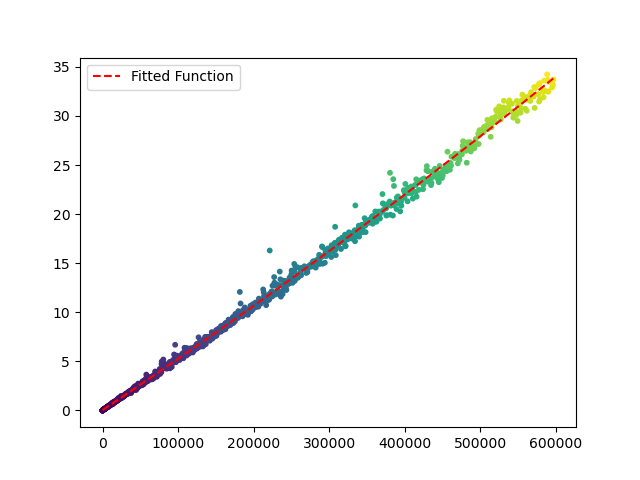
Tempo de execução Radix Sort: 0.059395 segundos

**Obs:** O Stooge Sort foi mantido rodando o caso teste por mais de 3 horas sem êxito.

**🡺 Conclusão**

Como esperado, o Radix Sort mostrou-se mais eficaz que o ineficiente Stooge Sort, já que, por conta das comparações que dependem do número de dígitos, ele consegue estabelecer uma complexidade linear de [](https://www.codecogs.com/eqnedit.php?latex=O(w%20%5Ccdot%20n)#0) enquanto o Stooge Sort, por ter que fazer a comparação de todos os dígitos, acaba com uma complexidade [](https://www.codecogs.com/eqnedit.php?latex=O((w%20%5Ccdot%20n)%5E%7B2.7%7D)#0). Assim, o Radix Sort é mais efetivo que qualquer outro método de ordenação para esse problema, já que ele naturalmente compara os dígitos das cartas enquanto outros sorts teriam que aplicar o seu próprio sistema de ordenação enquanto comparam os dígitos.

O gráfico mostrou resultados inusitados no Stooge Sort, já que o gerador de casa teste utilizado aumentava de 3 em 3 linhas e uma carta a cada 20 casos, causando esse comportamento estranho.



(Gráfico Radix e Stooge, Eixo X representa o número de entradas [número de cartas por mão \* número de mãos] e o Eixo Y milissegundos, feito a partir de casos teste a parte do Runcodes)