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

**Relatório - Aula 5**

**Alunos NUSP**

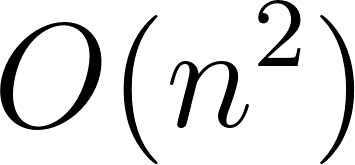
Pedro Henrique de Sousa Prestes 15507819

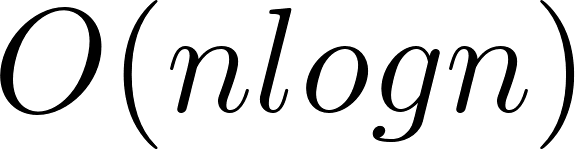
Pedro Lunkes Villela 15484287

**Trabalho 5 - Bubble and Quick**

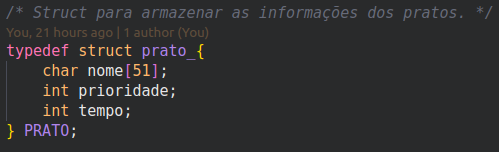
**Bubble Sort x Quick Sort**

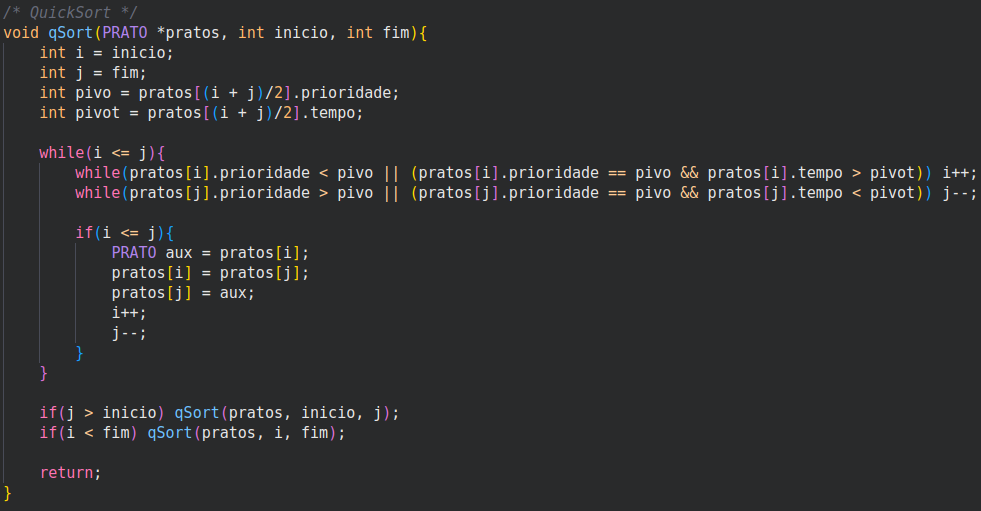
**🡺 Comentário**

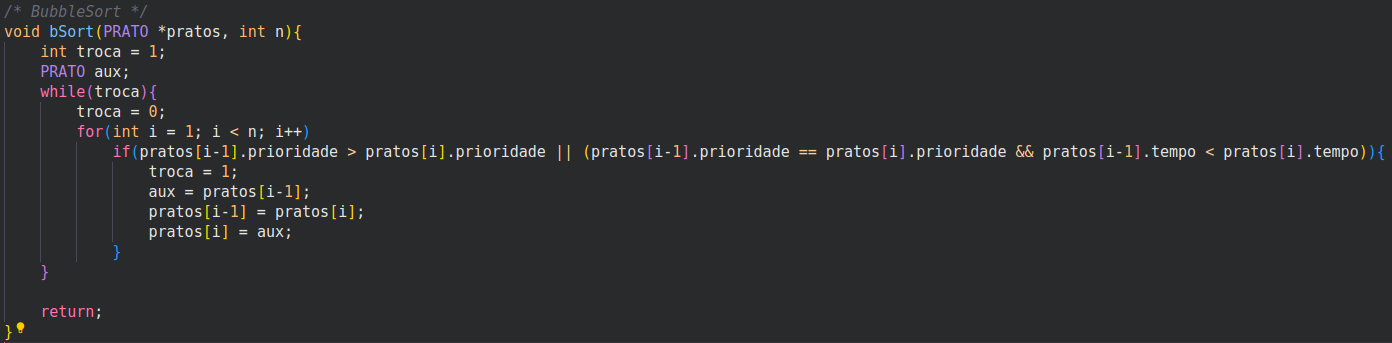
❖ **Bubble Sort:** Seu funcionamento é simples, consiste em comparar, a cada iteração, um elemento do vetor com seu sucessor e trocá-los se necessário. Sua complexidade média é [](https://www.codecogs.com/eqnedit.php?latex=O(n%5E%7B2%7D)#0);

❖ **Quick Sort:** Consiste em uma “melhoria do BubbleSort” que faz comparações e trocas pelo vetor em volta do valor de uma posição pré-determinada chamada pivô, é um algoritmo do tipo “dividir para conquistar”, pois divide o vetor e ordena suas partes de forma independente para produzir o resultado final. Sua complexidade média é [](https://www.codecogs.com/eqnedit.php?latex=O(n%20log%20n)#0) com pivô central.

Para comparar os dois sorts, foram utilizados casos de pratos de comida em um restaurante que devem ser ordenados a partir de seus valores de prioridade, caso dois pratos tenham prioridades iguais, o tempo de preparo deverá ser considerado como critério para ordenação.

**🡺 Código**

****

****

**🡺 Saída**

Seguem alguns casos testados por nós comparando o desempenho de ambos algoritmos.

**1° Caso**

Tempo de execução BubbleSort: 0.000003 segundos

Tempo de execução QuickSort: 0.000002 segundos

Tempo de execução BubbleSort: 0.000003 segundos

Tempo de execução QuickSort: 0.000002 segundos

Tempo de execução BubbleSort: 0.000002 segundos

Tempo de execução QuickSort: 0.000002 segundos

**3° Caso**

Tempo de execução BubbleSort: 0.002434 segundos

Tempo de execução QuickSort: 0.000040 segundos

Tempo de execução BubbleSort: 0.001080 segundos

Tempo de execução QuickSort: 0.000039 segundos

Tempo de execução BubbleSort: 0.001069 segundos

Tempo de execução QuickSort: 0.000039 segundos

**9° Caso**

Tempo de execução BubbleSort: 0.001056 segundos

Tempo de execução QuickSort: 0.008021 segundos

Tempo de execução BubbleSort: 0.001128 segundos

Tempo de execução QuickSort: 0.008488 segundos

Tempo de execução BubbleSort: 0.001403 segundos

Tempo de execução QuickSort: 0.015640 segundos

**10° Caso**

Tempo de execução BubbleSort: 307.033382 segundos

Tempo de execução QuickSort: 0.008341 segundos

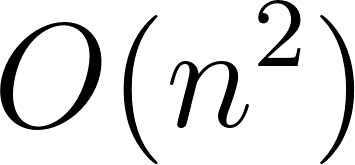
Tempo de execução BubbleSort: 304.393536 segundos

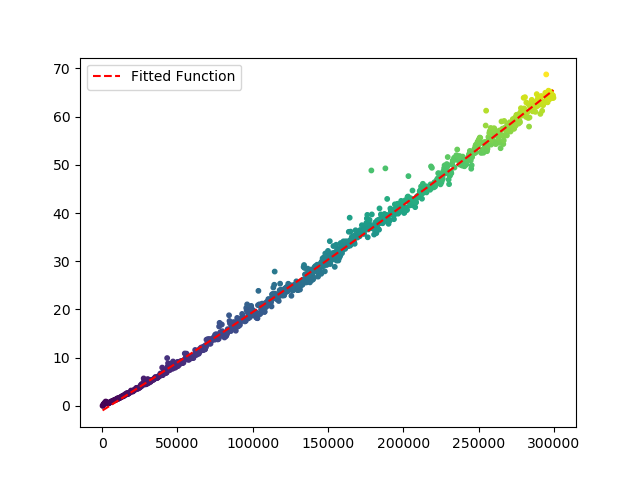
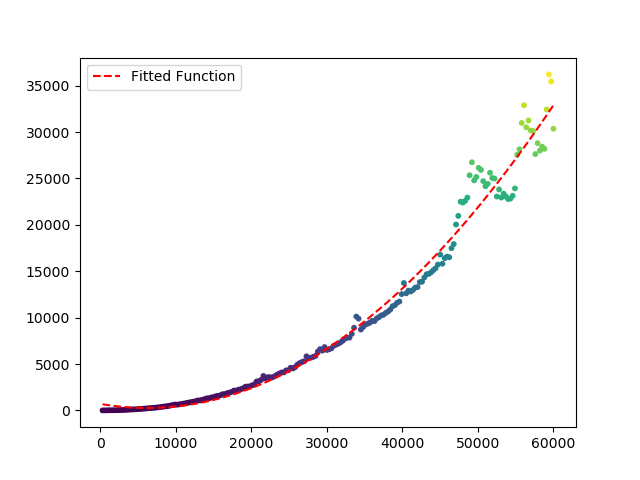
Tempo de execução QuickSort: 0.008396 segundos

Tempo de execução BubbleSort: 295.862082 segundos

Tempo de execução QuickSort: 0.008358 segundos

**🡺 Conclusão**

O algoritmo de Quick Sort mostrou-se muito eficaz em comparação ao Bubble Sort, com exceção para o caso de listas já ordenadas. A escolha do pivô central pode não ser a mais eficaz, porém reduz a chance de ocorrer uma complexidade de [](https://www.codecogs.com/eqnedit.php?latex=O(n%5E%7B2%7D)#0) no pior caso, principalmente em relação a aplicações do Quick Sort em que o pivô escolhido está nas extremidades.



(Gráfico Bubble e Quick, Eixo X representa o número de entradas e o Eixo Y milissegundos, feito a partir de casos teste a parte do Runcodes)