## Autores

Erick Jonathan Alves Viana da Silva

Henry Teruo Kimura

## Código em Python

```
def partition(arr, low, high): #declara uma função chamada partition que aceita três parâmetros: arr (uma lista), low (o índice mais baixo do subarray) e high (o índice mais alto do subarray). Essa função é responsável por particionar a lista arr em torno de um pivô selecionado.
    pivot = arr[high]  # Escolhe o último elemento como pivô
    i = low - 1  # Inicializa o índice do menor elemento
    for j in range(low, high):  # Itera sobre os elementos do subarray
        if arr[j] <= pivot:  # Se o elemento atual for menor ou igual ao pivô
            i += 1  # Incrementa o índice do menor elemento
            arr[i], arr[j] = arr[j], arr[i]  # Troca os elementos arr[i] e arr[j]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]  # Troca arr[i + 1] e arr[high], colocando o pivô na posição correta
    return i + 1  # Retorna o índice do pivô após a partição


def quick_sort(arr, low, high): #Esta linha declara uma função chamada quick_sort que aceita três parâmetros: arr (uma lista), low (o índice mais baixo do subarray) e high (o índice mais alto do subarray). Essa função é responsável por ordenar a lista arr usando o algoritmo de ordenação rápida (quicksort).
    if low < high:  # Verifica se ainda há elementos para ordenar
        pi = partition(arr, low, high)  # Obtém o índice do pivô após a partição
        quick_sort(arr, low, pi - 1)  # Ordena o subarray antes do pivô
        quick_sort(arr, pi + 1, high)  # Ordena o subarray depois do pivô


# Input dos números
arr = input("Digite os números separados por espaço: ").split()  # Recebe uma lista de números como string e a divide em uma lista de strings
arr = [int(num) for num in arr]  # Converte os elementos da lista para inteiros


n = len(arr)  # Obtém o tamanho da lista
quick_sort(arr, 0, n - 1)  # Chama a função quick_sort para ordenar a lista
print("Lista ordenada:")
for i in range(n):  # Itera sobre a lista ordenada
```

```
print("%d" % arr[i])  # Imprime cada elemento da lista ordenada
```

## Explicando o Quick Sort

**Escolha do Pivô:** O algoritmo seleciona um elemento da lista, chamado de pivô. A escolha do pivô pode afetar o desempenho do algoritmo. Uma escolha comum é selecionar o último elemento da lista, mas outras estratégias também são possíveis.

**Particionamento:** Os elementos da lista são rearranjados de forma que todos os elementos menores que o pivô fique antes dele, e todos os elementos maiores fiquem depois dele. Após essa operação, o pivô está em sua posição final na lista ordenada e é chamado de ponto de divisão.

**Recursão:** O algoritmo é aplicado recursivamente às sublistas geradas pelo processo de particionamento. Ou seja, aplica-se o Quick Sort para as sublistas à esquerda e à direita do pivô (elementos menores e maiores que o pivô, respectivamente).

**Combinação:** Como as sublistas são ordenadas recursivamente, quando o processo recursivo termina para todas as sublistas, a lista inteira estará ordenada.

A chave para a eficiência do Quick Sort está no passo de particionamento, onde a lista é dividida de forma que elementos menores que o pivô fique à esquerda e elementos maiores fiquem à direita. Isso permite que o algoritmo seja aplicado de forma recursiva em sublistas menores, reduzindo gradualmente o problema original em subproblemas menores e mais simples.