Trabalho 01

- Neste trabalho utilizaremos a linguagem Python na implementação de métodos de ordenação
- Faremos a comparação de um método eficiente (Quicksort) com outro pouco eficiente (Bubble Sort)

Quicksort

- Baseado em trocas a longa distância
- Procedimento
  - Particionar arquivo em porções desordenadas, cada vez menores
  - Cada pivô da partição está na posição correta
  - Recursivamente, o arquivo está ordenado

Quicksort

- Passos do algoritmo
  - Tomar aleatoriamente uma chave X
  - Percorrer o arquivo da esquerda para a direita até achar alguma chave maior ou igual à X
  - Percorrer o arquivo da direita para a esquerda até achar alguma chave menor ou igual à X
  - Trocar os elementos
  - Repetir até que haja cruzamento de percurso
  - Aplicar o mesmo procedimento à esquerda e à direita de X

Quicksort

Aplicar o mesmo procedimento às duas partições

Quicksort

- Ponto chave
  - Escolha do pivô
  - Pior caso: escolher sempre o maior ou o menor valor
    - Desempenho parecido com o bubble sort
  - Melhor caso: escolher sempre o valor médio
    - Desempenho parecido com a busca binária.

Quicksort

- Algoritmo básico

```
Ordenar (vet, 0, n-1)

Ordenar (vet, i, j)
    Escolher x
    Repetir
        Enquanto vet[i] < x, i++
        Enquanto vet[j] > x, j--
        Se i <= j
            Trocar vet[i] com vet[j]
            i++; j--
    Enquanto i < j
    Ordenar porção entre o início e x
    Ordenar porção entre x e o final
```

Quicksort
em Python

```
def partition(array, start, end):
    pivot = array[start]
    low = start + 1
    high = end

    while True:
        # Se o valor corrente é maior que o pivô, ele está no lugar certo,  basta
        # mover o ponteiro high à esquerda. Senão, ele está na posição e
        # errrada e o laço deve ser interrompido
        while low <= high and array[high] >= pivot:
            high = high - 1

        # Mesmo raciocício se aplica com low
        while low <= high and array[low] <= pivot:
            low = low + 1
```

Quicksort
em Python

```
        # Nesse ponto, ou achamos dois valores que precisam ser trocados, ou
        # então low > high e não há troca a fazer
        if low <= high:
            # Executando a troca
            array[low], array[high] = array[high], array[low]
        else:
            # Saindo do loop
            break
    # Colocando o pivô na posição correta
    # (Todos à esquerda do pivô são menores e todas à direita são maiores)
    array[start], array[high] = array[high], array[start]

    return high
```

Quicksort

em Python

```
def quick_sort(array, start, end):
    if start >= end:
        return

    p = partition(array, start, end)
    quick_sort(array, start, p-1)
    quick_sort(array, p+1, end)


array = [29,99,27,41,66,28,44,78,87,19,31,76,58,88,83,97,12,21,44]
print(array)
quick_sort(array, 0, len(array) - 1)
print(array)
```

Atividades da aula

- Atividade 1
  - Implemente uma versão do programa que leia valores até que o valor 0 (zero) seja lido
  - Mostre o vetor lido e, em seguida, mostre o vetor ordenado

Atividades da aula

- Atividade 2
  - Implemente uma versão do programa que gere um vetor com 100 valores gerados randomicamente. Números gerados devem estar no intervalo de 1 a 100
  - Dica: use o módulo random

Atividade para entregar

- Implemente uma função que receba uma lista de inteiros e ordena essa lista utilizando o Bubble Sort
- Escreva um programa que gere uma lista com 10.000 números inteiros (gerados randomicamente no intervalo entre 1 e 10.000)
- Computar o tempo gasto para ordenar a lista usando o Quicksort
- Computar o tempo gasto para ordenar a lista usando o Bubble sort
- Imprimir os resultados obtidos, em milisegundos
- Dica
  - https://pt.stackoverflow.com/questions/97364/medir-o-tempo-de-execu%C3%A7%C3%A3o-de-uma-fun%C3%A7%C3%A3o

Atividade para entregar

- A entrega do trabalho deve ser feita via email para:
  - com220unifei@gmail.com

- Assunto do email: Trabalho 01
- No corpo do email você deve colocar seu nome e seu número de matrícula

A resolução deste trabalho está em :
https://github.com/joaolucas2012/Unifei/blob/master/Git%20Codes/Git%20codes/Python/Works/Work1/quickandbubble.py