**Relatório Trabalho 1 | Análise de Algoritmos | 2019.1**
**Bruno Miranda Marinho | 1613071**

### Tarefa 1

### Código da função principal

```
def linear_selection(values, k):
    size = len(values)
    if k < 0 or k > size:
        raise Exception('Parameter k out of limits -> K: {}, SIZE: {}'.format(k, size))
    if size < 5:
        values.sort()
        return values[k-1]
    # slice
    parts = slice_in_groups_of_five(values)
    # find the median of each group
    medians_array = find_medians_array(parts, size)
    # find the median of medians
    all_median = len(medians_array) // 2
    median = linear_selection(medians_array, all_median)

    # partition original data around the median of medians
    lesser, greater = partition_set(values, median)

    lesser_size = len(lesser)
    # base case for the recursion
    if lesser_size == k-1:
        return median
    # recursive calls to find the element
    if lesser_size > k-1:
        return linear_selection(lesser, k)
    if lesser_size < k-1:
        return linear_selection(greater, k-lesser_size-1)
```

### Explicação

**O código implementado satisfaz uma equação de recorrência de forma:**

$$T(n) \leq cst \cdot n + T(n/5) + T(7n/10)$$

*A lista inicial é dividida em grupos de 5.*
*Cada lista é ordenada individualmenta.*
*A mediana de cada grupo é encontrada.*

### Como rodar o programa

**1.** Para rodar qualquer arquivo do trabalho, é necessário extrair os arquivos da pasta *'source'* contida no arquivo **.zip,** e pela linha de comando navegar para o diretório onde a pasta source está localizada.

**2.** Para rodar o código implementado para a seleção linear, basta rodar:

**$** make run

Este comando é um alias para rodar o arquivo *LinearSelection.py* e utiliza o arquivo

“numbers.txt” como fonte de dados. Caso seja necessário, é possível alterar o arquivo mencionado, mantendo o formato já especificado no exemplo (abaixo) que o acompanha.

1 2 3 4 5 6 7 8 81 65 41 99 312 32 61 24 654 23 11 58 33 123 34 12 | Exemplo

O código foi escrito e testado em Python3, então não é garantido que funcione em versões anteriores.

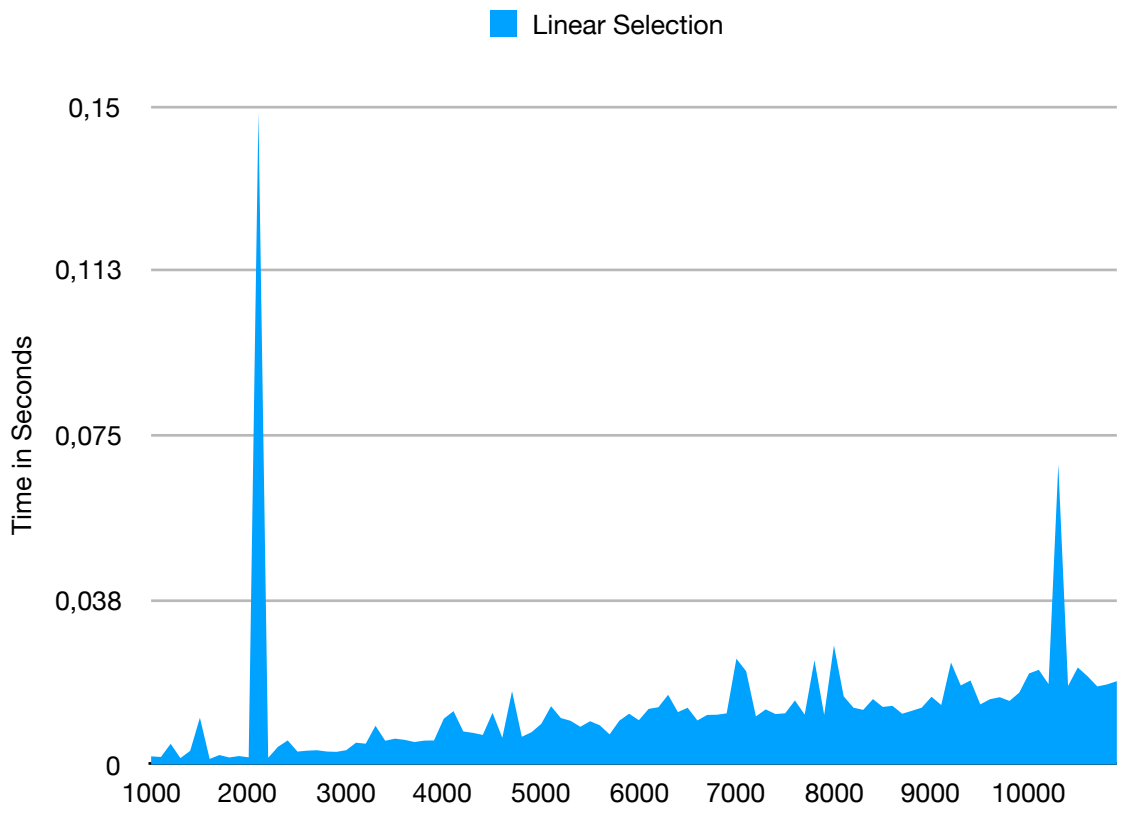
**3.** Para rodar os testes unitários escritos para cada função utilizada no algoritmo principal de seleção linear, basta rodar:

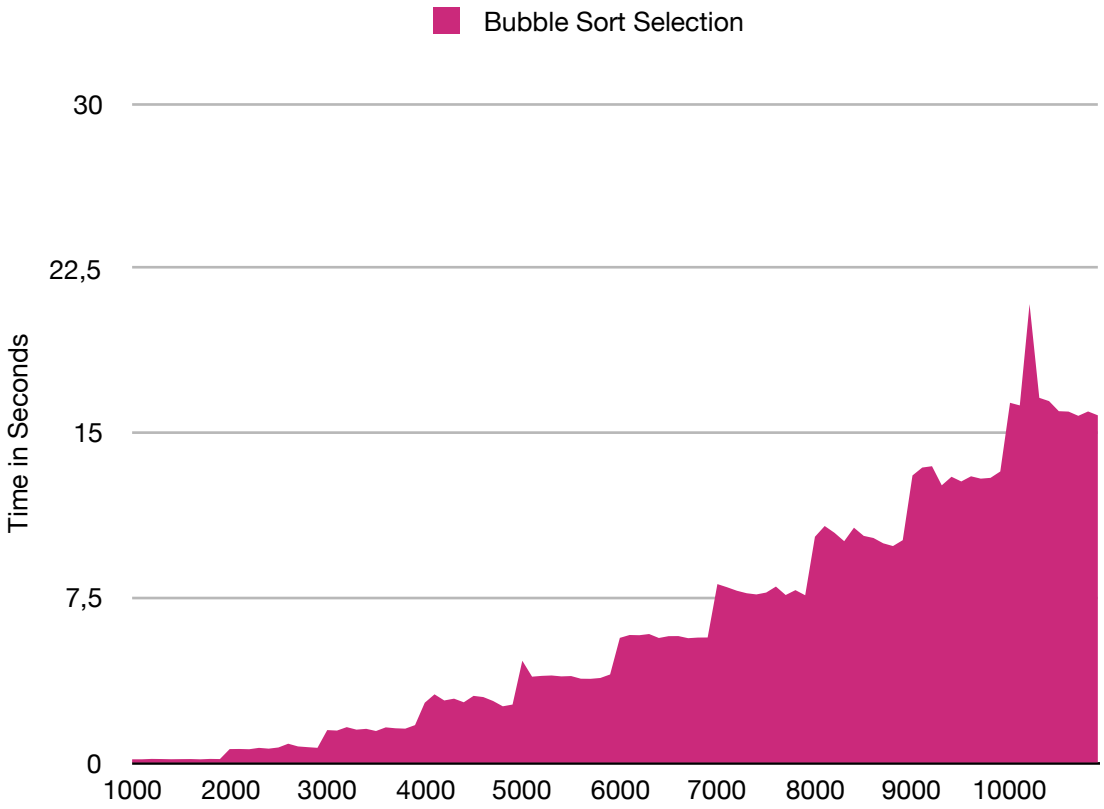
**$** make tests

Este comando é um alias para rodar o arquivo *UnitTests.py* e os dados utilizados estão contidos no mesmo.

**Tarefa 2**

Para gerar os gráficos utilizei os dados gerados no arquivo *ComparisonResult.csv*, porém foram necessárias algumas alterações de formato dos números e configurações de range de cada gráfico, uma vez que a diferença era muito gritante e foi necessário (para melhor visualizar as curvas) que os gráficos fossem separados. O algoritmo de BubbleSort se mostrou quadrático enquanto o de seleção linear mostrou um crescimento linear, como é possível ver abaixo:




Bubble Sort Selection
Time in Seconds
30
22,5
15
7,5
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000