ICMC - Instituto de Ciências Matemáticas e de Computação
BCC - Bacharelado em Ciências da Computação
Disciplina: Laboratório de Introdução a Ciência da Computação
Docente: Leonardo Pereira
Aluno: Bernardo Marques Costa (**Número USP** 11795551)

ANÁLISE DE ALGORITMO E COMPLEXIDADE

INTRODUÇÃO

O intuito deste trabalho é apresentar o tempo de execução para dois algoritmos simples: **bubblesort** e **inversor de arrays.** Será apresentado os algoritmos em si e os gráficos construídos com os seguintes dados: número de inputs fornecidos (tamanho do vetor que sofrerá ação do algoritmo) e o tempo de execução (tempo médio que o algoritmo levou para processar o input fornecido, foram utilizadas 1000 execuções para tornar um valor médio aceitável).

ALGORITMOS E CONTAGEM DE OPERAÇÕES

- **Bubblesort**

```c
void bubblesort(int *vec, int size){

    for(int i = 0; i < size; i++){   // primeira vez -> a + c
                                     // apos a primeira -> n(2a + c)

        for(int j = 0; j < size - 1; j++){ // primeira vez    -> n(a + c) = an + cn
                                           // apos a primeira -> n(n - 1)(2a + c))  = (n² - n)(2a + c) = 2an² + cn² - 2an - cn
            if(vec[j] > vec[j+1]){         // n(n - 1)(2p + a + c)  = (n²  - n)(2p + a + c) = 2pn² + an² + cn² -2pn - an - cn
                //swap
                int temp =  vec[j];        // n(n - 1)(p + a)   = (n² - n)(p + a)   = pn² + an² -pn - an
                vec[j] = vec[j + 1];       // n(n - 1)(2p + 2a) = (n² - n)(2p + 2a) = 2pn² + 2an² -2pn - 2an
                vec[j + 1] = temp;         // n(n - 1)(p + 2a)  = (n² - n)(p + 2a)  = pn² + 2an² - pn - 2an
            }
        }
        // f(n) = a + c + 2an + cn + an + cn + 2an² + cn² - 2an - cn + 2pn² + an² + cn² - 2pn -an -cn + pn² + an² -pn - an +
    }   // '-> + 2pn² + 2an² -2pn - 2an + pn² + 2an² - pn - 2an
}
        // considerando a = c = p = C
        // f(n) = 16Cn² - 11Cn + 2C

```

- **Inverter array**

```c
void invertVector(int *vec, int size){

    for(int i = 0; i < size/2; i++){  // primeira vez -> a + c
                                      // apos a primeira -> n/2(2a + c)

        int temp = vec[i];              // n/2(p + a)
        vec[i] = vec[size - i - 1];     // n/2(2a + 2p)
        vec[size - i -1] = temp;        // n/2(2a + p)
    }
}       // f(n) = a + c + 2an/2 + cn/2 + pn/2 + an/2 + 2an/2 + 2pn/2 + 2an/2 + pn/2
        // f(n) = 7an/2 + 4pn/2 + cn/2 + a + c
        // considerando a = c = p = C
        // f(n) = 12Cn/2 + 2C => f(n) = 6Cn + 2C

```

OBTENDO OS DADOS

Para executar a análise dos dados, foi utilizado um programa que requisitava a seguinte linha de comando para rodar o binário e salvar os dados no arquivo

```
./<programa> <Nome da função desejada> <Número de repetições> <Número de dados>
```

Caso a linha de comando executada fosse diferente da desejada pelo programa, seria apresentada a seguinte saída de erro:

```
Usage: ./a.out < function > < N executions > < Number of data >
        - function available: bubblesort invert
        - N executions: integer
        - Number of data: 10, 100, 1000, 5000, 10000, 100000, 1000000
```

A partir da execução do programa é construído um arquivo **bubblesort.dat** ou **invert.dat**, um arquivo semelhante a um CSV com dados correlacionais: números de inputs e tempo de execução
Obtemos os dois data frame sobre os algoritmos **bubblesort** e **inversor de arrays**, a partir da utilização da biblioteca **pandas** de python:

Bubblesort

	Number of Inputs	Time
0	10	0.000001
1	100	0.000041
2	1000	0.003982
3	5000	0.088282
4	10000	0.375590

Inversor de arrays

	Number of Inputs	Time
0	10	0.000001
1	100	0.000001
2	1000	0.000003
3	5000	0.000016
4	10000	0.000035

GRÁFICO DO TEMPO DE EXECUÇÃO

A partir da utilização da biblioteca **matplotlib.pyplot** de python, podemos construir os seguintes dois gráficos, entre número de inputs e tempo (em segundos):