Relatório de atividade de laboratório nº01 - ECOM13

Nome: Felipe dos Santos
Matrícula: 2019002970

Trechos de códigos

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

void bubbleSort(int vetor[], int tam)
{
    int aux;
    for(int i = 0; i < tam; i++)
    {
        for (int j = 0; j < tam - 1 - i; j++)
        {
            if (vetor[j+1] < vetor[j])
            {
                aux = vetor[j];
                vetor[j] = vetor[j+1];
                vetor[j+1] = aux;
            }

        }
    }
}

```

Figura 01 - bubbleSort

```cpp
void selectionSort(int vetor[], int tam)
{
    int min, aux;
    for (int i = 0; i < (tam-1); i++)
    {
        min = i;
        for (int j = (i+1); j < tam; j++)
        {
            if (vetor[j] < vetor[min])
                min = j;
        }
        aux = vetor[i];
        vetor[i] = vetor[min];
        vetor[min] = aux;
    }

}
```

Figura 02 - Função selectionSort

```cpp
int main()
{
    int N, aux;
    int *v1, *v2;

    cout << "Entre com o valor de N: ";
    cin >> N;

    v1 = new int[N];
    v2 = new int[N];
    //randomização dos valores nos vetores, com inteiros entre 0 e 999
    for (int i = 0; i < N; i++)
    {
        aux = rand() % 1000;
        v1[i] = aux;
        v2[i] = aux;
    }
    //visualização da saída ordenada
    /*
    bubbleSort(v1, N);
    selectionSort(v2, N);

    cout << "bubbleSort: ";
    for (int j = 0; j < N; j++)
    {
        cout << v1[j] << " | ";
    }

    cout << endl << endl << "selectionSort: ";
    for (int k = 0; k < N; k++)
    {
        cout <<  v2[k] << " | ";
    }
    cout << endl;
    */
    //calculo do tempo de execução de cada função
    clock_t tini, tfim, tms;
    tini = clock();
    //BubbleSort(v1, N);
    //selectionSort(v2, N);
    tfim = clock();
    tms = ((tfim - tini)*1000/CLOCKS_PER_SEC);
    cout << "Tempo total: " << tms << "ms" << "\n";
    return 0;
}
```

Figura 03 - Função main

Tabelas

Entrada	Tempo gasto (ms)	
N	BubbleSort	SelectionSort
1000	0	0
5000	47	31
10000	204	110
25000	1420	641
50000	6110	2569
75000	13876	5814
100000	25100	10329

Figura 04 - Comparação Algoritmos de Ordenação

Gráficos

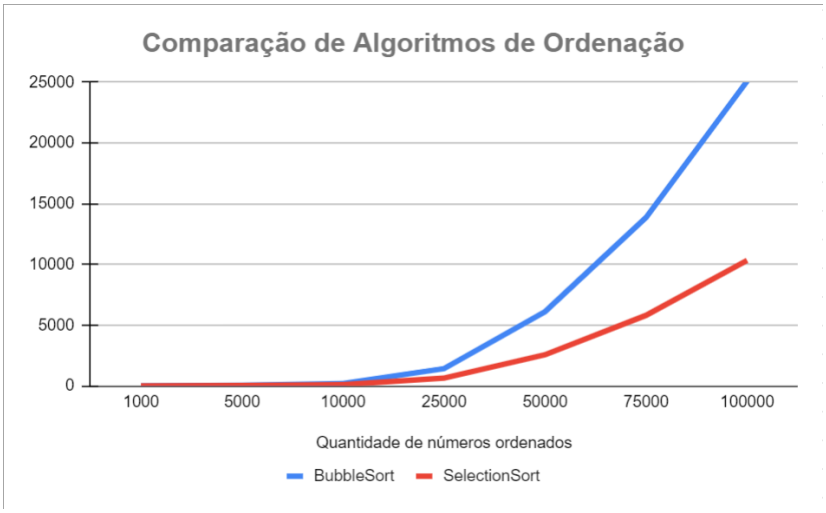

Figura 05 - Comparação Algoritmos de Ordenação

Conclusão

Com relação aos algoritmos de ordenação e seus gráficos de tempo de execução para a mesma entrada, ficou evidente que o método Selection Sort apresentou tempo de execução inferior ao método Bubble Sort. Com entradas de até 10 inteiros, o método Selection Sort ainda era mais eficiente, porém a diferença era de pequena magnitude, conforme mostrado na Figura 05. A partir dos valores de entrada acima de 10000, a diferença entre os dois métodos se tornou maior, intensificando-se cada vez mais até o maior valor de entrada testado (100000). Quanto a execução, o código foi executado para um método por vez, sendo comentadas as linhas referentes a um método na execução do outro. Foi observado também que a carga de trabalho sendo executada no momento da execução do Algoritmo de Ordenação tem influência nos valores de tempo de execução. Para obter os resultados mais fiéis possíveis, softwares em segundo plano foram desativados no momento de execução de ambos os métodos.

Para finalizar, fazendo uma referência à aula, conforme citado pelo professor, para pequenos valores de entrada, muitos algoritmos resolvem o problema, porém com a utilização de entradas maiores os problemas podem não ser resolvidos com eficiência ou até mesmo não serem resolvidos. O experimento acima demonstra o que foi dito.