SSN COLLEGE OF ENGINEERING, KALAVAKKAM
(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LAB EXERCISE 1

Name: Jayannthan P T *Dept: CSE 'A'* *Roll No.: 205001049*

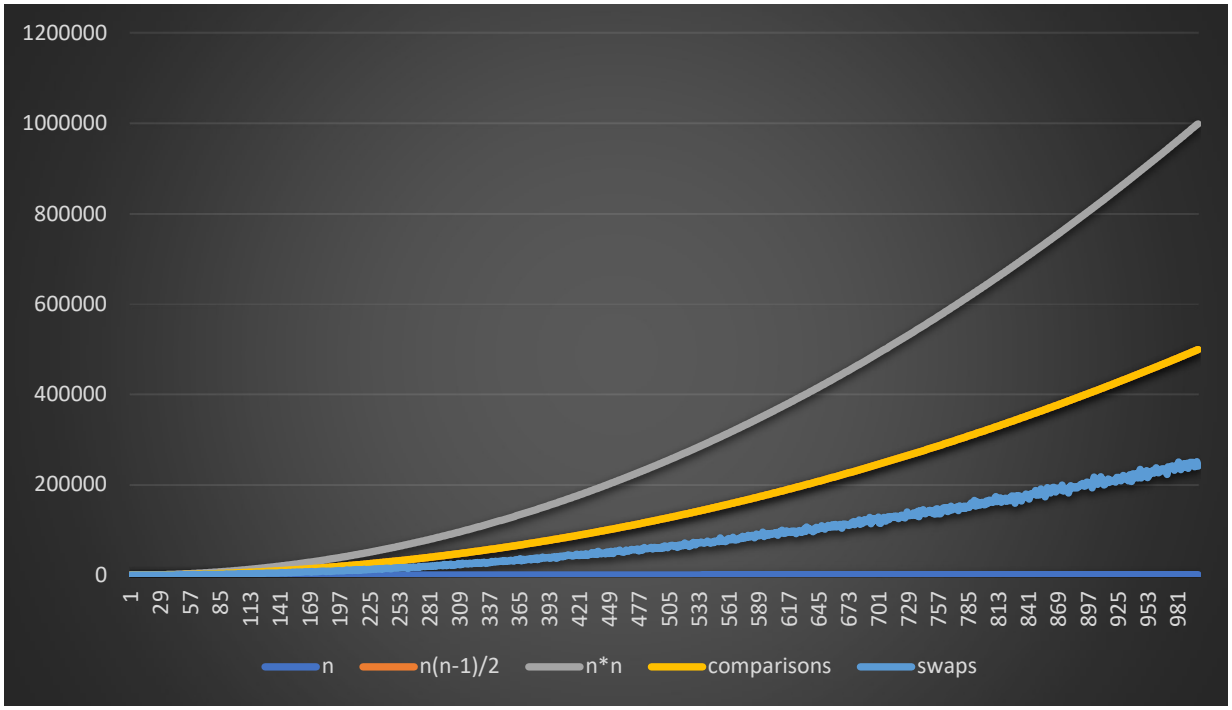
Implement Bubble Sort

1. Measure the number of Comparisons and swap for the program and plot a chart.

Code:

```cpp
#include <iostream>
#include <fstream>

using namespace std;
void bubblesort(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        a[i] = rand() % 10000;
    }
    cout << '\n';
    int cmp = 0, swaps = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n - i - 1; j++)
        {
            cmp++;
            if (a[j] > a[j + 1])
            {
                swaps++;
                swap(a[j], a[j + 1]);
            }
        }
    }
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }
    cout << '\n';
    std::ofstream myfile("bubblesort.csv", std::ios::app);
    myfile << n << "," << (n * (n - 1)) / 2 << "," << n * n << "," << swaps << "\n";
    cout << n << " " << swaps << '\n';
    myfile.close();
}

int main()
{
    // int n = 100;
```

```
    for (int i = 1; i <= 1000; i++)
    {
        int a[i];
        bubblesort(a, i);
    }
    return 0;
}
```

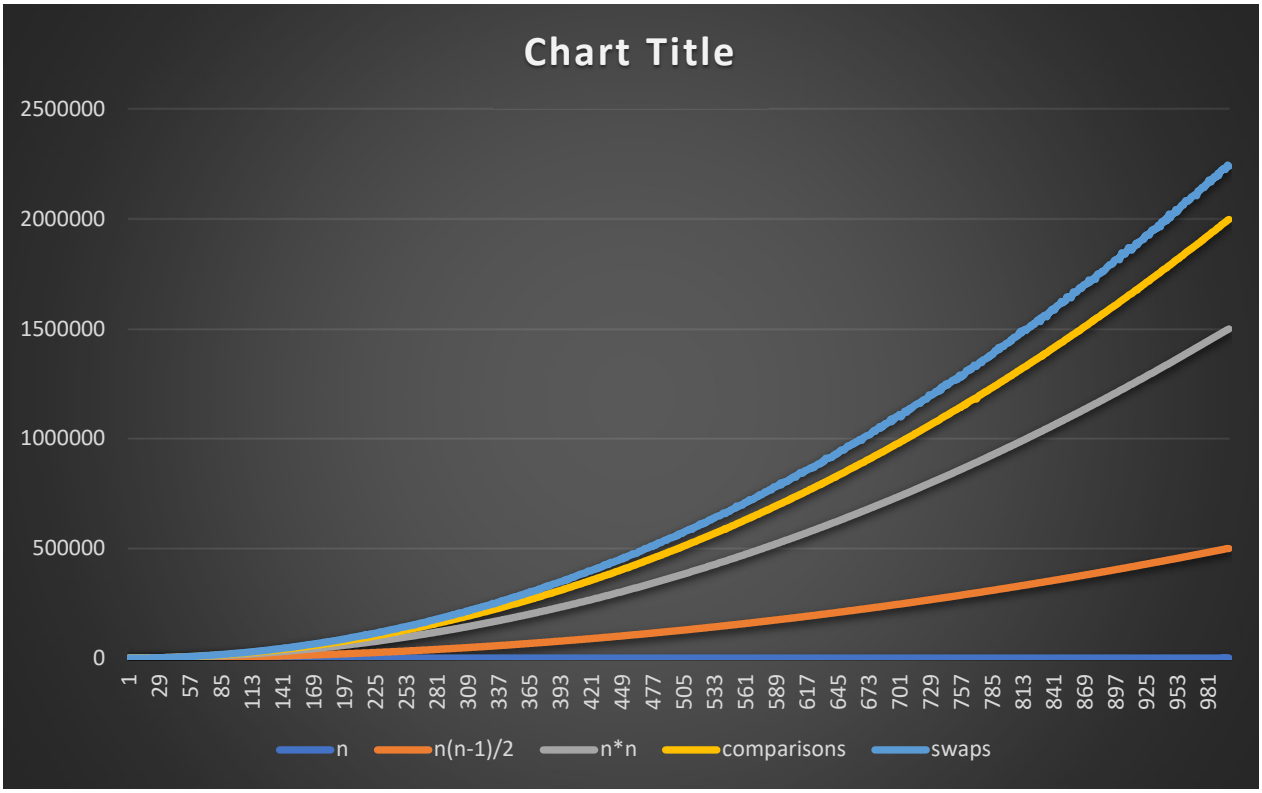


2. Modify the program to have best case Efficiency.

Code:

```
#include <iostream>
#include <fstream>

using namespace std;

bool checker(int a[], int n)
{
    for (int i = 1; i < n - 1; i++)
    {
        if (a[i] < a[i - 1])
        {
            return false;
        }
    }
    return true;
}

void bubblesort(int a[], int n)
{
    // int a[n];
    for (int i = 0; i < n; i++)
    {
```

```
        a[i] = rand() % 10000;
    }
    cout << '\n';
    int comparison = 0, swaps = 0;
    for (int i = 0; i < n; i++)
    {
        if (!checker(a, n))
        {
            for (int j = 0; j < n - i - 1; j++)
            {
                comparison++;
                if (a[j] > a[j + 1])
                {
                    swaps++;
                    swap(a[j], a[j + 1]);
                }
            }
        }
        else
            break;
    }
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }
    cout << '\n';
    std::ofstream myfile("bubblesorttemp.csv", std::ios::app);
    myfile << n << "," << (n * (n - 1)) / 2 << "," << n * n << "," << comparison << "," <<
swaps << "\n";
    cout << n << " " << swaps << '\n';
    myfile.close();
}

int main()
{
    // int n = 100;
    for (int i = 1; i <= 1000; i++)
    {
        int a[i];
        bubblesort(a, i);
    }
    return 0;
}
```



3. Check whether the algorithm has the 2 properties.
    - In place sorting takes place in bubble sort.

4. Implement recursive bubble sort.

```
void bubblesortrecursive(int a[], int n)
{
    if (n == 1)
    {
        return;
    }
    for (int i = 0; i < n; i++)
    {
        if (a[i] > a[i + 1])
        {
            swap(a[i], a[i + 1]);
        }
    }
    bubblesortrecursive(a, n - 1);
}
```

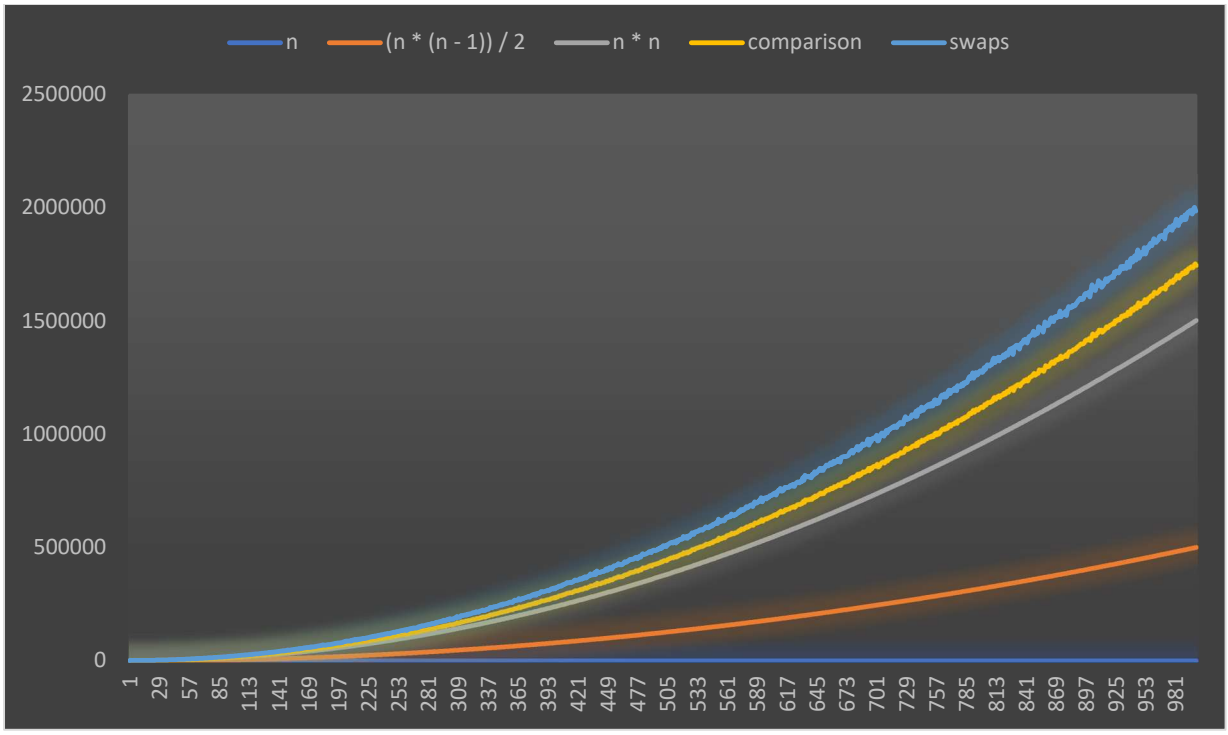
Implement Insertion Sort

1. Measure the number of Comparisons and swap for the program and plot a chart.

Code:

```cpp
#include <bits/stdc++.h>
#include <iostream>
#include <fstream>
using namespace std;

void swap(int arr[], int init, int fin)
{
    for (int i = fin; i >= init; i--)
    {
        arr[i] = arr[i - 1];
    }
}
void printarray(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }
    cout << endl;
}

void insertionSort(int a[], int n)
{
    int swaps = 0, comparison = 0;
    for (int i = 1; i < n; i++)
    {
        int big = a[i];
        int j = i - 1;
        comparison++;
        while (big < a[j] && j >= 0)
        {
            swaps++;
            a[j + 1] = a[j];
            --j;
            comparison++;
        }
        a[j + 1] = big;
        // printarray(a,n);
    }
    cout << swaps << " " << comparison << endl;
    std::ofstream myfile("insertion.csv", std::ios::app);
    myfile << n << "," << (n * (n - 1)) / 2 << "," << n * n << "," << comparison << "," <<
swaps << endl;
    // swapcomparison[0] = swaps;
    // swapcomparison[1] = comparison;
}

int main()
```

```
{
    std::ofstream myfile("insertion.csv", std::ios::app);
    myfile << "n"
           << ","
           << "(n * (n - 1)) / 2"
           << ","
           << "n * n"
           << ","
           << "comparison"
           << ","
           << "swaps" << endl;

    for (int i = 1; i <= 1000; i++)
    {
        int n = i;
        // int swapcomparison[2] = {0, 0};
        int a[n];
        for (int i = 0; i < n; i++)
        {
            a[i] = rand() % 10000;
        }
        cout << endl;
        cout << "Before Sort:";
        printarray(a, n);
        insertionSort(a, n);
        cout << "After Sort:";
        printarray(a, n);
        // fout << swapcomparison[0] << ", "
        //      << swapcomparison[1]
        //      << "\n";
    }
    return 0;
}
```



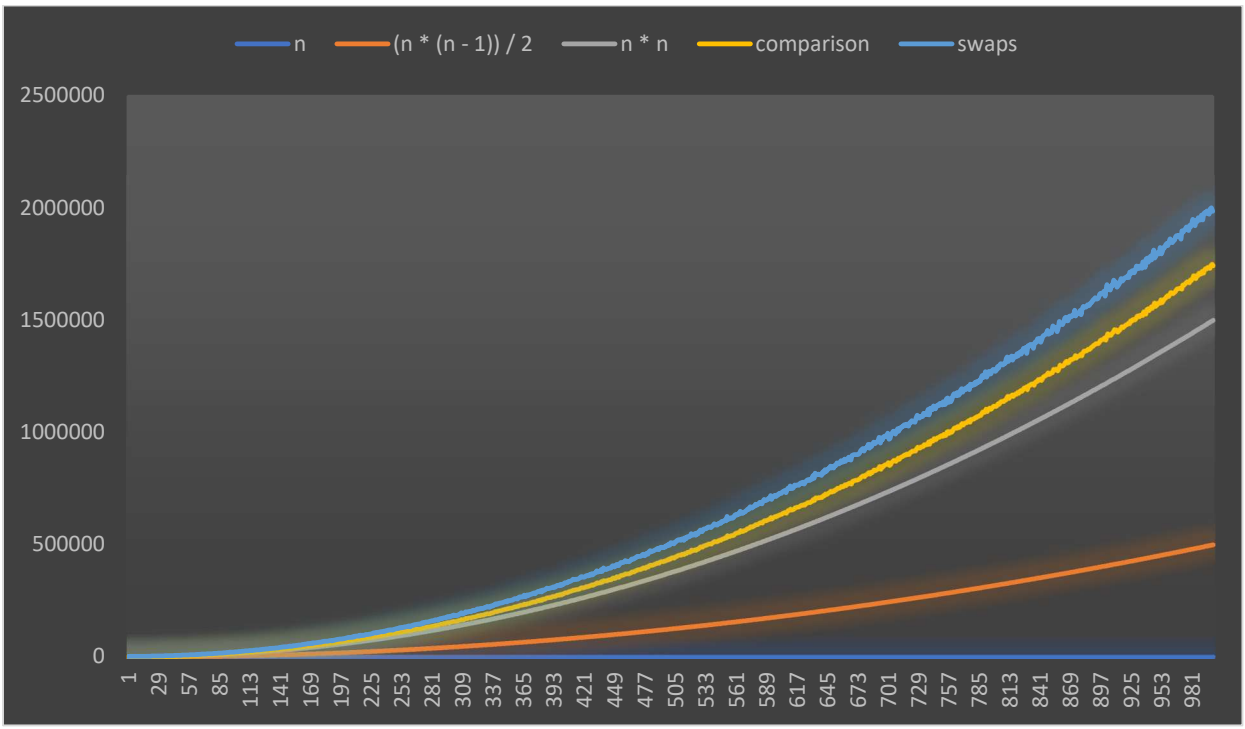
2. Modify the program to have best case Efficiency.

Code:

```
#include <bits/stdc++.h>
#include <iostream>
#include <fstream>
using namespace std;

bool checker(int a[], int n)
{
    for (int i = 1; i < n - 1; i++)
    {
        if (a[i] < a[i - 1])
        {
            return false;
        }
    }
    return true;
}

void swap(int arr[], int init, int fin)
{
    for (int i = fin; i >= init; i--)
    {
        arr[i] = arr[i - 1];
    }
}
void printarray(int a[], int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }
    cout << endl;
```

```
}

void insertionSort(int a[], int n)
{
    int swaps = 0, comparison = 0;
    for (int i = 1; i < n; i++)
    {
        if (!checker(a, n))
        {
            int big = a[i];
            int j = i - 1;
            comparison++;
            while (big < a[j] && j >= 0)
            {
                swaps++;
                a[j + 1] = a[j];
                --j;
                comparison++;
            }
            a[j + 1] = big;
        }
        else
        {
            break;
        }
        }
    // cout << swaps << " " << comparison << endl;
    // std::ofstream myfile("insertion.csv", std::ios::app);
    // myfile << n << "," << (n * (n - 1)) / 2 << "," << n * n << "," << comparison << ","
<< swaps << endl;
    // swapcomparison[0] = swaps;
    // swapcomparison[1] = comparison;
}

int main()
{
    // std::ofstream myfile("insertion.csv", std::ios::app);
    // myfile << "n"
    //        << ","
    //        << "(n * (n - 1)) / 2"
    //        << ","
    //        << "n * n"
    //        << ","
    //        << "comparison"
    //        << ","
    //        << "swaps" << endl;

    for (int i = 1; i <= 1000; i++)
    {
        int n = i;
        // int swapcomparison[2] = {0, 0};
        int a[n];
        for (int i = 0; i < n; i++)
        {
```

```
            a[i] = rand() % 10000;
        }
        cout << endl;
        cout << "Before Sort:";
        printarray(a, n);
        insertionSort(a, n);
        cout << "After Sort:";
        printarray(a, n);
        // fout << swapcomparison[0] << ", "
        //      << swapcomparison[1]
        //      << "\n";
    }
    return 0;
}
```



3. Check whether the algorithm has the 2 properties.
    - In place sorting takes place in bubble sort.

4. Implement recursive bubble sort.

```
void insertionSortrecursive(int a[], int n)
{
    if (n == 1)
    {
        return;
    }

    insertionSortrecursive(a, n - 1);
    int last = a[n - 1];
    int j = n - 2;
```

```
    while (j > 0 and a[j] > last)
    {
        a[j + 1] = a[j];
        j--;
    }
    a[j + 1] = last;
}
```