**DESIGN AND ANALYSIS OF**

**ALGORITHM**

**Practical File**

**NAME- SUYASH GAHTORI**

**ROLL NO- 21/18007**

**COURSE- BSc(H) Computer Science**

# PRACTICAL 1 (a)

**Objective**

Implement Insertion Sort (The program should report the number of comparisons)

# Code

#include <iostream> #include <cstdlib> using namespace std;

void InsertionSort(int array[], int n)

{

int i , j , key; int count = 0;

for (i = 1 ; i < n ; i++)

{

key = array[i]; j = i - 1;

while (j >= 0 && array[j] > key)

{

count++;

array[j + 1] = array[j]; j = j - 1;

}

array[j + 1] = key;

}

cout << "\n" << n; cout << "," << count;

}

int main()

{

cout << "Number of Elements , Number of Comparisons";

for (int x = 0 ; x < 100 ; ++x)

{

int n = rand()%971 + 30; int\* array = new int[n];

for (int i = 0 ; i < n ; ++i)

{

array[i] = rand();

}

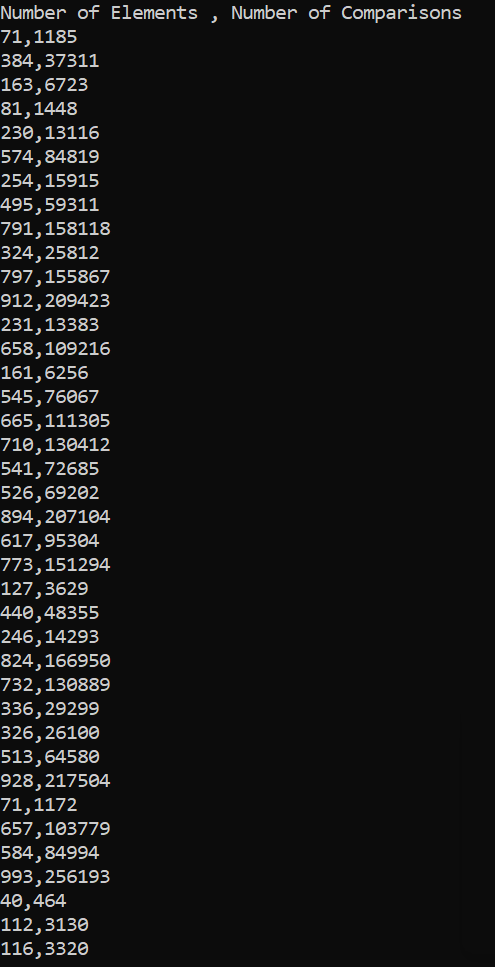
InsertionSort(array,n);

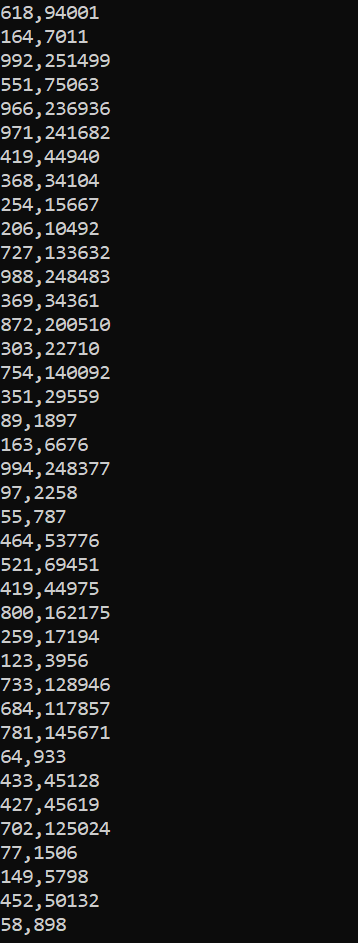
}

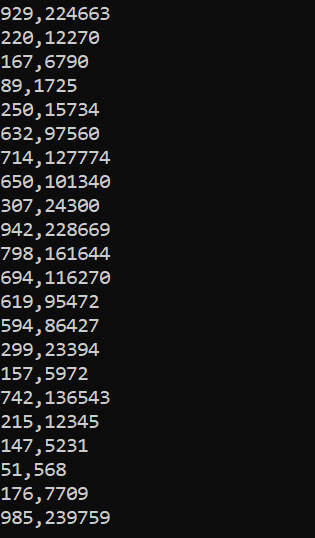
cout << endl; return 0;

}

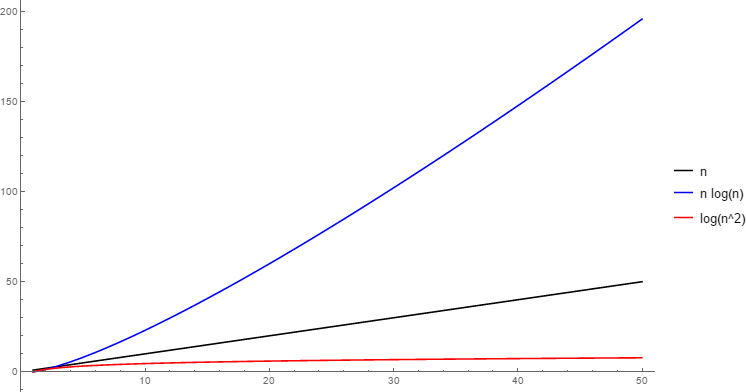
# Output







**Graph**



# PRACTICAL 1 (b)

**Objective**

Implement Merge Sort (The program should report the number of comparisons)

# Code

#include <iostream> #include <fstream> #include <vector> using namespace std;

vector <int> merge(vector <int> &array1,vector <int> &array2,int\* &count){ int lengthOfArray1=array1.size();

int lengthOfArray2=array2.size(); vector <int> res(

lengthOfArray1

+

lengthOfArray2

);

int leftPointer = 0; int rightPointer = 0; int resPointer = 0;

while ( leftPointer < lengthOfArray1 && rightPointer < lengthOfArray2)

{

if (array1[leftPointer]<array2[rightPointer])

{

res[resPointer] = array1[leftPointer];

resPointer+=1; leftPointer+=1;

\*count+=1;

}

else

{

res[resPointer] = array2[rightPointer]; resPointer+=1;

rightPointer+=1;

\*count+=1;

}

}

while(leftPointer<lengthOfArray1)

{

res[resPointer] = array1[leftPointer]; resPointer+=1;

leftPointer+=1;

\*count+=1;

}

while(rightPointer<lengthOfArray2)

{

res[resPointer] = array2[rightPointer]; resPointer+=1;

rightPointer+=1;

\*count+=1;

}

return res;

}

//1 1 2 1 3

vector <int> mergeSort(vector <int> &array,int start,int end,int \* &count){ if (end-start==0)

{

vector <int> r; r.push\_back(array[start]); return r;

}

int mid = (start+end)/2;

vector <int> left = mergeSort(array,start,mid,count); vector <int> right= mergeSort(array,mid+1,end,count); return merge(left,right,count);

}

int main(int argc, char const \*argv[]){ ofstream MyFile("Coordinates.txt"); int \* count = new int(0);

cout<<"Size of Array,No. of Comparisons\n"; for(int x=0; x < 100; ++x)

{

\*count = 0;

int n = rand() % 971 + 30; vector <int> vArray; for(int i=0; i<n; ++i)

{

vArray.push\_back(rand());

}

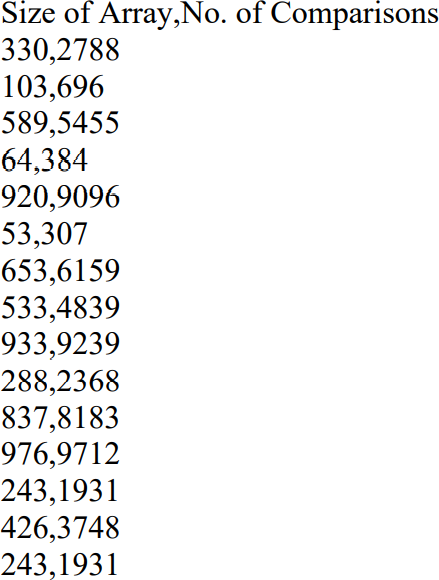
vector res = mergeSort(vArray,0,n-1,count); MyFile <<n<<","<<\*count<<"\n"; cout<<n<<","<<\*count<<endl;

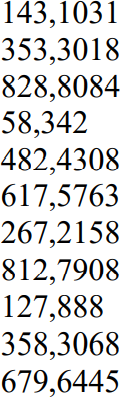
}

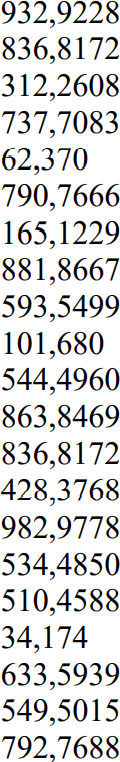
MyFile.close(); return 0;

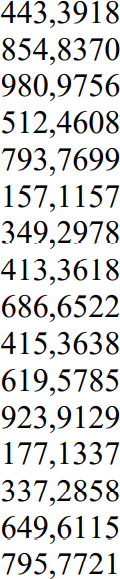
}

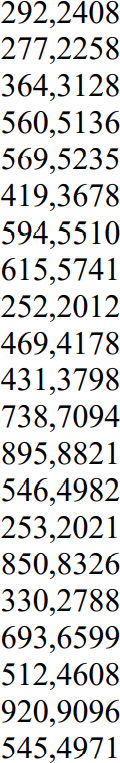
**OUTPUT**

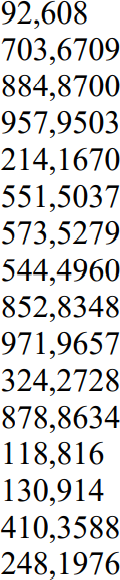




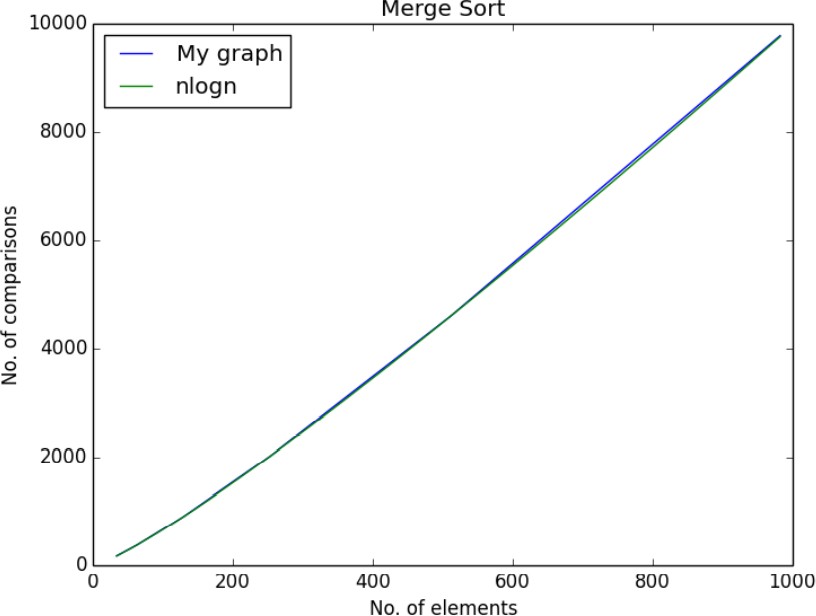








**GRAPH OF (n log n) VS Merge Sort**



# PRACTICAL 2

**Objective**

Implement Heap Sort (The program should report the number of comparisons)

# Code

#include <bits/stdc++.h> using namespace std; int cnt = 0;

void heapify(int arr[], int n, int i)

{

cnt++;

int largest = i;

int left = 2 \* i + 1; int right = 2 \* i + 2;

if (left < n && arr[left] > arr[largest]) largest = left;

if (right < n && arr[right] > arr[largest]) largest = right;

if (largest != i)

{

swap(arr[i], arr[largest]); heapify(arr, n, largest);

}

}

void heapsort(int arr[], int n)

{

// Building max-heap

for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);

// heap sort

for (int i = n - 1; i >= 0; i--)

{

swap(arr[0], arr[i]);

// heapify root element heapify(arr, i, 0);

}

}

int main()

{

int size;

ofstream fout("MyExcel.csv"); fout << "Size"

<< ","

<< "Comparisons" << endl; srand(time(0));

for (int i = 0; i < 100; i++)

{

size = rand() % 971 + 30; int Array[size] = {0};

for (int j = 0; j < size; j++)

{

Array[j] = rand() % 10000;

}

// cout << "Unsorted array" << endl;

// for (int i = 0; i < size; i++)

// {

// cout << Array[i] << " ";

// }

heapsort(Array, size);

fout << size << "," << cnt << endl; cnt = 0;

// cout << "Sorted array" << endl;

// for (int i = 0; i < size; i++)

// {

// cout << Array[i] << " ";

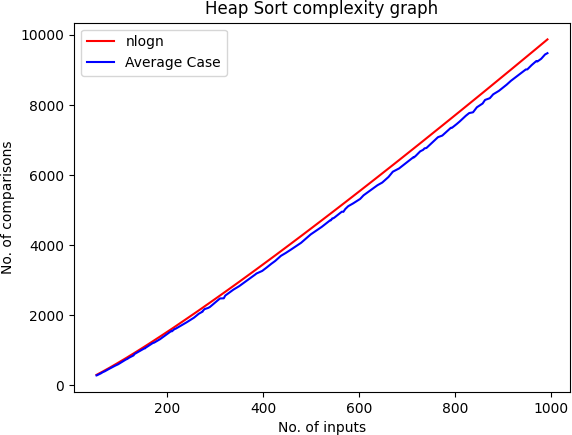
// }

}

return 0;

}

# Output



**PRACTICAL 3**

# Objective

Implement Randomised Quick Sort (The program should report the number of comparisons)

# Code

#include <bits/stdc++.h> using namespace std; int cnt = 0;

int Partition(int arr[], int low, int high)

{

int pivot = arr[high]; int i = (low - 1);

for (int j = low; j <= high - 1; j++)

{

if (arr[j] <= pivot)

{ i++;

swap(arr[i], arr[j]); cnt++;

}

}

swap(arr[i + 1], arr[high]); cnt++;

return (i + 1);

}

int partitionRandom(int arr[], int low, int high)

{

srand(time(NULL));

int random = low + rand() % (high - low); swap(arr[random], arr[high]);

cnt++;

return Partition(arr, low, high);

}

void quickSort(int arr[], int low, int high)

{

if (low < high)

{

int pi = partitionRandom(arr, low, high); quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

int main()

{

int size;

ofstream fout("MyExcel.csv"); fout << "Size"

<< ","

<< "Comparisons" << endl;

// srand(time(0)); size = 30;

for (int i = 0; i < 100; i++)

{

int Array[size] = {0};

for (int j = 0; j < size; j++)

{

Array[j] = rand() % 10000;

}

// cout << "Unsorted array" << endl;

// for (int i = 0; i < size; i++)

// {

// cout << Array[i] << " ";

// }

quickSort(Array, 0, size - 1);

fout << size << "," << cnt << endl;

// cout << "Size: " << size << " cnt: " << cnt << endl; cnt = 0;

// cout << "Sorted array" << endl;

// for (int i = 0; i < size; i++)

// {

// cout << Array[i] << " ";

// }

if (i < 20) size += 9; else

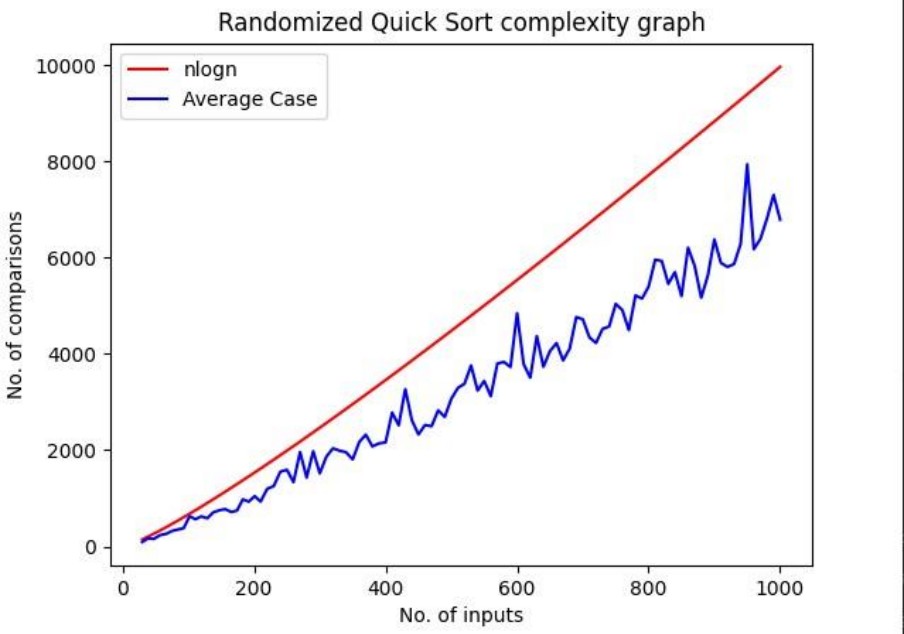
size += 10;

}

return 0;

}

# Output



**PRACTICAL 4**

**Objective** Implement Radix Sort **Code**

#include <iostream>

using namespace std;

// function to get maximum value in arr[] int getMax(int arr[], int n)

{

int mx = arr[0];

for (int i = 1; i < n; i++) if (arr[i] > mx)

mx = arr[i];

return mx;

}

void countSort(int arr[], int n, int exp)

{

int output[n]; // output array int i, count[10] = { 0 };

// Store count of occurrences in count[] for (i = 0; i < n; i++)

count[(arr[i] / exp) % 10]++;

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

for (i = n - 1; i >= 0; i--) {

output[count[(arr[i] / exp) % 10] - 1] = arr[i]; count[(arr[i] / exp) % 10]--;

}

for (i = 0; i < n; i++)

arr[i] = output[i];

}

// The main function that sorts arr[] of size n using Radix Sort

void radixsort(int arr[], int n)

{

// Find the maximum number to know number of digits int m = getMax(arr, n);

for (int exp = 1; m / exp > 0; exp \*= 10) countSort(arr, n, exp);

}

// function to print an array void print(int arr[], int n)

{

for (int i = 0; i < n; i++) cout << arr[i] << " ";

}

// Driver Code int main()

{

int arr[] = { 170, 45, 75, 90, 802, 24, 2, 66 };

int n = sizeof(arr) / sizeof(arr[0]);

// Function Call radixsort(arr, n); print(arr, n); return 0;

}

# Output

