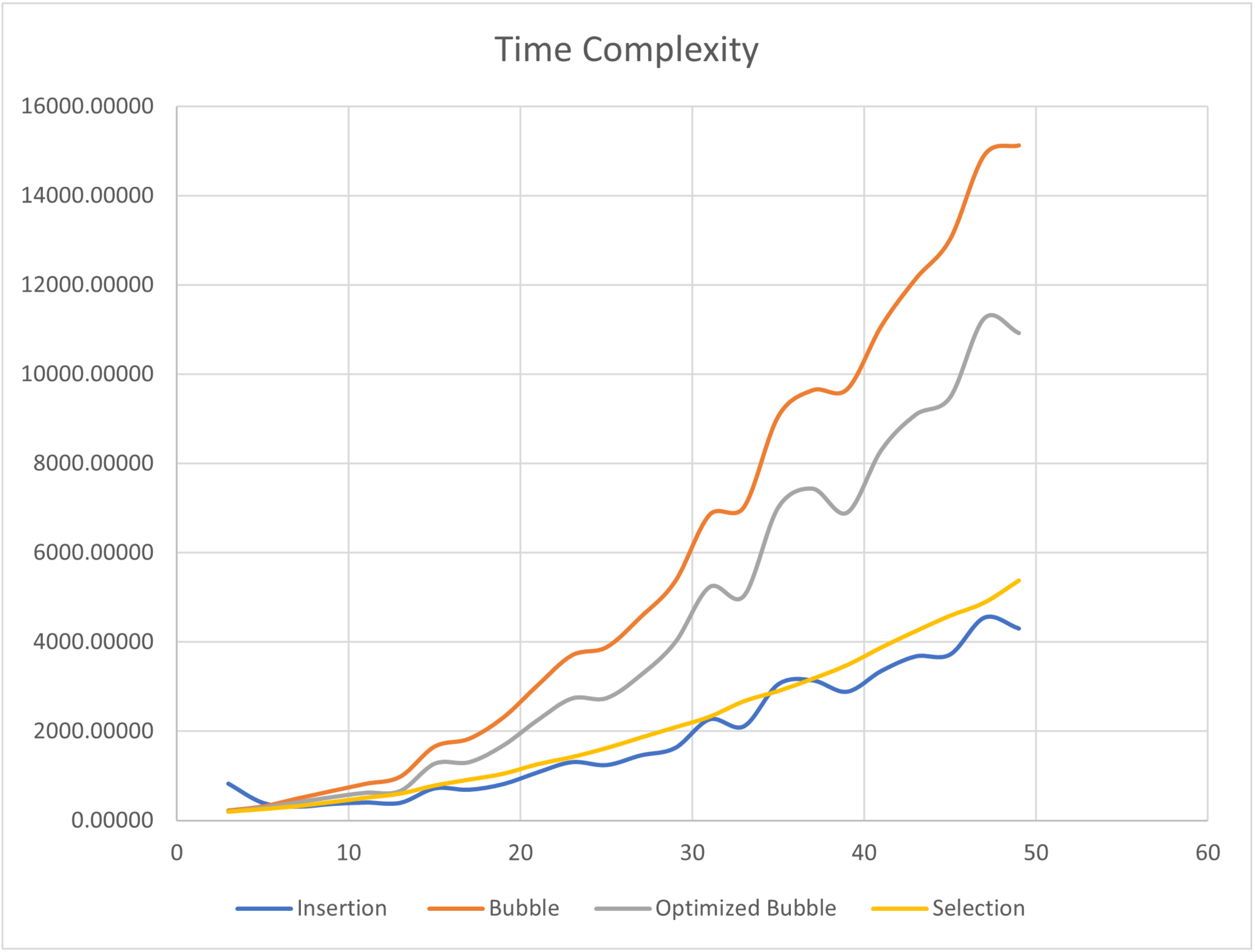
Inclass Lab(week 2)

220689N MKSL Weerasiri

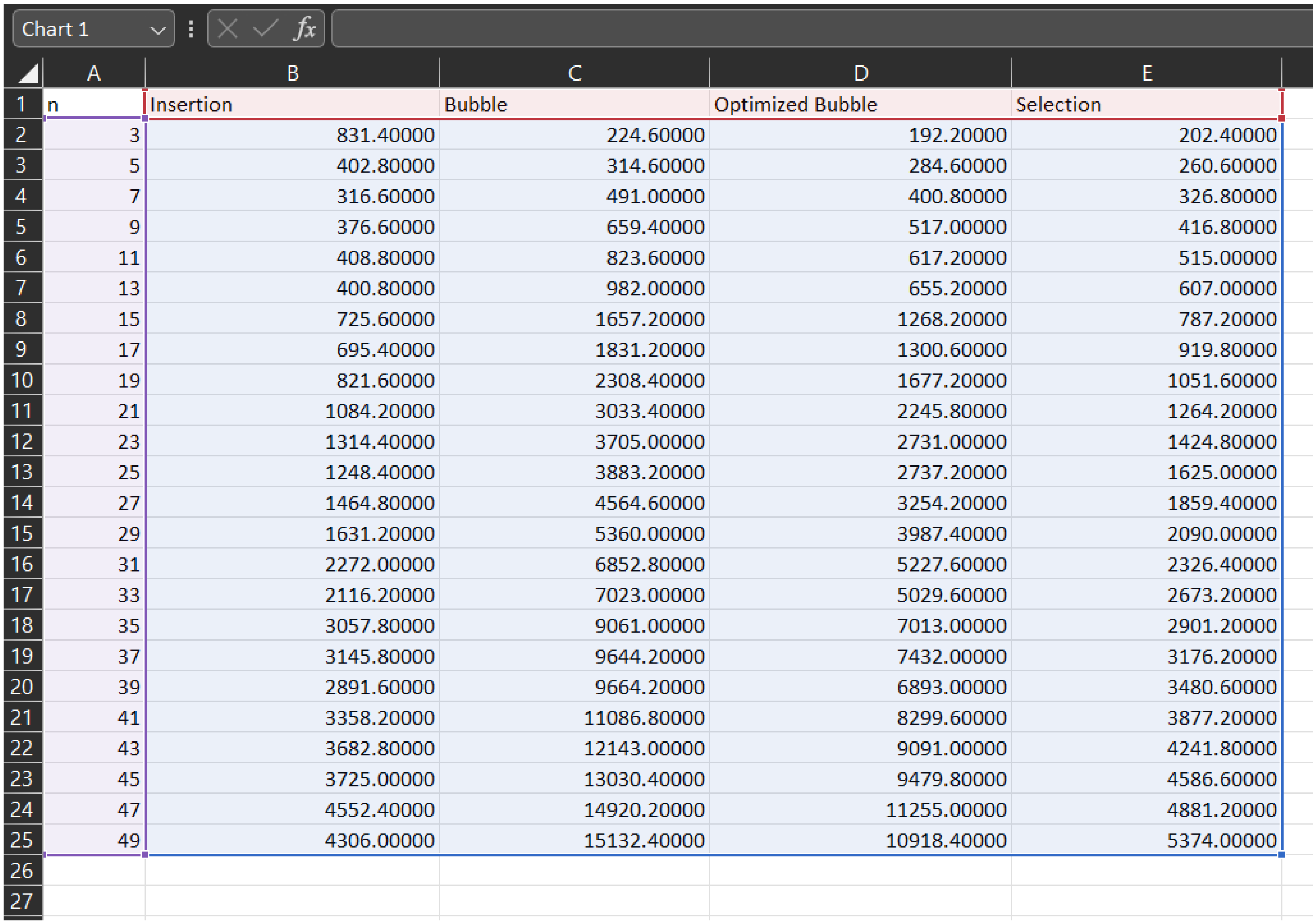
Graph:



Time: nanoseconds

Graph By MS Excel

<u>Data:</u>



n	Insertion	Bubble	Optimized Bubble	Selection
3	831.40000	224.60000	192.20000	202.40000
5	402.80000	314.60000	284.60000	260.60000
7	316.60000	491.00000	400.80000	326.80000
9	376.60000	659.40000	517.00000	416.80000
11	408.80000	823.60000	617.20000	515.00000
13	400.80000	982.00000	655.20000	607.00000
15	725.60000	1657.20000	1268.20000	787.20000
17	695.40000	1831.20000	1300.60000	919.80000
19	821.60000	2308.40000	1677.20000	1051.60000
21	1084.20000	3033.40000	2245.80000	1264.20000
23	1314.40000	3705.00000	2731.00000	1424.80000
25	1248.40000	3883.20000	2737.20000	1625.00000
27	1464.80000	4564.60000	3254.20000	1859.40000
29	1631.20000	5360.00000	3987.40000	2090.00000
31	2272.00000	6852.80000	5227.60000	2326.40000
33	2116.20000	7023.00000	5029.60000	2673.20000
35	3057.80000	9061.00000	7013.00000	2901.20000
37	3145.80000	9644.20000	7432.00000	3176.20000
39	2891.60000	9664.20000	6893.00000	3480.60000
41	3358.20000	11086.80000	8299.60000	3877.20000
43	3682.80000	12143.00000	9091.00000	4241.80000
45	3725.00000	13030.40000	9479.80000	4586.60000
47	4552.40000	14920.20000	11255.00000	4881.20000
49	4306.00000	15132.40000	10918.40000	5374.00000

Data By the below Code

Ran on
https://www.tutorialspoint.com/compile_cpp_online.php

Code:

```
#include <iostream>

#include <vector>

#include <chrono>

using namespace std;

void print(int n,vector<int> arr){

   /*

   Print a given vector seperated by a space

   Inputs : int (number of elements in the vector), vector<int> (vector)

   Outputs : NULL

   */

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

      std::cout<<arr[i]<<" ";

   }

   std::cout<<"\n";

}

vector<vector<int>> makeRandomArrays(int start_size,int end_size,int step, int
value_limit){

   /*

   Make random numbered vector collection

   Inputs : int (starting size), int (ending size), int (stepping size), int(maximum
value expected in a vector)

   Outputs : vecotr<vector<int>> vector of vectors of random numbers*/

   vector<vector<int>> arrays;

   vector<int> sample;
```

```
    for(int i=start_size;i<end_size+1;i=i+step){
        sample.clear();
        for(int j=0;j<i;j++){//create a random numbered vector
            sample.push_back(rand()%(value_limit+1));
        }
        arrays.push_back(sample);
    }
    return arrays;
}
void swap(int &a,int &b){
    /*
    Swap given two numbers
    Inputs : int (first number), int (second number)
    Outputs : NULL
    */
    int temp=a;
    a=b;
    b=temp;
}


void runtheProgramInsertion(int n,vector<int> array){
    /*
    Sorting numbers using the insertion sort
    Inputs : int (number of elements in the vector), vector<int> (vector)
    Outputs : NULL
    */
    bool is_inserted;
    for(int i=0;i<n;i++){
        is_inserted=false;
        int temp=array[i];//keep the current value out of the vector
        for(int j=i-1;j>=0;j--){
            if(array[j]>temp){
                array[j+1]=array[j];//shift the higher values to the right
            }else{
                array[j+1]=temp;//insert the key value
                is_inserted=true;
                break;
            }
        }
        if(!is_inserted){
            array[0]=temp;//insert the value to the front of the vector
        }
    }
}
void runtheProgramBubble(int n,vector<int> array){
    /*
    Sorting numbers using the Bubble sort
    Inputs : int (number of elements in the vctor), vector<int> (vector)
    Outputs : NULL
    */
    for(int i=0;i<n;i++){
        for(int j=0;j<n-1;j++){
            if(array[j]>array[j+1]){//check wheather the next value is greater than
current value
                swap(array[j+1],array[j]);// swap values
            }
        }
    }
}
void runtheProgramOptimizedBubble(int n,vector<int> array){
    /*
    Sorting numbers using the Optimized Bubble sort
    Inputs : int (number of elements in the vctor), vector<int> (vector)
    Outputs : NULL
    */
    bool is_swapped;
    for(int i=0;i<n;i++){
        is_swapped=false;
        for(int j=0;j<n-i-1;j++){
            if(array[j]>array[j+1]){//check wheather the next value is greater than
current value
                swap(array[j+1],array[j]);// swap values
            is_swapped=true;
          }
        }
        if(!is_swapped){//no swaps, that means the numbers are sorted
          break;
        }
    }
}
void runtheProgramSelection(int n,vector<int> array){
    /*
    Sorting numbers using the Selection sort
    Inputs : int (number of elements in the vctor), vector<int> (vector)
    Outputs : NULL
    */
    int index_minimum;
    for(int i=0;i<n;i++){
        index_minimum=i;//keep minimum value's index
        for(int j=i+1;j<n;j++){
          if(array[j]<array[index_minimum]){//check for minimum value
            index_minimum=j;
          }
        }
        if(index_minimum!=i){//If minimum index is changed
          swap(array[index_minimum],array[i]);//then only we have to swap
        }
    }

}

int main() {
    //Get the values
    vector<vector<int>> arrays=makeRandomArrays(3,50,2,100);

    double sum_duration;
    vector<double> avg_duration;
    string topic;//Name of the Sorting Algorithm

    for(int sorting=0;sorting<4;sorting++){
        avg_duration.clear();
        for(int t=0;t<arrays.size();t++){
            sum_duration=0.0f;

            for(int i=0;i<5;i++){//5 times

                auto start = chrono::high_resolution_clock::now();//start time


                switch(sorting){
```

```
    case 0://Insertion
    runtheProgramInsertion(arrays[t].size(),arrays[t]);
    topic="\n\n\nInsertion\n#########\n";
    break;
    case 1://Bubble
    runtheProgramBubble(arrays[t].size(),arrays[t]);
    topic="\n\n\nBubble\n#########\n";
    break;
    case 2://Optimized Bubble
    runtheProgramOptimizedBubble(arrays[t].size(),arrays[t]);
    topic="\n\n\nOptimized Bubble\n#########\n";
    break;
    case 3://Selection
    runtheProgramSelection(arrays[t].size(),arrays[t]);
    topic="\n\n\nSelection\n#########\n";
    break;
    default:
    break;
}
auto end = chrono::high_resolution_clock::now();//end time

// Calculating total time taken by the program.
double duration =
chrono::duration_cast<chrono::nanoseconds>(end - start).count();
```

```
                sum_duration=sum_duration+duration;




            }
            avg_duration.push_back(sum_duration/5.0f);//get average
        }
        cout<<topic;//print name of the sorting algorithm
        for(int i=0;i<avg_duration.size();i++){
            printf("%.20f\n",avg_duration[i]);
        }

    }
    return 0;
}
```