U19CS076 DAA ASSIGNMENT 2

KRITHIKHA BALAMURUGAN

- Insertion sort

Q1. Analyze the time complexity of the above algorithm using the RAM model

U19CS076

1. Insertion Sort

1.1 RAM Model

Insertion_sort (arr, count)

		Cost of instruct
1.	START	
2.	for i = 1 to count :	c_1
3.	set v = arr[i]	c_2
4.	set j = i − 1	c_3
5.	while (j >= 0 and arr[j] > v)	c_4
6	arr[j+1] = arr[j]	c_5
7.	set j = j − 1	c_6
8.	set arr[j+1] = v	c_7
9.	STOP	

Total cost ⇒

	Best	Worst
2.	n	
3.	n − 1	
4.	n − 1	
5.	$\sum_{i=1}^{n-1} 1$	$\sum_{i=1}^{n-1} i$
6.	$\sum_{i=1}^{n-1} 0$	$\sum_{i=1}^{n-1} (i-1)$
7.	$\sum_{i=1}^{n-1} 0$	$\sum_{i=1}^{n-1} (i-1)$
8.	n − 1	

Best Case Time Cost

$$T(n) = c_1 n + c_2 (n-1) + c_3 (n-1) + c_4 (n-1) + c_7 (n-1)$$

$$= [c_1 + c_2 + c_3 + c_4 + c_7] n - (c_2 + c_3 + c_4 + c_7)$$

U19CS076

$T(n) = an + b$

For worst case

$$T(n) = c_1 n + c_2(n-1) + c_3(n-1) + c_4\left(\sum_{i=1}^{n-1} i\right) + c_5\left(\sum_{i=1}^{n-1}(i-1)\right) + c_6\left(\sum_{i=1}^{n-1}(i-1)\right) + c_7(n-1)$$

Let $S_1 = \sum_{i=1}^{n-1} i$

$$\begin{aligned} S_1 &= 1 + 2 + \cdots + n-1 \\ + \quad S_1 &= (n-1) + (n-2) + \cdots + 3 + 2 + 1 \end{aligned}$$

$$2S_1 = n + n + \cdots + n \quad (n-1 \text{ times})$$

$$S_1 = \frac{n(n-1)}{2}$$

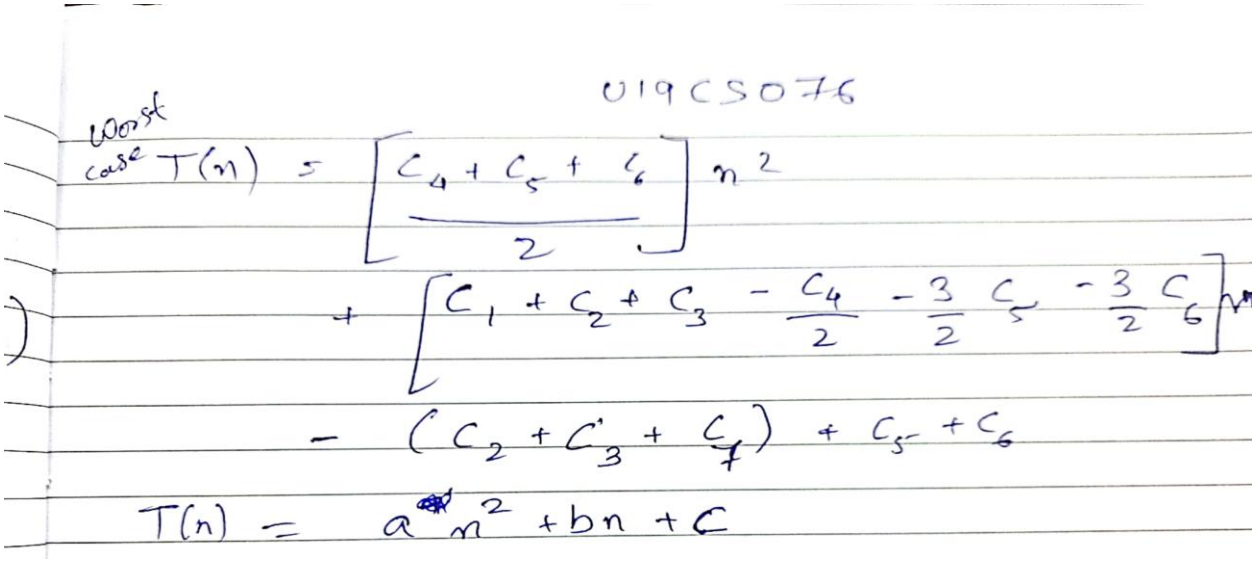
$$S_2 = \sum_{i=1}^{n-1}(i-1)$$

$$\begin{aligned} S_2 &= 0 + 1 + 2 + \cdots + n-2 \\ + \quad S_2 &= n-2 + n-3 + \cdots + 0 \end{aligned}$$

$$2S_2 = n-2 + n-2 + \cdots \quad (n-1) \text{ times}$$

$$S_2 = \frac{(n-1)(n-2)}{2}$$

$$T(n) = c_1 n + c_2 n - c_2 + c_3 n - c_3 + \frac{c_4 n^2}{2} - \frac{c_4 n}{2} + \frac{c_5 n^2}{2} - \frac{3c_5 n}{2} + \frac{2c_5}{n} + \frac{c_6 n^2}{2} - \frac{3c_6 n}{2} + \frac{2c_6}{2} + c_7 n - c_7$$



Q2. Implement the above algorithm using the programming language of your choice.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>

clock_t begin, end;
double time_;
void insertion_sort(long long data[], long long count)
{
   long long i, v, j;
   for (i = 1; i < count; i++)
   {
      v = data[i];
      j = i-1;
      while (j>=0 && data[j]>v)
```

```
        {
            data[j+1] = data[j];
            j = j-1;
        }
        data[j+1] = v;
    }
}
long long count(char file[])
{
            FILE *fp = fopen(file, "r");
            long long count = 0;
            char b[100];
            while(fscanf(fp, "%s\n", &b) == 1)
                                count++;
            fclose(fp);
            return count;
}
void insertion_sort_desc(long long data[], long long count)
{
    long long i, v, j;
    for (i = 1; i < count; i++)
    {
        v = data[i];
        j = i-1;
        while (j>=0 && data[j]<v)
```

```
        {
            data[j+1] = data[j];
            j = j-1;
        }
        data[j+1] = v;
    }
}
int main()
{
        long long j;
        long long n;
        long long *data;            //array to hold data
        int i;
        char filename[15];
        FILE *fp;
        printf("********************TIME SUMMARY***************\n");
        for(i=0;i<10;i++)
        {
                sprintf(filename, "File %d.txt", i+1);
                n = count(filename);
                printf("------------------------File %d.txt----------------------\n",i+1);
                printf("File %d has %lld elements\n",i+1,n);

                fp = fopen(filename, "r");
                data=(long long*)malloc(n*((long long)sizeof(long long)));
```

```
for(j=0; j<n; j++)
{
        fscanf(fp, "%lld", &data[j]);
}
begin= clock();
insertion_sort(data, n);
end = clock();
fclose(fp);
time_ = ((double)(end-begin)) / CLOCKS_PER_SEC;
printf("Average case : %0.10lf\n", time_);

sprintf(filename, "File %d_asc.txt", i+1);
fp = fopen(filename, "r");
for(j=0; j<n; j++)
{
        fscanf(fp, "%lld", &data[j]);
}
begin = clock();
insertion_sort(data, n);
end = clock();
fclose(fp);
time_= ((double)(end-begin)) / CLOCKS_PER_SEC;
printf("Best case  : %0.10lf\n",time_);

fp = fopen(filename, "r");
```

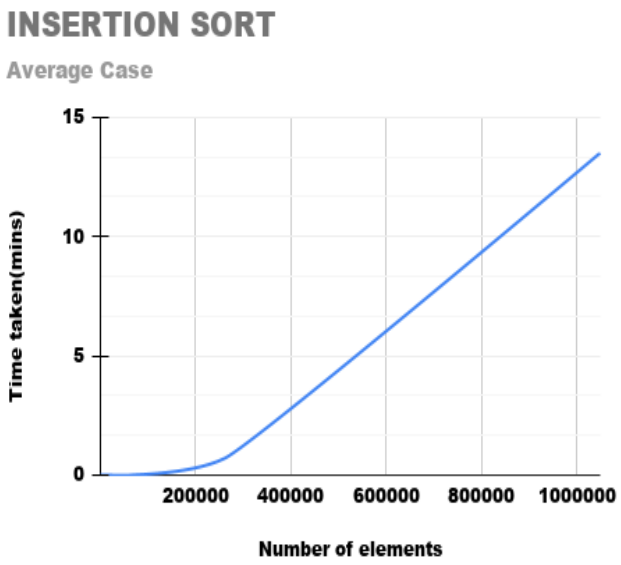
```
                for(j=n-1; j>=0; j--)
                {
                        fscanf(fp, "%lld", &data[j]);
                }
                begin = clock();
                insertion_sort_desc(data, n);
                end = clock();
                fclose(fp);
                time_ = ((double)(end-begin)) / CLOCKS_PER_SEC;
                printf("Worst case  %0.10lf\n\n", time_);
                free(data);
        }
}
```

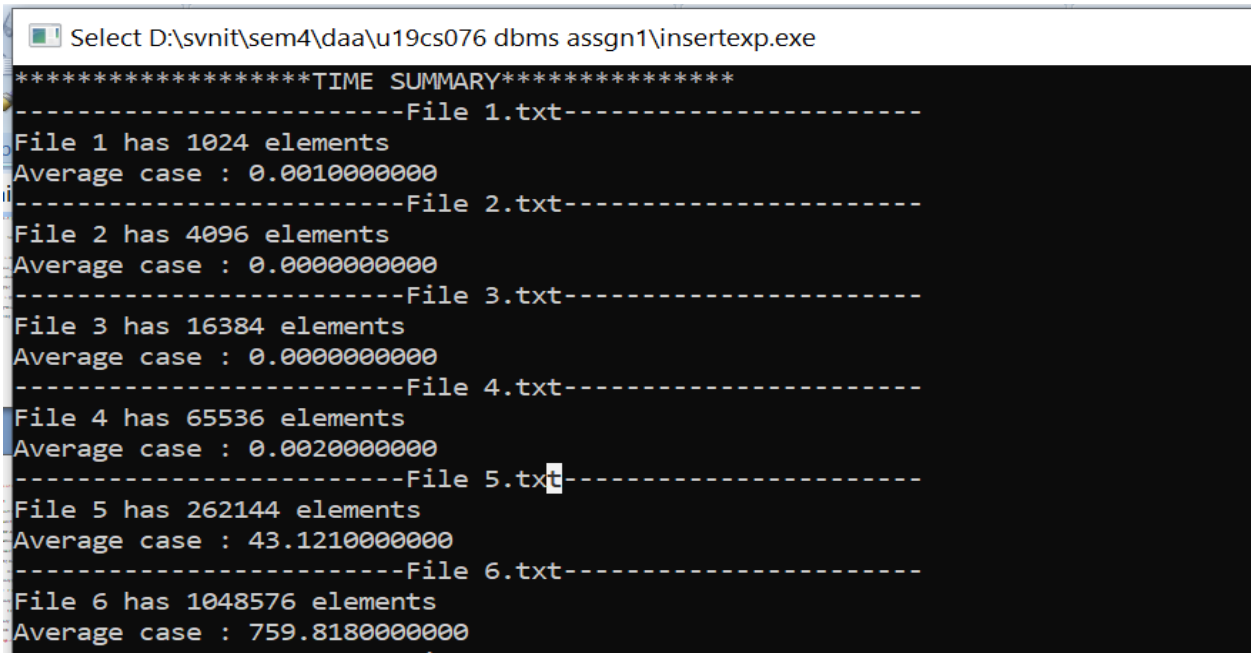
Q3.Provide the details of Hardware/Software you used to implement the algorithm and to measure the time.

Compiler	Dev C++ 5.11
OS Name	Microsoft Windows 10 Home (i5 8th Gen)
Version	10.0.19042 Build 19042
System Name	DESKTOP-BLE6CMQ
System Model	HP Pavilion x360 Convertible 14-ba1xx
System Type	x64-based PC
Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s)
BIOS Version/Date	Insyde F.54, 04-12-2019
Installed Physical Memory (RAM)	8.00 GB
Total Physical Memory	7.88 GB

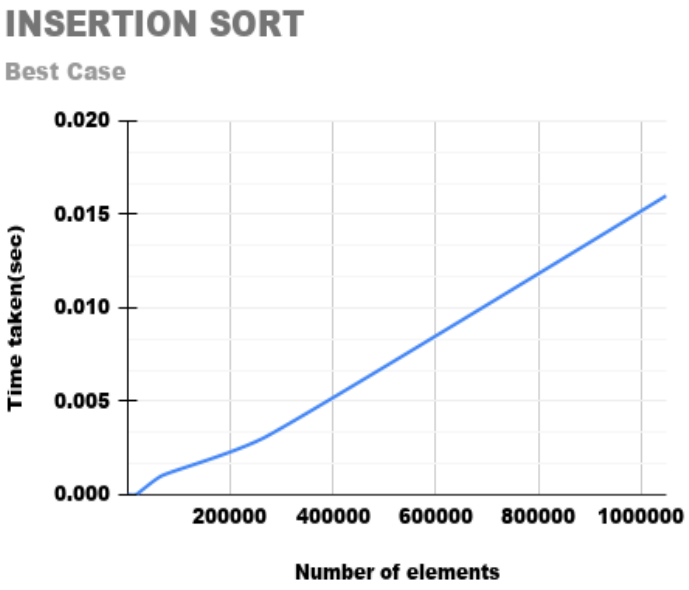
Available Physical Memory	1.75 GB
Total Virtual Memory	12.4 GB
Available Virtual Memory	4.59 GB
Page File Space	4.50 GB

Q5. Measure the average-case time (considering current data of ten files) of insertion sort for all ten files. Plot a graph.



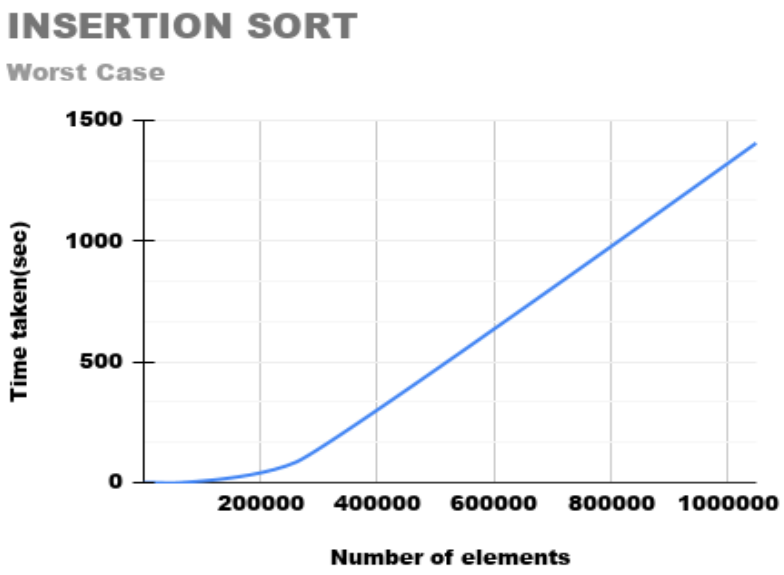


Q6. Measure the best-case time of insertion sort for all ten files. Plot a graph.




```
D:\svnit\sem4\daa\u19cs076 dbms assgn1\insertexp.exe
**********TIME SUMMARY***************
-------------------------File 1.txt----------------------
File 1 has 0 elements
Best case  : 0.0000000000
-------------------------File 2.txt----------------------
File 2 has 1024 elements
Best case  : 0.0000000000
-------------------------File 3.txt----------------------
File 3 has 4096 elements
Best case  : 0.0000000000
-------------------------File 4.txt----------------------
File 4 has 16384 elements
Best case  : 0.0010000000
-------------------------File 5.txt----------------------
File 5 has 65536 elements
Best case  : 0.0010000000
-------------------------File 6.txt----------------------
File 6 has 262144 elements
Best case  : 0.0010000000

--------------------------------
Process exited after 0.1169 seconds with return value 0
Press any key to continue . . .
```

Q7. Measure the worst-case time of insertion sort for all ten files. Plot a graph.

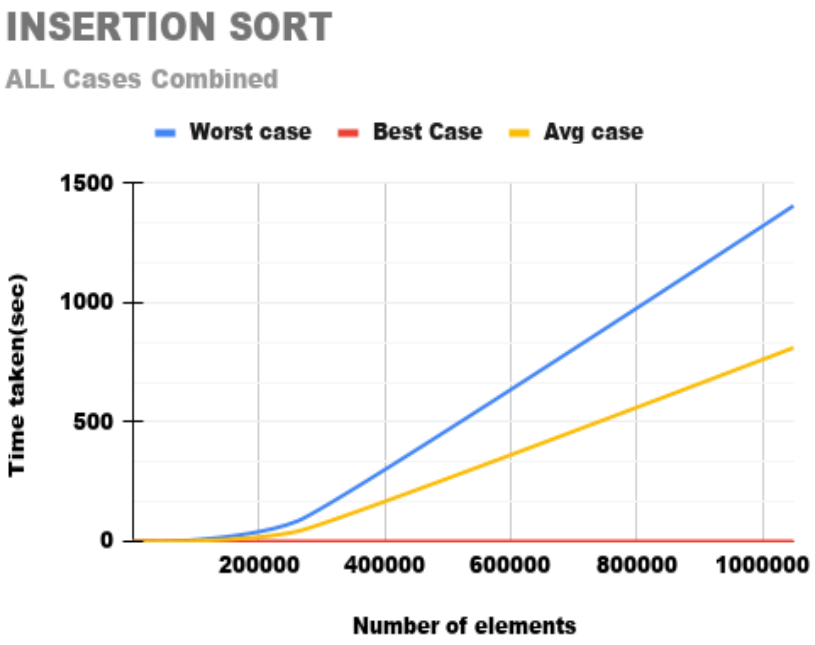



```
D:\svnit\sem4\daa\u19cs076 dbms assgn1\insertexp.exe
********************TIME SUMMARY***************
-------------------------File 1.txt-----------------------
File 1 has 1024 elements
Worst case  0.0020000000

-------------------------File 2.txt-----------------------
File 2 has 4096 elements
Worst case  0.0000000000

-------------------------File 3.txt-----------------------
File 3 has 16384 elements
Worst case  0.0020000000

-------------------------File 4.txt-----------------------
File 4 has 65536 elements
Worst case  0.0080000000

-------------------------File 5.txt-----------------------
File 5 has 262144 elements
Worst case  85.5310000000

-------------------------File 6.txt-----------------------
File 6 has 1048576 elements
Worst case  1444.6990000000
```

ALL CASES COMBINED OUTPUT

D:\svnit\sem4\daa\u19cs076 dbms assgn1\insertexp.exe

```
********************TIME SUMMARY***************
-------------------------File 1.txt-----------------------
File 1 has 1024 elements
Average case : 0.0010000000
Best case  : 0.000000
Worst case  0.0020000000

-------------------------File 2.txt-----------------------
File 2 has 4096 elements
Average case : 0.0000000000
Best case  : 0.000000
Worst case  0.0000000000

-------------------------File 3.txt-----------------------
File 3 has 16384 elements
Average case : 0.0000000000
Best case  : 0.000000
Worst case  0.0020000000

-------------------------File 4.txt-----------------------
File 4 has 65536 elements
Average case : 0.0060000000
Best case  : 0.002000
Worst case  0.0060000000

-------------------------File 5.txt-----------------------
File 5 has 262144 elements
Average case : 44.3700000000
Best case  : 0.000000
Worst case  99.4820000000

-------------------------File 6.txt-----------------------
File 6 has 1048576 elements
Average case : 720.4100000000
Best case  : 0.003000
Worst case  1444.6990000000
```



Q8. Assume that you don't know the time complexity of above algorithms.

8.1. Can you predict the same based on your implementation?

By observing Graphs of 3 cases we can conclude:

Best case is a Straight Line -> An+B

Average Case is a Parabolic graph -> An^2+Bn+C

Worst Case is a Parabolic graph -> An^2+Bn+C

So we can predict values if we know the input N by using extrapolating graphs.

8.2. Do they match with theoretical time complexity? Yes/No.

Yes, they would match with theoretical values too.

8.3. If yes, then write the time complexity of each algorithm. If no, then write the difference.

According to observations and graphs given above:

Time Complexity of Best Case-> An+B -> BIG THETA(n)

Time Complexity of Worst Case-> An^2+Bn+C -> BIG THETA(n^2)