**PROBLEM STATEMENT:** Draw a table comparing the time complexity of sorting algorithms for selection, insertion and bubble sort with respect to input data. Input data should be taken from a file and output result will be stored in output file. Programs should be written using dynamic memory allocation.

**Algorithm:**

STEP 1: Open the file from where data will be taken.

Repeat step 2 to step 4 as long as the user wants

STEP 2: Dynamically allocate space to store the required file data.

STEP 3: Store the data from the file to the allocated space.

STEP 4: Call the sort function on the dynamically allocated space.

End of loop.

Step 5: End.

Algorithm for sort function

Step 1: Start<-clock()

Step 2: Apply insertion sort on the array.

Step 3: Store the sorted array in a file.

Step 4: Display time required <-clock()-start.

Step 5: Start<-clock()

Step 6: Apply bubble sort on the array.

Step 7: Store the sorted array in a file.

Step 8: Display time required <-clock()-start.

Step 9: Start<-clock()

Step 10: Apply selction sort on the array.

Step 11: Store the sorted array in a file

Step 12: Display time required <-clock()-start.

Step 13: end.

**Program Code**

#include<stdio.h>

#include<time.h>

#include<malloc.h>

#include<stdlib.h>

void swap(int \*, int \*);

void insertionsort(int [], int );

void selectionsort(int [],int );

void bubblesort(int [], int );

void sorts(int [], int );

void display(int a[], int);

int main()

{

FILE \*fptr;

fptr = fopen("/home/dhiraj/input.txt","r");

if(fptr==NULL)

{

printf("not found");

exit(0);

}

int \*a,i,n,multi,j=1;

a=(int\*)malloc(sizeof(int));

printf("\n\nNAME OF ALGORITHM \t\t NO. OF DATA \t\t TIME REQUIRED \n\n");

do{

printf(" ENTER NUMBER OF DATA OR ZERO TO EXIT");

scanf(" %d", &n);

if(n!=0)

{

a=(int\*)realloc(a,n\*sizeof(int));

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

fscanf(fptr, "%d,", &a[i] );

sorts(a,n);

}

}while(n!=0);

fclose(fptr);

return 0;

}

void sorts(int a[], int n)

{

clock\_t start, end;

start= clock();

insertionsort(a,n);

end=clock();

start=end-start;

printf("INSERTION SORT\t\t\t %d \t\t\t %.\*f \n",n,7, (float)start/CLOCKS\_PER\_SEC);

display(a,n);

start=clock();

bubblesort(a,n);

start=clock()-start;

printf( "BUBBLE SORT\t\t\t %d \t\t\t %.\*f \n",n,7,(float)start/CLOCKS\_PER\_SEC);

display(a,n);

start=clock();

selectionsort(a,n);

start=clock()-start;

printf( "SELECTION SORT\t\t\t %d \t\t\t %.\*f \n",n,7 ,(float)start/CLOCKS\_PER\_SEC);

display(a,n);

}

void swap(int \*a , int \*b )

{

int temp= \*a;

\*a=\*b;

\*b=temp;

}

void bubblesort(int a[], int n)

{

int i,j, temp;

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

{

for(j=0;j<n-1-i;j++)

{

if(a[j]>a[j+1])

{ temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

}

void insertionsort(int a[], int n)

{

int i,j,temp;

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

{

j=i-1;

temp=a[j+1];

while((temp<a[j])&&(j>=0)){

a[j+1]=a[j];

j--;

}

a[j+1]=temp;

}

}

void selectionsort(int a[], int n)

{

int i,j,temp,k;

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

{ temp=i;

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

{

if(a[j]<a[temp])

temp=j;

}

k=a[temp];

a[temp]=a[i];

a[i]=k;

}

}

void display(int a[], int n)

{

FILE \*f;

f=fopen("/home/dhiraj/output.txt","a+");

int i;

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

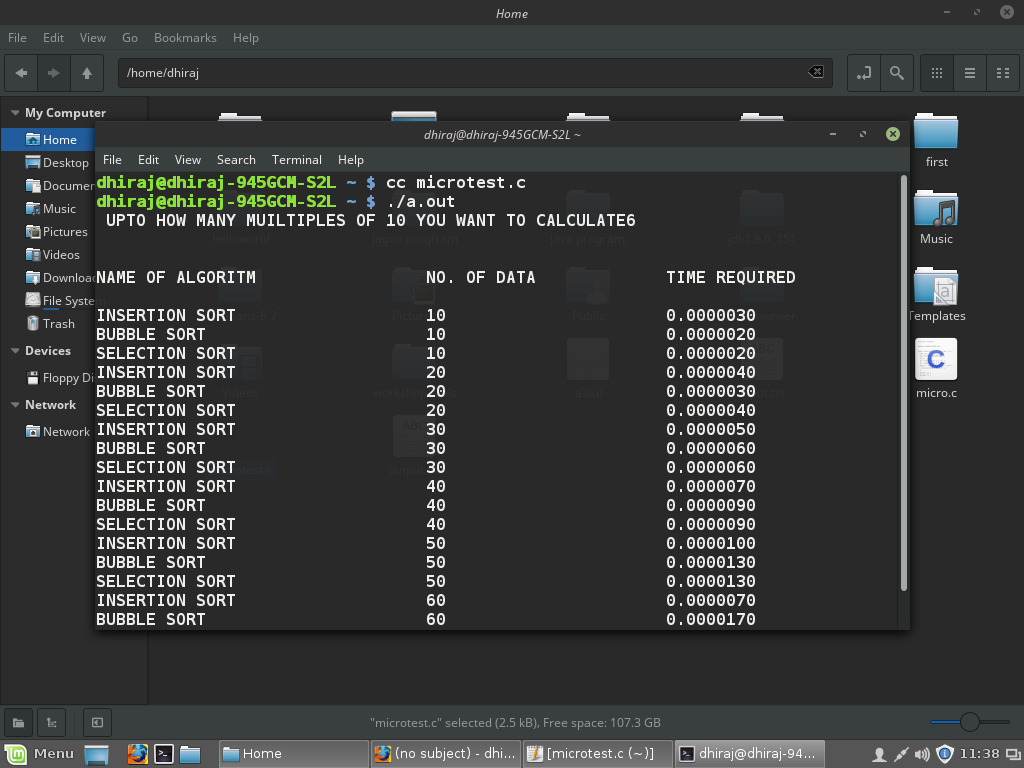
fprintf(f," %d,", a[i]);

fprintf(f,"\n \n \n");

fclose(f);

}

**OUTPUT**

****