**LAB 5- Selection Sort**

**AIM:** Sort a given set of N integer elements using **Selection Sort** technique and compute its time taken. Run the program for different values of N and record the time taken to sort.

Plot a graph of the time taken versus N  using MS Excel.  The program should allow both manual entry of the array elements and also reading of array elements using random number generator.  Note: In the record book students should

- Handwrite  the Algorithm

- Handwrite the Program

- Pasting of the printout of the Output and Graph or Handwriting of the Output and Graph.

Note: N value should be in the range

**ALGORITHM :** sel\_sort(a[0….n-1]

**//**Sorts a given array by selection sort

**//**Input : An array a[0….n-1] of orderable elements

//Output : Array a[0….n-1] sorted in ascending order

**for** i🡨0 to n-2 **do**

       small\_pos🡨i

**for** j🡨i+1 to n-1 **do**

**if** a[j]<a[small\_pos]

             small\_pos🡨j

**end if**

**end for**

    swap a[i] and a[small\_pos]

**end for**

**Program:**

#include<stdio.h>

#include<time.h>

#include<stdlib.h> /\* To recognise exit function when compiling with gcc\*/

void selsort(int n,int a[]);

void main()

{

   int a[15000],n,i,j,ch,temp;

   clock\_t start,end;

   while(1)

   {

 printf("\n1:For manual entry of N value and array elements");

 printf("\n2:To display time taken for sorting number of elements N in the range 500 to 14500");

 printf("\n3:To exit");

     printf("\nEnter your choice:");

     scanf("%d", &ch);

     switch(ch)

     {

       case 1:  printf("\nEnter the number of elements: ");

scanf("%d",&n);

printf("\nEnter array elements: ");

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

{

  scanf("%d",&a[i]);

}

start=clock();

selsort(n,a);

end=clock();

printf("\nSorted array is: ");

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

printf("%d\t",a[i]);

printf("\n Time taken to sort %d numbers is %f Secs",n, (((double)(end-start))/CLOCKS\_PER\_SEC));

break;

     case 2:

      n=500;

      while(n<=14500) {

      for(i=0;i<n;i++)

  {

    //a[i]=random(1000);

    a[i]=n-i;

  }

      start=clock();

      selsort(n,a);

          //Dummy loop to create delay

  for(j=0;j<500000;j++){ temp=38/600;}

              end=clock();

printf("\n Time taken to sort %d numbers is %f Secs",n, (((double)(end-start))/CLOCKS\_PER\_SEC));

            n=n+1000;

  }

      break;

   case 3: exit(0);

   }

   getchar();

    }

}

void selsort(int n,int a[])

{

     int i,j,t,small,pos;

     for(i=0;i<n-1;i++)

      {

       pos=i;

       small=a[i];

       for(j=i+1;j<n;j++)

       {

  if(a[j]<small)

  {

    small=a[j];

    pos=j;

  }

       }

       t=a[i];

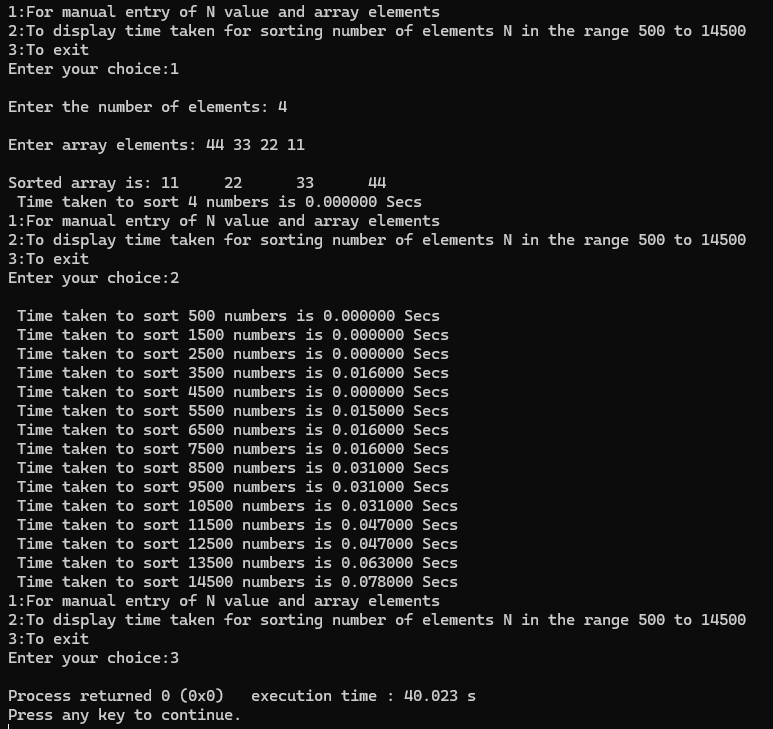
       a[i]=a[pos];

       a[pos]=t;

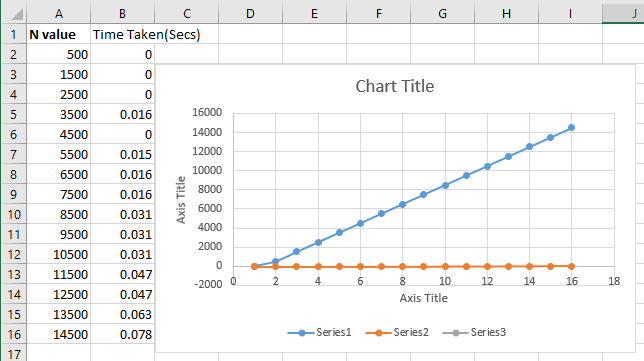
    }

}

**Output:**



**Graph Screenshot:**



**Merge Sort**

**AIM:** Sort a given set of N integer elements using **Merge Sort** technique and compute its time taken. Run the program for different values of N and record the time taken to sort.  Plot a graph of the time taken versus N using MS Excel.  The program should allow both manual entry of the array elements and also reading of array elements using random number generator.

Note: In the record book students should

- Handwrite  the Algorithm,

- Handwrite the Program

- Pasting of the printout of the Output and Graph or handwriting of the Output and Graph.

**ALGORITHM** **:** combine(a[0….n-1],low,mid,high)

//merge two sorted arrays where first array starts from *low* to *mid* and second starts from *mid+*1 to *high*

//Input : ***a*** is a sorted array from index position low to mid

//            ***a*** is a sorted array from index position mid+1 to high

//Output: Array a[0….n-1] sorted in nondecreasing order

i🡨low

j🡨mid+1

k🡨low

**while** i<=mid and j<=high **do**

**if** a[i]<a[j]

          c[k]🡨a[i]

           k🡨k+1

           i🡨i+1

**else**

          c[k]🡨a[j]

          k🡨k+1

          j🡨j+1

**end if**

**end while**

**if** i>mid

**while** j<=high **do**

          c[k]🡨a[j]

          k🡨k+1

          j🡨j+1

**end while**

**end if**

**if** j>high

**while** i<=mid **do**

          c[k]🡨a[i]

          k🡨k+1

          i🡨i+1

**end while**

**end if**

**for** i🡨low to high **do**

     a[i]🡨c[i]

**end for**

**ALGORITHM**: split(a[0….n-1],low,high)

//Sorts array a[0….n-1] by recursive mergesort

//Input :An array a[0….n-1] of orderable elements

//Output : Array a[0….n-1] sorted in nondecreasing order

**if** low<high

    mid🡨(low+high)/2

    split(a,low,mid)

    split(a,mid+1,high)

    combine(a,low,mid,high)

**end if**

**Program:**

#include<stdio.h>

#include<time.h>

#include<stdlib.h> /\* To recognise exit function when compiling with gcc\*/

void split(int[],int,int);

void combine(int[],int,int,int);

void main()

{

   int a[15000],n, i,j,ch, temp;

   clock\_t start,end;

   while(1)

   {

 printf("\n1:For manual entry of N value and array elements");

 printf("\n2:To display time taken for sorting number of elements N in the range 500 to 14500");

 printf("\n3:To exit");

     printf("\nEnter your choice:");

     scanf("%d", &ch);

     switch(ch)

     {

       case 1:  printf("\nEnter the number of elements: ");

scanf("%d",&n);

printf("\nEnter array elements: ");

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

{

  scanf("%d",&a[i]);

}

start=clock();

split(a,0,n-1);

end=clock();

printf("\nSorted array is: ");

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

printf("%d\t",a[i]);

printf("\n Time taken to sort %d numbers is %f Secs",n, (((double)(end-start))/CLOCKS\_PER\_SEC));

break;

     case 2:

      n=500;

      while(n<=14500) {

      for(i=0;i<n;i++)

  {

    //a[i]=random(1000);

    a[i]=n-i;

  }

      start=clock();

      split(a,0,n-1);

           //Dummy loop to create delay

  for(j=0;j<500000;j++){ temp=38/600;}

      end=clock();

printf("\n Time taken to sort %d numbers is %f Secs",n, (((double)(end-start))/CLOCKS\_PER\_SEC));

            n=n+1000;

  }

      break;

   case 3: exit(0);

   }

   getchar();

    }

}

void split(int a[],int low,int high)

{

 int mid;

 if(low<high)

 {

  mid=(low+high)/2;

  split(a,low,mid);

  split(a,mid+1,high);

  combine(a,low,mid,high);

 }

}

void combine(int a[],int low,int mid,int high)

{

 int c[15000],i,j,k;

 i=k=low;

 j=mid+1;

 while(i<=mid&&j<=high)

 {

  if(a[i]<a[j])

  {

   c[k]=a[i];

   ++k;

   ++i;

  }

  else

  {

   c[k]=a[j];

   ++k;

   ++j;

  }

 }

 if(i>mid)

 {

  while(j<=high)

  {

   c[k]=a[j];

   ++k;

   ++j;

  }

 }

 if(j>high)

 {

  while(i<=mid)

  {

   c[k]=a[i];

   ++k;

   ++i;

  }

 }

 for(i=low;i<=high;i++)

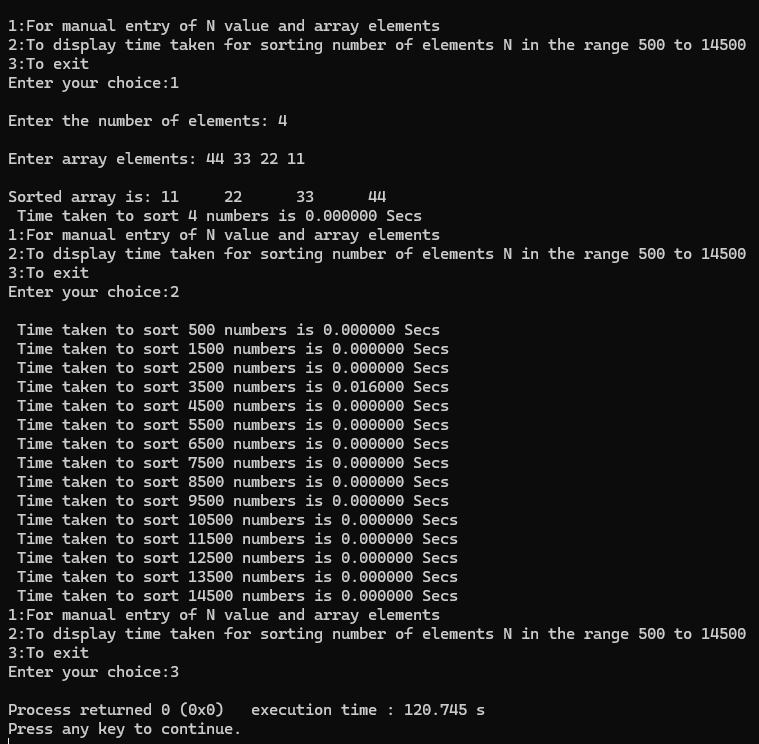
 {

  a[i]=c[i];

 }

}

**Output:**



**Graph Scree shot: It can be observed from the graph below that time taken by Selection sort is more when compared to Merge sort.**

