A NEW SORTING PROGRAM

Stephen.P
1JB10CS094
IV SEM 'B' CSE
SJBIT

I am overwhelmed that I have designed a new sorting program.

- I learned that sorting technique as 'CONVERGING SORT" which is a recursive selection sort.

Here is the pseudo-code :

```
- Algorithm Rec_sort()
// Purpose : To sort the given array of elements
             in ascending order.
// Input : The array elements with initial (low)
           and final indices (high)
// Output : The sorted array.
    {  if (low < high)
       {  select (a, low, high);
          sort (a, low+1, high-1);
       end if
end.  }
```

Algorithm Select( )

// Purpose : To select the least and max. element in the array, and to swap them as shown

// Input : The array (a) with low and high indices.

// Output : The array with the least index having least element and the highest index having highest element.

assign min = low
max = low

for (i = low + 1; i <= high; i++)
if (a[min] > a[i]) min = i;
end for
swap (a[min], a[low]);
for (i = low + 1; i <= high; i++)
if (a[max] < a[i]) max = i;
end for
swap (a[max], a[high]);

end.

Here is the program in C */

```
#include<stdio.h>
#include<conio.h>
void select(int a[], int low, int high)
{ int i, min, max;
  min = low;
  max = low;

  for(i=low+1; i<=high; i++)
      if(a[min] > a[i]) min=i;
  i=a[low];      /* Swapping a[min] and a[low] */
  a[low]=a[min];
  a[min]=i;

  for(i=low+1; i<=high; i++)
      if(a[max] < a[i]) max=i;
  i=a[max];      /* Swapping a[max] with a[high] */
  a[max]=a[high];
  a[high]=a[i];

  return;
}

void sort(int a[], int low, int high)
{ if(low < high)
  { select(a, low, high);
    sort(a, low+1, high-1);
  }
}
```

```
int main()
{  int a[20], n;
   int i;
   printf("Enter the number of elements: ");
   scanf("%d", &n);
   printf("Enter the elements: \n");
   for(i=0; i<n; i++)
      scanf("%d", &a[i]);
   sort(a, 0, n-1);
   printf("\n\n The sorted elements : ");
   for(i=0; i<n; i++)
      printf("%d\t", a[i]);
   getch();
}
```

ANALYSIS FOR TIME COMPLEXITY :- (TRIED)

- The array 'a', low and high indices are the inputs.
- select(a, low+1, high-1) is the basic operation executed for more number of times.
- It depends only on the size of the input array.
- Mathematically, it can be represented as

$$T(n) = \begin{cases} 0 & \text{if } !(low < high) \\ T(n-2) + C_n & \text{otherwise} \end{cases}$$

$T(n-2)$ – Time required to sort $n-2$ elements.

C_n – Time required to select and swap the least and highest elements with the lowth and highth position of the array.

$$T(n) = T(n-2) + C_n$$

$$= T(n-4) + 2C_n \qquad | \; T(n-2) = T(n-4) + C_n$$

$$T(n) = T(n-6) + 3C_n \qquad | \; T(n-4) = T(n-6) + C_n$$

After k substitutions,

$$T(n) = T(n-2k) + kC_n$$

when $k = \frac{n}{2}$,

$$T(n) = T(n-n) + \frac{n}{2}(n)C$$

$$= T(0) + \boxed{\frac{n^2}{2}C}$$

$$\boxed{T(n) = \Theta(n^2)}$$

ConsoleProject | (Global Scope) | swap(int i, int j)

```c
#include <stdio.h>

int ar[8] = { 8, 7, 6, 5, 4, 3, 2, 1};

void swap(int i, int j)
{
    if (i != j && ar[i] != ar[j])
    {
        int temp;
        temp = ar[i];
        ar[i] = ar[j];
        ar[j] = temp;
    }
}

void SortEnds (int low, int high)
{
    int nLow = low, nHigh = high;
    for (int k = low; k <= high; k++)
    {
        if (ar[k] < ar[nLow])
            nLow = k;
        if (ar[k] > ar[nHigh])
            nHigh = k;
    }

    swap(nLow, low);

    if (low == nHigh) swap(nLow, high);
    else if (nLow == high) swap(nLow, nHigh);
    else swap(nHigh, high);
}
```

Miscellaneous Files | (Global Scope)

```c

void ConvergingSort (int low, int high)
{
    if (low <= high)
    {
        SortEnds(low, high);
        ConvergingSort(low + 1, high - 1);
    }
}

void print()
{
    for (int i = 0; i < sizeof(ar) / sizeof(ar[0]); i++)
        printf("%d    ", ar[i]);
    printf("\n");
}

int main()
{
    ConvergingSort(0, sizeof(ar) / sizeof(ar[0]) - 1);
    print();
    return 0;
}

```