DSA- Assignment-6

P. Manoj
AP19110010054
CSE-G1

①

```
#include <stdio.h>
int main()
{
int i, low, high, mid, n, key, arr[100], temp, i, one, two,
                                        sum, Product;
Printf("Enter the number of elements in array")
Scanf("%d", &n);
 Printf("Enter %d intugers.", n);
 for(i=0; i<n; i++)
Scanf("%d", &arr[i]);
 for(i=0; i<n; i++)
 {
  if(i=i+1; j<n; j++)
  {
   if(arr[i]<arr[j])
  {
   if(arr[i]<arr[j])
  {
   if(temp=arr[j]);
    { arr[i]=arr[j];
     arr[j]=temp;
    }
```

```
    }
}
Printf("In elements of array is sorted in
                                    descending order
                                    : \n");
for (i=0; i<n; i++)
{
  Printf("%d", arr[i]);
}
Printf("Enter value to find");
Scanf("%d", &key);
low=0
high=n-1;
mid=(low+high)/2;
while (low<=high)
{
  if (arr[mid]>key)
  {
    low = mid+1;
  }
  else if (arr[mid]==key)
  {
    Printf("%d found at location
                    ", key, mid+1);
    break;
  }
  else
    high = mid-1;
    mid = (low+high)/2;
}
```

```
if (low > high)
{
  Printf("Not found %d is not present in list.\n",
                                        key);
}
Printf("\n");
Printf("Enter two locations to find sum and
                                Product of element");
scanf("%d", &one);
scanf("%d", &two);
sum = (arr[one] + arr[two]);
Product = (arr[one] * arr[two]);
Printf("The sum of elements = %d", sum);
Printf("The Product of elements = %d", Product);
return 0;
}
```

output :-

Enter number of elements in array 5

Enter 5 integer

9
7
5
4
2

Element of array is sorted in descending order.

Enter value to find :
5 found at location 3
Enter two locations to find sum and
Product of elements
2
4
The sum of elements = 89
The product of elements = 10

(2)

```
# include <stdio.h>
# include <conio.h>
# define MAX_SIZE 5
void merge_sort [MAX_size];
void merge_array (int, int, int, int);
int arr_sort [MAX_SIZE];
int main[]
{ int i, k, Pro = 1;
Printf ("Sample merge sort example functions
                and array \n");
Printf ("\n Enter %d Elements for sorting
                \n", MAX_
                SIZE);
for (i=0 ; i<MAX_SIZE ; i++)
{
```

```
Printf ("\t%d", arr-sort[i]);
}
merge-sort(0, max-size-1);
Printf ("\n sorted data:");
for (i=0; i<MAX-SIZE; i++)
{
Printf ("%d," arr-sort[i]);
}
Printf (" find the Product of kth element
                from first and last k\n");
Scanf ("%d", &k);
Pro = arr-sort[k] * arr-sort[MAX-SIZE-k-1];
Printf ("Product = %d", Pro);
getch();
}
void merge-sort(int i, int j)
{ int m;
  if (i<j)
  { m=(i+j)/2;
    merge-sort(i,m);
    merge-sort(m+1, j);
    // merging two arrays.
    merge-array(i, m, m+1, j);
```

```
}
void merge_array (int a, int b, int c, int d)
{
  int t[50];
  int i=a, j=c, k=0;
  while (i<=b && j<=d)
  {
    if (arr_sd+[i]<arr_sd+[j])
      t[k++]= arr_sd+[i++];
    else
      t[k++]= arr_sd+[j++];
  }
  // collect remaining elements:
  while (i<=b)
    t[k++]= arr_sd+[i++];
  for [i=a, j=a, i<=d; i++; j++)
    arr_sd+[i]= t[j];
}
```

output:

Enter 5 elements for sorting

9
7
4
6
2

your data: 9 7 4 6 2

sorted data: 2 4 6 7 9

Product = 36.

③ Insertion sort:

Insertion sort works by inserting the set of values in existing sorted file. It constructs the sorted array by inserting a single element at a time. This process continues untill whole array is sorted in same order. The primary concept behind insertion sort is to insert each item into its appropriate place in final list.

working of insertion sort

→ It uses two sets of arrays where one stores the sorted data and unsorted data.

→ The sorting algorithm works untill there are elements in unsorted set.

→ The first elements of unsorted portion has array index (if LB=0)

→ After each iteration, it chooses the first element of unsorted position and inserts into proper place in sorted set.

Advantage of Insertion sort:

→ Easily implimented and very efficient

when used with small sets of data.

→ It is considered to be fine sorting technique as the list can be sorted as new elements are received.

→ It is faster than other sorting elements.

Complexity of Insertion Sort:-

The best case complexity of insertion sort is O(n) times ie when array is previously sorted in the same way. when array is sorted in reverse order, the first element in unsorted array is to be compared with each element in the sorted set.

Example:

arr[] = 46 22 11 20 9

// find minimum element in arr[0 ... 4] and place at begining

9 46 22 11 20

// Find the minimum element in arr [1 ... 4] and place at begining of arr [1 ..... 4]

9 11 46 22 20.

Selection Sort:-

The selection sort performs sorting by searching for minimum value numbers and placing it into the first & last position according to the order. The process of searching the minimum key and placing it in the proper position is continuous until all elements are placed.

working of selection sort:-

→ Suppose an array Arr with n elements in memory.

→ In the first pass, then the Arr[Pos] is supposed and swapped with in Arr[0]. then for Arr[0] is sorted.

→ In the pass (n-1) the same process is performed to sort the n number of elements.

Advantages of selection sort:-

→ The main advantage of selection sort is that is performs well on a small list

→ Further more, because it is in place sorting algorithm no additional storage is required.

Complexity of selection Sort:-

As the working of selection sort doesn't depend on Signal order of elements in the array. So there is not much difference between best case and worst case complexity of selection sort. the selection sort selects the minimum value element in the selection process. At the 'n' number of elements are scanned there for n-1 comparision are made in first pass. Thus running time complexity of selection sort is $O(n^2) = (n-1)+(n-2)+\ldots$

$+2+1 = n(n-1)/2$

$= O(n^2)$

Example:-

13 12 14 6 7

let us loop for i = 1 (second element of array) to u (last element of array)

i = 1 since 12 is smaller than 13, move 13 and insert 12 before 13

do same for i = 2, i = 3, i = 4

sorted array.

6 7 12 13 14.

④

```
# include <stdio.h>
# include <conio.h>
int main()
{
int arr[50], i, j, n, temp, sum=0, Product=1;
Printf("Enter total number of elements to
store:")
scanf("%d", &n);
Printf("Enter %d elements:", n);
for(i=0; i<n; i++)
scanf("%d", &arr[i]);
Printf("In sorting array using bubble sort
                technique In");
for(i=0; i<(n-1); i++);
{
  for(j=0; j<(n-1-i); j++)
  {
    if arr[j] > arr[j+1]
    {
      temp = arr[j];
      arr[j] = arr[j+1]
      arr[j+1] = temp;
    }
  }
}
```

```
Printf ("All array elements stored \n");
Printf (" Array elements in ascending order:
                                    \n \n");
for (i=0; i<n; i++)
{
  Printf ("%d \n", arr[i]);
}
Printf ("array elements in alternate order
                                    \n");
for ( i=0 ; i<=n ; i= i+2)
{
  Printf ("%d \n", arr[i]);
}
 for (i=1 ; i<=n ; i= i+2)
{
 sum = sum + arr[i] ;
}
 Printf ("the sum of odd position elements
                arr = %d \n", sum);
 for (i=0 ; i<=n ; i=i+2)
 {
   Product = arr[i];
 }
   Printf ("the Product of even
                Position elements are
                = %d \n", Product);
 getch();
 return 0 ();
}
```

Output:

Enter 5 elements

8
6
4
3
2

Sorting array using bubble sort technique

All array elements sorted successfully.

elements in ascending order.

2
3
4
6
8

array elements in alternate order

2
4
8

The sum of odd position element is 9

Product of even position element are 6, 4.

⑤

```
# include <stdio.h>
# include <stdio.h>
void binary search (int, arr[], int num, int
                    first, int last)
{
   int mid;
   if (first > last)
     {
```

```
Printf(" Number is not found ");
}
else
{
  mid= (first + last) /2;
}
 if (arr [mid]==num)
{
 Printf(" Element is found at index
                          %d", mid);
 exit (0);
}
  else if (arr [mid]>num)
{
   Primary search (arr, num , first mid-1);
}
else
{
   Binary search ( arr : num; mid+1, last);
}
}

void main(){
  int arr[100], beg, mid, end, i, n, num;
Printf (" Enter size of an array:")
scanf (" %d ", &n);
```

```
Printf ("Enter the value in sorted sequence\n");
for (i=0; i<n; i++)
{
    scanf ("%d", &arr[i]);
}
    beg=0
    end= n-1;
    Printf ("Enter a value to be search: ");
    scanf ("%d", &num);
    Binary search (arr, num, beg, end);
}
```

Output:-

```
Enter size of an array 5
enter the value in sorted sequence
4
5
6
7
8
Enter a value to search: 5
Element is found at index: 1
```