~~Chemi~~

DSA - Assignment-6

D.Tarun Venkat Sai
AP19110010504
CSE-H

1) Take the elements from the user and sort them in decenting order and do the following.

a) using Binary Search find the element and the location in the array where the element is asked from user.

b) Ask the user to enter any two locations print the sum and product of values at those locations in the sorted array.

Sol:
```
# 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 integers," n);
 for (i=0; i<n; i++)
 Scanf ("%d", &arr[i]);
 for (i=0; i<n; i++)
 {
  if(j=i+1; j<n; j++)
 {
   if(arr[i] <arr[j])
   { if (temp = arr[i]);
   { arr[i]= arr[j];
     arr[j]= temp;
   }
  }
 }
```

```
Printf("\n elements of array is sorted in decending 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 %d", key, mid+1);
    break;
  }
   else
   high=mid-1;
    mid=(low +high/2 ;
  }

 if (low >high)
{
 Printf("Not found! %d isn't present in list\n", key);
}
Printf ("/n");
Printf("Enter two locations to find sum
       and product of the elements")
```

Scanned with CamScanner

```
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 integers

2

4

5

7

9

Element of array is sorted in ascending order

2 4 5 7 9 Enter value to find 7

7 found in location 4

Enter two locations to find sum and Product

4

7

Sum = 14

Product = 45

(2) Solve the array using merge sort where elements are taken from the product of the K^th elements from first and last where K is taken from the user.

Sol:
```
#include <stdio.h>
#include <conio.h>
#define MAX_Size 7
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++)
{scanf("%d", &arr_sort[i]);
 Printf("\n your data:");
}
for (i=0, i<MAX-SIZE; i++
{
 Printf("\t%d", arr_sort[i]);
}
merge-sort(0, Max_size-1);
Printf("\nSorted data:");
for (i=0; i<MAX-SIZE; i++)
{
 Printf("\t%d", arr_sort[i]);
}
```

```
Printf("Find the product of the ~~kth~~ Kth element
            from first & last where K \n");
Scanf("%d", &K);
Pro = arr_sort [K]* arr_sort [MAX_SIZE-K-1];
Printf (" Produce = %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~~ int d)
{
  int t[50];
  int i=a, j=c, K=0;
  while (i<b&& j<=d)
 {if (arr_sort[i] <arr_sort [j])
     ~~t[K++]=a~~
     t [K++]= arr_sort [i++];
  else
     t[K++]= arr_sort[j++];
 }
```

```
// collect remaining elements.
while (i<=b)
    t[k++] = arr_sort[j++];
for [i=a, j=a; i<=d; i++, j++)
   arr_sort[i] = t[j];
}
```

<u>output:</u>

Sample merge sort example functions and array.

Enter 5 elements for sorting.

2

4

6

8

9

your data: 2 4 6 8 9

Sorted data: 2 4 6 8 9

Find the product of Kth elements from first and last where K = 4

Product = 81

③ Discuss insertion sort and selection sort with examples.

Sol: Insertion sort :-

Insertion sort works by inserting the set of values in the existing sorted file. It constructs the sorted array by inserting a single element at a ~~sorted~~ time. This process continues until whole array is sorted in same order. The primary concept behind insertion sort is to insert each item into its appreciate place in the final list. The insertion sort method saves an effective amount of memory.

Working of insertion sort

-> It uses two sets of arrays where one stores the sorted data and other on unsorted data.

-> ~~Let~~ The sorting algorithm works until there are elements in the unsorted set.

-> Lets assume there are 'n' numbers elements in the array. Initially, the element with index 0 (LB=0) exists in the sorted set remaining elements are in the unsorted partition of the list.

-> The first element of the unsorted portion has array index 1 (if LB=0)

-> After each interaction, it chooses the first element of the insorted position and inserts it into the proper place in the sorted set.

Advantages of Insertion Sort:-

-> Easily implemented and very efficient when used with small sets of data.

-> The additional memory space requirement of insertion sort is less [i.e, (O(1))].

-> It is considered to be line sorting tech-niques as the list can be sorted as the new elements are recieved

-> It is faster than other sorting algorithms.

Complexity of Insertion Sort:-

The best case complexity of insertion sort is O(n) times, i.e. when the array is previously sorted. In the same way, when the array is sorted in the reverse order, the first element in the unsorted array is to be composed

with each element in the sorted set.
So, in the worst case, running time of insertion sort is quadratic, i.e $O(n^2)$. In average case also it has to make the minimum $(k-1)/2$ Comparisions.
Hence, the average case also has quadratic running time $O(n^2)$.

Example:-

arr[ ] = 46 22 11 20 9

→ Find the 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

→ Find the minimum element in arr[2---4] and

9 11 20 46 22.

→ Find the minimum element in the array a[3----4] and insert at the begining of the array [3.----4]

∴ Sorted array

9 11 20 22 46.

Selection sort:

The It perform sorting by searching for the minimum value number and Placing it into the first or last

Position accordingly to the order. The process of searching the minimum key and placing it in the proper position is continued until all the elements are placed at right position.

Working of the selection sort:

- Suppose an array Arr with n elements in the memory.
- In the first pass, the smallest key is searched along with its position, then the Arr [Pos] is supposed swapped with Arr [0]. Therefore Arr [0] is sorted.
- In the second pass, again the position of the smallest value is determined in the sub array of (n-1) elements interchange the Arr [Pos] with Arr [1]
- In the pass (n-1), the same process is performed to sort the n number of elements.

Advantages of selection sort:

- The main advantages of selection sort is that is performs well on a small list.
- Furthermore, because it is an in-place sorting algorithm. no additional temporary

Storage is required beyond what is needed to hold the original list.

Complexity of Selection sort

As the working of selection sort does not depend on the original orders of the elements in the array. So there is not much difference b/w best case and worst case complexity of selection sort. The selection selects the minimum value element, in the selection process. At the 'n' number of elements are scanned, i.e. n-1 comparisions are made in the first pass. then, the elements are interchanged. Similarly in the second pass also to find the second smalled element we require scanning of rest n-elements and the process is continued till the whole array sorted. Thus running time complexity of selection sort is:

$O(n^2) = (n-1)+(n-2)+\dots+2+1 = n(n-1)/2 = O(n^2)$

Example:

13 12 14 6 7

let us loop for i=1 (second element of the array) to 4 (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) Sort the array using bubble sort. where elements are taken from the user and display the elements.

(i) in alternate order.

(ii) Sum of elements in odd positions and Products of elements in even positions

(iii) elements which are divisible by m where m is ~~&~~ taken from user.

Sol:

```
#include <stdio.h>
#include <conio.h>
int main()
{
 int arr[50], i,j, n, temp, sum=0, Product=1;
 Printf("Enter total no.of elements to store:");
 Scanf(" %d", &n);
 Printf("Enter %d elements:", n);
 for (i=0; i<n; i++){
 Scanf ("%d", & arr[i]);
 Printf ("\n Sorting array using bubble technique\n");
 for (i=0; i<(n-1); i++);
 {
 for (j=0; j<(n-i-1); j++)
 {  if (arr [j] > arr[j+1]
   {
```

```
        temp = arr [j];
         arr [i] = arr [j+1]
          arr [j+1] = temp;
        }
    }
 }

Printf ("All array elements sorted Successfully:\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)
   {
     Perint f("%d "\n", arr [i]);
   }
   for (i=1; i<=n; i= i+2)
   {
     Sum = Sum + arr [i];
   }
  Printf( "The sum of odd position elements are = %d \n", sum);
  for (i=0, i<=n, i=i+1)
 {
   Product *= arr [i];
 }
 Printf ("The product of even position elements
                                 are = %d \n", product );
  getch ();
  return 0();
 }
```

Output:

Enter total number of elements to store: 5

Enter 5 elements

6

4

7

2

3

Sorting array using bubble sort technique

All array elements sorted successfully

Array elements in ascending order.

2

3

4

6

7

array elements in alternate order

2

4

7.

The sum of odd position element is 10

The Product of event " " " 56

(5) Write a Recursive program to implement binary Search?

Sol:

```
#include <stdio.h>
#include <conio.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 the size of an array");
  Scanf("%d",&n);
```

```
Printf(" Enter the values 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);
}
```

Outputs:

Enter the Size of an array
Enter the values in Sorted Sequence
4
5
6
7
8
Enter a value to search: 6
Element is found at index: 2