Assignment - 6 [DSA]
(Searching and sorting)

Jaya Harshith Ambati
AP19110010515
CSE - H

D. Take the elements from the user and sort them in descending 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.

Solution:

```c
#include <stdio.h>
int main
{
    int i, low, high, mid, n, key, arr[100], temp, one, two, sum, product;
    printf("Enter 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("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)
```

```c
        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
                            the list", key);
    }
    printf(" \n");
    printf("Enter two locations to find sum and
                        product of elements");

    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 of the program:-

Enter number of elements in array = 5
Enter 5 integers

10
9
8
7
6

Element of array is sorted in descending order

10 9 8 7 6 Enter value to find = 8

8 found at location 2

Enter two locations to find sum and product of elements

2
4

The sum of elements = 14
The product of elements = 48

2). Sort the array using merge sort where elements are taken from the user and find the product of Kth elements from first and last where K is taken from the user.

Solution:-

```c
#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 ("Simple merge sort example functions of array\n");
    printf ("\nEnter %d elements for sorting\n", max_size);

    for (i=0; i< max_size; i++)
    {
        scanf ("%d", & arr_sort [i]);
    }
    printf (" In your data = ");
    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 ("\t %d", arr_sort [i]);
    }
}
```

```c
printf ("Find the product of the k^{th} element
                  from first & last where 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_sort[i] < arr_sort[j])
            t[K++] = arr_sort[i++];
        else
            t[K++] = arr_sort[j++];
    }
}
```

```
while (i <= lr)
        t[K++] = arr_sort [s++];
    for (i=a, s=0; i <= d; i++, s++)
        arr_sort [i] = t[s];
}
```

output of the program :-

Simple merge sort example. functions & array

Enter 5 elements for sorting

5
4
3
1
2

Your data = 5 4 3 1 2

Sorted data = 1 2 3 4 5

Find the product of Kth elements from first and last where K=2

Product = 10

③. Discuss insertion sort & selection sort with examples.

Solution:

→ Definition of 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 time. This process continues until while array is sorted in some order.

(*) The primary Concept behind insertion sort is to insert each item into its appropriate place in the final list. The insertion sort method saves an effective amount of memory.

→ Working of the insertion sort:

(*) It uses two sets of arrays where one stores the sorted data and other on unsorted data.

(*) The sorting algorithm works until there are elements in the unsorted set.

(*) Let's assume there are 'n' number elements in the array. Initially, the element with inter 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 iteration, it chooses the first element of the unsorted partition 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 live sorting technique 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 reverse order, the first element of the unsorted array is to be compared 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 $(K-1)/2$ comparisons. Hence, the average case also has quadratic running time $O(n^2)$.

Example:

25	15	30	9	99	20	26

15	25	30	9	99	20	26

15	25	30	9	99	20	26

9	15	25	30	99	20	26

9	15	25	30	99	20	26

9	15	20	25	30	99	26

9	15	20	25	26	30	99

☐ Unsorted list   ☐ sorted list.

Scanned with CamScanner

→ Definition of selection sort:

* The selection sort perform sorting by searching for the minimum value number and placing it into the first or last position according to the order (descending or ascending). The process of searching minimum key and placing it in the proper position is continued until the all elements are placed at right position.

→ Working of 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 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-arrary 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 advantage of selection sort is that it performs well on a small list.

* Furthurmore, because it is an in-place sorting algorithm, no additional temporary storage is required beyond what is needed to hold original list.

→ **Complexity of selection sort:**

(✳) As the working of selection sort does not depend on the original order 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 sort selects the minimum value element, in the selection process. All the n number of elements are scanned. Therefore $n-1$ comparisions are made in the first pass. Then, the elements are interchanged. Similarly in the second pass also to find second smallest element we require scanning of rest $n-1$ elements & the process is continued till the whole array sorted.

Thus, running time complexity of selection sort is

$$O(n^2) = (n-1) + (n-2) + \cdots + 2 + 1 = n(n-1)/2 = c(n^2).$$

Example:

0	1	2	3	4
17	16	3	13	6

Pass 1

17	16	3	13	6

↑ min   ↑ Loc

Pass 2

3	16	17	13	6

↑ min   ↑ Loc

Pass 3

3	6	17	13	16

↑ ↑ min Loc

Pass 4

3	6	13	17	16

↑ ↑ min Loc

Pass 5

3	6	13	16	17

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 product of elements in even positions

(iii). Elements which are divisible by m where m is taken from the user.

Solution :-

```c
#include <stdio.h>
#include <conio.h>
int main ()
{
    int arr[50], i, j, m, 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 ("\n sorting array using bubble sort technique \n");
    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[i] = arr[j+1]
                arr[j+1] = temp;
            }
        }
    }
}
```

```c
printf ("All array elements sorted successfully =\n").
printf ("Array elements in ascending order =\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 are=%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 of the program:-

Enter total number of elements to store = 5

Enter 5 elements

5

4

3

2

1

Sorting array using bubble sort technique.

All array elements sorted successfully.

Array elements in ascending order

1

2

3

4

5

Array elements in alternate order

1

3

5

The sum of odd position element is = 6

The product of even position element are = 15

5). Write a recursive program to implement binary search.

Solution:

```c
#include <stdio.h>
#include <stdlib.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)
    {
        binary search (arr, num, first, mid-1);

    }
    else
    {
        binary search (arr, num, mid+1, last);

    }

}
```

```c
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 search =");
    scanf("%d", &num);
    binary search(arr, num, beg, end);
}
```

Output of the program :-

Enter the size of an array 5
Enter the values in sorted sequence
1
2
3
4
5

Enter a value to search = 5
Element is found at index 4