

## Data Structures Assignment - 7

1. Assume that there is a list (22, 22, 22, 22, 22, 22, 22)

What happens when selection sort is applied on the list?

Example?

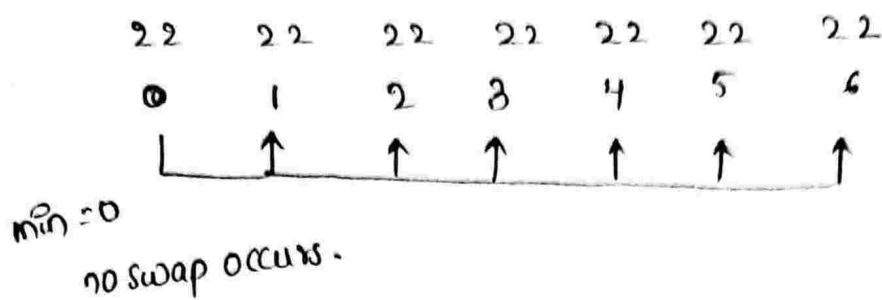
A:- Selection Sort:-

In selection sort algorithm, we search for the lowest element and arrange it to the proper location. We swap the current element with the next lower number.

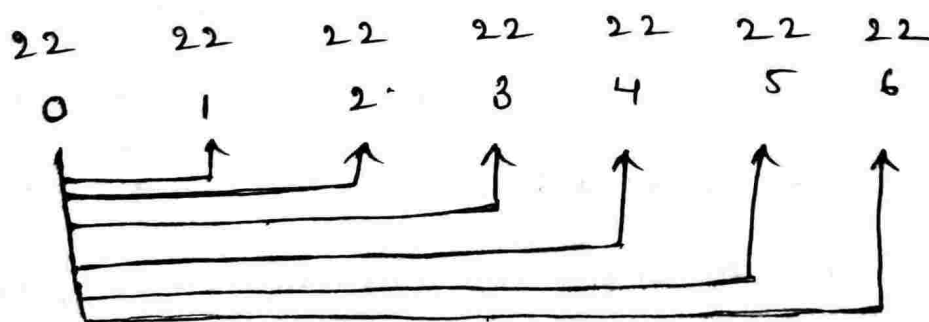
Algorithm Steps:-

1. At the first step the whole array will be unsorted.
2. Then find the minimum number or value from that unsorted part and swap it with first element of the list.
3. Now take the second index and remain the 1<sup>st</sup> sorted value as same, and again check the minimum value from the remaining unsorted part and again the swap the number with the second element from the left side.
4. Continue the steps by swapping the numbers by taking (or) increasing the numbers for comparing.

Here the given list  $\{ 22, 22, 22, 22, 22, 22, 22 \}$



We applied Selection Sort for the list given as we said in the steps we will check for the minimum number with the 1<sup>st</sup> element. But there will be no change here because all the numbers are equal. So we cannot swap any number. We taken index [0] as comparable element, we compared with the right side elements but there will be no change.



\* If we apply selection sort for equal elements then there will be no change in the elements.

\* It has  $(n-1)$  swaps.

2. select the following list of names using Insertion Sort:  
Varun, Amar, Karthik, Ramesh, Bhuvan, Dinesh, Firoz and Ganesh?

A:- Insertion Sort:-

It shifts the elements instead of swapping. It is good for small elements.

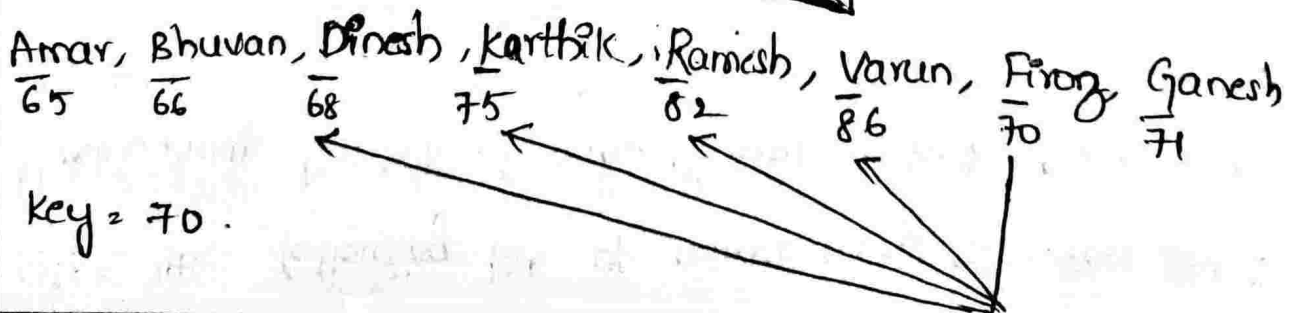
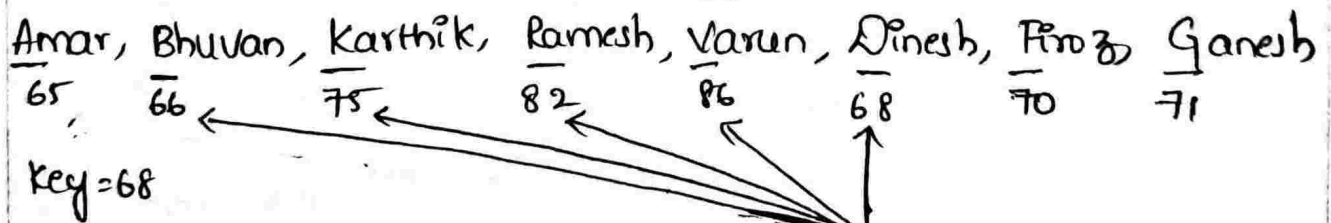
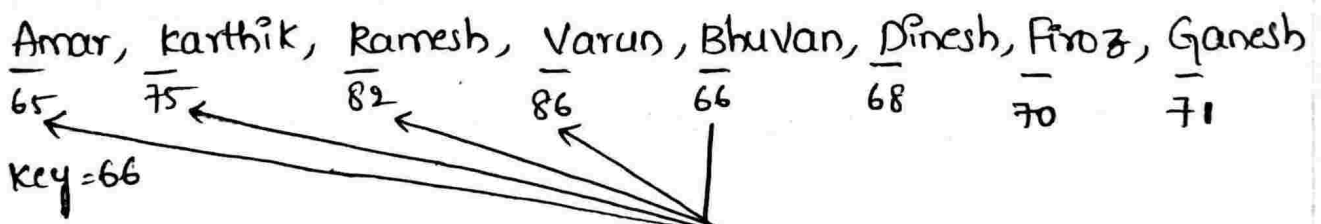
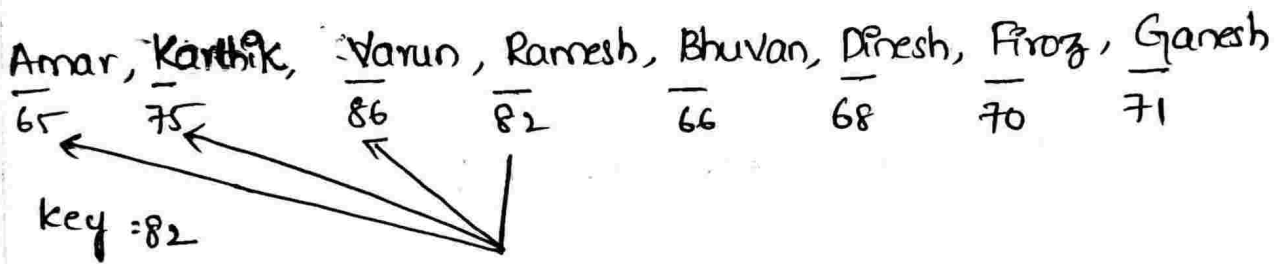
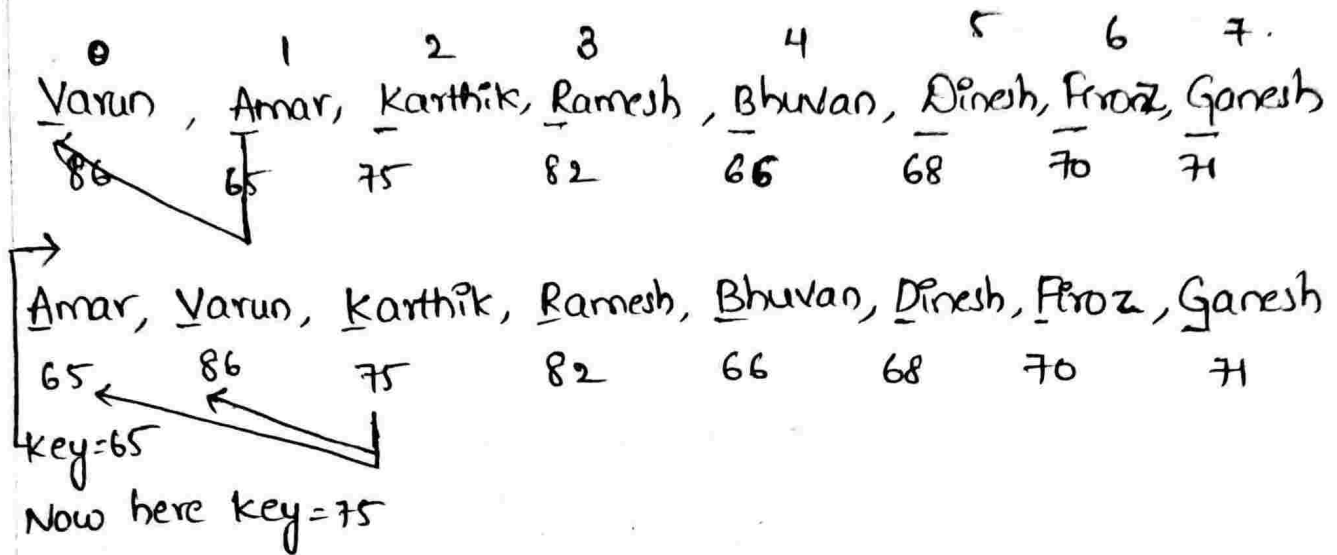
→ To Sort an array to make it in Ascending order.

Algorithm steps:-

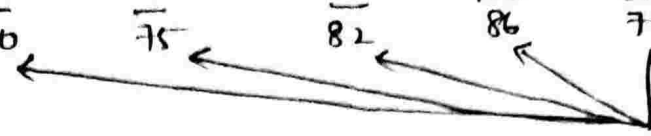
1. Iterate from  $a[1]$  to  $a[n]$  over the array.
2. Every Element should be Compared with previous Element and it get shifted.
3. If we taken a Key Element if ~~we~~ it is greater than the right side element then we shift the elements.
4. For the string we take ASCII values.
5. Then the Sorted list should be in the Ascending Order.

Here the given list.

{ Varun, Amar, Karthik, Ramesh, Bhuvan, Dinesh, Firoz, Ganesh }.



Amar, Bhuvan, Dinesh, Froz, Karthik, Ramesh, Varun, Ganesh  
 $\overline{65}$     $\overline{66}$     $\overline{68}$     $\overline{70}$     $\overline{75}$     $\overline{82}$     $\overline{86}$     $\overline{71}$



Key = 71

Amar, Bhuvan, Dinesh, Froz, Ganesh, Karthik, Ramesh, Varun  
 $\overline{65}$     $\overline{66}$     $\overline{68}$     $\overline{70}$     $\overline{71}$     $\overline{75}$     $\overline{82}$     $\overline{86}$

∴ Here the list sorted using Insertion Sort.

- 3) Sort the following numbers using Quick Sort: 67, 54, 9, 21, 12, 65, 56, 43, 34, 79, 70 and 45.

Quick Sort:-

It is an popular technique.

- \* It follows divide and Conquer as merge Sort but it sorts only when it needs.

Algorithm Steps:-

- \* Consider an unsorted list.
- \* Identify the 1<sup>st</sup> element as Key/pivot Element.
- \* start Comparing key with right most Elements.
- \* As long as there are bigger Elements towards right keep coming left (Move R to L).
- \* When small Element is found Swap it.
- \* start Comparing the key with left most Element.

\* As long as there is a Smaller Element towards Left, keep coming right (Move L to R).

\* When a bigger Elements is found, perform Swap.

\* Repeat the steps 4 to 8 till the key Element reaches its own/appropriate place.

\* Further divide the list into two Sublists

\* Again apply these steps 2 and to on the sublists.

Given List:-

← R to L

{ 67, 54, 9, 21, 12, 65, 56, 43, 34, 79, 70, 45 }

key

L to R →

45, 54 9 21 12 65 56 43 34 79 70 67

← R to L key

45 54 9 21 12 65 56 43 34 67 70 79

L to R → key

45 ~~54~~ 9 21 12 65 56 43 54 67 70 79

key

45 34 9 21 12 65 56 43 | 54 67 | 70 79

key                      key

L → R →

43 34 9 21 12 65 56 45 54 67 70 79

← R to key

43 34 9 21 12 | 45 | 56 | 65 | 54 67 70 79

L to R → key

43 12 9 21 34 | 45 | (56)<sup>L to R →</sup> 65 (54) (67) (70) (79)

43 12 9 21 34 | 45 | (54) (65) (56) (67) 70 79

(43) 12 9 21 34 | 45 | (54) (65)<sup>key</sup> 56 | 67 | 70 79  
← R to L

34 12 9 21 (43)<sup>key</sup> 45 | 54 | 56 65 67 70 79

(34)<sup>L to R →</sup> 12 9 21 | 43 | 45 54 56 65 67 70 79  
<sup>key</sup> ← R to L

21 12 9 (34)<sup>key</sup> 43 45 54 56 65 67 70 79

→ L to R  
(21) 12 9 | 34 | 43 45 54 56 65 67 70 79  
← R to L

9 12 21 | 34 | 43 45 54 56 65 67 70 79

Here the given list is sorted using quick sort.

4) Implement Linear Search and Binary Search using Recursion.

Program:-

Class Test

{

Static int a[] = {12, 34, 54, 2, 3}

/\* Recursion method to search x in a[l...r] \*/

Static int research(int a[], int l, int r, int x)

{

```

        if (l < 1)
            return -1;
        if (a[l] == x)
            return l;
        if (a[r] == x)
            return r;
        return recSearch(a, l+1, r-1, x);
    }

    public static void main(String[] args)
    {
        int x = 3;
        int index = recSearch(a, 0, a.length-1, x);
        if (index != -1)
            System.out.println("Element " + x + " is present at index " +
                                index);
        else
            System.out.println("Element " + x + " is not present");
    }
}

```

```

class BinarySearch {
    static int binarySearch(int a[], int l, int r, int x)
    {
        if (r >= l) {
            int mid = l + (r - l) / 2;

```



```

        if (a[mid] == x)
        if (a[mid] > x)
            return binarySearch(a, mid + 1, r, x);
    }

```

```

public static void main (String args[])
{

```

```

    BinarySearch ob = new BinarySearch();

```

```

    int a[] = { 2, 3, 4, 10, 40 };

```

```

    int n = a.length;

```

```

    int x = 10;

```

```

    int result = ob.binarySearch(a, 0, n-1, x);

```

```

    if (result == -1)

```

```

        System.out.println("Element not present");

```

```

    else

```

```

        System.out.println("Element found at index " + result);

```

```

    }

```

```

}

```

out put:-

element found at index 3

5. Explain, in brief, the various factors that determine the selection of an algorithm to solve a Computational Problem.

A: An algorithm is a sequence of steps to solve a particular problem. One problem can have 'n' no. of solutions (i.e., multiple solutions). To find out best solution among 'n' solutions, we apply some analysis.

Analysis of algorithms is the determination of the amount of time space resources required to execute it.

The performance of an algorithm can be measured by two properties : Time and Space.

Time Complexity of an algorithm to (run as a fun) quantifies the amount of space or memory taken by an algorithm to run as function of the length of the input.

Space Complexity:-

An algorithm represents the amount of memory space needed the algorithm in its life cycle.

Space needed by an algorithm is equal to the sum of the following two component.

$$\text{Space (SP)} = \text{Fixed part (A)} + \text{Variable part SP (I)}$$

A fixed part that is a space required to store certain data and variables, that are not dependent of the size of the problem. A variable part is a space required by variables, whose size is totally dependent on the size of the problem.

eg:- recursion stack space, dynamic memory allocation etc. Space can be calculated based on amount of space required

- To store program instructions.
- To store constant values.
- To store variable values.
- And for few other things like function calls, jumping statement, etc.

eg:-

```
int sum(int a[], int n)
{
    int sum = 0, i;
    for (i = 0; i < n; i++)
        sum = sum + a[i];
    return sum;
}
```

In the above piece of code it requires.

→  $n * 4$  bytes of memory to store array variable 'a[]'

- 4 bytes of memory for integer parameter 'n'.
- 8 bytes of memory for local integer variables 'sum' & 'i' (4 bytes each).
- 4 bytes of memory for return values.

In case of Time Complexity, the running Time of an algorithm depends upon the following.

- Whether it is running on single processor machine or multi processor machine.
- Whether it is a 32-bit machine or 64 bit machine.
- Read and write speed of the machine.
- The amount of time required by an algorithm to perform Arithmetic operations, logical operations, return value and assignment operations etc.
- Input data.