19BQ1A05E3

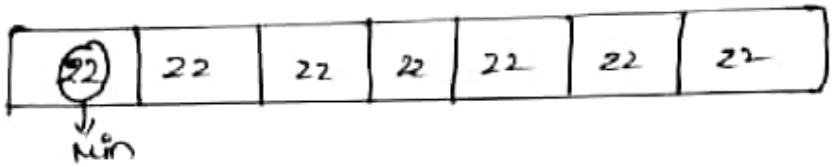
M. Venkata Gopi Jayaram
19BQ1A05E3
II CSE-C

DATA STRUCTURES Assignment-1

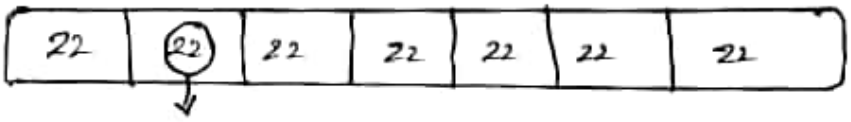
① Assume that there is a list [22, 22, 22, 22, 22, 22, 22] what happens when selection sort is applied on the list? explain.

Ans Selection sort: selection sort is an algorithm that we select and search for the lowest element. Then the lowest in a element is swapped with current element

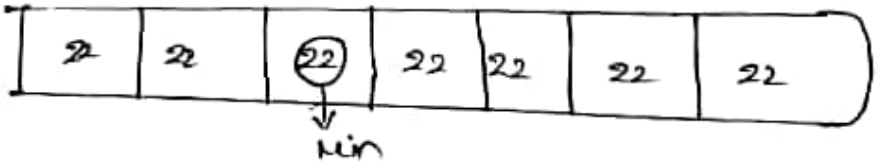
given array,



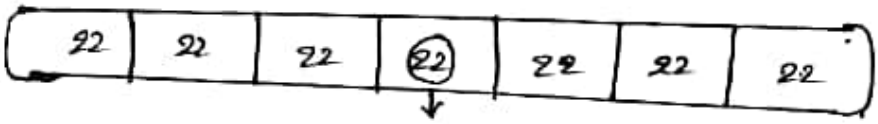
Here no swap



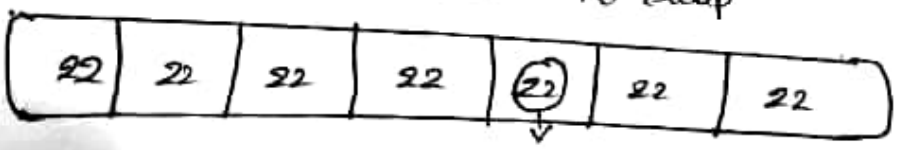
Here, no swap



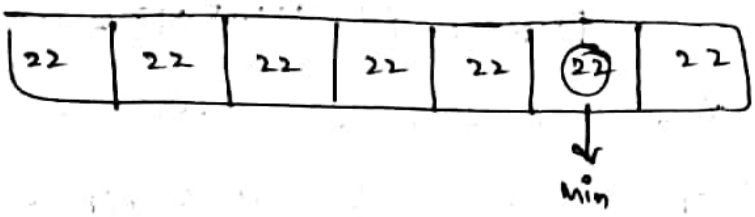
Here no swap



Here, no swap



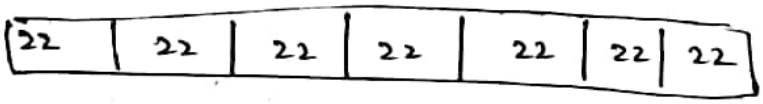
Here, no swap



Here, no swap.

In the above list all the elements are same so, there are no swappings at all.

Output :-



---

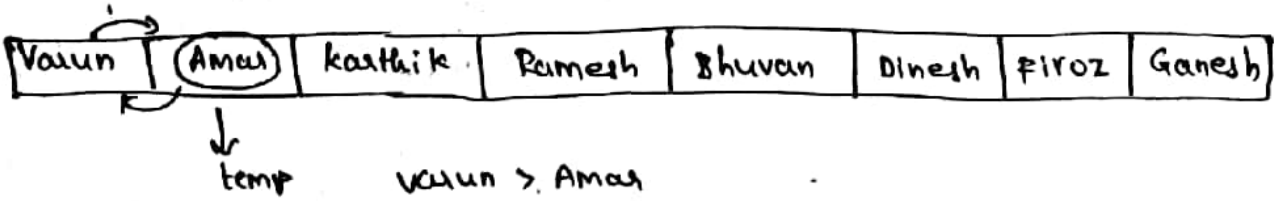
(2) Sort the following list using Insertion sort

Varun Amar karthik Ramesh Bhuvan Dinesh Firoz Ganesh.

Ans:- Insertion sort:- It is also a sorting algorithm. But it is more effecient because it replaces sorting swapping with shifting

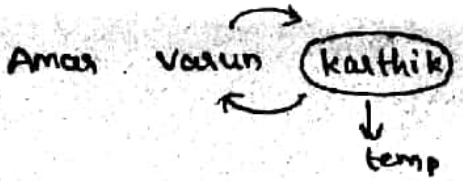
Here every element is compared to its previous element If we found any bigger element before the key, then we shift their places.

Given array,



varun > Amar

so, shift varun right and insert Amar at 0^{th} pos



Ramesh Bhuvan Dinesh Firoz Ganesh

varun > kaithik

shift varun right and insert kaithik at 1st position.

Amar ~~varun~~ kaithik

Amar kaithik varun (Ramesh) Bhuvan Dinesh Firoz Ganesh

↓
temp.

Varun > Ramesh

Shift Ramas varun Right & insert Ramesh at 2nd pos

Amar kaithik Ramesh varun (Bhuvan) Dinesh Firoz Ganesh

↓
temp

varun > Bhuvan , Ramesh > Bhuvan , kaithik > Bhuvan

shift kaithik, Ramesh , varun to right &

insert Bhuvan at position 1.

Amar Bhuvan kaithik Ramesh varun (Dinesh) Firoz Ganesh

↓
temp

shift kaithik , Ramesh , varun to right and

insert Dinesh at 2nd position.

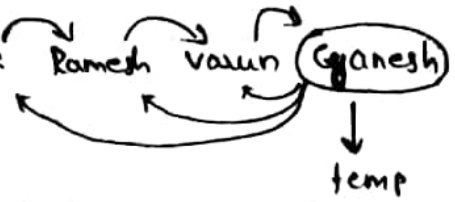
Amar Bhuvan Dinesh kaithik Ramesh varun (Firoz) Ganesh

↓
temp

shift kaithik , Ramesh, varun to right and

insert Firoz at 3rd position.

Amar Bhuvan Dinesh Firoz Karthik Ramesh Varun Ganesh



Shift karthik, Ramesh, varun to Right and insert Ganesh at 4th pos.

Amar	Bhuvan	Dinesh	Firoz	Ganesh	karthik	Ramesh	varun

This is the sorted list.

② Sort the following numbers using Quick sort :

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

Ans:-

Quick sort:- It is based on devide and conquer principle. Take first element of the list as pivot. Swappings are done untill pivot element reaches its correct position. Then, again take the two sublists and repeat the process untill we get a sorted list.

Given array:-

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

↓ pivot

compare from ~~left~~ to right to left for smaller

Swap (67, 45)

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

↓ pivot

Compare from left to right for bigger elem

Swap (79, 67)

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

pivot.

67 is in correct position

Now divide left sublist and Right sub list

L.S.L

(45) 54 9 21 12 65 56 43 (34)

pivot

Compare left from right.

Swap (45, 34)

34 (54) 9 21 12 65 56 43 (45)

pivot

left to right Swap (54, 45)

34 (45) 9 21 12 65 56 (43) 54

pivot

Right to left (Swap (45, 43))

34 43 9 21 12 (65) 56 (45) 54

pivot.

left to Right .

Swap (65, 45)

R.S.L

(70) 79

pivot

Right to left

No swap

70	79

34 43 9 21 12 (45) 56 65 54

Pivot

Left to Right to left

NO swap

So 45 is in correct position

Now divide Left sub list & Right sub list.

L.S.L

(34) 43 9 21 (12)

Pivot

R → L swap (34, 12)

12 (43) 9 21 (34)

Pivot

L → R (swap (34, 43))

12 (34) 9 (21) 43

pivot

R → L (swap (34, 21))

12 21 9 (34) 43

Pivot

34 in correct position

R.S.L

(56) 65 (54)

Pivot

Right to left

swap (56, 54)

54 (65) (56)

Pivot

left to right

swap (65, 56)

54	56	65

Left sub list

(12) 21 9

pivot

Right sub list

(43)

Left sub list

(12) 21 (9)

↓
pivot

Right to left

(swap 12 & 9)

9 (21) (12)

↓
pivot

Left to Right

swap (21, 12)

9	12	21

The final sorted list is

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

(H) Implement Linear and Binary search using recursion

Ans:-

Linear search:-

It is used to find position of an element in the given list. It is also called as sequential search.

Algorithm:-

i) Take a list of elements

ii) Compare the key with all the elements in the list sequentially

Program:-

```
import java.util.Scanner;
Public class linearSearch
{
  public static void main (string args[])
  { int a[], n, i, key, pos;
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter size of array");
    n = sc.nextInt ( );   a = new Int[n];
    System.out.println ("Enter elements of array");
    for (i=0; i<n; i++)
        a[i] = sc.nextInt();
    System.out.println ("The array is : ");
    for (i=0 ;i<n ;i++)
        system.out.print ( a[i] + "    ");
    System.out.println();
  || pos = linearSearch (a, 0, n-1, key)
    System.out.println ("Enter search key");
     key = sc.nextInt ();
    pos = linearSearch (a, 0, n-1, key);
    if (pos == -1)
       System.out.println (" key not found!");
    else
       System.out.println (" key found at : " + (pos+1));
}
```

```
public static int/void linearSearch (int a[], int lb, int ub, int key)
  {
      if (lb > ub)
         return -1;
      else if (a[lb] == key)
         return lb;
      else
         return linearSearch (a, lb+1, ub, key);
  }
}
```

output:-

```
Enter size of array
5
Enter elements of array
5
9
12
14
15
The array is:
5  9  12  14  15
Enter search : 14
key found at 4
```

* Binary search:-

It is also used to find position of an element in the given list. It is based on divide and conques principle.

It reduces no. of comparisons when compared to linear search.

Algorithm:-

i) Take array of elements.

ii) Find mid position

iii) If key is found at mid, return mid.

iv) If key is greater than mid element and repeat the procedure (steps 1 to 3)

v) If key is less than mid element then take left sublist and repeat the procedure (step 1 to 3)

program:-

```
import java.util.Scanner;
Public class Binarysearch
{
  public static void main (String args[])
  {
    int a[], key, pos, n, i;
    Scanner sc = new Scanner (System.in);
    System.out.println (" Enter size of array");
    n= sc.nextInt ();
```

```
Public static int binarysearch ( int a[], int lb, int ub, int key)
{
    int mid=0;
    if (ub>=lb)
    {
        mid = lb + (ub-lb)/2;
        if (a[mid]==key)
            return mid;
    }
    if (a[mid]> key)
        return binarysearch (a, lb, mid-1, key);
    else
        return binary search (a, mid+1, ub, key);
}
Public static void sort ( int a[], int n)
{
    int i, j
    for (i=0 ; i<n-1; i++)
        for (j=0; j<n-i-1 ;j++)
            if (a[j] > a[j+1])
            { int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
    }
}
```

Output:-

```
Enter size of array
5
Enter elemets of array
5 4 3 2 1
```

```
The array is
5 4 3 2 1
The sorted array is
1 2 3 4 5
```

```
Enter search key
3
key found at 3
```

(5) Explain in brief, the various factors that determine the selection of any algorithm to solve a computational problem.

Analysis of algorithms is the determination of the amount of time and space recources required to solve a computational problem or any other

Usually, the efficiency or running time of an algorithm is stated as a function relating the input length to the number of steps, known as time complexity, or volume of memory, known as space complexity.

By considering algorithm for a specific problem we can begin to develop pattern recognition, so that similar types of problems can be solved with the help of this algorithm.

Efficiency of Algorithm is measured in two diffent stages

1) Space complexity:-

It represents the total amount of memory needed for an algorithm to solve a problem.

space = fixed part + variable part

It depends on processor, hardware, os etc.

Eg:- If we compare bubble sort and merge sort Bubble sort requires less space compared to merge sort.

2) Time complexity:-

The amount of time required to for an algorithm to solve a problem.

It is mainly based on processor, clock speed, os etc.

There are 3 types of time complexities

i) Best case (Omega notation) :- The minimum number of steps taken to solve a problem.

ii) Average case (Theta notation) :- The Average number of steps taken to solve a problem

* iii) Worst case (Big O notation):- The maximum number of steps taken to solve a problem

usually, Big-O notation is the most used one. Because, algorithm perform may vary with different types of input data.

So, Based on the two complexities we find a better algorithm to solve a computational problem.

```
a = new int[n];
System.out.println("Enter elements of array");
for (i=0; i<n; i++)
    a[i] = sc.nextInt();
System.out.println("The array is :");
for (i=0; i<n; i++)
    System.out.print(" a[i] + "  ");
system.out.print();
Sort (a, n);
System.out.println("The sorted array is");
for (i=0; i<n; i++)
    System.out.print( a[i] + "  ");
system.out.println();
System.out.println("Enter search key");
key = sc.nextInt();
pos = binarySearch(a, 0, n-1, key);
if (pos == -1)
    system.out.println("No key found");
else
    system.out.println("key found at " +(pos+1));
}
```