DAY - 3

#Days_Until_I_get_Placed.

TOPIC - Array | Searching...

- Defined as an operation of finding a particular element or a group of elements in array.
- Several searching algorithms include :-
  1) Linear Search
  2) Binary Search
  3) Tenary Search

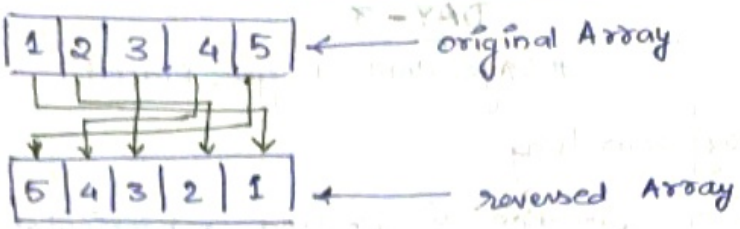
1) Linear Search :- Linear Search is defined as a sequential search algorithm that starts at one end and goes through each element of a list until the desired element or group of elements is found.
   - time Complexity is O(N).

2) Binary Search :- - It is a searching algorithm used in sorted array.
   - In this algorithm, the element is found by repeatedly dividing the search interval in half and deciding the next interval to find the element.
   - time Complexity is $O(\log_2 N)$.

3) Ternary Search :- - It is a divide and conquer algorithm that can be used to find an element in an array.
   - similar to binary search but in this we divide the given array into three parts and determine which has the key (searched element).
   - time complexity is $O(\log_3 N)$.

★ Array Reverse in C/C++ / Java / Python / JavaScript :-

- Reversing an array is changing the possition of each number of the given array to its opposite position from end
- If a number is at position 1 then its new position will be Array.length. for position 2 will be Array.length - 1.



DAY - 4

#Days_Until_I_get_Placed.

* Program for array left rotations by d positions.

Let's take an example. an array of integers arr[] of size N.

Input :- arr[] = {1,2,3,4,5,6,7}, d = 2

Output : 3 4 5 6 7 1 2

for doing so, Approach 1 (Using temp Array).

In this suppose we have arr[] = [1,2,3,4,5,6,7], d = 2.

1) storing elements from 2nd index to last.

temp[] = [3,4,5,6,7]

2) storing first 2 elements in temp[] array.

temp[] = [3,4,5,6,7,1,2]

3) Copying all the elements of temp[] array into original array.

arr[] = temp[] so arr[] = [3,4,5,6,7,1,2]

Approach 2 (Rotate one by One):

1) arr[] = [1,2,3,4,5,6,7], d = 2.

2) Rotate to left by one position.

arr[] = {2,3,4,5,6,7,1}

3) Rotate again to left by one position.

arr[] = {3,4,5,6,7,1,2}

So the array becomes arr[] = {3,4,5,6,7,1,2}

Approach 3 - The juggling Algorithm

Let arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14} and d = 10

1) first step is {0, 5, 10}
   - » rotating this set by d position in cyclic order.
   - » arr[0] = arr[0 + 10]
   - » arr[10] = arr[(10 + 10) % 15]
   - » arr[5] = arr[0]
   - » This set becomes {10, 0, 5}
   - » Array arr[] = {10, 1, 2, 3, 4, 0, 6, 7, 8, 9, 5, 11, 12, 13, 14}

2) Second step : is {1, 6, 11}
   - » Rotate this set by d position in cyclic order.
   - » This set becomes {11, 1, 6}
   - » Array arr[] = {10, 11, 2, 3, 4, 0, 1, 7, 8, 9, 5, 6, 12, 13, 14}

3) Third step : » {2, 7, 12}
   - » Rotate this set by d position in cyclic order.
   - » This set becomes {12, 2, 7}
   - » Array arr[] = {10, 11, 12, 3, 4, 0, 1, 2, 8, 9, 5, 6, 7, 13, 14}

4) fourth step : » {3, 8, 13}
   - » Rotate this set by d position in cyclic order.
   - » This set becomes {13, 3, 8}
   - » Array arr[] = {10, 11, 12, 13, 4, 0, 1, 2, 3, 9, 5, 6, 7, 8, 14}

5) fifth step : » {4, 9, 14}
   - » Rotate this set by d position in cyclic order.
   - » This set becomes {14, 4, 9}
   - » Array arr[] = {10, 11, 12, 13, 14, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

★ Print array after it is right rotated K-times :-

eg :- Array [] = {1, 3, 5, 7, 9} , K = 2

Output : { 7, 9, 1, 3, 5

Explanation :-

After 1st rotation - {9, 1, 3, 5, 7}

After 2nd rotation - {7, 9, 1, 3, 5}

APPROACH 2 :- Reversing the array

- reversing the array 3 times

eg:- Array [] = {1, 3, 5, 7, 9, 11}

Output = {7, 9, 11, 1, 3, 5}

Approach 3 : Recursive Approach

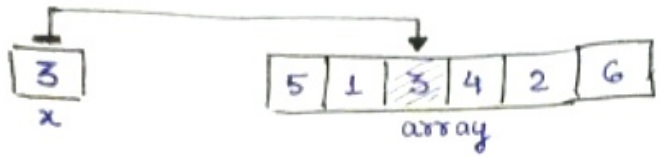
- Pseudocode

```
function rotateArray(arr, n, k):
    // Reduce the number of rotations
    k = k % n

    // Reverse the first part of array
    reverse(arr, arr+n-k)
    // Reverse the second part of array
    reverse(arr+n-k, arr+n)

    // Reverse the entire array
    reverse(arr, arr+n)

arr = {1,3,5,7,9}
n = size(arr)
k = 2

rotateArray(arr, n, k)

for i = 0 to n-1:
    print arr[i]
```

★ Search, Insert and Delete in an Unsorted Array.

1) **Search Operation :-** - In an unsorted array the search operation can be performed by linear traversal from the first element to the last element.

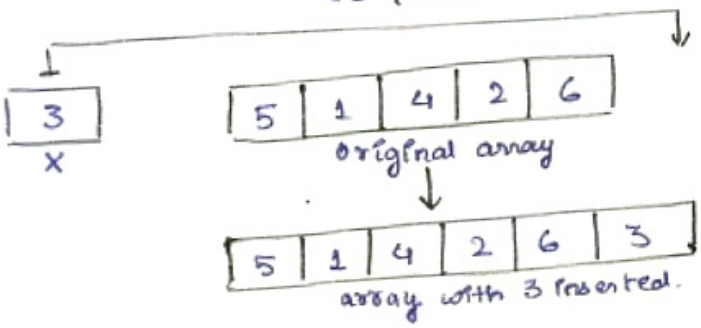
eg:-



2) **Insert Operation :-**

1) **Insert at the end :-** In an unsorted array, the insert operation is faster as compared to a sorted array because we don't have to care about the position at which the element is to be placed.

eg:-



2) **Insert at any position :-** Insert operation is an array at any position can be performed by shifting elements to the right, which are on the right side of the required position.