03 Introduction To Array

14:08

Agenda :

1. Introduction To Arrays
2. Problems
3. Dynamic Arrays

Introduction To Arrays

Space Complexity

- Space complexity is the max space(worst case) that is utilised at any point in time during running the algorithm.
- We also determine Space Complexity using Big O.
- We only consider the space utilised by our program and not the Input Space since it is not in our control.

Array is the collection of same types of data. The datatype can be of any type i.e, int, float, char, etc. Below is the declaration of the array:

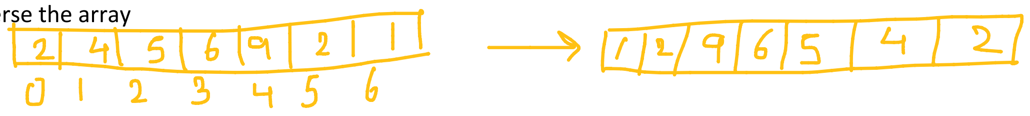
Note: Array indexing starts with 0.

Why indexing starts at 0 ?

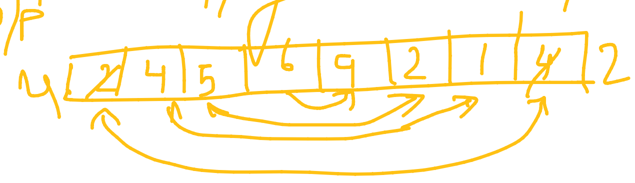
An array arr[i] is interpreted as *(arr+i). Here, arr denotes the address of the first array element or the 0 index element. So *(arr+i) means the element at i distance from the first element of the array.

Problems

Q1. Reverse the array

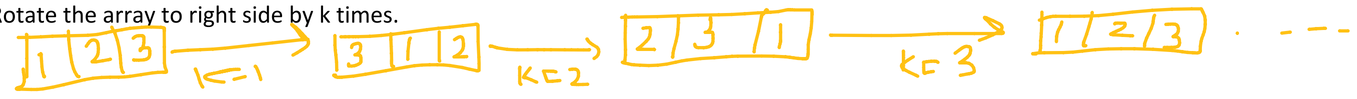


We can keep on swapping the value from first till mid to get desired o/p



Q2. Reverse the array in range ex: [index=4 to index=10]

Q3. Rotate the array to right side by k times.



Brute Force :

For k iterations keep on shifting each element of the array. Resultant array will contain array shifted with k positions.

Optimized approach :

After K rotations, last K elements become 1st K elements and rest elements will go at back.

For example - Suppose we have an array arr as shown below and k = 3.

1 2 3 4 5 6 7

After 1st rotation, k=1:

7 1 2 3 4 5 6

After 2nd rotation, k=2:

6 7 1 2 3 4 5

After 3rd rotation, k=3:

5 6 7 1 2 3 4

So, we have observed that last 3(K=3) elements i.e, 5 6 7 comes in front and rest elements appear at the end.

Therefore, we will first reverse the entire array, then reverse first K elements individually and then next N-K elements individually.

Edge case needs to be handled where size can be equal to n so we can k = k%n

Q4. Find sum of max and min of an array.

Q5. Given an array A and an integer B, find the number of occurrences of B in A.

Q6. You are given an integer array A. You have to find the second largest element/value in the array or report that no such element exists.

Q7. Given an integer array A of size N. In one second, you can increase the value of one element by 1. Find the minimum time in seconds to make all elements of the array equal.

Q8. You are given an integer array **A** of length **N**.
You are also given a 2D integer array **B** with dimensions **M x 2**, where each row denotes a [L, R] query.
For each query, you have to find the sum of all elements from L to R indices in A (0 - indexed).
More formally, find A[L] + A[L + 1] + A[L + 2] +... + A[R - 1] + A[R] for each query.

Dynamic Array In Java

- A dynamic array is an array with a big improvement: automatic resizing.
- It expands as you add more elements. So you don't need to determine the size ahead of time.

- **ArrayList<String> al = new ArrayList<>(); //Arraylist is created**
- `public Object set(int index, Object element)`
- **al.add("50"); //50 is inserted at the end**
- **al.clear(); // al={}**
-
```
for (int i = 0; i < al.size(); i++) {
        System.out.print(al.get(i) + " ");
}  //iterating the Arraylist
```

Strengths:

Fast lookups: Just like arrays, retrieving the element at a given index takes
O(1) time.
Variable size: You can add as many items as you want, and the dynamic array will expand to hold them.

Weaknesses:
Slow worst-case appends:

Usually, adding a new element at the end of the dynamic array takes O(1) time.
But if the dynamic array doesn't have any room for the new item, it'll need to expand, which takes O(n) time.
It is because we have to take a new array of bigger size and copy all the elements to a new array and then add a new element.
So, Time Complexity to add a new element to Dynamic array is O(1) amortised.
Amortised means when most operations take O(1) but some operations take O(N).