DSA Topics Preparation

Part-1

Topic- ARRAY

All detailed revision of array which consists of topic wise revisions and solving problems recently asked in interviews-

Just simplified my experience here…

Hope it goona help you all…

Save this pdf and thanks me later

@himanshu_shekhar16

@himanshushekar

Reference taken form GFG, Leetcode, and InterviewBit

What is an Array?

An array is a collection of items of same data type stored at contiguous memory locations.

This makes it easier to calculate the position of each element by simply adding an **offset** to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array). The base value is index 0 and the difference between the two indexes is the **offset.**

Is the array always of fixed size?

In C language, the array has a fixed size meaning once the size is given to it, it cannot be changed i.e. you can't shrink it nor can you expand it. The reason was that for expanding if we change the size we can't be sure (it's not possible every time) that we get the next memory location to us for free. The shrinking will not work because the array, when declared, gets memory statically allocated, and thus compiler is the only one that can destroy it.

Types of indexing in an array:

- 0 (zero-based indexing): The first element of the array is indexed by a subscript of 0.
- 1 (one-based indexing): The first element of the array is indexed by the subscript of 1.
- n (N-based indexing): The base index of an array can be freely chosen. Usually, programming languages allowing n-based indexing also allow negative index values, and other scalar data types like enumerations, or characters may be used as an array index.

How an Array is initialized?

By default the array is uninitialized, and no elements of the array are set to any value. However, for the proper working of the array, array initialization becomes important. Array initialization can be done by the following methods:

1. Passing no value within the initializer: One can initialize the array by defining the size of the array and passing no values within the initializer.
Syntax:
int arr[5] = { };

2. By passing specific values within the initializer: One can initialize the array by defining the size of the array and passing specific values within the initializer.
Syntax:
int arr[5] = { 1 , 2 , 3 , 4 , 5 };

Note: The count of elements within the "{ }", must be less than the size of the array
If the count of elements within the "{ }" is less than the size of the array, the remaining positions are considered to be '0'.
Syntax:
int arr[5] = { 1 , 2 , 3 } ;

3. By passing specific values within the initializer but not declaring the size: One can initialize the array by passing specific values within the initializer and not particularly mentioning the size, the size is interpreted by the compiler.
Syntax:
int arr[ ] = { 1 , 2 , 3 , 4 , 5 };

4. Universal Initialization: After the adoption of **universal initialization** in C++, one can avoid using the equals sign between the declaration and the initializer.
Syntax:
int arr[ ] { 1 , 2 , 3 , 4 , 5 };

Insertion in Array:

We try to insert a value to a particular array index position, as the array provides random access it can be done easily using the assignment operator.

Pseudo Code:
// to insert a value= 10 at index position 2;

arr[2] = 10;

Time Complexity:
- O(1) to insert a single element
- O(N) to insert all the array elements [where N is the size of the array]

Access elements in Array:
Accessing array elements become extremely important, in order to perform operations on arrays.

Pseudo Code:

// to access array element at index position 2, we simply can write

return arr[2] ;

Time Complexity: O(1)

Searching in Array:

We try to find a particular value in the array, in order to do that we need to access all the array elements and look for the particular value.

Pseudo Code:

// searching for value 2 in the array;

```
Loop from i = 0 to 5:
    check if arr[i] = 2:
        return true;
```

Time Complexity: O(N), where N is the size of the array.

Here is the code for working with an array:

Types of arrays :

1. One dimensional array (1-D arrays)
2. Multidimensional array

Advantages of using arrays:

- Arrays allow random access to elements. This makes accessing elements by position faster.
- Arrays have better cache locality which makes a pretty big difference in performance.
- Arrays represent multiple data items of the same type using a single name.

Disadvantages of using arrays:

You can’t change the size i.e. once you have declared the array you can’t change its size because of static memory allocation. Here Insertion(s) and deletion(s) are difficult as the elements are stored in consecutive memory locations and the shifting operation is costly too.

PREVIOUSLY ASKED PROBLEMS

- Cyclically rotate an array by one
- Find minimum and maximum element in an array
- Subarray with given sum
- Missing number in array
- Find duplicates in an array
- Sort an array of 0s, 1s and 2s
- Peak element
- Count pairs with given sum
- Wave Array
- First Repeating Element
- Implement two stacks in an array
- Array Subset of another array
- Alternate positive and negative numbers
- Move all negative elements to end
- Max sum path in two arrays
- Non-Repeating Element
- Sum Pair closest to X

- Minimum number of jumps
- Kadane's Algorithm
- Minimize the Heights II
- Kth smallest element
- Majority Element
- Trapping Rain Water
- Maximum Product Subarray
- Minimum Platforms
- Find Missing And Repeating
- Smallest Positive missing number
- Triplet Sum in Array
- Row with max 1s
- Spirally traversing a matrix
- Maximum Index
- Coin Change
- Stock buy and sell
- Stock span problem
- Max sum in the configuration
- Subarrays with equal 1s and 0s

- Smallest Positive Integer that can not be represented as Sum
- Common elements
- Minimum element in a sorted and rotated array
- Sort The Array
- Chocolate Distribution Problem
- Quick Sort
- Subarray with 0 sum
- Longest consecutive subsequence
- Reverse a String
- Factorials of large numbers
- Partition Equal Subset Sum
- Longest alternating subsequence
- Find the Frequency
- Cyclically rotate an array by one
- Find minimum and maximum element in an array
- Subarray with given sum
- Missing number in array
- Find duplicates in an array
- Sort an array of 0s, 1s and 2s

- Peak element
- Count pairs with given sum
- Wave Array
- First Repeating Element
- Implement two stacks in an array
- Array Subset of another array
- Alternate positive and negative numbers
- Move all negative elements to end
- Max sum path in two arrays
- Non-Repeating Element
- Sum Pair closest to X
- Minimum number of jumps
- Kadane's Algorithm
- Minimize the Heights II
- Kth smallest element
- Majority Element
- Trapping Rain Water
- Maximum Product Subarray
- Minimum Platforms

- Find Missing And Repeating
- Smallest Positive missing number
- Triplet Sum in Array
- Row with max 1s
- Spirally traversing a matrix
- Maximum Index
- Coin Change
- Stock buy and sell
- Stock span problem
- Max sum in the configuration
- Subarrays with equal 1s and 0s
- Smallest Positive Integer that can not be represented as Sum

Feel Free to connect with me –

https://www.linkedin.com/in/himanshushekhar16/