Array: Interview-Focused Training Content for Students

---

1. What is an Array?

Definition

An array is a collection of elements of the same type stored in contiguous memory locations. Each element is accessed by its index, starting from 0.

Declaration (C example)

```
int arr[10]; // declares an array of size 10
```

Traversal

Use loops to visit each element.

```
for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
}
```

Time Complexities

Operation	Complexity
Access	O(1)
Traversal	O(n)
Search (linear)	O(n)
Insert/Delete	O(n)

Real-life Uses

Arrays are used widely in daily life and software systems for the following:

- **Storing Marks/Temperatures**: When you want to record values that belong to the same category (e.g., daily temperatures), arrays provide a convenient way.
- **Timetables and Seat Booking Systems**: Arrays can be used to represent seat positions in a bus or cinema hall.

Technical Uses

Arrays are fundamental in programming and algorithm design:

- **Stacks and Queues**: Implemented using arrays for fixed-size data.
- **Static Lookup Tables**: Used in compilers and interpreters for symbol resolution.
- **Sorting/Searching Algorithms**: Arrays form the base for implementing merge sort, quicksort, binary search, etc.
- **Graphs and Matrices**: Adjacency matrices or multi-dimensional arrays for graph algorithms.

---

2. Solutions to Important Questions

Reverse an Array or String

Problem: Reverse all elements so the last becomes first and so on.

Example:

```
Original: [1, 2, 3, 4, 5]
Reversed: [5, 4, 3, 2, 1]
```

Algorithm:

1. Use two pointers: one at the start and one at the end.
2. Swap elements at these pointers.
3. Move start forward and end backward until they meet.

Code:

```
void reverse(int arr[], int n) {
    int start = 0, end = n - 1;
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}
```

Time Complexity: O(n)

Find Maximum or Minimum Element

Problem: Find the largest and smallest element in the array.

Example:

```
Array: [10, 4, 2, 20, 5]
Max: 20, Min: 2
```

Algorithm:

1. Initialize max and min to the first element.
2. Iterate through the array.
3. Update max if current element is greater.
4. Update min if current element is smaller.

Code:

```
void findMaxMin(int arr[], int n, int *max, int *min) {
    *max = arr[0];
    *min = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > *max)
            *max = arr[i];
        if (arr[i] < *min)
            *min = arr[i];
    }
}
```

Time Complexity: O(n)

Find 3rd Largest Element

Problem: Find the third largest unique element in an array.

Example:

```
Array: [10, 20, 5, 8, 15]
3rd Largest: 10
```

Algorithm:

1. Initialize first, second, and third to the lowest possible value.

2. Iterate through the array.
3. Update first, second, and third whenever a larger element is found.

Code:

```
int thirdLargest(int arr[], int n) {
    int first = INT_MIN, second = INT_MIN, third = INT_MIN;
    for (int i = 0; i < n; i++) {
        if (arr[i] > first) {
            third = second;
            second = first;
            first = arr[i];
        } else if (arr[i] > second && arr[i] != first) {
            third = second;
            second = arr[i];
        } else if (arr[i] > third && arr[i] != second && arr[i] != first) {
            third = arr[i];
        }
    }
    return third;
}
```

Time Complexity: O(n)

Remove Duplicates from Array

Problem: Remove duplicates so only unique elements remain.

Example:

```
Original: [1, 2, 2, 3, 4, 4]
After: [1, 2, 3, 4]
```

Algorithm:

1. Create a temporary array to store unique elements.
2. Traverse each element in the original array.
3. Check if it is already in the temp array.
4. If not, add it.

Code:

```
int removeDuplicates(int arr[], int n) {
    if (n == 0 || n == 1)
        return n;

    int temp[n];
    int j = 0;

    for (int i = 0; i < n; i++) {
        int isDuplicate = 0;
        for (int k = 0; k < j; k++) {
            if (arr[i] == temp[k]) {
                isDuplicate = 1;
                break;
            }
        }
        if (!isDuplicate)
            temp[j++] = arr[i];
    }

    for (int i = 0; i < j; i++)
        arr[i] = temp[i];

    return j;
}
```

Time Complexity: O(n^2)

3. Frequently Asked Array Questions

1. Find max element.
2. Find min element.
3. Reverse array.
4. Check if array is sorted.
5. Find second largest element.
6. Find k-th largest or smallest element.
7. Remove duplicates.
8. Rotate array by k positions.
9. Move zeros to the end.
10. Merge two sorted arrays.
11. Find missing number from 1 to n.
12. Find element appearing once when others appear twice.
13. Find all pairs with given sum.
14. Find majority element.
15. Find subarray with maximum sum.
16. Count even and odd numbers.

17. Check for palindrome array.
18. Find minimum number of swaps to sort.
19. Find duplicate number in restricted range.
20. Find longest increasing subsequence.

Prepared for classroom explanation and interview preparation.