Sheryians Coding School

Live Cohort

Day 80

Quick Sort

Definition:

Quick Sort is a Divide and Conquer algorithm that sorts an array by selecting a pivot, partitioning elements around it, and recursively sorting the subarrays. It's efficient for general-purpose sorting.

Code(Examples):

```
let arr = [18, 5, 3, 40, 10, 30];
quickSort(arr, 0, arr.length - 1);
console.log(arr); // Output: [3, 5, 10, 18, 30, 40]

function quickSort(arr, first, last) {
    if (first < last) {
        let pivotIndex = findPivotIndex(arr, first, last);
        quickSort(arr, first, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, last);
    }
}

function findPivotIndex(arr, first, last) {
    let pivot = arr[last];
    let i = first - 1;

    for (let j = first; j < last; j++) {
        if (arr[j] < pivot) {
            i++;
            swap(arr, i, j);
        }
```

```
    }

    i++;
    swap(arr, i, last); // Place pivot correctly
    return i;
}

function swap(arr, i, j) {
    let temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

Use Case:

- Suitable for sorting large datasets.
- Efficient on average due to O(n log n) performance.
- Common in libraries and system-level sorting functions.
- Works well when auxiliary memory is limited (in-place sort).

Interview Q&A:

Q: When should you use innerText vs innerHTML?

A:

- Best: O(n log n)
- Average: O(n log n)
- Worst: $O(n^2)$ – when pivot selection is poor (e.g., always picks smallest or largest element)

Q: When should you use innerText vs innerHTML?

A: No, Quick Sort is not stable, meaning it may change the relative order of equal elements.

Cyclic Sort

Definition:

Cyclic Sort is an index-based sorting algorithm used for arrays containing integers from 1 to n with no duplicates. It places elements directly at their correct index positions in linear time.

Code(Examples):

```
let arr = [8, 5, 7, 2, 1, 3, 4, 6];
let i = 0;

while (i < arr.length) {
    let correctIdx = arr[i] - 1;

    if (arr[i] != arr[correctIdx]) {
        let temp = arr[i];
        arr[i] = arr[correctIdx];
        arr[correctIdx] = temp;
    } else {
        i++;
    }
}

console.log(arr); // Output: [1, 2, 3, 4, 5, 6, 7, 8]
```

Use Case:

- Ideal for arrays with numbers in the range 1 to n and no duplicates.
- Often used in problems like finding missing numbers, duplicates, or mismatched indices in a sequence

Interview Q&A:

Q1: What is the time complexity of Cyclic Sort?

A:

- Time: O(n) – because each element is placed at its correct position with at most one swap
- Space: O(1) – in-place sorting

Q2: Is Cyclic Sort a stable algorithm?

A: No, Cyclic Sort is not stable.

Q3: Why is Cyclic Sort faster than other sorting methods for 1 to n arrays?

A: Because it places each number directly at its correct index using the property arr[i] - 1, minimizing operations and avoiding recursion.