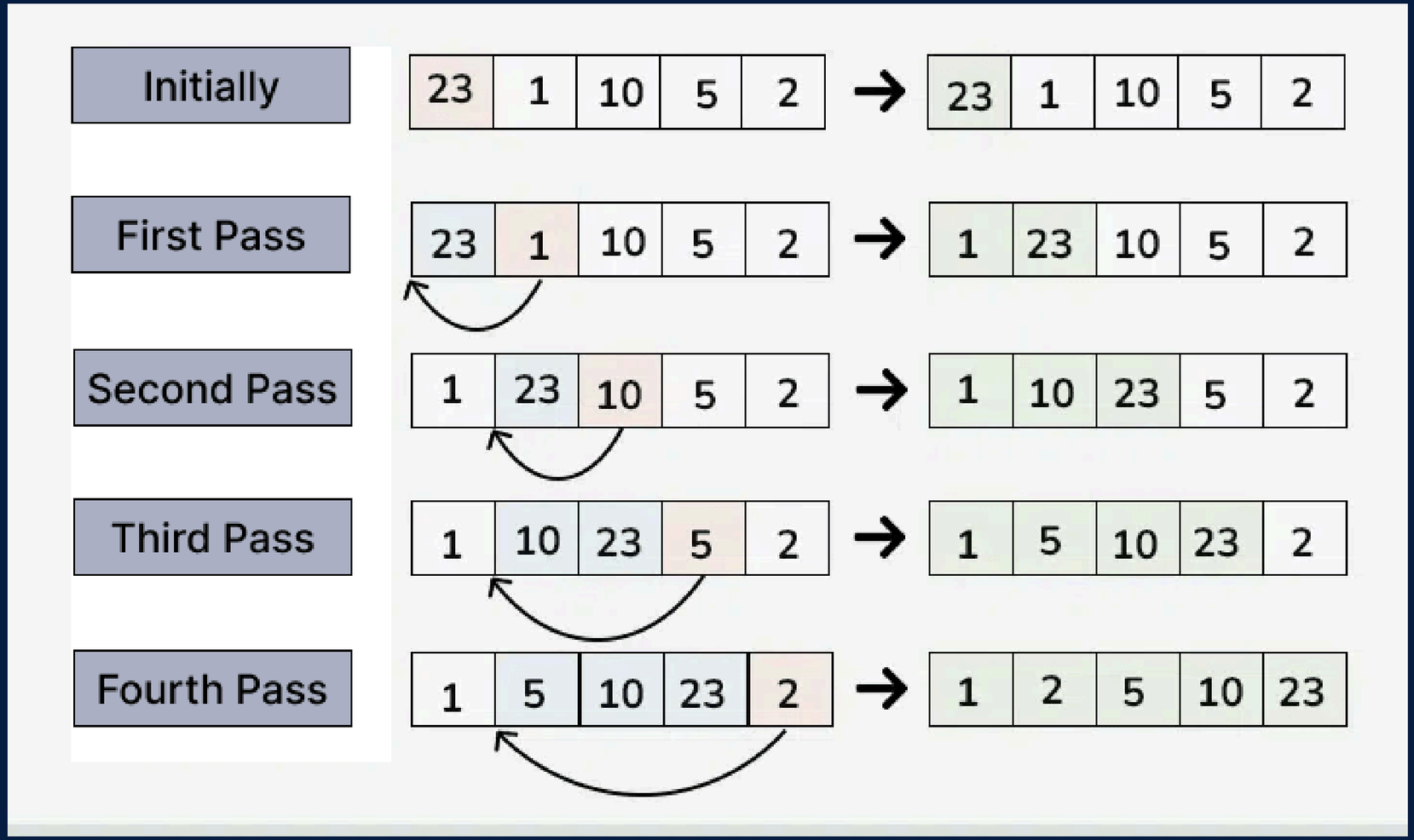
Competitive programming 101
SORTING

By: Wafae Boumajjane & Hamza Bouali

INSERTION SORT

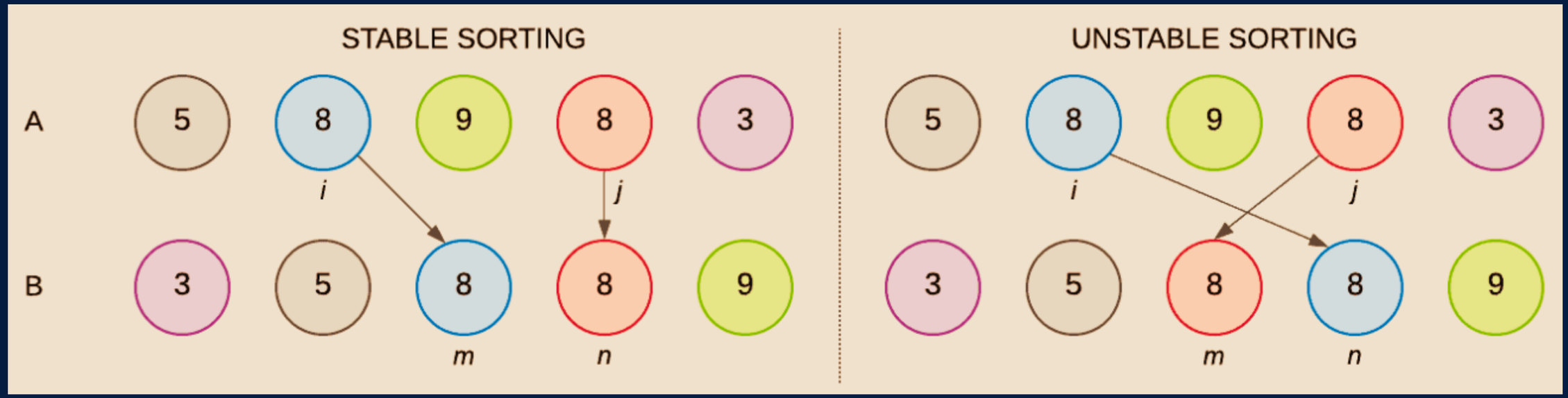


INSERTION SORT

Nested Loop

```
for (i from 1 to arr.length){
    j = i-1
    while (j >= 0 and arr[j+1] < arr[j]){
        tmp = ar[j+1]
        arr[j+1] = arr[j]
        arr[j] = tmp
        j -= 1
    }
}
```

STABLE AND UNSTABLE



So is the insertion sort stable or unstable ?

TIME COMPLEXITY

So what is the time complexity ?

Best and Worst case scenarios ?

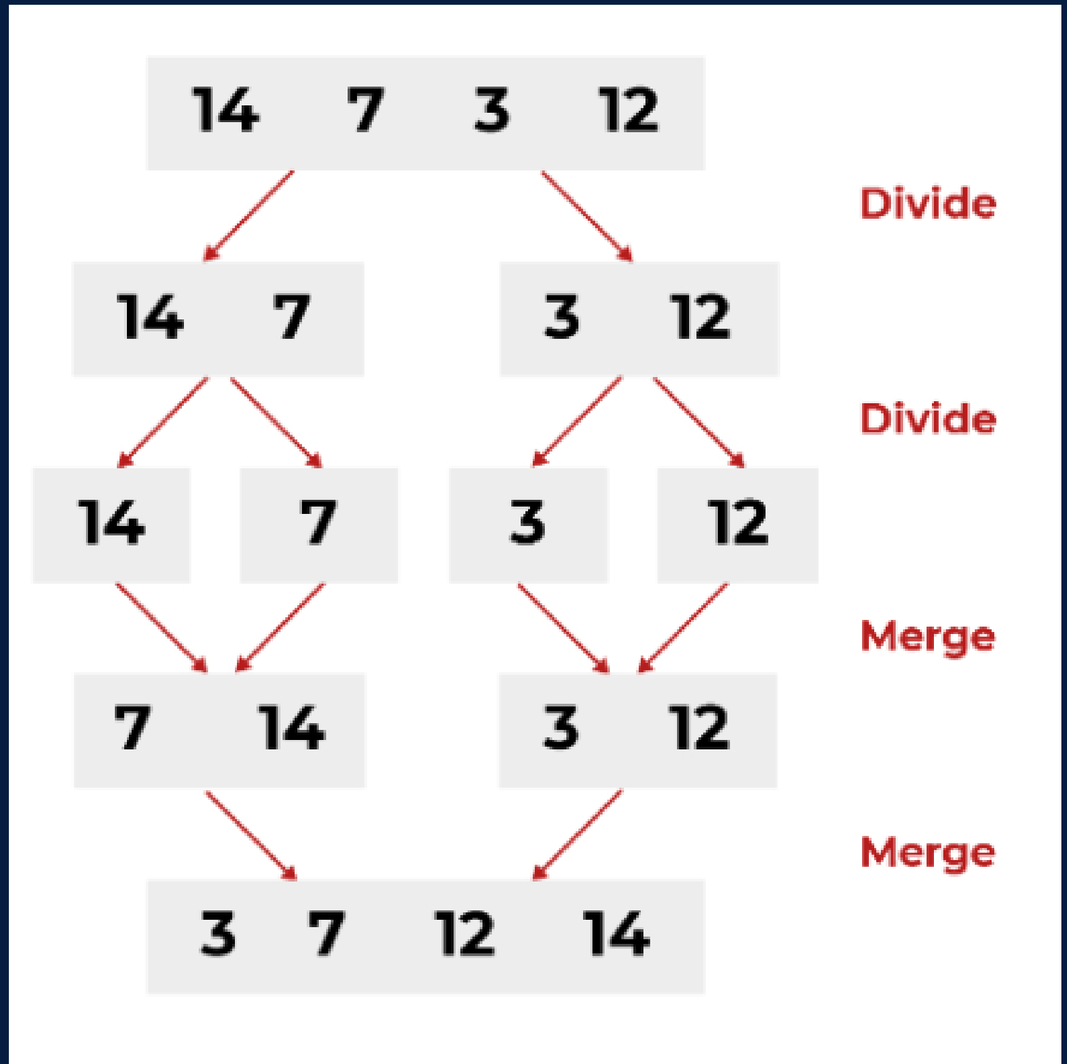
TIME COMPLEXITY

- Best case: O(n) , If the list is already sorted, where n is the number of elements in the list.
- Worst case: $O(n^2)$, If the list is in reverse order

CODE

MERGE SORT

Two branch decision tree



MERGE SORT

```
int[] mergeSort(int[] arr, int s, int e){
    if (e - s +1 <= 1)
        return arr;

    m = (s+e)/2;
    mergeSort(arr, s, m);
    mergeSort(arr, m+1, e);

    merge(arr, s, m, e);

    return arr;
}
```

MERGE SORT

Stable or unstable?

TIME AND SPACE COMPLEXITY

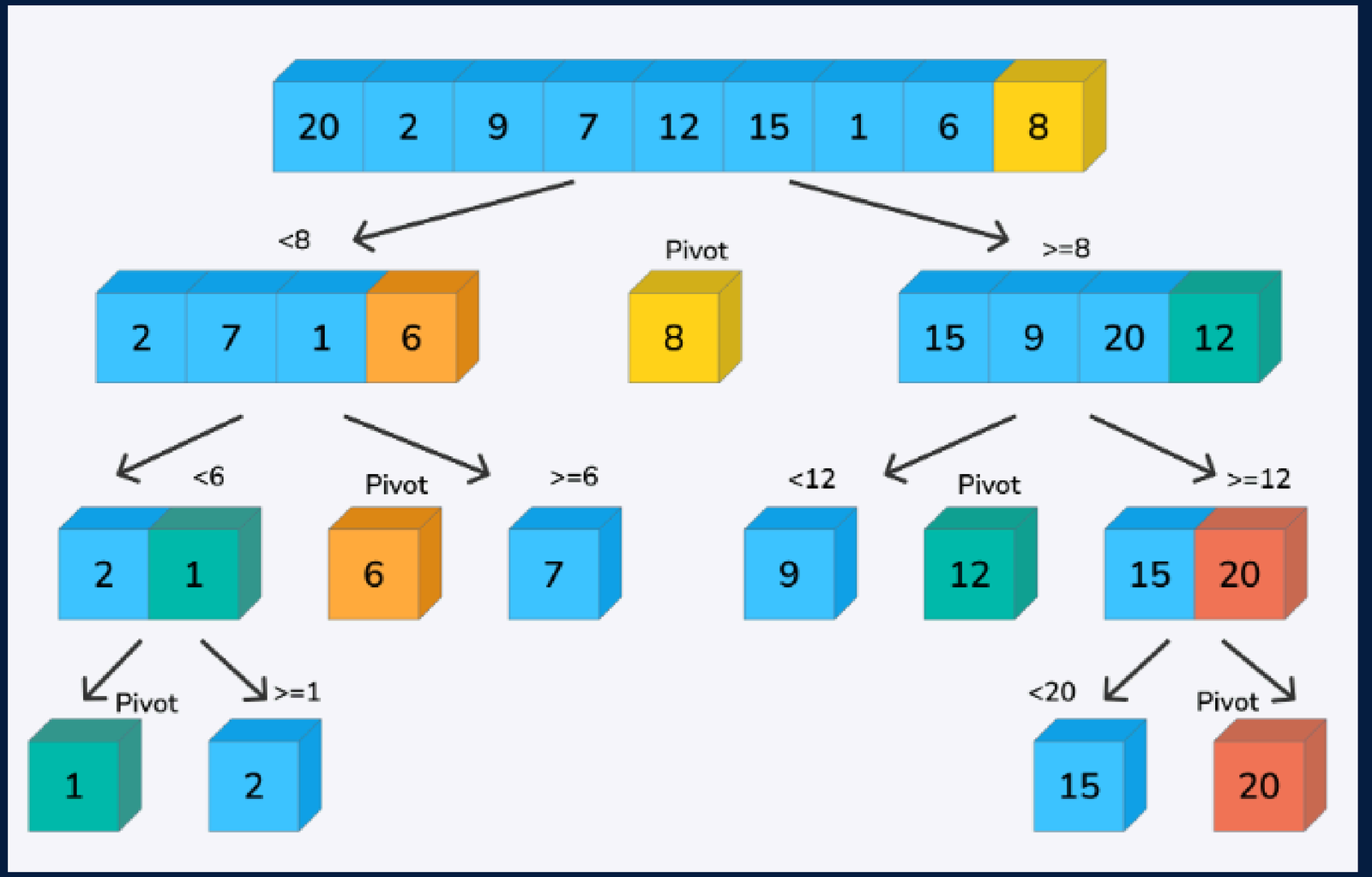
So what are the time and space complexities ?

A liittlee bit of math

TIME AND SPACE COMPLEXITY

- Time Complexity: O(n log n), When the array is already sorted or nearly sorted.
- Auxiliary Space: O(n), Additional space is required for the temporary array used during merging.

QUICK SORT


20 2 9 7 12 15 1 6 8
<8
Pivot
>=8
2 7 1 6
8
15 9 20 12
<6
Pivot
>=6
<12
Pivot
>=12
2 1
6
7
9
12
15 20
Pivot
>=1
<20
Pivot
1
2
15
20

QUICK SORT

```
int[] quickSort(int[] arr, int s, int e){
    if (e-s+1 <= 1)
        return arr
    pivot = arr[e]
    //Elements smaller than pivot on left side
    for i from s to e {
        if (arr[i] < pivot){
            tmp = arr[left]
            arr[left] = arr[i]
            arr[i] = tmp
            left += 1
        }
    }
    //Move pivot in between left & right sides
    arr[e] = arr[left]
    arr[left] = pivot

    quickSort(arr, s, left-1)
    quickSort(arr, left+1; e)
    return arr
}
```

CODE

TIME COMPLEXITY

So what is the time complexity ?

Best and Worst case scenarios ?

TIME COMPLEXITY

- Best case: O(n log n) , Occurs when the pivot element divides the array into two equal halves.
- Average Case O(n log n), On average, the pivot divides the array into two parts, but not necessarily equal.
- Worst case: $O(n^2)$, Occurs when the smallest or largest element is always chosen as the pivot (e.g., sorted arrays)

QUICK SORT

Stable or unstable?

PRACTICE

Implement the 3 sorts