DIVIDE AND CONQUER

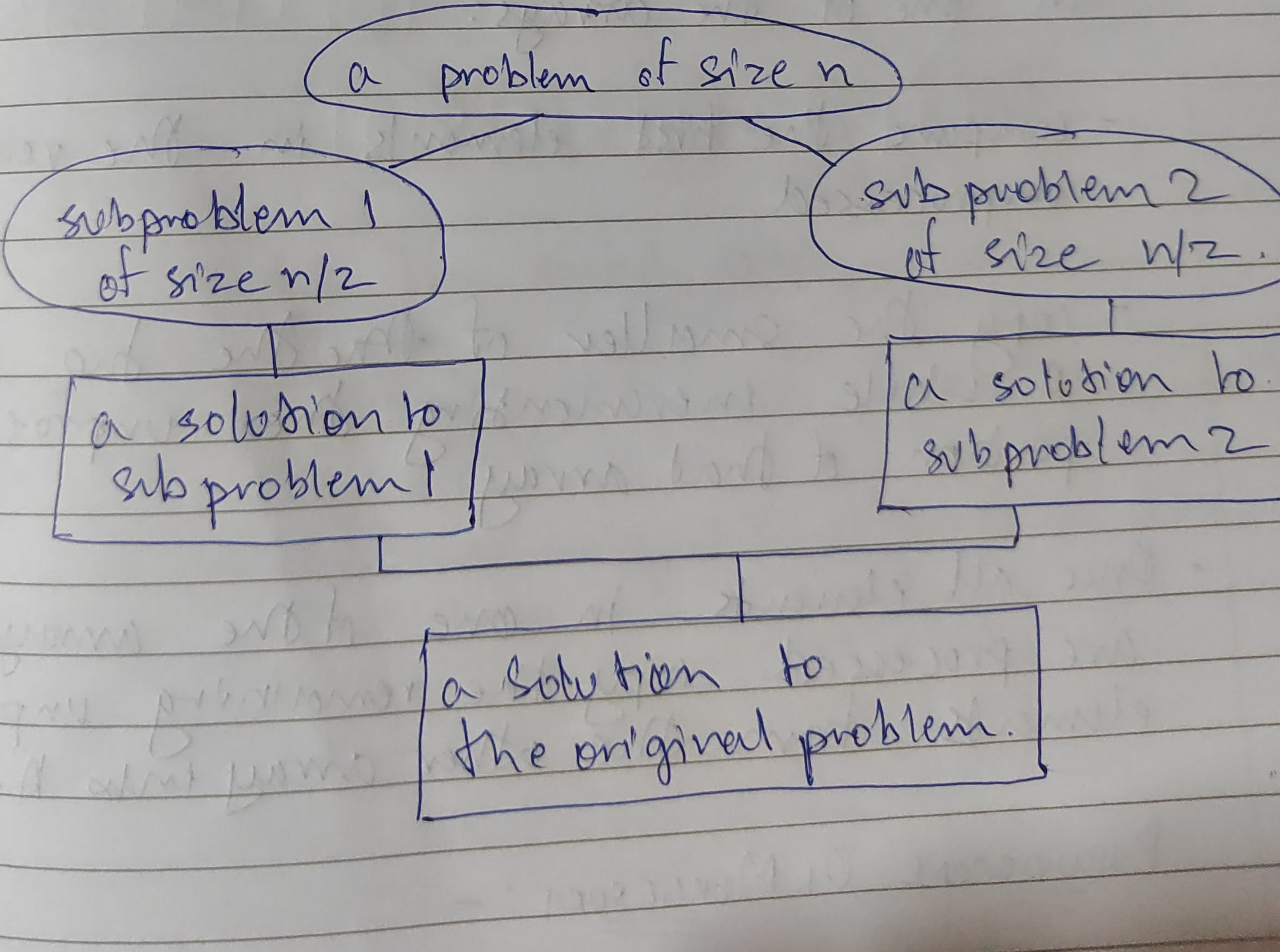
The most well known algorithm design strategy

1. Divide instance of problem into two or more smaller instances.

2. Solve smaller instances recursively

3. Obtain solution to original (larger) instance by combining these solutions.

DIVIDE AND CONQUER TECHNIQUE.


a problem of size n
subproblem 1 of size n/2
sub problem 2 of size n/2.
a solution to sub problem 1
a solution to sub problem 2
a Solution to the original problem.

MERGESORT

- Split array A [0..n-1] in two about equal halves and make copies of each half in arrays B and C

- Sort arrays B and C recursively.

- Merge sorted arrays B and C into array A as follows

  • Repeat the following until no element remain in one of the arrays:

    - compare the first elements in the remaining unprocessed

    - copy the smaller of ~~the~~ the two into A while incrementing the unprocessed portion of that array

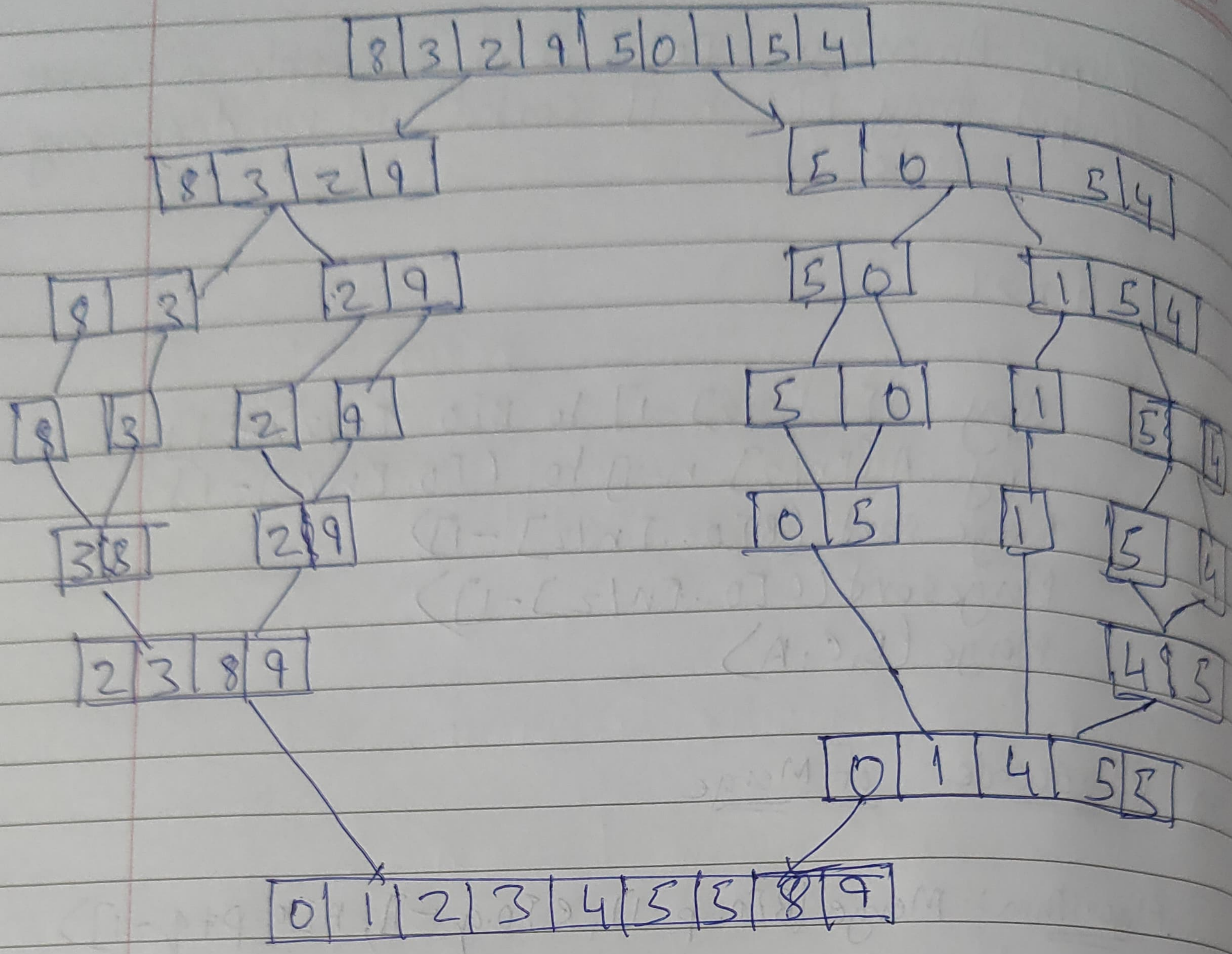
  • Once all elements in one of the arrays are processed copy the remaining unprocessed elements from the other array into A.

PSEUDOCODE OF MERGESORT :-

```
ALGORITHM :- Mergesort (A [0..n-1])
    // Sort array A[0..n-1] by recursive mergesort.
// Input: An array A[0..n-1] of orderable elements
// Output: Array A[0..n-1] sorted in nondecreasing
           order.

If n>1

    copy A[0..[n/2]-1] to B[0..[n/2]-1]
    copy A[[n/2]..n-1] to C[0..[n/2]-1]
    Merge sort (B[0..[n/2]-1])
    Merge sort (C[0..[n/2]-1])
    Merge (B, C, A)
```

→ **Pseudocode of Merge**

```
Algorithm:- Merge (B[0..p-1], C[0..q-1], A[0..p+q-1])

    // Merges two sorted arrays into one
       sorted array.
    // Input: Arrays B[0..p-1] and C[0..q-1] both
              sorted.
    // Output: Sorted array A[0..p+q-1] of the
               elements of B and C
        i ← 0; j ← 0; k ← 0

    while i<p and j<q do.
     if B[i] ≤ C[j]
         A[k] ← B[i]; i ← i+1
     else A[k] ← C[j]; j ← j+1
        k ← k+1
     if i=p, copy C[j..q-1] to A[k..p+q-1]
      else copy B[i..p-1] to A[k..p+q-1].
```


8 3 2 9 5 0 1 5 4
8 3 2 9
5 0 1 5 4
8 3
2 9
5 0
1 5 4
8
3
2
9
5
0
1
5
4
3 8
2 9
0 5
1
5
4
2 3 8 9
4 5
0 1 4 5 5
0 1 2 3 4 5 5 8 9