Resources:

[https://github.com/donnemartin/system-design-primer#company-engineering-blogs](https://github.com/donnemartin/system-design-primer%252525252525252523company-engineering-blogs)

<https://haseebq.com/how-to-break-into-tech-job-hunting-and-interviews/>

Algorithms:

* For recursive solution to count sum ALWAYS use the weight+<recursive call>
* If it needs to check if we reach leaves then pass the sum as additional argument
* For DP use recursive solution with memoization.
* For DP if it needs to find optimistic sum - it has to use range [-sum , +sum] in the dp table
* For recursive call it needs to make as number calls as there is branches in algorithms.
* For DP if it needs to count something we can use as LIS approach Lis[i] = max(Lis[i], Lis[j] + 1) where j < I and we can’t consider any variants and our previous values is based on earlier one.
* For DP if it needs to count some optimistic difference we can use the table dp as dp[arr[i]] = dp[arr[j]- diff] + 1
* For recursive complete search we can use a value in array as indicator of use this value: at the beginning we set up value in an array to same neutral and at the end we reset it back.
* If need to find some number we can try to search this value with binary search.
* Use binary search to find some value and use some functionality to check if this value is appropriate.
* For combination:
* 0. If len(cur) == n: append it to result list

1. Call method recursively with i +1
2. Create new list appending i-th item
3. Call method recursively with i + 1 and new list

- If we use recursive solution to find combination number for restriction of some item in combination we can use array with setting up item frequency. If we do the same for consecvative restriction we will use argument.

- to check if one of numbers is less than 0 to use Xor for logical operations: a < 0 xor b < 0.

* to count factors name we use cycle to sqrt(n) + 1 and if a % b=0 than factor+=1 and a // b != b factors += 1. For squared N we have odd factor’s number.
* If we work with number and it needs to sort item we can try to use bucket sort with bucket as our number
* If need to evaluate item in sequence : find difference between them We can use pattern "stack".
* To find middle of the list it need to have two pointers: slow and fast.Fast is slow.next.next. then when fast one reaches the end the slow will points to middle of the list.
* To valid sequence (,) we can use counter by increasing it if we meet ( and decrease it otherwise.
* To get sorted sequence we can use stack put stack item and pop up it if it’s less/greater than current item.
* If we calculate min, max consecutive sum let’s use prefix sum and calculateboth variant at once. sum[i+1] - sum[i] if sum[0] = a[0]

Use a & (-a) to get rightmost bit 1, use bitmask having xored result of number without duplicates in array, then use a & (-a) xoring only items with bit 1, then we get 2 numbers (x , x ^ bitmask)