Resources:

<https://github.com/donnemartin/system-design-primer#company-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 memorization.
* 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