**DSA Patterns.**

𝟭. 𝗧𝘄𝗼 𝗣𝗼𝗶𝗻𝘁𝗲𝗿𝘀:  
  
- Merge Two 2D Arrays by Summing Values→ <https://lnkd.in/eXvrAqWb>  
- Merge Sorted Array→ <https://lnkd.in/eC3G62RG>  
- Sort Array by Parity→ <https://lnkd.in/eynqARXQ>  
- Sort Array by Parity II→ <https://lnkd.in/eK_QtUuv>  
- Rearrange Array Elements by Sign→ <https://lnkd.in/ek4BFtgT>  
- Remove Duplicates from Sorted Array→ <https://lnkd.in/e7deX-a5>  
- Remove Element→ <https://lnkd.in/e_GkgjCe>  
- Partition Array According to Given Pivot→ <https://lnkd.in/e5jEEbV6>  
- Rotate Array→ <https://lnkd.in/ewG8uGJg>  
- Apply Operations to an Array→ <https://lnkd.in/eji38k8D>  
- Find All K-Distant Indices in an Array→ <https://lnkd.in/eUqSKHkM>  
- Two Sum→ <https://lnkd.in/eaC5hfaa>  
- 3Sum→ <https://lnkd.in/eS4usEnF>  
- 3Sum Closest→ <https://lnkd.in/eVZJfjua>  
- 4Sum→ <https://lnkd.in/eduk3Tks>  
- Sort Colors→<https://lnkd.in/eM7FTf3s>  
- Container With Most Water→<https://lnkd.in/eNgDn4Ce>  
- Watering Plants II→<https://lnkd.in/e5xujpGH>  
- Next Permutation→<https://lnkd.in/eJry4e2n>  
- Next Greater Element III→<https://lnkd.in/en23T757>  
  
  
→ 2 Pointers on Strings  
Don’t skip these either:  
  
- Reverse String→<https://lnkd.in/esmGPN6i>  
- Reverse Prefix of Word→<https://lnkd.in/etVjF63e>  
- Reverse Vowels of a String→<https://lnkd.in/eV43Q--K>  
- Reverse Words in a String→<https://lnkd.in/eu2JSHUZ>  
- Reverse Words in a String III→<https://lnkd.in/ewi34Qev>  
- Valid Palindrome→<https://lnkd.in/e36HfMg3>  
- Valid Palindrome II→<https://lnkd.in/e32MME4r>  
- Lexicographically Smallest Palindrome→<https://lnkd.in/eaMF9tZa>  
- Merge Strings Alternately→<https://lnkd.in/eAHxRwm5>  
- Largest Merge of Two Strings→<https://lnkd.in/enrz8GRA>  
- Shortest Distance to a Character→<https://lnkd.in/eXZbpUPz>  
- DI String Match→<https://lnkd.in/e-j9qAHc>  
- Make String a Subsequence Using Cyclic Increments→<https://lnkd.in/e8uQvFG2>  
- Count Binary Substrings→<https://lnkd.in/ewKQy_GF>  
- Minimum Length of String After Deleting Similar Ends→<https://lnkd.in/etXGbT7f>  
- String Compression→<https://lnkd.in/e58HVg8S>  
- Separate Black and White Balls→<https://lnkd.in/euuvp_Ey>  
- Move Pieces to Obtain a String→<https://lnkd.in/ebqBUCUG>  
- Sentence Similarity III→<https://lnkd.in/eGUuqmsE>

𝟮. 𝗜𝗻𝘁𝗲𝗿𝘃𝗮𝗹𝘀:

[Non-overlapping Intervals](https://leetcode.com/problems/non-overlapping-intervals/) -> <https://leetcode.com/problems/non-overlapping-intervals/description/>  
Merge Intervals -> <https://leetcode.com/problems/merge-intervals/>

[Insert Interval](https://leetcode.com/problems/insert-interval/) -> <https://leetcode.com/problems/insert-interval/description/>

- <https://lnkd.in/eGv_iaGZ>  
 - <https://lnkd.in/e2i7354g>

𝟯. 𝗔𝗿𝗿𝗮𝘆:

- <https://lnkd.in/erqPMHrx>  
 - <https://lnkd.in/dz99BXqf>  
 - <https://lnkd.in/dASWh_wV>

**𝟰. 𝗗𝘆𝗻𝗮𝗺𝗶𝗰 𝗣𝗿𝗼𝗴𝗿𝗮𝗺𝗺𝗶𝗻𝗴:**

**(i).DP on sub sequences:**

**(ii).0/1 Knapsack:**

**(iii). 1D DP:**

**Here, you have to apply DP on 1D array. Below are some common examples:**[**https://leetcode.com/problems/climbing-stairs/description/**](https://leetcode.com/problems/climbing-stairs/description/)

[**https://www.geeksforgeeks.org/problems/geek-jump/1**](https://www.geeksforgeeks.org/problems/geek-jump/1)

[**https://www.geeksforgeeks.org/problems/minimal-cost/1**](https://www.geeksforgeeks.org/problems/minimal-cost/1)

**(iv).2D/3D DP:**

**Here, you have to apply DP on 2D/3D matrices. Below are some common examples:**[**https://www.geeksforgeeks.org/problems/geeks-training/1**](https://www.geeksforgeeks.org/problems/geeks-training/1)

[**https://leetcode.com/problems/unique-paths/description/**](https://leetcode.com/problems/unique-paths/description/)

**-** [**https://lnkd.in/euzKZUSF**](https://lnkd.in/euzKZUSF) **-** [**https://lnkd.in/eEGGuzZZ**](https://lnkd.in/eEGGuzZZ) **𝟱. 𝗗𝗙𝗦-𝗕𝗙𝗦:  
 -** [**https://lnkd.in/euvQM2R3**](https://lnkd.in/euvQM2R3) **-** [**https://lnkd.in/eYNV-u6H**](https://lnkd.in/eYNV-u6H) **-** [**https://lnkd.in/e-RCFMkC**](https://lnkd.in/e-RCFMkC) **-** [**https://lnkd.in/efMF4sWe**](https://lnkd.in/efMF4sWe) **𝟲. 𝗕𝗶𝗻𝗮𝗿𝘆 𝗦𝗲𝗮𝗿𝗰𝗵:**finding a presence of an element in an array:

[Binary Search](https://leetcode.com/problems/binary-search/):<https://leetcode.com/problems/binary-search/description/>  
[**https://lnkd.in/eRyYVw9E**](https://lnkd.in/eRyYVw9E)  
 **-** [**https://lnkd.in/dp4NUqcg**](https://lnkd.in/dp4NUqcg) **-** [**https://lnkd.in/dTDhWNqi**](https://lnkd.in/dTDhWNqi) **𝟳. 𝗧𝗿𝗲𝗲 𝗧𝗿𝗮𝘃𝗲𝗿𝘀𝗮𝗹:  
 -** [**https://lnkd.in/egpV9fyR**](https://lnkd.in/egpV9fyR) **-** [**https://lnkd.in/eH6PsJC7**](https://lnkd.in/eH6PsJC7) **-** [**https://lnkd.in/dhfbbMAx**](https://lnkd.in/dhfbbMAx)

𝟵. 𝗕𝗮𝗰𝗸𝘁𝗿𝗮𝗰𝗸𝗶𝗻𝗴:

**Backtracking is an extended/optimized version of recursion where you control decisions, restore results after performing an operation. Backtracking also known as controlled recursion.**

**Backtracking explanation:** [**https://medium.com/leetcode-patterns/leetcode-pattern-3-backtracking-5d9e5a03dc26**](https://medium.com/leetcode-patterns/leetcode-pattern-3-backtracking-5d9e5a03dc26)

**(i). Subsets: in this pattern, you will be given a array of characters, numbers or string and will be asked to print all the subsets of a stream. No duplicates are allowed.**

[**https://leetcode.com/problems/subsets/description/**](https://leetcode.com/problems/subsets/description/)

**(ii). Subsets II: in this pattern, you will be given characters, numbers or string that has duplicates and will be asked to print all the subsets of a stream. No duplicates are allowed. For this pattern, implementation will be same as first pattern but you have to implement a logic to avoid duplicates by performing below steps:  
Sort a stream.**

**If adjacent elements are same, skip an iteration.**

[**https://leetcode.com/problems/subsets-ii/**](https://leetcode.com/problems/subsets-ii/)

**(iii). Combination sum:** in this pattern,we are given an array of integers and target sum. We have to print subsets of an array that has sum equals target sum. You can choose same element multiple times.   
  
<https://leetcode.com/problems/combination-sum/>  
<https://leetcode.com/problems/letter-combinations-of-a-phone-number/description/>

**(iv). Combination Sum II:** in this pattern,we are given an array of integers and target sum. We have to print subsets of an array that has sum equals target sum. we are not allowed to use one number twice.  
  
[**https://leetcode.com/problems/combination-sum-ii/description/**](https://leetcode.com/problems/combination-sum-ii/description/) **(v) Permutations I:** in this pattern, we are given an array of integers or string. We have to print all its permutations.  
  
<https://leetcode.com/problems/permutations/description/>  
 **(vi) Permutations II:** in this pattern, we are given an array of integers or string with duplicates. We have to print all its unique permutations. We have to add a logic to avoid swapping duplicates.

[**https://leetcode.com/problems/permutations-ii/description/**](https://leetcode.com/problems/permutations-ii/description/)

**(vii) Palindrome Partitioning: in this pattern, you will be given a string you have to find all possible palindromic partitions.**[**https://leetcode.com/problems/palindrome-partitioning/description/**](https://leetcode.com/problems/palindrome-partitioning/description/)

**Useful Links:  
 -** [**https://lnkd.in/e7c6XbdG**](https://lnkd.in/e7c6XbdG) **-** [**https://lnkd.in/eGyS6q3R**](https://lnkd.in/eGyS6q3R) **-** [**https://lnkd.in/eTZDncpG**](https://lnkd.in/eTZDncpG)

**Sliding Window:**

1.There is one pattern where you need to shift window from left to right based on conditions. Here, questions would be finding longest substring without repeating characters, length of subarray containing almost 2 distinct numbers.  
  
[Longest substring without repeating characters:](https://leetcode.com/problems/longest-substring-without-repeating-characters/description/)

[Max consecutive ones III:](https://leetcode.com/problems/max-consecutive-ones-iii/description/)

[Find length of the longest subarray containing atmost two distinct integers:](https://www.geeksforgeeks.org/problems/fruit-into-baskets-1663137462/1)

2. There is a pattern where you have to get count of subarrays, substrings like number of substrings containing all 3 chars,   
  
[Number of Substrings Containing All Three Characters](https://leetcode.com/problems/number-of-substrings-containing-all-three-characters/):

**Two Pointer:**

**Graphs:**

1. **BFS:**

BFS is a traversal technique in graph. BFS do visit closest nodes first and then visit closest of closest.

<https://www.geeksforgeeks.org/problems/detect-cycle-in-an-undirected-graph/1>

1. **DFS:**

DFS traverses graph path-by-path. It picks one path and traverse thoroughly. Then it picks next path. DFS is good fit for problems where you have to find connected components.

Problems:

<https://leetcode.com/problems/number-of-provinces/description/>

1. **Flood Fill:**

Flood fill is used to traverse graph stored in matrix. Here, graph moves in 8 directions.by adding and substracting i and j, you will move in cells.

Problems:

<https://leetcode.com/problems/flood-fill/description/>

1. **Flood Fill (multi-source BFS):**

there is one more pattern where you have to start from multiple sources, here you can use multisource BFS where you can start BFS from multiple nodes.  
Problems:  
<https://leetcode.com/problems/rotting-oranges/description/>

<https://leetcode.com/problems/01-matrix/description/>

<https://leetcode.com/problems/map-of-highest-peak/description/>  
  
1. Two Pointer Problems:  
 - [**https://lnkd.in/dK\_fB-Eg**](https://lnkd.in/dK_fB-Eg)  
  
2. Backtracking Pattern:  
 - [**https://lnkd.in/dDGsdfps**](https://lnkd.in/dDGsdfps)  
  
3. Dynamic Programming Patterns:  
 - [**https://lnkd.in/dX7a4cau**](https://lnkd.in/dX7a4cau)  
  
4. Dynamic Programming Patterns 2:  
 - [**https://lnkd.in/db2tAp27**](https://lnkd.in/db2tAp27)  
  
5. Powerful Ultimate Binary Search Template:  
 - [**https://lnkd.in/dxk7kdeb**](https://lnkd.in/dxk7kdeb)  
  
6. A general approach to backtracking questions:  
 - [**https://lnkd.in/drsHxsZh**](https://lnkd.in/drsHxsZh)  
  
7. Binary Tree Traversal & Views:  
 - [**https://lnkd.in/dxGcKx65**](https://lnkd.in/dxGcKx65)  
  
8. Graph For Beginners [Problems | Pattern | Sample Solutions]:  
 - [**https://lnkd.in/dkpyiB3R**](https://lnkd.in/dkpyiB3R)  
  
9. A comprehensive guide and template for monotonic stack based problems:  
 - [**https://lnkd.in/dtmFMzDJ**](https://lnkd.in/dtmFMzDJ)  
  
10. All Types of Patterns for Bits Manipulations and How to use it:  
 - [**https://lnkd.in/d-rfVNx2**](https://lnkd.in/d-rfVNx2)  
  
11. Collections of Important String questions Pattern:  
 - [**https://lnkd.in/dCy\_j-vw**](https://lnkd.in/dCy_j-vw)  
  
12. Leetcode Pattern 1 | BFS + DFS == 25% of the problems:  
 - [**https://lnkd.in/dtaEpzrC**](https://lnkd.in/dtaEpzrC)  
  
13. Template that can solve most 'substring' problems:  
 - [**https://lnkd.in/dEbVbBu4**](https://lnkd.in/dEbVbBu4)  
  
14. C++ Maximum Sliding Window Cheat sheet Template:  
 - [**https://lnkd.in/dPiMzzpA**](https://lnkd.in/dPiMzzpA)

2. Skipping mock interviews - Solving problems is one thing, explaining them under pressure is another. Try these free mock interviews Websites and avoid end day preparation.  
  
1- https://www.remasto.com/   
2- Interviewbit.com  
3- https://www.pramp.com/  
4- https://lnkd.in/gD2PXXvm  
5- https://lnkd.in/gFFWX2qA  
  
3. Ignoring time complexity - Recruiters care as much about your thought process as your solution. Top 10 interview experience to have an idea how to deliver what you think.  
  
1-https://lnkd.in/gkXGMq3C  
2-https://lnkd.in/gFsaydjj  
3-https://lnkd.in/gACnXiMH  
4-https://lnkd.in/gv5a62Sq  
5-https://lnkd.in/gej3Kyxt  
6-https://lnkd.in/giddDSCS  
7-https://lnkd.in/g9FZbrEK  
8-https://lnkd.in/gHdhsMgT  
9-https://lnkd.in/gHa4B-wz  
10-https://lnkd.in/gxD3\_g9x  
  
Get some amazing resources - https://lnkd.in/gf87BYSe