**Top 75 questions**

[**https://leetcode.com/list/xi4ci4ig/**](https://leetcode.com/list/xi4ci4ig/)

**Pep Level code:**

[**https://github.com/rajneeshkumar146/pepcoding-Batches/tree/master/2022**](https://github.com/rajneeshkumar146/pepcoding-Batches/tree/master/2022)

**Java tips:**

**Lists**

List<List<Integer>> result=new ArrayList<List<Integer>>(); // creating list of list

List<Integer> tempResult=new ArrayList<Integer>(); // creating list

result.add(new ArrayList<>(tempResult)); //adding list to a list object

* To get minimum , use

int min = (int) 1e9;

**instead of**

int min = Integer.MAX\_VALUE;

because it gives buffer.

* How to **use dictionary** (HashMap) in Java

HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();

for(int i : nums){  
 if(map.containsKey(i)){  
 map.put(i, map.get(i) + 1);  
 }else{  
 map.put(i,1);  
 }

}

for(int i : nums){  
 if(map.get(i) == 1){  
 return i;  
 }  
 }

* **returning a list directly:**

return new int[] {2,3};

//this will return a list of int[] with two items: 2 and 3.

**Recursion**:

1. Euler for recursion: https://www.youtube.com/watch?v=R7qja\_gZrvI

left print -> Left call -> print -> Right call

2. Use Faith and Expectation:

Faith is if run(n) will do its job, than run(n-1) will do its job automatically and than add a base case

* **Differnece between Flood fill and Maze Path**  is :

In maze path - you move in specific direction, but in flood fill you can move in **any direction** to reach from one point to other, so you have to manage the **visited cells**(mark them as true before loop and mark them as false after loop ends) in flood fill to avoid crossing the same cell.

* **Combination**- hamesha forward move karna hota

**Arrangement/Permutation**- direction backward kar sakte ho beech mai

E..g say string is **abcde …**you haveto find combination so possible combination can be :

ab, abc, abce -- these are all combinations because we are moving in same direction

**Another example:**

Permutation of two things from three given things a, b, c is ab, ba, bc, cb, ac, ca

Combination of two things from three given things a, b, c is ab, bc, ca

**If the order is important**, **then the problem is related to permutation**, and the possible number of samples will be, XY, YX, YZ, ZY, XZ, ZX. In this case, XY is distinct from the sample YX, YZ is distinct from the sample ZY and XZ is distinct from the sample ZX.

If the order is unnecessary, then the question is relevant to the combination, and the possible samples will be XY, YZ and ZX.

**Point:** Permutations are always more than Combinations

However, abec, abdc - are arrangements (not combinations) because they are not forward looking in 1 direction

* **For question –CoinChange\_GetMinCoins\_LC322 (Leetcode 322)**

**Icon

Description automatically generated with medium confidence**

Say coins are {2,3,5,7} and target sum is 10, we have to find minimum number of coins required

We start with 2 in the loop in solution code, remaining number is 8(10-2) – now our funciton will return minimum no. of coins required to get 8 (10-2).

We have to add 1 because we have used 1 coin already i.e 2.

Base condition return 0 since minimum coins required to get 0 is 0.

* You can get the row and column of a cell in a 2D matrix. By its serial number with the following logic

**int row = index / nums.length** (size of 2Darray)

**int row = index % nums.length** (size of 2Darray)

* You can also **store 2D array into 1D array:**

Now, for this r and c can be found using above forumales (r= idx / nums.length and c= idx % nums.length)  
But how would you find the idx:

using formula  **R\*N+C**  where N is number of columns.

**Note:** (Check Level up lecture of 1st Apr).

* **Working with charcters**

Below code will store occurences of each character in a string and than remove duplicate charcters and store the count.

int[] freq = new int[26];

        // Below for loop with store the occurences count of charcters in a

string in freq[] array

        // e.g. "sendmoremoney" will store [0,0,0,0,3.....] 3 because 'e'

occured 3 times

        for (int i = 0; i < superString.length(); i++) {

            freq[superString.charAt(i) - 'a']++;

        }

        superString = "";

        // Below For loop will remove the duplicate values and order them

alphabetically "demnorsy"

        for (int i = 0; i < 26; i++) {

            if (freq[i] > 0) {

                superString += (char) (i + 'a');

            }

}

       System.out.println(superString);

* **Bit manipulation**

1 byte= 8 bit

1 int = 4 byte= 32 bit

1 long = 8 byte = 64 bit

Most significant bit -> 1001 <- Least significant bit(LSB)

**^ (XOR)**

0 ^ 0 = 0

0 ^ 1 = 1

1 ^ 1 = 0

1 ^ 0 = 1

**! (Complement)**

!0 = 1

!1 = 0

**~ (1’s complement) – reverses the bits**

m = 100101

~m =011010

**-(2’s complement)**

Turns the number negative

**<< Left shift**

X=1011

X << 2 = 101100 (appends zeroes to the right and 2 digits from left will be dropped)

**>> Right shift**

X=1011

X >> 2 = 111011 (Right shift is speical here, if MSB is 1 than 1s will be appended on the left and if MSB is 0 than 0 will be appeneded and 2 bits from right will be dropped)

Y = 0011

X >> 2 = 000011

**-Getting the last set bit**

**n & (-n)**

* **Important Formula**

Shifting bits towards **left** by **n** digits -> will result in final number as **num\* (2^n)**

E.g.

5 is 101 -> shifting by 2 bits will result in 10100 - which is 5\*(2^2) = 20

Similarly , Shifting bits towards right by **n** digits -> will result in final number as **num/ (2^n)**

* **Tricks to find number of set bits(1) in a binary number**(See BitManiplationQuestions.java and video of 6th Apr )

1. Hamming weight method

Logic of hamming weght…when you AND a number ‘n’ with ‘n-1’ , number of set bits reduces by 1

So keep doing AND till all 1s becomes 0s.

Text, letter, whiteboard

Description automatically generated

Other appraoch is to AND with ‘1’ so what will happen that if you get 1 that means last most bit is 1, keep counting and keep ‘right shifting’ the number by 1.

All Palendromic partitions Euler diagram:

AllPallendromicPartitions\_LC131

Graphical user interface

Description automatically generated with low confidence

Original String is “**pepcocp”..** We take substrings which are palaledromic and start by taking first character which is also a pallndrome **e..g** We start with **pepcocp** whichbreak into “p” and “pep” (only pallendrome). When we take “p” - we pass rest of the string i.e. “epcocp” into recursive call. Again we take first bit i.e. “e” and now we don’t have any substring which is pallndrome

* Another way of creating array with values :

new int[] { a, b };