DSA Curriculum

Week 1: Basic of DSA

▼ Day 1: Fundamentals of Java [Operators, if-else, increment-decrement]

▼ Day 2: Loops [while loop, for-loop, do-while loop, switch]

▼ Day 3: 1-D Array [defining & initialising, iterations, looping on Arrays]

▼ Day 4: 1-D array continued [Solve More problems on Arrays]

In Class

- Sum and Mean

Post Class

- Replace Element
- Alternate Sum Product
- Is this repeated

In Class[Approach Discussed]

- Increasing Array

▼ Day 5: Functions [Significance, defining, function return type, passing arguments]

In Class

- Pokemon Master

Post Class

- Help Sherlock
- Penny and Charity
- Rotation Policy

▼ Contest For this week

▼ So many chocolates? (Contest)

- Hint 1 - Each guest is giving some amount of chocolates to Solo. We have to tell the total number of chocolates that Solo receives.
- Hint 2 - Add the chocolates he receives from each of the guests to get the total chocolates he receives.

▼ Divide (Contest)

- Hint 1 - We have to distribute N candies among M people such that each of them get equal number of candies.
- Hint 2 - Candies can be distributed if n%m is equal to zero.

▼ Happy Balloons (Contest)

- Hint 1 - Find the number of odd positions that have odd values stored at them and the number of even positions that have even values stored at them.
- Hint 2 - Check for the equality of a[i]%2 and i%2

▼ Strange Number (contest)

- Hint 1 - If the sum of digits is divisible by 9 then the number is divisible by 9.
- Hint 2 - Nth strange number is (N-1)th multiple of 9

▼ N-N (Contest)

- Hint 1 - We have to find Nth multiple of N considering 1 as first multiple.
- Hint 2 - (N-1)*N is the Nth multiple of N considering 1 as first multiple.

▼ Moving right (Contest)

- Hint 1 - We have to start jumping towards right starting from any index and have to find the maximum number of jumps keeping in mind the fact that we can jump to a building only if its height is less than or equal to the height of the current building.
- Hint 2 - Whenever height of current building is greater than the height of previous building then we have to start jumping from this index. We

can update the answer according to the number of jumps made till previous building.

- Hint 3 - We can have one counter for tracking the number of jumps made to a particular building. This counter is initialized to zero in the case when the height of current building is greater than the height of previous building.

Week 2: Sorting & Time complexity

Day 1: Time Complexity [best-worst-avg-case explaining through loops]

▼ Day 2: Simple Sorting algos [bubble,selection,insertion]

In Class

- Selection Sort
- Bubble Sort

Post Class

- Bubble Sort(Descending Order)

In Class [Approach Discussed]

- Insertion Sort

▼ Day 3: Recursion

In Class

- Power function

Post Class

- Fibonacci Numbers
- Sum of Digits
- Sum of product of digits of a given number

In Class[Approach Discussed]

- Factorial

▼ Day 4: Recursion Continued

In Class

- Tower of Hanoi

Post Class

- Number of ways
- Candy Crush

In Class[Approach Discussed]

- Welcome Problem

▼ Day 5: Merge-sort [algo explanation, time complexity discussion]

In Class

- Merge Sort

Post Class

- Shopping
- Even odd separate sorting

▼ Contest For this week

▼ Tribonacci Number: Easy-version (Recursion)

- Hint 1 - Any number in tribonacci series is equal to the sum of the preceding three elements. It is similar to the Fibonacci series where any number is equal to the sum of the preceding two elements.
- Hint 2 - Suppose T(n) gives the nth tribonacci number. Then the recurrence for finding T(n) is T(n) = T(n-1) + T(n-2) + T(n-3)

▼ Food Line (Group Contest April '21)

- Maximum number of people will be happy only when all the people standing in front of him take less time in taking food as compared to him.
- We can sort the array having the person with minimum time to take in the first position. For each index we need to know whether the prefix sum till previous index is less than the time required to take food by the person at this index.

▼ Easy - Peasy (Contest)

- Hint 1 - Arr[i] + Arr[j] will only be odd when one of them is odd and the other is even.
- Hint 2 - If in the given array both odd and even numbers are present then we can set any position for a given element through a number of swaps i.e we can make the array increasing which will be the lexicographically smallest array.

▼ Kill the monster(Contest)

- Hint 1 - Eliminate those weapons which are not heavier than D as they cannot be used to kill the monster.
- Hint 2 - Sort the weapons based on their attack powers in the descending order i.e. weapon with highest power must come first and so on. Then select first k weapons whose sum of attack power exceeds the health of the monster.

▼ Inversion Count

- Hint 1 - A simple solution would be for each array element count all elements less than it to its right and add the count to output. The complexity of this solution is O(n^2), where n is the size of the input.
- Hint 2 - This solution idea is based on the divide and conquer algorithm. Here, we will be dividing our array into two halves similar as in the merge sort algorithm. For each half, we will be getting the inversion counts using recursion. The total counts of inversion will be the sum of inversions in the first half, second half as well as the inversion counts during the process of merging. During the merge process, the inversion count is found by comparing elements of both the halves. We will use two pointers that will compare the indexes of both the array halves.

Week 3: Standard sorting algo continued and Searching

▼ Day 1: Quick-sort [algo explanation, time complexity discussion]

In Class

- Implementing Quick Sort

Post Class

- Sort 0's, 1's and 2's
- Maximum Force

▼ Day 2: Time Complexity Analysis of all sorting algos

▼ Day 3: Buffer day [implementation of sorting algos]

▼ Day 4: Linear Search, Binary Search

In Class

- Searching an element in a sorted array

Post Class

- Square root of an integer
- Minimum Element in Sorted and Rotated Array

▼ Day 5 - Binary Search Continued

In Class

- Shipping Parcels

Post Class

- Min Cut Tree
- K-sum

In Class[Approach Discussed]

- Kth Smallest Difference

▼ Contest For this week

▼ Is this repeated? (Contest)

- Hint 1 - We have to find whether 3 occurences of any number is present consecutively in the array or not.
- Hint 2 - We can iterate through 1 to N-2 and at each time we can check if the current element is equal to the next two elements or not.

▼ Pair sum (Contest)

- Hint 1 - A simple solution will be to consider all the pairs (a[i],a[j]) and then find the sum corresponding to max(a[i],a[j]). Time complexity of this soultion is O(n^2) where n is the length of the array.
- Hint 2 - We can sort the array. The maximum element in the array will be present at a[n] after sorting. It will contribute (n-1)*a[n] to the required sum as it will act as the maximum element for all the pairs (a[0],a[n]), (a[1],a[n]), ... ,(a[n-1],a[n]).

  Element at ith index of sorted array will contribute (i-1)*a[i] to the required sum as this element will be greater than all the elements present before this index and will act as the maximum while forming a pair with these elements.

▼ Weird Product (Contest)

- Hint 1 - A simple solution will be to take every number in the range [C,D] for each nubmer in range[A,B]. Time Complexity of this solution is O((B-A)*(D-C)).
- Hint 2 - If we look closely there will be 5 possible cases:-
  A<=0, B>=0 and C<=0, D>=0:- the answer will be maximum of A*C, B*D;
  B<=0 and C>=0:- answer will be B*C*
  *A>=0 and D<=0:- answer will be A*D
  B<=0 and D<=0:- answer will be A*C*
  *A>=0 and C>=0:- answer will be B*D

▼ Candy Love(contest)

- A simple solution to start from the candy having sweetness 1 and go till the candy having sweetness 1000000000 and stop at candy whose price is more than T. The candy previous to the candy where we stopped will be the required answer.
- Hint 2 - In the previous approach, we performed a linear search. We can optimize our solution by performing a binary search.
- Hint 3 - In binary search, we have to set low =1 and high = 1000000000 and if the price for the candy having sweetness

mid[(low+high)/2] is less than T then it means that we have to set low to mid+1, otherwise set high to mid-1.

▼ Play Play (Contest)

- Hint 1 - For each A[i], the player will visit the monster in fix time. For the first monster, the player will visit the monster in 1st sec, N+1, 2*N+1, and so on. For the second monster, the player will visit in the 2nd sec, N+2, 2*N + 2...
  For each monster, we can find the first occurrence at which the monster's health will be 0 and the player will visit the monster at that time. Then we just need to find the minimum among them. We can find the first occurrence at which the monster's health will be 0 through linear search.
  Total time complexity:- O(N*A[i]).
- Hint 2 - In order to optimize the solution, we can find the first occurrence at which the monster's health will be 0 through binary search.

  Total time complexity:- O(N*log(A[i]))

▼ Counters and Lines (Group Contest April '21)

- Hint 1 - We can perform linear search to find the maximum number of people that can line up in front of the K-th counter such that each counter gets at least one person and no counter worker is angry. We can start iterating from 1 and stop on the number n when n number of people cannot line up in front of the Kth counter. In this case maximum of n-1 number of people can line up in front of people.
- Hint 2 - In the previous approach we performed a linear search. We can optimise our solution by perfoming a binary search.
- Hint 3 - In binary search, we have to set low =1 and high = 1000000000 and if we can accommodate mid number of people on the kth counter then it means that we have to set low to mid, otherwise set high to mid.

Week 4: Advanced Data types and Intro to Basic Data Structures

▼ Day 1 - Multidimensional Arrays

In Class

- Max sum column

Post Class

- Good Cells
- Row with maximum 1's
- A Boolean Matrix Problem

In Class[Approach Discussed]

- Diagonal Sum

▼ Day 2 - Strings

In Class

- Odd Characters
- Is palindrome?

Post Class

- Palindrome
- String Sum
- Reverse

▼ Day 3 - Strings Continued

In Class

- Pangram Checking

Post Class

- Repeating character - First apperance leftmost
- Anagram
- Longest Distinct Characters in a string

In Class[Approach Discussed]

- Longest Common prefix in an array

▼ Day 4 - STL[Vector, Set, Iterator]

In Class

- Pair Sum in Vector

Post Class

- Addition of Common Elements
- Pair Sum Existence-Revisited
- Bubble Sort in pair Array

▼ Day 5 - STL continued.[Map, Iterator]

In Class

- Max freq

Post Class

- Subarray with Given Sum
- Largest subarray of 0's and 1's
- Maximum subarray sum modulo M

In Class[Approach Disussed]

- Remove duplicates from array

▼ Contest For this week

▼ Smallest String There Is (Group Contest: April '21)

- Hint 1 - The lexicographically minimum string of length n we can have when the adjacent characters can be the same is aaaaa...(n times).
- Hint 2 - The lexicographically minimum string of length n we can have when the adjacent characters are not same is abab....

▼ Red (Contest)

- Hint 1 - Check if all the three characters 'r', 'g', and 'b' are present in the string or not.
- Hint 2- Use map to check the condition in Hint 1.

▼ Flips (Contest)

- Hint 1 - The only thing we need to check is "vowel-odd pair" as the rest combinations are possible. So while traversing the string we can check if the current element is vowel or a odd number. If it is then we need to flip the coin.

▼ Simple Cakes (Simple Contest)

- Hint 1 - Find the number of unique elements in the array.
- Hint 2 - A simple solution would be sort the array and count different numbers, i.e start traversing and check for each i from 1<=i<=n-1 if a[i]==a[i+1] than continue else increase the count of different flavor.
- Hint 2 - Use unordered map.

▼ Candies and chocolates (Contest)

- Hint 1 - We already know the final position of each candy.
  So you just have to traverse the string and maintain track of the curr final position.
  Whenever you get a candy you find the time needed for it to go to its final position which will be difference in its position right now and final position.
- Hint 2 - But there is a catch, the time taken by the candy to reach its final position cannot be less than (time taken by the previous candy+1).So the actual time Ti is max(T(i-1)+1,diff(Finalpos-initialpos)).
- Hint 3 - yi=finalpos
  xi=initialpos
  Let's manage the second case when diff <= ti . The candy with number $(i - 1)$ will be on yi-th position by $t(i - 1)$-th second, so $t_i \geq t(i - 1) + 1$. Let's consider such moment of time p, when i-th candy stand right after $(i - 1)$-th, but not on yi-th position. After that, in $(p + 1)$-th moment of time $(i - 1)$-th candy and the chocolate in front of it will swap their positions, but i-th candy will save its position. Then since $p + 2$-th second till $t_i - 1$ both will change their positions. Finally, at $(t(i - 1) + 1)$ - th second i-th candy will occupy its position. Therefore, $t_i = t(i - 1) + 1$ in this case.

▼ Kth Row of Pascal's Triangle

- Hint 1 - In a pascal's triangle In row 0 (the topmost row), there is a unique nonzero entry 1. Each entry of each subsequent row is constructed by adding the number above and to the left with the number above and to the right, treating blank entries as 0. For example, the initial number in the first (or any other) row is 1 (the sum of 0 and 1), whereas the numbers 1 and 3 in the third row are added to produce the number 4 in the fourth row.
- Use 2 dimensional vector where each 1 dimensional contains the elements present in each row.

Week 5:

▼ Day 1 - STL continued[STL functions like lower bound, upper bound, sort]

In Class

- Smaller Elements

Post Class

- Floor and Ceil
- Shopping
- Mutating Array

▼ Day 2 - Linked List

In Class

- Insert node at the given position

Post Class

- Print the linked list
- Delete the Kth node from the end
- Intersection of two linked list

▼ Day 3 - Linked List continued

In ClasssMerge two sorted linked list

- Reversing the linked list

Post Class

- Palindrome List
- Merge two sorted linked list

▼ Day 4 - Linked List continued[Double/circular]

In Class

- Insert node at kth position in double linked list

Post Class

- Deletion in double linked list
- Reversing a double linked list

In Class[Approach Disucssed]

- Insertiong in circular linked list

▼ Day 5 - Stack

In Class

- Array implementation of Stack

Post Class

- Stack implementaion using linked list
- Stack Operations[Do it using STL]

▼ Contest For this week

▼ Sort it (Contest)

- Hint 1 - Let's say the count of 0s in the given array be X. We know that in the sorted array the first X elements will be 0s. So If we swap all the 1s which are less than X with the 0s greater than X we will get a sorted array.

▼ Simple Binary (Simple contest)

- Hint 1 - If the total 1s in the given array is not equal to total 0's then our answer will always be -1.

If one of the subarrays has equal 0s and 1s then the rest array will also have the same number of 0s and 1s.

- Hint 2 - While traversing the array we can manage a count of 1's and 0's which are encountered till now and whenever these two counts are equal we can say that we get one subarray that has equal 1's and 0's. i.e we find the smallest prefix array that has equal 0s and 1s and increases our answer by 1 and removes this prefix array..

▼ Missing Integer (Contest)

- Hint 1 - Use sets from STL
- Hint 2- Insert ith element in the set when you reach ith index while traversing the array. And keep another variable which stores the smallest non- negative integer that is missing in subarray A[1]...A[i]. This variable is incremented by 1 when the value stored in this variable is found in the set.

▼ Game time (contest)

- Hint 1 - Naruto will only win if the string has already all the same characters or the length of the array is odd. Because Sasuke will never remove the only different element as it is obvious if he does remove it then Naruto will certainly win.

▼ List shuffle

- Hint 1 - Find the middle element of the list.
- Hint 2 - After finding the middle element start traversing the list using 2 pointers where 1 pointer points to the start of the list and another pointer points to the middle of the list.

▼ Pair Hate (Contest)

- Hint 1 - We will use a stack to store the elements.
- Hint 2 - When the upcoming element and the top of stack is the same we will pop out the element from stack and move to the next upcoming element if it is not the same we just store it in the stack.

▼ Same Pair (Contest)

-

Week 6:

▼ Day 1 - Applications of Stack

In Class

- Infix to Postfix

Post Class

- Nearest smaller element
- Stock span problem
- Greater is Better

In Class[Approach Discussed]

- Height Problem

▼ Day 2 - Queue

In Class

- Array Implementation of queue

Post Class

- Linked List Implementation of queue
- Operation on queue[Do it using STL]

▼ Day 3 - Applications of Queue

In Class

- Operations on Dequeue

Post Class

- Generate Binary Numbers
- Reverse First K elements of queue

In Class[Approach Discussed]

- Maximum of all subarrays of size K

▼ Day 4 - Buffer Day for Linked List, Stack and Queue

▼ Day 5 - Maths[Modular Arithmetic, Modular Exponentiation, GCD, Modular Multiplicative Inverse]

In Class

- Modular exponentiation
- GCD
- Modular Multiplicative Inverse

Post Class

- Identical Groups
- Count Occurence of X
- Favourable Multiple
- Sum of divisors

▼ Contest For this week

▼ Two sets(contest)

- Hint 1 - It turns out that the strategy depends on n mod 4.
- Hint 2 - The total sum of all these numbers, which is n(n+1)/2. The parity of this sum depends on n mod 4: if n is congruent to 0 or 3 mod 4, the total sum is even; otherwise the total sum is odd.

  When the total sum is odd, it is not possible to divide the set into two parts with equal sums. The minimum difference you can expect is therefore 1.

  If n=0 mod 4: you already figured out how to do that.

  If n=1 mod 4: divide 2,…,n using your strategy, and add the extra 1 to any side. This achieves the minimum difference 1.

  If n=2 mod 4: divide 3,…,n using your strategy, and add the extra 1 and 2 to each side to get the minimum difference 1.

  If n=3 mod 4: divide 4,…,n using your strategy, and add 1 and 2 to one side, and 3 to the other side to get the minimum difference 0.

▼ Elections (Contest)

- Hint 1 - Let's assume the highest score among the given A[i] is of the index x. So, the candidate with rank Ax will have the smallest score. Since a negative score is not possible, only Bx candidates can have a score less than Bx, i.e., Bx-1, Bx-2...,0.
  So, answer is Ax + Bx.

▼ Help Nobita (Contest)

- Hint 1- GCD(A,B) has a special property so that it can always be represented in the form of an equation i.e. Ax+By = GCD(A,B).
- Hint 2- Find GCD(x,y) and output yes n%GCD(x,y) is equal to zero.

▼ Weird Series (Contest)

- Hint 1 - For a number N, the first occurrence of a number divisible by N in the given series will be less than the Nth term(if exist) of the series because after the Nth term X%M(X is the term in the series) will start repeating. So for the given integer N, we can check for all the terms from 1 to Nth term whether the current one is divisible by N or not.

▼ Play Play (Contest)

- Hint 1- If P*Q = N, We need to find the largest value for P so that Q will be minimum
- Hint 2 - In other words, we have to find the largest factor of N which is smaller than K. After this Q can be calculated easily.

▼ Margin(Contest)

-

▼ Lexo Sequence (Group Contest April '21)

- Hint 1 - The simplest approach is to generate all possible subsequences of length K from the given string and store all subsequences in a vector. Now, sort the vector and print the string at 0th position for lexicographically the smallest subsequence. Time Complexity: O(2N).

- Hint 2 - To optimize the above approach, the idea is Stack Data Structure to keep track of characters in increasing order, to get the lexicographically smallest subsequence.
- Hint 3 - Follow the steps below to solve the problem:
  - Initialize a stack, say answer, such that at any index of the string, the stack should contain the smallest K-length subsequence up to the current index.
  - Traverse the string and perform the following steps:
    If the stack is empty, then push the current character into the stack. Otherwise, iterate until the current element of the string S[i] is less than the top element of the stack and keep popping from the stack to ensure that the size of the stack is at most K.
    If the size of the stack is less than K after the above step, then push the current character into the stack.
  - After completing the above steps, print the characters stored in the stack in reverse order to obtain the resultant subsequence string.

Week 7:

▼ Day 1 - Maths[GCD, Primality Testing, Seive of Eratosthenes]

In Class

- Check if prime
- Number of primes

Post Class

- Good Team Leader
- Sum of Prime
- Kth prime factor of N

▼ Day 2 - Buffer Day for Maths

▼ Day 3 - Greedy

In Class

- Minimum Absolute Difference in Array

Post Class

- Permutation Game
- Best Score
- Maximize diff

▼ Day 4 - Greedy Continued

In Class

- Cost of Stock
- Maximum Contiguous Subarray Sum

Post Class

- Largest Number with given sum
- Pairs sum divisble by K
- Toy Company Greedy

In Class[Approach Discussed]

- Minimum Number of Coins

▼ Day 5 - DP

In Class

- Staircase Problem

Post Class

- Longest Chain Subsequence
- Adjacent numbers in subsequence differs by 1
- Sum of all substrings
- Stickler Theif

In Class[Approach Discussed]

- Increasing Subsequences

▼ Contest For this week

▼ Cut the rope (Contest)

- Hint 1 - Consider it as 11 points to be selected from N-1 points. So the total number of possible ways will be:- (N-1)C(11).

▼ XOR Problem (Contest)

- Hint 1 - Rewriting equation we have $a \oplus x = a - x$. If you look in the xor definition, it is easy to see, that $a \oplus x \geq a - x$, no matter a and x (just look at the each bit of the $a \oplus x$).
- Hint 2 - The equality handles only if bits of x form a subset of bits of a. So the answer is 2t, where t is the number of bits in a (also known as popcount).

▼ Increasing array (Contest)

- Hint 1 - Initialize counter to 1. For each ith index keep on incrementing counter from the value at which it was left on previous index until it divides a[i] or becomes greater than a[i]. The array cannot be sorted in increasing order if it becomes greater than a[i].

▼ Closest Prime (Contest)

- Hint 1 - It's a simple brute force question, one can check all the numbers greater or smaller than the given number until it encounters a prime, as the maximum gap between two primes in the given range will be less than 500.

▼ GCD Pairs (Contest)

- Hint 1 - The simplest method to solve this problem is to use two loops to generate all possible pairs of elements of the array and calculate and compare the GCD at the same time. We can use the Extended Euclidean algorithm for efficiently computing GCD of two numbers. Time Complexity: O(N^2 * log(max(a, b))). Here, log(max(a, b)) is the time complexity to calculate GCD of a and b.

- Hint 2 - In order to optimize solution we can maintain a count array to store the count of divisors of every element. We will traverse the given array and for every element, we will calculate its divisors and increment at the index of count array. The process of computing divisors will take O(sqrt(arr[i])) time, where arr[i] is element in the given array at index i. After the whole traversal, we can simply traverse the count array from last index to index 1. If we found an index with a value greater than 1, then this means that it is a divisor of 2 elements and also the max GCD.
- Hint 3 - : The most optimized approach is based on the idea of Sieve Of Eratosthenes. First let's solve a simpler problem, given a value X we have to tell whether a pair has a GCD equal to X. This can be done by checking that how many elements in the array are multiples of X. If the number of such multiples is greater than 1, then X will be a GCD of some pair. Now for pair with maximum GCD, we maintain a count array of the original array. After that follow a Sieve-like approach for loop.
  The step by step algorithm of this approach:
  Iterate 'i' from MAX (maximum array element) to 1.
  Iterate 'j' from 'i' to MAX. We will check if the count array is 1 at index 'j'.
  Increment the index 'j' everytime with 'i'. This way, we can check for i, 2i, 3i, and so on. If we get 1 two times at count array that means 2 multiples of i exists. This makes it the highest GCD.

▼ Divide by 3 (Contest)

- Hint 1 - For a number to be divisible by 3 the sum of its digits must be divisible by 3 too. So what we need is to find the sum of the digits and if the remainder is 1(when divided by 3) then we need to delete either 1 element whose remainder is 1 or 2 elements whose remainder is 2. The same process can be followed when the remainder is 3.

Week 8:

▼ Day 1 - DP Continued

In Class

- 0-1 Knapsack Problem

Post Class

- Max Sum Path
- Subset Sum
- Subset with equal Sum

In Class[Approach Discussed]

- DP Grid 2

▼ Day 2 - DP Continued

In Class

- Matrix Chain Multiplication

Post Class

- Longest Common Substring
- Shortest Common Supersequence
- Coin Change - Minimum Number of Coins

In Class[Approach Discussed]

- Minimum Moves

▼ Day 3 - Buffer Day for DP and Greedy

▼ Day 4 - Pattern Matching - KMP, Z Algorithm

In Class

- Distinct Pattern Search

Post Class

- Naive Pattern Search
- Luckiest Substring
- Cute Strings

▼ Day 5 - Tree[Traversals]

In Class

- Level order traversal of a tree

Post Class

- Postorder Traversal
- Count Leaves in Binary Tree
- Two Trees are Identical or Not

In Class[Approach Discussed]

- Inorder Traversal

▼ Contest For this week

▼ Simple Deforestation (Contest)

- Hint 1 - M-1 edges are required to divide a tree into M subtrees.

▼ Sara and Coins (Contest)

- Hint 1 - For numbers from 1 to N, we can easily count the number of ith set digits by observing the pattern. The formula for the ith(i>=0) set digits is:-

  ((N+1)/(2^i+1))*2^i + max(0,(N+1)%2^i - 2^i)

- Hint 2 - Now the only thing left is to calculate the number of set digits for every possible i and then take a minimum of set bits and unset bits (N-set bits).

▼ Random Sequences (Contest)

- Hint 1 - In order to find the number of arrays having sum N we can make use of the number of arrays having sum N-3,N-4,N-5,.....,1,0.
- Hint 2 - If dp[i] stores the number of arrays having sum i then dp[i] = dp[i-3]+dp[i-4]+dp[i-5]+...+dp[1]+dp[0]

▼ Minimum hurdles (Contest)

- Hint 1 - This can be solved using a 2D-DP.
- Hint 2 - All we need is to mark the top row and the first column hurdles as a single path first. Then traverse through the matrix row-

wise and for each use:- dp[i][j]=min(dp[i][j-1], dp[i-1][j])+a where a is the count of hurdles in the current cell.

▼ Nice permutations (Contest)

- Hint 1 - Dynamic Programming can be used.
- Hint 2 - Use 3D DP in which one state tells us the counts of numbers, one state tells us the count for which Pi>i and one state tells us the count for which Pi<i.
- Hint 3 - dp[i][j][k] += dp[i-1][j][k] + dp[i-1][j][k-1]*j + dp[i-1][j-1][k]*k + dp[i-1][j-1][k-1]*(i-j-k) where i tells us the count of numbers, j tells us the count for which Pi>i and k tells us the count for which Pi<i.

▼ Crazy Math (Contest)

-

Week 9

▼ Day 1 - Tree Continued[Construction of tree from traversals, height of tree, mirror of tree]

In Class

- Tree from Inorder and Preorder

Post Class

- Height of Binary Tree
- Mirror of binary tree
- Sum of Deepest Leaves

In Class[Approach Discussed]

- Foldable or Not

▼ Day 2 - Tree Continued[LCA, Diameter]

In Class

- Lowest Common Ancestor

Post Class

- Right View of Binary Tree
- Maximize Sum
- Maximum Width of Binary Tree

In Class[Approach Discussed]

- Diameter of Binary Tree

▼ Day 3 - BST

In Class

- Insertion in BST

Post Class

- Minimum in BST
- Lowest Common Ancestor in a BST
- Is BST?

In Class[Approach Discussed]

- Deletion in BST

▼ Day 4 - Buffer Day for Trees

▼ Day 5 - Heaps

In Class

- Kth smallest element

Post Class

- Kth largest element in a stream
- Max in Queue

▼ Contest For this week

▼ Max permute (Contest)

- Hint 1 - arr[i] = i+1 for all i except i=1 and i=n. arr[1]=1 and arr[n]=2. This array will give us the maxium sum.

- Hint 2 - The maximum sum that can be obtained using the array discussed above is 0+2+3+4+....+1 = 1+2+3+...+(n-1) = n(n-1)/2.

▼ Max Candies (Contest)

- Hint - First we initialize max sum to a[0]+a[n-1]. And then for each i from 1 to (n-1) add min(a[i],a[i-1] to max sum.

▼ Longest substring (contest)

- Hint 1 - The brute force approach would be to check for all subarrays which contain at most k consonants and find the longest among them. (TLE)
- Hint 2 - The optimized approach would be to use two pointers(Sliding window) one pointing to the end of the subarray and one to the beginning, move the end pointer until we have <= k consonants, once the number of consonants becomes greater than k move the starting pointer forward until we again have a number of consonants <= k, then note the length of longest subarray and print the value.

▼ Level sum (Contest)

- Hint 1- Use BFS for level order traversal and when you encounter the required node, find the sum of all the nodes available at that level.

▼ Subtree Search

-

▼ Possible sum

-

Week 10:

▼ Day 1 - Heaps Continued

In Class

- Find medium in a stream

Post Class

- Minimum Cost of ropes

▼ Day 2 - Graph[DFS,BFS]

In Class

- BFS
- DFS Basic

Post Class

- is connected?
- Has Path
- Hamiltonian Path

▼ Day 3 - Graphs Continued[Cycles in Graph]

In Class

- Cycles in Undirected Graph

Post Class

- Detect Cycle in a directed graph
- Count Bad Vertices
- Shortest Cycle(easy version)

▼ Day 4 - Graphs Continued[DFS, BFS on matrix]

In Class

- Find the number of islands

Post Class

- Number of Components
- Find whether path exist
- Rotten Oranges

▼ Day 5 - Graphs Continued[Dijkstra Algorith, Flood Fill Algorithm]

In Class

- Dijsktra Algorithm

Post Class

- Replace O's with X's
- Minimum Cost Path
- Snake and Ladder Problem

In Class[Approach Discussed]

- Flood Fill Algorithm

▼ Contest For this week

▼ Simple Pairs (Simple Contest)

- Hint 1 - When we take sum of the absolute differences then one thing which can be observed is that half elements are taken positive and the other half negatives i.e We divide the array into two N/2 parts and take the difference such that the diff is maximum.
- Hint 2 - To do this we will arrange elements such that one part contains N/2 maximum elements of the array.

▼ Remember (Contest)

- Hint 1- We just need to count the number of divisors for N.

▼ Jump Jump (Contest)

- Hint 1 - After a finite number of jumps, you will be at your starting index. That is after K moves the values will repeat.
- Hint 2 - Find the number of jumps after which we end up at starting index. Then take K%number of jumps. We then perform the jump from the starting index K%number of jumps times.

▼ Disconnected Kingdoms (Contest)

- Hint 1 - Find the number of connected components.
- Hint 2 - The number of roads to be built such that all the kingdoms are connected to one another is (number of connected components - 1).

▼ Reversed path (Contest)

- Hint 1 - Consider all possible paths for 1 to N (including the reverse edges) and for each path calculate the total edges reversed during the path.
- Hint 2 - Consider all possible paths for 1 to N (including the reverse edges) and for each path calculate the total edges reversed during the path.

▼ Merge Trees (Contest)[Difficult]

-

Week 11 :

▼ Day 1 - Graphs Continued[Minimum Spanning Tree, Floyd Warshall, Topological Sorting]

In Class

- Minimum Spanning tree

Post Class

- Bipartite Graph
- Lexicographically Smallest Topo Order

In Class[Approach Discussed]

- Floyd Warshall Algorithm
- Topo-sort

▼ Day 2 - Buffer Day for Graphs

▼ Contest for this week

▼ Simple Numbers (Simple Contest)

- Hint 1 - if(a%k!=0) number of divisors are b/k - a/k.
  if(a%k==0) number of divisors are b/k-a/k-1.

▼ Lexo Small (Contest)

- Hint 1 - A simple solution will be:- sort the string and then check how many characters from the start are equal.

Time complexity:-O(|S|*log|S|)

- Hint 2 - We can initialize an array arr[26] to maintain the count of each element then find the smallest non zero element of the array.
  Time complexity:-O(|S|)

▼ Number of Strings (Contest)

- Hint 1- Find the number of strings of length N such that the character 'a' does not appear in these strings and subtract if from the total number of strings of lenght N to get the required number of strings.
- Hint 2 - Required number of strings = 26^N[Total number of strings of lenght N - 25^N[Number of strings of length N that don't have 'a']

▼ FoodFest (Contest)

- Hint 1 - For each time T, you can check whether the team can finish at least P dishes or not. If the team can eat at least P dishes in T time then the answer holds for all values greater than T similarly, if the team can not eat P dishes in T time then the answer will hold for all values less than T.
- Hint 2 - Now it's a simple binary search problem. Use binary search to find different values of T and for each value check if it is valid or not.
  Time complexity:- O(NlogP)

▼ Random Sequences

- Hint 1 - In order to find the number of arrays having sum N we can make use of the number of arrays having sum N-3,N-4,N-5,.....,1,0.
- Hint 2 - If dp[i] stores the number of arrays having sum i then dp[i] = dp[i-3]+dp[i-4]+dp[i-5]+...+dp[1]+dp[0]

▼ Connect them (Contest)

- Hint 1 - Create a graph using all the N points on 2D plane and assign the weight to each of them using min(abs(x1- x2), abs(y1- y2)).
- Hint 2 - Apply kruskal algorithm on this graph.

- Sara and Factors (Contest)[Difficult]
- Divide Array (Contest)[Some what high level for week 2]
- Camp Setup (Contest)[Not that good]
- Room (Contest)[Difficulty - 5/5]
- Sara and Apples (Contest)[Difficulty-5/5]
- Zero Window (Contest)
- Sequence Counter (Contest)
- Happiness Everywhere! (Contest) [Difficult]

- Max tickets (Contest)

END OF DSA