T&P

Phase-2

→ Bit manipulation

- XOR of sum of pairs
- Sum of XOR of pairs
- Max subarray XOR

} Easy.

→ Recursion

- Subset with sum k — E

print all valid porentheses — EM

- Magic square → M

→ • String portioning ← H

H • No.of ways of placing blocks such that there is atleast k dist b/w each block

~~wen~~

M • Given N-tasks a(i) = time taken to complete i^th task in k-workers k

M • 2 sorted arrays → A, B.
median of (A+B)

Strings

→ Given 2 strings A, B

M check if B is substring of A

(Rabin karp, KMP, Rolling Hash)

→ longest palindromic substring → M

→ ⊗ checking prime → E

→ void generating all primes → E

<u>Arrays</u>

→ Subset sum K – E

→ array rotation & search in rotated sorted array

→ first missing +ve integer – M

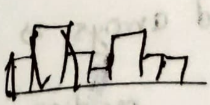
→ longest subset with equal 0's & 1's → E

→ Sum of OR of all subarrays → M

(bit manipulation)

→ populate array such that every element in the new array is the first greater element on right M

Os Max rectangular area under histogram HM



→ for all subarrays print medians M

Linked list

→ joining linked list connected at a node

→ cycle detection in SLL

→ Duplicates +2

Trees

→ Sum of all nodes

→ traversals (Inorder, pre, post)

→ Max sum given by any path

→ No.of nodes at a distance of K ↗H

→ verical level order

→ diagonal level order

→ left view, right view, top view, bottom view

BST

→ Check BST

→ Least common ancestor of given two nodes

→ +2

Tries

→ Given a list of words and Q queris find all the words that have 'Q' as a prefix

E

→ Max value of arr[i] ^ arr[j] (i != j)

M

→ Given N×M matrix with 0's & 1's

E find distinct rows using trie.

Heaps

→ Given an array of size N, find } ★

E → K^{th} smallest elements.

→ K sorted array

H

DP : Dynamic programming

→ Staircase (move to n^{th} step with (1, 2) steps)

→ tiles (1×5), (5×1)

→ no. of binary strings having length N

→ no. of ways to reach N using a dice

→ + (min cost)

factorial ✓

→ Painting houses H

→ Jobs and Machines H

→ Maximum subarray sum M

→ Longest Increasing Subsequence M

→ Knapsack

^

factional Integer

M → coins and chage

→ divide array into two subsets A, B Such that

min |sum(A) - sum(B)|

①

~~Graphs~~

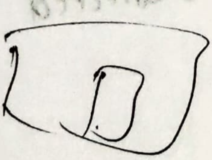
→ Given a matrix:-

move from src to dest

move only down or right

B →

H → cal sum (mat[i:j] : mat(k:l))



H → largest Submatrix sum

G → longest common subsequence in two strings

M → longest increasing path in a matrix.

E → Unique paths from src to dest

Graphs

E → check if there exists a path from src to dest

E (BFS , DFS travel)

M → length of shortest path from Src to dest

E → longest path length in a graph

M → Number of Islands

M → Minimum weighted path (Dijkstras)

H → Knight on a chess board (weighted & unweighted)

M → Topo-logical sort

E → Detect cycle in an undirected graph

E → Minimum spanning tree

E → is bipartite graph

LRU cache