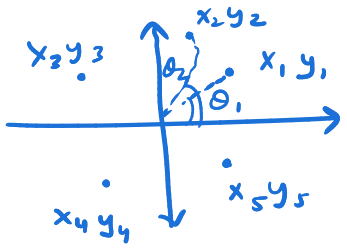
Problem Solving Domain

Chapter 1 : Introduction to PS

1. Understanding the templates | nxn grid → m is even/odd, n is even/odd → Student swap, Chocolate break, Chessboard cutting
2. Time complexity — 10^8 ops ~ 1 sec — given 'n', figure out mathematical class.
3. Recursion overview ⇒ factorial (linear), fibonacci (tree), GCD
4. Sorting algos. → $O(n^2)$ | bubble sort
   $O(n \log n)$ | Mergesort quicksort.
5. Imp. f^n of Qs → partition method $O(n)$ | Rearrangement: alternate +ve -ve, odd even, push zeros to the end.
6. Custom sort → comparator f^n. | numbers as strings → form largest no. Sort strings based on length or reverse of them for rhymes.

HW: (x_i, y_i) ⇒ n points | sort them based on the polar angle.

o/p: $[(x_1,y_1), (x_2,y_2) \dots (x_5,y_5)]$

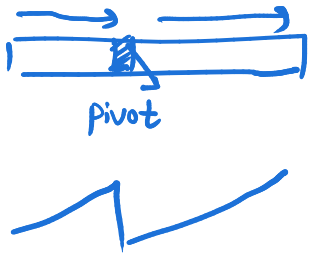


7. Searching
   - linear search (unsorted) — $O(n)$ to search, $O(1)$ to insert.
   - binary search — 1 time effort $O(n \log n)$ to sort, $O(\log n)$ to search
   - BS as a method to identify best value x in range 1 to n for some b^n. f(x)

H.W Aggressive Cows SPOJ

8. Hashing
   - table → double the table | $O(1)$ amortized insertion time
   - collision methods ⇒ probing
   - Use sets & maps which are in-built.
     - sets: uniqueness
     - maps: frequency

9. Rotations → left/right | Reversal algo O(n)



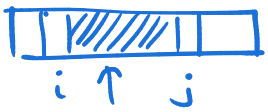
⇐ values graph in rotated array.

10. Two pointer method i, j ⇒ index

Case 1 : i → ← j Typical : Two sum

Case 2 : i → j → Typical : → Dejavu: partition function
→ Maximum sum subarray / Kadane's problem.

11. Sliding window technique i, j → window



This window is imp.

Window : set

Typical : longest non-repeating char substring in string

12. prefix sum arrays & difference arrays | $\int$ and $\frac{d}{dx}$ of arrays

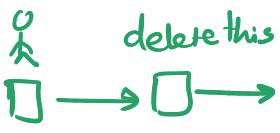
↓ cumulative freq. ↓ array of deltas

Typical : Bulb toggle | pattern: You will get alot of range queries to update your array.

H.W. Rainwater Trapping Problem Keep track of them in diff arr, Use prefix array to fix it eventually.

Chapter 2 Linked list (unpopular in interviews)

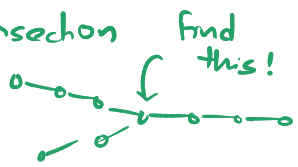
1. CRUD on data structure
   - Insert length of list, search, sum of ele
   - Traverse →
   - Update / search for a key & replace it
   - Delete



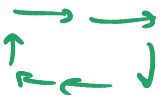
2. Specifics on LL: Reversing

Sorting → Mergesort

Intersection — find this!



3. loops : → → → → | slow & fast pointer.



4. Circular LL, Doubly LL. ← not discussed (ss)

Chapter 3: Stacks & Queues

1. Used STL | we did implement queues once.

2. Design questions in stacks & queues (ss) Imp!!

3. Basic classic f^n : prefix infix postfix of expression

Balanced pair (checking)

4. V.V.Imp. NGR NGL NSR NSL → Max area under histo.
   → Stock span problem

Chapter 4: Dynamic Programming

min, max, longest, ... ↑ #ways ↑

1. How to identify DP problem? ⇒ Optimization OR Combinatorial

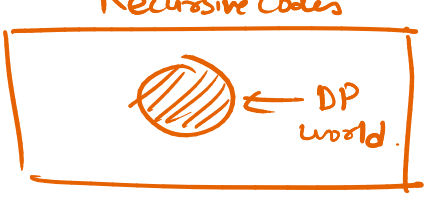
   a) Greedy → Counter case: Optimal strategy every step isn't optimal overall.

   Eg: change=6 coins={1, 3, 4}

   b) Try for recursion. |

   c) If not recursion, its a adhoc OR some other domain problem.

2. For DP to show its power, you need duplicate recursive calls. Cache them.



Recursion + Memoization → DP.

We need this first

To master this

3. Revisited Recursion. → Backtracking → Sudoku
                                        → Knights Tour
                                        → N Queens.

HW. Try to generate all possible subsets of an array.

i/p: [1,2,3] o/p. [[], [1], [2], ..[2,3], [1,]] i/p: n, o/p: 2^n | Power set.

4. Templates of DP: many variations!!
   a) Fibonacci
   b) Knapsack
   c) LCS
   d) MCM

5. Class2 Standalone DP problems.
   a) Min cost path ↑→ (2 dirs allowed)
   b) Word Break problem
   c) Edit distance problem.
   d) Gantt chart based scheduling.

HW: CSES problem set on DP.

Chapter 5: Trees

1. Types/Terminology: full, perfect, complete, skewed
2. Recursive codes: many variations
3. Traversals: Depth ⇒ Pre in post
   Breadth ⇒ level order ⇒ variations!!
4. Construction → a) Given 2 traversals, can you make a tree?
   b) Catalan number (variation of C_n)
5. Hierarchy → LCA, cousins, siblings, etc.
6. path ⇒ node to x, node to all leaves, diameter of tree.

Chapter 6: BST and Heaps.

1. BST → CRUD | predecessor & successor logic.
   → quickselect algo to find median in O(n) time.

2. Heap → heapify O(n) (array represented) | extract min again & again from min heap → Heapsort.

Imp: Heaps are priority queues. Classic: Join the ropes

Chapter 7: Graphs.

1. Graph terminology

2. C → add_edge | 2 representations: matrix list

   R → BFS → shortest path (advantage)

   DFS → disconnected graphs can be handled easily.

   U → search for a key

   D → If you delete a node, remove all edges dependant on that node.

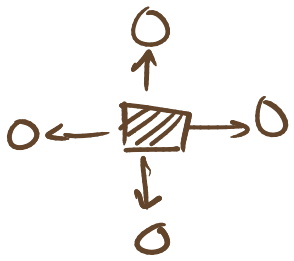
3. Many variations of BFS & DFS.

   a) Identifying graph is the key.

   b) print the path from src to dest. | all paths.

   c) Cycles

   d) V.V. Imp Grids as graphs !!



   all 4 directions mean 4 connected cells are my neighbours.

HW.. SPOJ: Prime path (you have mastered BFS)

SPOJ: MM Knight moves (mastered BFS)

No. of components (DFS is done!!)

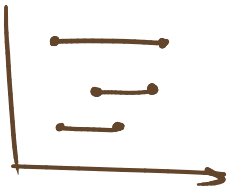
Max area island (DFS you killed it!)

→

NOT DONE. (not so popular in interviews)

(elaborate it tom)

Max jobs ⇒



① A huge section on graphs.

② Divide and Conquer :
- a) Binary exponentiation
- b) Fast Fourier Transforms

③ Greedy :
- a) Scheduling Problems.

④ DP :
- a) Digit DP
- b) Bit DP

⑤ String matching:
- a) KMP, Z, etc. | We have done Rabin-Karp

⑥ Range Queries
- a) Sqrt Decomposition $O(\sqrt{n})$
- b) Segment Trees
- c) Fenwick Trees / B trees (DBMS use B+ trees)

⑦ Number Theory
- a) GCD LCM
- b) Prime number
- g Modular arithmetic
- d) Algebra & Geometry ⇒ Convex Hull problem.

⑧ Game Theory
- a) A performs a move
  B performs a move
  A ..
  B ...

  ———
  who wins ?

A majority of GT problems can be solved with pure logic. =