**Random:**

1. **3 sum, 2sum, 4 sum**: 2 sum uses 2 pointers, 3 sum: select each number and do 2 sum on remaining, 4 sum: do loop each element and do 3 sum on rest (O(n cube)), or do 2 sum, and another 2 sum inside the first one (O n2).

1. <https://leetcode.com/problems/basic-calculator-ii/>: lot of edge cases like ‘"0-2147483647"’
   1. String s = “”+char; and int a = char - ‘0’;
   2. Getting number from chars is running following in loop until char is a digit: **num = num \* 10 + char;**
   3. Character.isDigit()
   4. **No need to store + or - into stack. Store numbers -n if negative and do sum on all.**
   5. for(int i :st){
   6. res+=i;
   7. }
2. **Basic calculator** (hard): use repetitive stacks for each ‘(‘ occurrence.
   1. <https://leetcode.com/problems/decode-string/>

1. <https://leetcode.com/problems/multiply-strings/>: multiplying result position will be i+j, i+j+1.
2. [~~https://leetcode.com/problems/water-and-jug-problem/~~](https://leetcode.com/problems/water-and-jug-problem/)~~: there is mathematical proof that to get z litres with x and y litres, z should be a multiple of GCD of (x,y). If then, ax + by = z. Ie 3\*3+-1\*5 = 4, ie fill 3 litre 3 times and pour from 5 litre once. To find gcd, use euclidean formula, recursively minus the smaller number from bigger number or modulus.~~
   1. ~~if (b==0) return a; else return generateGCD(b,a%b);~~
3. <https://leetcode.com/problems/populating-next-right-pointers-in-each-node/> folowup: in O(1) space.
4. <https://leetcode.com/problems/contiguous-array/>
5. **~~beautiful trees (Bitonic Sequence)~~**~~, if given array, height of trees should be increasing and then decreasing. Find the minimum number of tree cuts to make beautiful.~~
   1. ~~Ex: 3, 17, 4, 12, 5, 6 : min cuts are 17 and 12.~~
   2. ~~Idea: 1. Find number of violations so far in increasing order in every index.~~
   3. ~~2. Find decreasing violations count. Both are in array. The find index where min abs difference is there. The find min cuts to make left side of index increasing and right side decreasing. Use stack.~~
6. <https://leetcode.com/problems/prison-cells-after-n-days/>: idea is since 8, number of cells is constant, the possible configurations are 2^8 = 256. So after some time, it will definitely reoccur.
7. <https://leetcode.com/problems/wiggle-sort-ii/>: idea: sort the array, then use two pointers one from mid and from end, place large pointer values in odd index and smaller ones in even index.
8. <https://leetcode.com/problems/minimum-size-subarray-sum/>
   1. <https://leetcode.com/problems/maximum-size-subarray-sum-equals-k/>
   2. <https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/> [link](https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/discuss/204290/Monotonic-Queue-Summary) change from above questions is this have negatives.
   3. <https://leetcode.com/problems/contiguous-array/>
9. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock/> : follow up 2, 3, 4
10. <https://leetcode.com/problems/pairs-of-songs-with-total-durations-divisible-by-60/>
11. <https://leetcode.com/problems/game-of-life/>: update an infinite matrix in place as keeping another infinite matrix is not possible.
12. Sort big file: a 20 GB file. : merge sort.
13. ~~Ants on a triangle: ant on each vertex, then find the probability that they collide if they start walking randomly.~~
    1. ~~P(clockwise) = (½)^3~~
    2. ~~P(anticlock) = (½)^3~~
    3. ~~P(same) = 2 \* (½)^3~~
    4. ~~P(collide) = 1- 1/4;~~
14. <https://leetcode.com/problems/rotate-image/>: reverse up to down and symmetry for clock wise. Reverse left to right and symmetry for anticlock wise.
15. ~~Sort stack by recursion~~

**Heap:**

1. **Kth Largest Element in a Stream**: **heap concepts:** <https://leetcode.com/problems/kth-largest-element-in-a-stream/>
   1. Queue<Integer> q = new PriorityQueue<>(k, (a,b) -> b-a); b-a for max heap. A-b for min heap.
   2. To keep heap of size k, poll and offer based on condition.
2. **378. Kth Smallest Element in a Sorted Matrix:** 2 ways: heap with 13 ms and binary search with 1 ms.
3. **373. Find K Pairs with Smallest Sums:** basic idea is to keep a max heap, such that whenever a value less than max value comes, poll it and offer the small value.

**Quick Select:**

1. <https://leetcode.com/problems/kth-largest-element-in-an-array/>
2. <https://leetcode.com/problems/k-closest-points-to-origin/>
3. <https://leetcode.com/problems/top-k-frequent-elements/>: use pq or **Bucket sort**
4. <https://leetcode.com/problems/sort-characters-by-frequency/>

**~~Sampling~~**~~:~~

* [~~https://leetcode.com/problems/random-pick-index/~~](https://leetcode.com/problems/random-pick-index/)~~:~~ **~~reservoir sampling~~**~~: ie equally selecting k items from a stream of n items where n is unknown.~~
* [~~https://leetcode.com/problems/random-pick-with-weight/~~](https://leetcode.com/problems/random-pick-with-weight/)~~: binary search~~

**Stack/monotone stack/Queue**: Queue<Integer> q = new LinkedList<Integer>();

Stack<Integer> st = new Stack<>();

1. Monotone stack: check similar questions in <https://leetcode.com/problems/sum-of-subarray-minimums/discuss/178876/stack-solution-with-very-detailed-explanation-step-by-step>
2. [https://leetcode.com/problems/**next-greater-element-i/**](https://leetcode.com/problems/next-greater-element-i/) : normal solution is O(n square), Followup: while using stack, the time complexity is O(m+n) as both arrays are read only once, and stack pop happens on avg only once for each element.
   1. space complexity: O(m+n): for map and stack of size n and result array of m size.
   2. Amotized O(n) as we pop and push all n elements only once. So it will be like 2\*n.
3. <https://leetcode.com/problems/next-greater-element-ii/> idea: loop twice.
4. <https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/> [link](https://leetcode.com/problems/shortest-subarray-with-sum-at-least-k/discuss/204290/Monotonic-Queue-Summary)
5. <https://leetcode.com/problems/largest-rectangle-in-histogram/submissions/> : brute force: create another array with sorted heights. Use it as outer loop and actual array as inner loop and find max area possible for each heights.
6. <https://leetcode.com/problems/trapping-rain-water/>
7. <https://leetcode.com/problems/minimum-cost-tree-from-leaf-values/discuss/339959/One-Pass-O(N)-Time-and-Space> : good one with explanation and similar questions to practise stack.
8. <https://leetcode.com/problems/remove-all-adjacent-duplicates-in-string-ii/>
9. <https://leetcode.com/problems/132-pattern/> : O(n^2) and O(n) using stack. Specific way. Check. Idea is s1,s3,s2 numbers given, need s1 < s3 < s2. So idea is start s3 with min\_value of int. Then add every s2 into stack while looping backwards and check and update s3 with largest value < s2. Then if new num < s3 return true; **Use array itself instead of stack to remove extra space.**
10. <https://leetcode.com/problems/longest-valid-parentheses/> : dp
11. LC239. Sliding Window Maximum
12. LC739. Daily Temperatures
13. LC901. Online Stock Span
14. LC907. Sum of Subarray Minimums

**LIST:**

1. **23. Merge k Sorted Lists:** catch is in splitting the list as in binary search
   1. int mid = beg +(end-beg)/2;
   2. ListNode node1 = helper(lists, beg, mid);
   3. ListNode node2 = helper(lists, mid + 1, end);
2. <https://leetcode.com/problems/copy-list-with-random-pointer/> catch is while looping for random pointer, we may get out of loop since next is null, but random will be self loop. So put a if check after loop to reassign random pointer if not already.
3. <https://leetcode.com/problems/linked-list-cycle-ii/discuss/44793/O(n)-solution-by-using-two-pointers-without-change-anything> : HARD, for idea check first equation in comment of above link. In brief its distance between start to entry point = meeting point to entry.
4. <https://leetcode.com/problems/convert-binary-search-tree-to-sorted-doubly-linked-list/> : check iterative way also.
5. <https://leetcode.com/problems/populating-next-right-pointers-in-each-node-ii/>: without extra space: idea: once next pointers are populated no need of queue.

**DP**: <https://leetcode.com/problems/longest-palindromic-subsequence/discuss/222605/DP-Problem-Classifications-Helpful-Notes>

<https://leetcode.com/discuss/general-discussion/458695/dynamic-programming-patterns>

1. <https://leetcode.com/problems/house-robber/discuss/156523/From-good-to-great.-How-to-approach-most-of-DP-problems.>
2. Memoization (top-down) and tabulation (bottom-up approach). Bottom-up means starting from bottom state to final state.
3. Bitonic sequence:
4. <https://leetcode.com/problems/maximal-square/> : easy dp on 2d array. Visualize example.
5. <https://leetcode.com/problems/minimum-swaps-to-make-sequences-increasing/>
6. <https://leetcode.com/problems/maximal-rectangle/> : idea: dp
7. <https://leetcode.com/problems/paint-fence/>
8. <https://leetcode.com/problems/unique-binary-search-trees/>: idea is consider each val as root, then max ways for it is cartesian product of left and right vals.
   * dp[level]+= dp[root-1]\*dp[level-root], root-1 gives prev val in series and other gives next val. Ie if 3 taken as root from 1,2,3,4,5, then dp[2]\*d[4]
   * Followup: print all. Use recursion.
9. [https://leetcode.com/problems/**house-robber/**](https://leetcode.com/problems/house-robber/) : both tabulation and memoization: catch is example 2,1,1,2 : max is 4.
10. **55. Jump Game:** can do in o(n) like similar to maximum subarray.
11. **45. Jump Game II**: good test case [2,3,0,1,4]. O(n) solution
12. **53. Maximum Subarray:** in O(n square ) solution, edge case input 1. O(n) approach is good. And find divide and conquer approach.
13. **523**. **Continuous Subarray Sum :** basic catch is dividing by zero. Kadane's algo ie in each loop consider max((sum + nums[i]), nums[i]);
14. <https://leetcode.com/problems/max-sum-of-rectangle-no-larger-than-k/>
15. <https://leetcode.com/problems/continuous-subarray-sum/>
16. Subsequence dp
17. <https://leetcode.com/problems/minimum-cost-tree-from-leaf-values/>
18. <https://leetcode.com/problems/word-break/> : O(2^n) brute-confirm, O(n^2) in dp. Can also do with BFS.
19. <https://leetcode.com/problems/word-break-ii/>
20. https://leetcode.com/problems/longest-valid-parentheses/
21. <https://leetcode.com/problems/longest-palindromic-substring/> IMP
    * <https://leetcode.com/problems/longest-palindromic-subsequence/>IMP
    * <https://leetcode.com/problems/longest-increasing-subsequence/>: IMPbrute force: O(2 raised to n), dp O(n2), dp with binary search O(nlogn)
    * Longest common subsequence
    * <https://leetcode.com/problems/delete-operation-for-two-strings/>
    * Count of palindromic substrings
    * Min deletion in a string to make it palindrome
    * Palindromic partitioning
22. Knapsack
    * 0/1 knapsack
    * Equal subset sum partition
    * Subset sum
    * Minimum subset sum difference
    * Count of subset sum
    * <https://leetcode.com/problems/last-stone-weight-ii/>
    * <https://leetcode.com/problems/target-sum/> : can use map for dp,
23. Unbounded knapsack(repetition of values allowed)
    * Unbounded knapsack
    * Rod cutting
    * <https://leetcode.com/problems/coin-change-2/> : knapsack, for brute force, think about how combinations are found from recursion.
    * <https://leetcode.com/problems/partition-equal-subset-sum/>: diff of this between coin change is, in coin change we can reuse.
    * <https://leetcode.com/problems/coin-change/> : for knapsack, declare state, then final state, then state transition. Like in ([link](https://leetcode.com/problems/coin-change-2/discuss/141076/Logical-Thinking-with-Clear-Java-Code))
    * Maximum ribbon cut
24. Fibonacci
    * Fibonacci numbers
    * Staircase
    * Number factors
    * Min jump to reach end
    * Min jumps with fee
    * House thief
25. Longest common substring
    * Longest common substring
    * Longest increasing subsequence: O(nlogn) is using binary search. complex.
    * Shortest common supersequence
    * Min deletions to make a sequence sorted
    * Longest repeating subsequence
    * Subsequence pattern matching
    * Longest bitonic subsequence
    * Longest alternating subsequence
    * Edit distance
    * String interleaving
26. Decision making
    * <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/> : hard.
    * <https://leetcode.com/problems/house-robber/> 2(circular condition), 3()
    * <https://leetcode.com/problems/delete-and-earn/>: convert que to house robber.
    * <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/>
    * <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown/>: hard in O(1) space dp.
    * <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/>: base case is if k>prices.length/2, then consider all transactions by a for loop.
27. Distinct ways
    * <https://leetcode.com/problems/domino-and-tromino-tiling/>
    * <https://leetcode.com/problems/minimum-swaps-to-make-sequences-increasing/>: very difficult- a particular idea. Find cost of keeping values of A[i-1] and B[i-1] increasing and use it in next.
28. <https://leetcode.com/problems/valid-parenthesis-string/> : dp as well as O(n) another solution. Also Can do validate parentheses without stack using simple counter.
29. Minimax dp
    * <https://leetcode.com/problems/stone-game/>
    * <https://leetcode.com/problems/predict-the-winner/>

**Sliding Window or 2 pointer**:

<https://leetcode.com/problems/minimum-window-substring/discuss/26808/Here-is-a-10-line-template-that-can-solve-most-'substring'-problems>: idea is have beg and end, and increase beg or end as needed.

1. <https://leetcode.com/problems/minimum-window-substring/>
2. <https://leetcode.com/problems/longest-continuous-increasing-subsequence/>
3. <https://leetcode.com/problems/longest-substring-with-at-most-k-distinct-characters/>
4. <https://leetcode.com/problems/longest-substring-with-at-least-k-repeating-characters/>
5. <https://leetcode.com/problems/longest-repeating-character-replacement/>
6. <https://leetcode.com/problems/minimum-number-of-k-consecutive-bit-flips/>
7. <https://leetcode.com/problems/unique-letter-string/>
8. https://leetcode.com/problems/substring-with-concatenation-of-all-words/
9. **Sliding Window: 1004. Max Consecutive Ones III:** basically can do with queue but 700ms runtime, with sliding window:
   1. **3. Longest Substring Without Repeating Characters**
   2. **340. Longest Substring with At Most K Distinct Characters:** Using int[] map = new int[256]; instead of map to keep track of count reduces runtime from 25ms to 7ms. Another **followup** in O(logn)
   3. <https://leetcode.com/problems/sliding-window-maximum/>: you can do it with a treemap (O(logn)) if no other way.
10. **567. Permutation in String:** can use **sliding window**, by modifying second approach given below.
    1. brute force: Find all anagrams and check if its a substring in other using string.contains(s)
    2. Second approach: iteratively, get count characters of s1 in a map. Then loop through each substring of s2 of length s1 sequentially and check if number of characters of that substring <= count in initial map.
    3. Doing it with array reduces runtime of 300ms. Arr[char - ‘a’]
11. **242. Valid Anagram:** use **new int[256] to include all unicode characters**. Map is not good.
12. <https://leetcode.com/problems/find-all-anagrams-in-a-string/>
13. 28. Implement strStr()

**BackTracking**:

1. <https://leetcode.com/problems/word-search/> remember to clear visited array when returning back in backtracking.
2. <https://leetcode.com/problems/n-queens/>
3. <https://leetcode.com/problems/sudoku-solver/>
4. <https://leetcode.com/problems/partition-to-k-equal-sum-subsets/> : kind of difficult one.
5. <https://leetcode.com/problems/minimum-knight-moves/>: try to find min from given x,y to 0,0 using bfs in symmetry, ie consider only x-1,y-2 and x-2,y-1 positions in each recursive case. Base case x+y == 0 return 0 and x+y == 2 return 2.
6. <https://leetcode.com/problems/permutations-ii/>
7. <https://leetcode.com/problems/permutation-sequence/>
8. <https://leetcode.com/problems/palindrome-partitioning-ii/>: idea check oddCount first, then add half of each char to list and generate its permutations and append reverse of it.
9. <https://leetcode.com/problems/combination-sum/>
10. <https://leetcode.com/problems/subsets/>

**Trees:**

1. <https://leetcode.com/problems/verify-preorder-sequence-in-binary-search-tree/>: **hard, idea:** keep low and high. Whenever new num < low, false, if its greater than peek(), then pop() once and set it as low, high = peek(), then loop until low<n and n<high.
2. <https://leetcode.com/problems/construct-binary-tree-from-preorder-and-postorder-traversal/>: check both recursive and iterative.
3. **144. Binary Tree Preorder Traversal:** iterative is not easy. Use stack to pop and use nodes to explore right nodes.
4. **208. Implement Trie (Prefix Tree):** i implemented with hashmap. Edge is if entered ‘apple’, then search for ‘app’ should be false. For that, add a ‘.’ to last map and search return true if map at the end of ‘app’ contains key ‘.’ . **implement using custom class with array.**
5. **652. Find Duplicate Subtrees:** beware of shapes validation along with sequence.
6. <https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree/>
7. 98. Validate Binary Search Tree : in BST, every element of left should be less than current. Equal value in left or right is not allowed.
   1. <https://leetcode.com/problems/validate-binary-search-tree/solution/>**:** using stack is one way with keeping only low key instead of both low and upper keys and other issimply by checking the array generated.
   2. <https://leetcode.com/problems/delete-node-in-a-bst/>
   3. <https://leetcode.com/problems/largest-bst-subtree/> in O(n) time
   4. [**https://leetcode.com/problems/recover-binary-search-tree/**](https://leetcode.com/problems/recover-binary-search-tree/) **O(n)**: simply do dfs a create array, then do O(n) loop or more better we can traverse the tree once and find first and second element that needs to be swapped in O(1) . O(1) space without recursion space: use **morris inorder traversal** (check tushar video for checking idea).
   5. **333. Largest BST Subtree**
   6. <https://leetcode.com/problems/serialize-and-deserialize-bst/submissions/> : good one: simple preorder for serialize and deserialize using queue.

**Segment or Finwick/Binary Indexed Tree**: BIT is easier to code and better in performance. <https://leetcode.com/problems/range-sum-query-mutable/>

1. Segment tree used when say we do range queries (such as sum, min, max between given indexes) and updates. Usually, range queries takes O(n) and update in O(1). Say if we do m queries, then it will take O(mn). Segment trees reduce query complexity to O(m\*logn) by creating a indexed tree form. **The way segment tree is created ie merge operation used while creating tree depends on the type of range queries we usually do.** <https://leetcode.com/articles/a-recursive-approach-to-segment-trees-range-sum-queries-lazy-propagation/>
2. Making segment tree: we keep it array tree[]. Children of node at tree[i] are at tree[2\*i+1] and tree[2\*i+2] when index starts from 0, for index starts from 1, 2\*i and 2\*i+1 are used.
3. BIT is better in space than ST. Segement tree takes O(nlogn) space while Binary Indexed tree takes O(logn) where n is number of queries. Query complexity is O(k+logn) for ST and O(logn) for BIT, where k is number of reported results. For update, both takes O(logn). **The parent child relaion is like this**: given node i, finding parent node is by removing the least significant 1 bit. Ie i = i- (i& -i), -i is done thorugh 2’s compliment. So similarly finding child given i is i+= i & -i. First node is dummy node. BIT finds sum from 0 to node i. So if need range sum, call Sum(j) - Sum(i).
4. **RMQ(range minimum query using Sparse table)** give O(nlogn) to create and O(1) retreival, while ST takes O(n) to create and O(logn) to retreive. Idea is create a 2D table M[i][j] which is minimum from ith index to 2^j th index.
5. <https://leetcode.com/problems/the-skyline-problem/>
6. <https://leetcode.com/problems/count-of-smaller-numbers-after-self/>
7. <https://leetcode.com/problems/largest-rectangle-in-histogram/submissions/> : stack is optimal, bu can be done by divide and conquer using segment tree. Check geeks for geeks. Why use segment tree, since we need to find min in each divided ranges and it will cost O(n) for single min fetch. Segment tree reduce it to O(logn). Read stack solution. Idea is pop from stack until elements are greater thatn or equal to curr. Then add curr to stack and continue; At end, if stack not empty, find area as array.length-stack,poll()

**Greedy**: Greedy problems usually look like "Find minimum number of *something* to do *something*" or "Find maximum number of *something* to fit in *some conditions*", and typically propose an unsorted input. The idea of greedy algorithm is to pick the *locally* optimal move at each step, that will lead to the *globally* optimal solution.

1. <https://leetcode.com/problemset/all/?topicSlugs=greedy>
2. **Broken calculator: idea is greedy**, ie given X and to reach Y, continuously take half of Y, until close to X, ie a = Y/2, then b = a/2,... and store them in a stack. Then using allowed operations reach each half stored in stack.
3. 1029. Two City Scheduling
4. <https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/> : multiple transactions instead of one….profit is sum of all peaks, since [1, 3, 2, 10]. Is best splitted as [1,3] [2,10] instead of [1, 10]. Visualize the peaks graphs.
5. <https://leetcode.com/problems/minimum-cost-to-connect-sticks/>
6. <https://leetcode.com/problems/minimum-domino-rotations-for-equal-row/>: can take 4 variables and do or count occurences. Do Math.min(countA[i],countB[i])-same[i]. Similar to finding union.
7. <https://leetcode.com/problems/gas-station/>: idea: sumGas should be >= sumCost, then its possible.
8. <https://leetcode.com/problems/partition-labels/>

**Strings/Arrays**: matrix position can be represented as rowNum \* (numOfCols) + colNum.

1. **Regex**: line.split(“\\W+”); for words
2. **Max common word: p.replaceAll("\\W+",”").toLowerCase().split("\\s+");**
3. **567. Permutation in String: TLE in brute force** : O(n!) as number of permutations possible is n!. Space complexity: O(n2) as n recursive stack and length of each stack is n. So n\*n.
4. <https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/>
5. **459. Repeated Substring Pattern:** shit hard
6. **KMP algo:** <https://leetcode.com/problems/implement-strstr/>
   1. O(n), idea is to create a prefix array or lps(longest proper suffix which is also suffix) ie the array is of size pattern string, then at each index, it tells size of longest prefix before that point which is also a suffix before that point.<https://www.youtube.com/watch?v=KG44VoDtsAA>
   2. <https://leetcode.com/problems/shortest-palindrome/> : brute force recusrive is O(n^2)
7. <https://leetcode.com/problems/reorganize-string/> : difficult : idea is lets say ‘a’ is a character with most count in string, if it is greater than half of length, then we cannot reorganize. Otherwise, from starting index 0, put all ‘a’ in even indexes.
   1. If count of ‘a’ was half of length, then index will reach end and then we just have to place all chars rest in between.
   2. Else Then ‘b’ is next char with max count, it won ever reach the b even it reaches end and start again from beginning.
   3. Examples: a\_a\_a\_a\_ if count > (length+1)/2 cannot create
8. <https://leetcode.com/problems/spiral-matrix/> : super solution in looping. If need to loop a value from 0 to 4, d =0; d= (d+1)%4.
9. <https://leetcode.com/problems/contiguous-array/> Super question. Idea is keep counter, and its index in map, so if in future same count reaches means, the number in between is same.
10. <https://leetcode.com/problems/next-permutation/>
11. <https://leetcode.com/problems/next-greater-element-iii/> variation of next permutation, also just find a number lesser from prev digit from right. Then replace min digit present to its right with it and sort that small portion.
12. <https://leetcode.com/problems/longest-absolute-file-path/> : “\t\nasds”, here \t is considered as single character.
13. <https://leetcode.com/problems/validate-ip-address/>: a.split(“.”, -1) wont work, use split(“\\.”, -1) -1 is limit. If limit == 0, pattern is applied as many times as possible and discard trailing spaces (ie “1.1.” will return [1, 1] while limit < 0 returns [1,1, ]. Ie limit < 0 means, apply pattern as many as possible but donot discard trailing spaces.
    1. NumberFormtException
    2. Integer.parseInt(hexdecimalString, 16).
14. <https://leetcode.com/problems/isomorphic-strings/>: init 2 arrays and While looping if same diff char has different values in both array, return false.
15. <https://leetcode.com/problems/missing-ranges/>: use long for int max and min cases.
16. **Prefix Sum**: <https://leetcode.com/problems/subarray-sum-equals-k/> : GOOD idea: use prefix sum, ie when we move forward save previous sums. So at a position, if map contains(sum-k), then array can be formed by remaining elements from after that sum position.
    1. <https://leetcode.com/problems/path-sum-iii/>
    2. <https://leetcode.com/problems/subarray-sums-divisible-by-k/>
17. <https://leetcode.com/problems/product-of-array-except-self/>GOOD idea: keep L and R arrays for multiply results in 2 directions. In O(1) space, use result array for keeping multiply result in one direction and calculate other in fly.
18. <https://leetcode.com/problems/maximum-product-subarray/>
19. <https://leetcode.com/problems/minimum-size-subarray-sum/>: 2 pointer/sliding window.

**Graph**: Find shortest path, think about BFS or **Dijkstra’s**

1. **Think about** dfs, bfs(can start from 2 states ie if matrix contains 0 and 1, can explore starting from all 0’s or all 1s), unionfind, djisktra, .
2. **Dijkstra’s shortest path algorithm: BFS + PQ**
3. **Things to think:** if given matrix contains 2 elements only, try starting dfs/bfs from both elements and see which will give result faster. About bfs: think priortyqueue. Union find.
4. <https://leetcode.com/problemset/all/?topicSlugs=graph>
5. <https://leetcode.com/problems/evaluate-division/>
6. <https://leetcode.com/problems/shortest-path-in-binary-matrix/> remember why this BFS ?
7. [https://leetcode.com/problems/number-of-connected-components-in-an-undirected-**graph/**](https://leetcode.com/problems/number-of-connected-components-in-an-undirected-graph/)
8. <https://leetcode.com/problems/critical-connections-in-a-network/> : using tarjans or by detecting cycle edges using rank.
   1. Strongly Connected components: In directed graph, these are subgraph, in which every vertex can reach other vertexes directly or through others.
   2. Weakly Connected components: Probably disconnected sub graphs.
9. **BFS:** 
   1. <https://leetcode.com/problems/word-ladder/> **simple BFS, Bi-directional BFS(re learn it) (reducing time to traverse the search,** ie 2 bfs at same, one from end and one from beginning, at end, add distance**),** if branching factor is b and distance from source to target node is d, the for bfs, O(b^d), while for birectional b^d + b^d which is lesser. Just consider beginset if its smaller than endset, else otherwise. Continue this.
10. <https://leetcode.com/problems/clone-graph/>
11. <https://www.geeksforgeeks.org/minimum-number-swaps-required-sort-array/>: simple: use selection sort, or consider n distinct elements as n nodes in graph. For cycle with k nodes, k-1 swaps is required.
12. [**h**ttps://leetcode.com/problems/pacific-atlantic-water-flow/](https://leetcode.com/problems/pacific-atlantic-water-flow/)
13. <https://leetcode.com/problems/word-search-ii/>
14. <https://leetcode.com/problems/word-search/>
15. <https://leetcode.com/problems/path-with-maximum-minimum-value/> : really good question. Think about the idea.
16. **Do BFS starting from all non required positions: good** <https://leetcode.com/problems/shortest-distance-from-all-buildings/>
    1. <https://leetcode.com/problems/as-far-from-land-as-possible/>
    2. <https://leetcode.com/problems/pacific-atlantic-water-flow/>
17. **Union-Find:** In a directed graph, a simple dfs a checking in a hashmap will detect cycle.
    1. If not using rank, then update parent of newnodes with parent of existing nodes. It will be faster.
    2. <https://leetcode.com/problems/friend-circles/discuss/101336/Java-solution-Union-Find>
    3. <https://leetcode.com/problems/friend-circles/>
    4. <https://leetcode.com/problems/most-stones-removed-with-same-row-or-column/>
    5. <https://leetcode.com/problems/graph-valid-tree/> : detect cycles using union find.
       1. <https://leetcode.com/problems/redundant-connection/>
    6. Union by rank: Initially all nodes are set with just that val. increase rank only when 2 sets representative has the same rank. Else parent will be set with higher rank.
    7. Find set(number): returns representative of the set in which number is present.
    8. Path compression: each node has pointers to root. When joins 2 sets, some leaf node may not point to new root, so repoint it to new root. This reduces finding representative of a set when Find Set(number) is called.
    9. Rank, data(node val), parent pointer(pointer from child to root)
    10. Space complexity= O(n) if n elements
    11. Time complexity, if m operations and n elements: O(m\*alpha(n)) = O(4m) = O(m) as alpha(n) approx = 4 in all normal use cases.
    12. <https://leetcode.com/problems/longest-consecutive-sequence/>: O(n): idea is to keep a hashset, then recurse through while removing from hashset once visited.
    13. **detect cycle in an undirected graph:** First, make all nodes as its own set. Then loop through each edge say s->e, with finding representative of s and e using findset. If both parents of s and e belongs to same set, loop.
18. **Dijkstras/Bellman ford/ Floyd warshall**
    1. **Bellman ford:** O(VE) ie more than djikstras. This is used as same as Djiskstras but nodes have negative weights. Its a dp.
    2. **Bellman ford**: <https://leetcode.com/problems/network-delay-time/> Bellman ford: O(VE), Djikstas O(VlogV + E)
    3. **Floyd Marshal:** to find shortest distance between every vertexes. O(n^3). Like find min distances between every cities. Its a dp.
       1. <https://leetcode.com/problems/find-the-city-with-the-smallest-number-of-neighbors-at-a-threshold-distance/>
    4. **Djikstras:** find shortest path to all nodes from source node. Ie it creates path from a->b whie prims/kruskals create spanning tree.
    5. <https://leetcode.com/problems/cheapest-flights-within-k-stops/> : djikstras or bfs plus PQ.
    6. <https://leetcode.com/problems/the-maze-ii/>
    7. <https://leetcode.com/problems/reachable-nodes-in-subdivided-graph/>
19. **Travelling Salesman:**
    1. <https://leetcode.com/problems/find-the-shortest-superstring/>
20. **Kriskals/prims:** 
    1. Kruskals: on connected and undirected graph : <https://leetcode.com/problems/connecting-cities-with-minimum-cost/>
       1. Sort edges by wights, then add edges on by one considering if it does form a cycle. Cycle check by union find. Ass V-1 edges..
    2. Both are greedy algo to generate MST(a tree with v-1 vertices, ie a spanning tree that touches all vertices with min cost). Kruskals complexity is almost O(ElogV). And prims O(E+VlogV) when fibonacci heap used, so prims is better in some cases..
    3. Kruskals better in sparse graph while prims in dense where more edges to vertices ratio.
    4. <https://leetcode.com/problems/optimize-water-distribution-in-a-village/>

**Topological Sort in BFS 3ms and DFS 55ms**: ie for edge uv, u is printed always first before v. Can be more than 1 topological sorting. Adjacency list and Adjacency matrix used for graph representation.

**Conditions**: In DAG. Start with vertex with degree 0, ie no other incoming edges

* Basically do a DFS, and whenever all child of node are visited, Add that node to Stack. Also keep a Visited map or array. At end, print from stack.
* **Detecting cycle** **in directed graph**: Do **dfs** keeping an array or map of visited. Initially say n nodes, then loop on n nodes, and call dfs on each of the n nodes. At end of each recursion, visited[node] is reset to false. Ie consider example 1->0->3->2->0, there is a cycle as when we dfs, we visit 0 again. It forms a tree. Ie if cycle present only if same node from any single path appears in the same path of the tree created. Check <https://leetcode.com/problems/course-schedule/>
  + **Using BFS**: do like topological sorting. So if cycle is present without any left out nodes, no nodes with indegree 0 and hence cycle. else Add every nodes with degree 0 to queue, then loop over it. In each node poped from queue, loop over its neighbours and reduce its indegree by 1. If indegree of that neighbour after reducing is 0, add it to queue. So basically, we should pop all nodes if no cycle. O(N+E)
* **Detecting cycle in an undirected graph**: dfs, but while expanding a node, does not consider parent node since we came from that. Also can do union find.
* Union-Find
* **Topological sorting with bfs:** (done based on fact: A DAG G has at least one vertex with in-degree 0 and one vertex with out-degree 0.): find indegrees of all nodes. Add nodes with indegrees 0 into queue. Loop through queue, then for each node poped from queue (node to a list say resultList to print at end), decrease indegrees by 1 for its neighbouring nodes and if indegree is 0, add that neighbor node to q. Count number of nodes poped from queue. At end, count should equal total nodes then print resultList, otherwise topological not possible.
* **Topological sort checking with dfs**: just check for cycle.

1. <https://leetcode.com/problems/course-schedule/> :
2. <https://leetcode.com/problems/course-schedule-ii/>: check no cycle present before returning result.
3. 261. Graph Valid Tree: any connected graph without simple cycles is a tree. Edge ex: 0->1 2->3. If graph contains unconnected parts, then its not a valid tree. Do dfs once and check if its visits every node. Since its undirected it should visit every node.
4. <https://leetcode.com/problems/minimum-height-trees/>
5. <https://leetcode.com/problems/reconstruct-itinerary/>: **Find Eulerian path(using** Hierholzer’s Algorithm - O(E)**)** is path which passes through every edge exactly once while hamiltonian path passes through every vertex exactly once. Hamiltonian cycle is path returns to starting vertex. Euler path is circuit or cycle when it returns to starting edge.
   1. For Eulerian cycle: every vertex should have even degree, ie 2,4,6 meaning one to reach and another to exit. For Eulerian path: exactly 2 vertex should have odd degree.
   2. Hierholzer’s Algorithm: do dfs and add each node from backwards. Reaches end since end node have odd degree, ie no exit.
6. <https://leetcode.com/problems/alien-dictionary>:idea: like course schedule 2, return “” if either no char with indegree 0 or final result string length > actual number of unique char, ie there is a cycle.graph, topological sort.
7. <https://leetcode.com/problems/is-graph-bipartite/>
8. <https://leetcode.com/problems/critical-connections-in-a-network/>
9. <https://leetcode.com/problems/cherry-pickup/>: good idea: start 2 paths , the to reduce state from 4d array to 3d, r+c = t is satisfied for every move since moving only in right and down. So r1+c1 = r2 + c2, ie r1 = r2+c2-c2;
10. <https://leetcode.com/problems/optimal-account-balancing/>: idea: create debt array where d[i]< 0 means d[i] gets that much money and d[i] > 0 means has to give. Then recursively try out every combinations to cancel out every pos and neg to zero.
11. https://leetcode.com/problems/sequence-reconstruction

**Interval**:

1. <https://leetcode.com/problems/insert-interval/>
2. <https://leetcode.com/problems/merge-intervals/>: sort by starting time. list.toArray(new int[s][]);
3. <https://leetcode.com/problems/meeting-rooms-ii/> : O(nlogn) use 2 arrays start and end. Then loop from start and increment rooms. If current start > end, that means, prev room got free, so rooms--;
4. <https://leetcode.com/problems/non-overlapping-intervals/>: greedy approach using end or start sorted.
5. <https://leetcode.com/problems/interval-list-intersections/>: find inersections of 2 lists. Only 3 conditions. Find max of start and min of end.
6. <https://leetcode.com/problems/task-scheduler/>: use a queu and map for cooldown.
7. <https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/>
8. <https://leetcode.com/problems/add-bold-tag-in-string/>
9. <https://leetcode.com/problems/set-intersection-size-at-least-two/>: good idea: sort by end, then keep start and end as last 2 points of each intervals. Another way would be, remove an interval if that completely overlaps another.

**Search:**

1. **Binary Search:** 
   1. **int beg = 0, end = arr.length-1, mid = beg + ((end - beg)/2);**
   2. **while(beg < end){**
   3. **if(arr[mid] == target)return true;**
   4. **if(target > arr[mid])**
   5. **beg = mid+1;**
   6. **else**
   7. **end = mid-1;**
   8. **mid = beg + ((end - beg)/2);**
   9. **}**
   10. **Beg <= end used when we are returning from inside the loop.**
2. **378. Kth Smallest Element in a Sorted Matrix:** binary search, we can say there are 2 kinds of binary search, one with index as search space where data is linear, and one with range of actual values as search space where data is like in this case. <https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/discuss/85173/Share-my-thoughts-and-Clean-Java-Code>
3. **Binary Search on large array: start + (end-start)/2**
4. <https://leetcode.com/problems/find-k-closest-elements/>
5. <https://leetcode.com/problems/find-the-duplicate-number/>
6. <https://leetcode.com/problems/random-pick-with-weight/>
7. <https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/>
8. <https://leetcode.com/problems/split-array-largest-sum/>: minimize the maximum in m groups.
9. <https://leetcode.com/problems/minimize-max-distance-to-gas-station/>: in questions its said, ans within 10^-6 is accepted. So in binary search while(l + 1e-6 < r) can be used.
10. <https://leetcode.com/problems/koko-eating-bananas/>
11. A\* search

**Divide and Conquer**:

* <https://leetcode.com/problems/the-skyline-problem/solution/>: idea: another sol is use priority queue to store heights and loop.

**SQL**

* <https://leetcode.com/problemset/all/?search=sql>
* If select query returns null, then creating a temp table and selecting it is good. As Select (select query) as column
* Use offset for nth highest questions
* <https://leetcode.com/problems/rank-scores/> : for ms sql, use dense\_rank over (partition) method, ex: dense\_rank over (order by col desc).

**SORT**

* Bubble, selection, insertion, counting, radix, bucket sort.
* Bucket sort: O(n) in best case, where all gets put into diff buckets. Then individually sort each buckets and merge.
* Counting: put all frequency into an aux array. Then copy that much frequency numbers into new array..
* In efficiency: heap < merge < quick. All are nlogn actually, but slight difference. Merge sort is faster than heap due to sequential reading but it uses more spaces(array) as compared to heap.
* <https://leetcode.com/problems/compare-strings-by-frequency-of-the-smallest-character/> : counting sort

**Bit-Manipulation**: for 32 bit: -2^31 and 2^31-1, since 0 is included as in 0-7 is 2\*3-1 numbers.,

Signed right shift operator (logical) >>: shifts bits to right filling left with sign bit present. Dividing by number which is power of 2.

Unsigned right shift operator(Arithmetic) >>>: shifts bit right.

Logical and Arithmetic left shift are same >>: multiplying by powers of 2. Ie 1 << 2 will be 1\*2^2.

* <https://leetcode.com/problemset/all/?topicSlugs=bit-manipulation>
* <https://leetcode.com/problems/single-number/>: idea xor
* Twos compliment to find negative num ie flip all bits and add 1.
* x^y == 1 if odd number of 1 present.
* **Bitmask**:
  + <https://leetcode.com/problems/maximum-students-taking-exam/>
* <https://leetcode.com/problems/majority-element/> : can be done by Boyer-Moore Majority Vote Algorithm, also by bit manipulation
* Swap using bit: a = a^b
  + a^b^b = a
  + a^b^a = b
* <https://leetcode.com/problems/single-number-ii/> : idea use 2 bitmaps
* <https://leetcode.com/problems/divide-two-integers/> : idea: decrease divident as factor of 2, as in 15 = 3 \* 4 + 3, we first get 1<<2 = 4 as anse, then 1, so 4+1 = 5. overflow when a= intmin and b = -1.
* <https://leetcode.com/problems/number-of-1-bits/> Hamming weight
* <https://leetcode.com/problems/power-of-two/>: check if y power of x: for 2, number of 1 bit ==1 or n&(n-1) == 0. Continuously dividing abd checking if equal will be O(Logxy.) or a = Math.log(y)/Math.log(x). Ie log y to base x. Then check if its an int. a==Math.floor(a); Another solution: while iteratively checking each multiplies, do it on squares and if number is passed, do binary search in between. O(LogLogxy.)
* <https://leetcode.com/problems/power-of-three/>: another way is find max power value and check if that val %n == 0.

**Math**:

* <https://leetcode.com/problems/rectangle-overlap/> : idea is to find cases where diagonals can never overlap.

**Design**

* LRU
* LFU
* <https://leetcode.com/problems/design-hashmap/>: collision resolution
  + Seperate chaining: bucket or linkedlists used to keep all elements hashed to a particular value. Load factor (avg length of chain) = N/table size, where N is number of items and table size is size of array in hashvalues are stored. So if array is size 2 and N =10, then load factor = 10/2 = 5.
  + Open addressing: if place is already occupied, search for next free space using linear probing(linear function), quadratic probing(f(i) = i^2), double hashing.
  + **Use prime number when using modulo as hash function as it have no factors, helps in keeping balance**. Say 12 is size of bucket, if numbers coming are multiples of 3, it will always be mapped to locations which are multiples of 3 repeatedly and other locations will be empty.
* <https://leetcode.com/problems/encode-and-decode-tinyurl/> : simple way is to keep counter and store in hashmap, but it will create new id for a string which is already encoded. Solution for that is also use another map. Check discuss.
* <https://leetcode.com/problems/design-hit-counter/> : good question. Ask about scaling when number of hits large.
* <https://leetcode.com/problems/design-log-storage-system/>:good ques: use string compareTo which returns neg if a<b and pos if a>b. Also can use TreeMap subMap.
* <https://leetcode.com/problems/shortest-word-distance-ii/>
* Trie: <https://leetcode.com/articles/implement-trie-prefix-tree/>
  + <https://leetcode.com/problems/add-and-search-word-data-structure-design/>
  + <https://leetcode.com/problems/word-search-ii/>
  + <https://leetcode.com/problems/design-search-autocomplete-system/>

**Concurrency**

* <https://leetcode.com/problems/print-in-order/>: use semaphore, or atomicInteger or volatile boolean and do while loop infiintely until bol value changed by other threads. **Volatile**: tells jvm to donot cache values and always take from memory.

**Random:**

* O(n): using stack sometimes results in O(n)
* If given a matrix both rows and col sorted matrix, a way would be to start from the top right.
* **BST**: The left subtree of a node contains only nodes with keys lesser than the node’s key. Ie, not just next left is smaller, but every nodes of left should be smaller.
* <https://www.mtu.edu/career/students/networking/interviews/prepare.pdf>

**Theory**:

1. Copy constructor in c++:
2. Random random = new Random();
   1. random.nextInt(50) generate 0(inclusive) - 50(exclusive), if we want 1 - 50, random.nextInt(50) + 1;
3. Comparing like new PriorityQueue<>(a-b) may result in overflow, so Use Integer.compare(a,b);
4. **When creating count for characters: do array[char-’a’]++; instead of map.**
5. Private inheritance: public and protected members of the base class becomes private.
6. **Friend Class** A friend class can access private and protected members of other class in which it is declared as friend.
7. **HashMap** and LinkedHashMap maintains O(1) insertion and lookup. Allow 1 null key. And any number of null values.
8. **TreeMap** does insertion and lookup in O(logn). Do not allow null key.And any number of null values. treemap.firstKey() and lastKey() for sorted.
9. If checking prime, then i <= (int)Math.sqrt(N)
10. **Initialize**: List<Integer> list = new ArrayList<Integer>(){{add(1);add(2);}};
11. Map<String, String> map = new HashMap<String, String>(){{put(“a”,”b”);}};
12. Arrays:
    1. Arrays.sort(ch); and collections.sort() does inplace.
    2. Sorting 2d array: Arrays.sort(costs, (int[] a, int[] b) -> ((a[0]- a[1]) - (b[0]- b[1])) );
    3. Arrays.sort((a,b) -> a-b) sorts in ascending
    4. Arrays.sort(arr, (a,b) ->a-b); wont work on int array as a,b are not objects.
    5. int[][] dir = {{0,1},{0,-1},{1,0},{1,-1}};
13. **String**: String.contains(“”)
    1. a.compareTo(b): 0 if equal, negative if a<b and positive if a>b.
    2. String.split(s, limit) limit < 0 means apply pattern as many but donot discard trailing spaces as in limit == 0. If limit > 0, apply atmost limit-1 times.
    3. StringBuilder sb = new StringBuilder(inputString);
    4. sb. [deleteCharAt(index)](http://docs.oracle.com/javase/6/docs/api/java/lang/StringBuilder.html#deleteCharAt%28int%29): inplace
    5. sb.append(“as”).append(‘s’);
    6. s.substring(0, indexNotIncluded)
    7. Char ch = (char)(count + '0'); // to convert a number to char.
    8. Replace character in a string : word.substring(0, j) + newch + (j==(wordlen-1)?"":word.substring(j+1, wordlen));
    9. ‘a’+1 gives b
    10. Int to char: String.valueOf(n).charAt(0)
    11. String is immutable, so passing in function and updating is not reference.
14. HashMap/TreeMap: Tremap is implemented by red black tree in java.
    1. To copy: newMap.putAll(oldMap);
    2. To get max and min key: The declaration should be TreeMap instead of Map.
       1. TreeMap<Integer, Integer> tree = new TreeMap<Integer, Integer>();
       2. int maxKey = tree.lastEntry().getKey();
       3. int minKey = tree.firstEntry().getKey();
    3. To get the first element: Map.Entry<String, Integer> entry = map.entrySet().iterator.next();
    4. Iterator<Map.Entry<String, String>> itr = gfg.entrySet().iterator();
       1. while(it.hasNext())it.next().getKey() or it.next().getValue()
    5. map.putIfAbsent(key, value)
    6. Map.Entry<String,String> entry : gfg.entrySet()
    7. Queue<Map.Entry<String, Integer>> q = new PriorityQueue<>((a,b) -> a.getValue() - b.getValue());
    8. TreeMap<Integer, String> internalMap = new TreeMap<>(Collections.reverseOrder());
    9. Sort by value:
       1. list.addAll(map.keySet());
       2. Collections.sort(list, (a,b) -> {
       3. int fa = map.get(a);
       4. int fb = map.get(b);
       5. if(fa == fb)return a.compareTo(b);
       6. else return fa-fb;
       7. });
    10. treemap.floorKey(val) : returns greatest key less than equal to val. To get greatest key less than val- treemap.lowerKey(val).
    11. Similarly ceilingKey(val) and higherKey(val).
15. Array, List:
    1. Collections.sort(integersList, Collections.reverseOrder());
    2. Arrays.sort(Integer[], (a,b)->a-b); cannot sort int[] with comparator
    3. Beware that using linkedlist will be more time consuming than arraylist.
    4. There is auto conversion from int to Integer but not from Integer to int.
    5. Set/Add: if list = [1,2,3] at indexes 0, 1, 2. Then if we call list.add(0, 4) to replace value at index 0 with 4, result is : [4,1,2,3]. So use set to replace and get: [4,2,3]
    6. ArrayList<Integer>[] matrix = new ArrayList[numCourses];
    7. list.remove(num) will removes value from index num if it is an int. To remove value num from list, convert it to Integer as new Integer(num) and pass.
    8. list.toArray(new int[s][]);
    9. list.toArray(new int[list.size()]) this does not work for primitives, so list.stream().mapToInt(i->i).toArray();
    10. if you put int[] in hashset, everyone will have diff hashcode even same objects. But List checks equals by the actual elements. Ie to hash a tuple, store as list.
16. Tree:
    1. in BST, every element of left should be less than current. Equal value in left or right is not allowed.
    2. A tree is undirected graph with any 2 vertices connected by only 1 path, ie given n nodes, n-1 total edges.
17. Short 2 bits, int 4, long 8, float 4, double 8, bit and boolean 1, char 2.
18. Create array[128] to keep all characters ascii. A 97 , z 122. A 65 z 90
19. Queue
    1. Deque<Object> st = new ArrayDeque<>(); ArrayDeque is better than LinkedList
    2. st.addFirst(): adds at top
    3. st.addLast(): adds at last
    4. st.pollFirst(): gets from head.
    5. st.pollLast: gets from tail.
20. Maven lifecycle: validate, compile, test, package, verify, install.
21. If asked to return res module 10^9 + 7, mod = 1e9 + 7. res%mod.

**Revise**: walmart: do completed questions again before going.

1. <https://leetcode.com/problems/longest-consecutive-sequence/>
2. <https://github.com/jayshah19949596/CodingInterviews/tree/master/LinkedIn%20Software%20Engineer%20-%20Entry%20Level>
3. how to sort a file which 10 times larger than memory. Non -coding.
4. Given a list of sticks with different height and a number K. Find out total number of ways to rearrange these sticks so that when looking from the left , at most K sticks are visible

**Links**

* <https://www.interviewbit.com/search/?q=Google>
* <https://www.interviewbit.com/search/?q=Microsoft>
* <https://www.geeksforgeeks.org/top-25-interview-questions/>
* <https://www.interviewbit.com/search/?q=Amazon>
* <https://www.geeksforgeeks.org/must-coding-questions-company-wise/>
* <https://www.geeksforgeeks.org/must-do-coding-questions-for-companies-like-amazon-microsoft-adobe/>
* <https://github.com/jayshah19949596/CodingInterviews>
* <https://www.bigocheatsheet.com/>
* grokking system design, system design primer, prepare OO design leetcode
* Ask if sys or OO design questions is there to recruiter.
* S

**Motivation**

* <https://leetcode.com/wxy9018/>
* <https://leetcode.com/flyingpenguin/>
* <https://leetcode.com/fighternan/>
* <https://leetcode.com/nevergiveup/>
* <https://leetcode.com/discuss/interview-experience/505108/nda-google-l4l5-japan-feb-2020-pending-offer>

**Amazon**:

* <https://www.amazon.jobs/en/landing_pages/in-person-interview>
* Try pramp, mock leetcode
* <https://docs.google.com/spreadsheets/d/1SbpY-04Cz8EWw3A_LBUmDEXKUMO31DBjfeMoA0dlfIA/htmlview?sle=true#gid=126913158>
* Revise amazon list solutions before interview with complexity and followup of most asked que.
* Behaviour: "I can't think of a specific example, but if I was in that situation, I'd handle it like so...."
* Give a lot of mocks as he <https://leetcode.com/kaipeng21/>
* Check how to crack amazon bar raiser.
* <https://www.amazon.jobs/en/principles>
* Prison cells
* Gcd of n numbers in array
* <https://www.geeksforgeeks.org/visitor-design-pattern/>
* Rotate linked list by k positions
  + Sys design:
    - Design an online bookstore system design
    - SDE1: <https://leetcode.com/discuss/interview-question/499558/Amazon-System-Design-Question>

**Google**: Cover ctci

* <https://www.interviewbit.com/search/?q=Google>
* <https://leetcode.com/problems/n-queens/>

**Vmware**

* Why vmware?
* Learn kubernetes, virtualization, why vmware, vmware products
* <https://masteringvmware.com/top-vmware-interview-questions/?fbclid=IwAR0BgdOQ8uUbxryUw90Uj-HiTxvV2N8HqUv70mxsbrkyWgbfClZ8gVpeHag>
* <https://blogs.vmware.com/careers/2013/10/9-tips-to-help-you-ace-a-technical-interview-at-vmware.html>
* <https://biginterview.com/best-questions-to-ask-end-interview/>
* <https://elink.io/p/introduction-to-vmware-software-products-and-services>
* <https://leetcode.com/discuss/interview-experience/427930/VMware-or-SWE-or-Palo-Alto-or-Nov-2019>
* <https://leetcode.com/discuss/interview-question/432071/VMWare-or-OA-2019-or-Min-Cost-to-Make-Non-Decreasing>
* bunch of questions about operating systems and networks, virtual memory concepts and networking concepts
* Saas in private and public clouds. My jobs is to increase performance, scalability.
* Learn little golang
* How to check what cause low sys performace:
  + Df -h, free -m, network issues.
* Kubernetes diag experience, micr service archi understanding, analysze java performance issue - gc, memory profiling, call, troubleshoot linux performace issues.
* hash, OS, mutex, semaphore, oop concepts
* Object oriented programming related to a hotel management system
* Network, storage and server virtualization.
* Aproach\_\_\_\_\_\_ 3 problem solving with 2 questions, 1 sys design.
* Integer to Roman
* skyline problem.
* number of islands 2. word break
* Find the greatest divisor of a number
* search a element in a cyclic sorted array .
* Largest rectangle in histogram.
* Print All the nodes which are at K distance from the leaf nodes.
* Integer to Roman
* inorder BST traversal with recursion and without recursion.
* find missing number,
* <https://leetcode.com/problems/valid-parentheses/>
* <https://leetcode.com/problems/clone-graph/>
* <https://leetcode.com/problems/maximum-width-of-binary-tree/>
* <https://www.geeksforgeeks.org/maximum-sum-path-across-two-arrays/>
* https://leetcode.com/problems/lru-cache/
* https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/
* Design::::::
* https://www.educative.io/courses/grokking-the-system-design-interview/m2yDVZnQ8lG

**Uber**

* **Why uber**
  + <https://eng.uber.com/kepler-data-visualization-traffic-safety/>
  + <https://eng.uber.com/engineering-maps/>
* Rounds: should nail first 2 rounds 1 hour each.
  + Senior software engineer: design or coding, 10:30 to 11:30 AM - Adam Kidder
  + Senior software engineer: deep experienced in go, systems, opensource. design or coding. 11:30 to 12:30 PM - Ed Barwani
  + Lunch
  + 2 Senior manager: gaming, visualizaion, part of movement team. 01:15 to 02:00 PM - Travis Gorkin and Chris Chen: hiring manager.
  + Roche is s/w developer in front end, 02:00 to 02:45 PM - Tony Cui and Roche Janken: probably bar raiser since i think from diff team.
  + 02:45 to 03:45 PM - Wesam Manassra: design or coding
* <https://leetcode.com/problems/bus-routes/>
* <https://leetcode.com/problems/critical-connections-in-a-network/discuss/382632/Java-implementation-of-Tarjan-Algorithm-with-explanation>
  + https://leetcode.com/problems/binary-tree-right-side-view
  + <https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree>
  + <https://leetcode.com/discuss/interview-experience/270705/Uber-or-SWE-or-Apr-2019-Fail>
  + Integer to english words
  + <https://leetcode.com/discuss/interview-question/426697/Uber-Online-Assessment-Questions>
  + <https://leetcode.com/discuss/interview-question/411074/Uber-or-Phone-Screening-Question>
  + Given array of stick lengths, return number of turns it takes to have no sticks left. Each turn you remove sticks of smallest length and cut that length from the rest of the sticks.
  + In a 2D vector, all vectors were of size two. Eg: {{1,3},{2,3},{1,4}}. The vectors represent the entry and exit time of a pedestrian crossing a road. Pedestrian 1 entered at time 1 and exited at time 3 and so on.. Find the interval during which maximum number of pedestrians were crossing the road.
  + Serialize and deserialize n ary general tree.
  + <https://leetcode.com/problems/path-with-maximum-minimum-value/>
  + Topological sort = O(v+E)
  + Permutations = O(n!)
  + <https://leetcode.com/problems/stream-of-characters/>
  + Onsite
  + Onsite
  + onsite
  + <https://leetcode.com/discuss/interview-experience/456323/Uber-SWE2-Reject-Unique-experience>
  + <https://leetcode.com/discuss/interview-experience/132801/Uber-or-Software-Engineer-or-New-York-Reject>
  + <https://leetcode.com/discuss/interview-experience/462380/Uber-or-SDE1-or-SF-or-Oct-2019-Offer>
  + <https://leetcode.com/problems/shortest-word-distance-ii>
  + <https://leetcode.com/problems/meeting-rooms-ii>
  + <https://leetcode.com/problems/word-search-ii>
  + <https://leetcode.com/problems/sliding-window-maximum>
  + <https://leetcode.com/problems/insert-delete-getrandom-o1> : Followup - edge cases and multithreading
  + <https://leetcode.com/problems/letter-combinations-of-a-phone-number/description/>
  + <https://leetcode.com/problems/lru-cache/>
  + India: <https://www.geeksforgeeks.org/segregate-even-and-odd-numbers/>
  + TSP
  + <https://leetcode.com/problems/squares-of-a-sorted-array>
  + Given a partially sorted array, find min range (i and j) such that sorting the array from i to j will sort the entire array. 581. Shortest Unsorted Continuous Subarray
  + <https://leetcode.com/problems/word-ladder/>
  + <https://leetcode.com/problems/asteroid-collision/>
  + <https://leetcode.com/discuss/interview-question/360770/Uber-or-Onsite-or-Jumping-Numbers>
  + <https://leetcode.com/discuss/interview-question/297202/Uber-or-Onsite-or-Surround-substrings>
  + <https://leetcode.com/problems/basic-calculator/>
  + <https://leetcode.com/problems/remove-invalid-parentheses/description/>
  + <https://leetcode.com/discuss/interview-question/498054/Uber-or-OA-2020-or-composeKPalindrome>
  + <https://leetcode.com/problems/path-sum/>
  + <https://leetcode.com/problems/3sum-smaller>
  + <https://leetcode.com/problems/target-sum/>
  + <https://leetcode.com/problems/shortest-path-in-a-grid-with-obstacles-elimination/>
  + <https://leetcode.com/problems/accounts-merge/>
  + <https://leetcode.com/problems/interval-list-intersections/description/>
  + <https://leetcode.com/problems/longest-consecutive-sequence/>
  + ALl interval questions
  + Subset sum
  + Pattern matching
  + <https://leetcode.com/problems/longest-common-subsequence/>
  + <https://leetcode.com/problems/search-a-2d-matrix/>
  + <https://leetcode.com/problems/lfu-cache/>
  + https://leetcode.com/problems/binary-tree-vertical-order-traversal/
  + Solve an equation in string with two variables x and y, with x given, solve y. Only +, - and multiplication in form of 'kx'.
  + Reinvent the wheel question (e.g. iterator class)
  + Construct binary tree from IOT and PrOT IOT iteratively Removing dups from a list
  + Minesweeper Graph problem, Spiral Matrix
  + Serialize and deserialize n ary general tree.
  + LRU
  + <https://leetcode.com/problems/pancake-sorting/>
  + <https://leetcode.com/problems/one-edit-distance/>
  + Find if tree is a valid BST
  + <https://leetcode.com/problems/longest-valid-parentheses/>
  + <https://leetcode.com/problems/generate-parentheses/>
  + <https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree/>
  + Design
    - Design hashmap
    - <https://leetcode.com/discuss/interview-experience/456079/Give-Away-How-to-prepare-for-System-Design-Interviews>
    - Amazon: think about a bookstore system. The database diagram, and some details.
    - Design a system to check the status (ping) of a website. Similar to [Status.io](http://status.io/).Big deal was how to schedule all the tasks every x seconds.
    - SWE:
      * Design Uber eats
      * How to implement caching
      * Geolocation based user tracking
      * Node Clustering and Database replication
    - The problem was about maintaining the count of rides a driver has completed. We had discussion around distributed hash tables, network failure, eventual consistency, queueing systems etc.
    - SDE2: Design a photo sharing application like instagram.
      * Design a meeting scheduler system like we've in outlook. Follow-up: How will you deal with recurring meetings?
    - Uber: <https://leetcode.com/discuss/interview-question/124558/Uber-or-Rate-Limiter>
    - Uber: Design a Elevator system considering yourself as a elevator company. Only classes and interfaces. Should be able to handle multiple elevators in a building, and elevators servicing only a subset of floors
    - Design a system of my choice, where if i need to add new code, new module, instead of if-else-if-else blocks i have to design my classes in such a way that, very minimal changes has to be done for the classes. basically, classes should be open for extension and closed for modification ( i dont know this at that time) I have to use stratergy pattern to dynamically select algorithms on the fly. I knew nothing. I failed here.
    - Uber architecture: design UBER.Which i fairly did with the help of quadtree
      * Design uber backend - define use cases, scope on your own, come up with various components, give high and low level design....
      * <https://leetcode.com/discuss/interview-question/124542/Design-Uber-Backend>
    - <https://leetcode.com/discuss/interview-question/341504/Uber-or-Implement-ScheduledExecutorService>
    - Design QR code system for grocery
    - Design google pay or paytm
    - <https://leetcode.com/discuss/interview-question/124936/Design-an-elevator-system>
    - implement a API rate limiter in the UI
    - Design a notification service.
    - Design message broker with high throughput and fault tolerant
    - Design slack
    - Design an object store system like S3
    - Design concurrent hashmap
    - Design a distributed system for sorting of large files.
    - Uber: L4: Design google auto compelete system in very detail. From user search to Trie update. Database to aggregator, ML algorithms to mapreduce.
      * Design social network network graph, and quering system.
    - L5: Design chat application.
    - Design distributed niotification service.
  + Behaiour
    - <https://leetcode.com/discuss/interview-question/437082/Amazon-Behavioral-questions-or-Leadership-Principles-or-LP>
    - <https://www.google.com/search?q=Dan+Croiter&oq=Dan+Croiter&aqs=chrome..69i57j0l7.688j0j4&sourceid=chrome&ie=UTF-8>
    - I can't think of a specific example, but if I was in that situation, I'd handle it like so...
    - Questions on conflicts within team and outside team. Strengths and Weakness.
    - questions around conflict resolution, taking ownership and more responsibility, mentoring new engineers, qualities of a great engineer and a few more. For every question, I was asked to provide an example from my experiences.
    - Tell me about a time when you had hard time working with someone in your team. How did you handle it?
    - Tell me about a challenging customer issue you dealt with.
    - Tell me about a time when you pushed for an idea but it was opposed by others.
    - Discussion on current project: end-to-end flow, what part you are responsible for, most challenging part for you etc.
    - What is one thing that excites you about working at Uber
    - handling disagreements
    - Example of Conflict resolution, Project I liked the most, What I liked about the project, what you are essentially looking for in the new assignments?

**Improve**: 2 coding, 1 HM, 1 behav, 1 design

* Sys oop design - prepare on white board
* Find edge cases
* Revise solution and complexity.
* Behevioural
* Resume
* Do all mediums, and biheart idea of all hard ones.
* Dfs of topoligical sort
* Bring rough book
* Be short while introducing yourself, no longer than 3 minutes (time it!)
* Clarify question before jumping.
* Hiring manage - checks fit for team
* Bar raise - checks fit for current level. Or uber - resume
* Ask lots of questions, and present multiple solutions. My general flow was to first ask as many questions about size limits, overflow, input validation, etc. that I could think of. Next, start spitballing as many possible solutions as you can think of and their big O complexity and trade-offs.
* Binary search
* Chekc uber linkedin
* <https://leetcode.com/tag/graph/>
* <https://leetcode.com/tag/union-find/>
* <https://leetcode.com/tag/breadth-first-search/>
* <https://leetcode.com/tag/depth-first-search/>

**In interview**:

* If feeling hopeless, close your eyes for a second, think to focus on problem and not on that you will loss. Open your eyes and think about problem only.

**Lessons Learned**:

* Do a lot of mock/pramp interviews. Since simulating that kind of thought is really imporant.
  + Asked in vmware question sprial matrix: done several times before, but forgot.
* Before interview, make sure that not be over confident