A BETTER VERSION ->> https://neetcode.io/

| Video Solution | Category | Name | Link | Notes |
|---|---|---|---|---|
| https://youtu.be/KLlXCFG5TnA | Arrays | Two Sum | https://leetcode. | use hash map to instantly check for difference value, map will add index of last occurrence of a num, don't use same element twice; |
| https://youtu.be/1pkOgXD63yU | Arrays | Best Time to Buy and Sell Stock | https://leetcode. | find local min and search for local max, sliding window; |
| https://youtu.be/3OamzN90kPg | Arrays | Contains Duplicate | https://leetcode. | hashset to get unique values in array, to check for duplicates easily |
| https://youtu.be/bNvIQI2wAjk | Arrays | Product of Array Except Self | https://leetcode. | make two passes, first in-order, second in-reverse, to compute products |
| https://youtu.be/5WZl3MMT0Eg | Arrays | Maximum Subarray | https://leetcode. | pattern: prev subarray cant be negative, dynamic programming: compute max sum for each prefix |
| https://youtu.be/lXVy6YWFcRM | Arrays | Maximum Product Subarray | https://leetcode. | dp: compute max and max-abs-val for each prefix subarr; |
| https://youtu.be/nIVW4P8b1VA | Arrays | Find Minimum in Rotated Sorted Arra | https://leetcode. | check if half of array is sorted in order to find pivot, arr is guaranteed to be in at most two sorted subarrays |
| https://youtu.be/U8XENwh8Oy8 | Arrays | Search in Rotated Sorted Array | https://leetcode. | at most two sorted halfs, mid will be apart of left sorted or right sorted, if target is in range of sorted portion then search it, otherwise search other half |
| https://youtu.be/jzZsG8n2R9A | Arrays | 3Sum | https://leetcode. | sort input, for each first element, find next two where -a = b+c, if a=prevA, skip a, if b=prevB skip b to elim duplicates; to find b,c use two pointers, left/right on remaining list; |
| https://youtu.be/UuiTKBwPgAo | Arrays | Container With Most Water | https://leetcode. | shrinking window, left/right initially at endpoints, shift the pointer with min height; |
| https://youtu.be/gVUrDV4tZfY | Binary | Sum of Two Integers | https://leetcode. | add bit by bit, be mindful of carry, after adding, if carry is still 1, then add it as well; |
| https://youtu.be/5Km3utixwZs | Binary | Number of 1 Bits | https://leetcode. | modulo, and dividing n; mod and div are expensive, to divide use bit shift, instead of mod to get 1's place use bitwise & 1; |
| https://youtu.be/RyBM56RIWrM | Binary | Counting Bits | https://leetcode. | write out result for num=16 to figure out pattern; res[i] = res[i - offset], where offset is the biggest power of 2 <= I; |
| https://youtu.be/WnPLSRLSANE | Binary | Missing Number | https://leetcode. | compute expected sum - real sum; xor n with each index and value; |
| https://youtu.be/UcoN6UjAI64 | Binary | Reverse Bits | https://leetcode. | reverse each of 32 bits; |
| https://youtu.be/Y0lT9Fck7qI | Dynamic Programming | Climbing Stairs | https://leetcode. | subproblem find (n-1) and (n-2), sum = n; |
| https://youtu.be/H9bfqozjoqs | Dynamic Programming | Coin Change | https://leetcode. | top-down: recursive dfs, for amount, branch for each coin, cache to store prev coin_count for each amount; bottom-up: compute coins for amount = 1, up until n, using for each coin (amount - coin), cache prev values |
| https://youtu.be/qiWnW0hdF1Y | Dynamic Programming | Longest Increasing Subsequence | https://leetcode. | recursive: foreach num, get subseq with num and without num, only include num if prev was less, cache solution of each; dp=subseq length which must end with each num, curr num must be after a prev dp or by itself; |
| https://youtu.be/Ua0GhsJSlWM | Dynamic Programming | Longest Common Subsequence | https://leetcode. | recursive: if first chars are equal find lcs of remaining of each, else max of: lcs of first and remain of 2nd and lcs of 2nd remain of first, cache result; nested forloop to compute the cache without recursion; |
| https://youtu.be/Sx9NNgInc3A | Dynamic Programming | Word Break Problem | https://leetcode. | for each prefix, if prefix is in dict and wordbreak(remaining str)=True, then return True, cache result of wordbreak; |
| https://youtu.be/GBKI9VSKdGg | Dynamic Programming | Combination Sum | https://leetcode. | visualize the decision tree, base case is curSum = or > target, each candidate can have children of itself or elements to right of it inorder to elim duplicate solutions; |
| https://youtu.be/73r3KWiEvyk | Dynamic Programming | House Robber | https://leetcode. | for each num, get max of prev subarr, or num + prev subarr not including last element, store results of prev, and prev not including last element |
| https://youtu.be/rWAJCfYYOvM | Dynamic Programming | House Robber II | https://leetcode. | subarr = arr without first & last, get max of subarr, then pick which of first/last should be added to it |
| https://youtu.be/6aEyTjOwJJU | Dynamic Programming | Decode Ways | https://leetcode. | can cur char be decoded in one or two ways? Recursion -> cache -> iterative dp solution, a lot of edge cases to determine, 52, 31, 29, 10, 20 only decoded one way, 11, 26 decoded two ways |
| https://youtu.be/IIEsdxuD4lY | Dynamic Programming | Unique Paths | https://leetcode. | work backwards from solution, store paths for each position in grid, to further optimize, we don't store whole grid, only need to store prev row; |
| https://youtu.be/Yan0cv2cLy8 | Dynamic Programming | Jump Game | https://leetcode. | visualize the recursive tree, cache solution for O(n) time/mem complexity, iterative is O(1) mem, just iterate backwards to see if element can reach goal node, if yes, then set it equal to goal node, continue; |
| https://youtu.be/mQeF6bN8hMk | Graph | Clone Graph | https://leetcode. | recursive dfs, hashmap for visited nodes |
| https://youtu.be/EgI5nU9etnU | Graph | Course Schedule | https://leetcode. | build adjacentcy_list with edges, run dfs on each V, if while dfs on V we see V again, then loop exists, otherwise V isnt in a loop, 3 states= not visited, visited, still visiting |
| https://youtu.be/s-VkcjHqkGI | Graph | Pacific Atlantic Water Flow | https://leetcode. | dfs each cell, keep track of visited, and track which reach pac, atl; dfs on cells adjacent to pac, atl, find overlap of cells that are visited by both pac and atl cells; |
| https://youtu.be/pV2kpPD66nE | Graph | Number of Islands | https://leetcode. | foreach cell, if cell is 1 and unvisited run dfs, increment count and marking each contigous 1 as visited |
| https://youtu.be/P6RZZMu_maU | Graph | Longest Consecutive Sequence | https://leetcode. | use bruteforce and try to optimize, consider the max subseq containing each num; add each num to hashset, for each num if num-1 doesn't exist, count the consecutive nums after num, ie num+1; there is also a union-find solution; |
| https://youtu.be/6kTZYvNNyps | Graph | Alien Dictionary (Leetcode Premium) | https://leetcode. | chars of a word not in order, the words are in order, find adjacency list of each unique char by iterating through adjacent words and finding first chars that are different, run topsort on graph and do loop detection; |
| https://youtu.be/bXsUuowmnoQ | Graph | Graph Valid Tree (Leetcode Premium) | https://leetcode. | union find, if union return false, loop exists, at end size must equal n, or its not connected; dfs to get size and check for loop, since each edge is double, before dfs on neighbor of N, remove N from neighbor list of neighbor; |
| https://youtu.be/8f1XPm4WOUc | Graph | Number of Connected Components in | https://leetcode. | dfs on each node that hasn't been visited, increment component count, adjacency list; bfs and union find are possible; |
| https://youtu.be/A8NUOmlwOlM | Interval | Insert Interval | https://leetcode. | insert new interval in order, then merge intervals; newinterval could only merge with one interval that comes before it, then add remaining intervals; |
| https://youtu.be/44H3cEC2fFM | Interval | Merge Intervals | https://leetcode. | sort each interval, overlapping intervals should be adjacent, iterate and build solution; also graph method, less efficient, more complicated |
| https://youtu.be/nONCGxWoUfM | Interval | Non-overlapping Intervals | https://leetcode. | instead of removing, count how max num of intervals you can include, sort intervals, dp to compute max intervals up until the i-th interval; |
| https://youtu.be/PaJxqZVPhbg | Interval | Meeting Rooms (Leetcode Premium) | https://leetcode. | sort intervals by start time, if second interval doesn't overlap with first, then third def wont overlap with first; |
| https://youtu.be/FdzJmTCVyJU | Interval | Meeting Rooms II (Leetcode Premium | https://leetcode. | we care about the points in time where we are starting/ending a meeting, we already are given those, just separate start/end and traverse counting num of meetings going at these points in time; for each meeting check if a prev meeting has finished before curr started, using min heap; |
| https://youtu.be/G0_I-ZF0S38 | Linked List | Reverse a Linked List | https://leetcode. | iterate through maintaining cur and prev; recursively reverse, return new head of list |
| https://youtu.be/gBTe7lFR3vc | Linked List | Detect Cycle in a Linked List | https://leetcode. | dict to remember visited nodes; two pointers at different speeds, if they meet there is loop |
| https://youtu.be/XIdigk956u0 | Linked List | Merge Two Sorted Lists | https://leetcode. | insert each node from one list into the other |
| https://youtu.be/q5a5OiGbT6Q | Linked List | Merge K Sorted Lists | https://leetcode. | divied and conquer, merge lists, N totalnodes, k-lists, O(N*logk). For each list, find min val, insert it into list, use priorityQ to optimize finding min O(N*logk) |
| https://youtu.be/XVuQxVej6y8 | Linked List | Remove Nth Node From End Of List | https://leetcode. | use dummy node at head of list, compute len of list; two pointers, second has offset of n from first; |
| https://youtu.be/S5bfdUTrKLM | Linked List | Reorder List | https://leetcode. | reverse second half of list, then easily reorder it; non-optimal way is to store list in array; |
| https://youtu.be/T41rL0L3Pnw | Matrix | Set Matrix Zeroes | https://leetcode. | use sets to keep track of all rows, cols to zero out, after, for each num if it is in a zero row or col then change it to 0; flag first cell in row, and col to mark row/col that needs to be zeroed; |
| https://youtu.be/BJnMZNwUk1M | Matrix | Spiral Matrix | https://leetcode. | keep track of visited cells; keep track of boundaries, layer-by-layer; |
| https://youtu.be/fMSJSS7eO1w | Matrix | Rotate Image | https://leetcode. | rotate layer-by-layer, use that it's a square as advantage, rotate positions in reverse order, store a in temp, a = b, b = c, c = d, d = temp; |
| https://youtu.be/pfiQ_PS1g8E | Matrix | Word Search | https://leetcode. | dfs on each cell, for each search remember visited cells, and remove cur visited cell right before you return from dfs; |
| https://youtu.be/wiGpQwVHdE0 | String | Longest Substring Without Repeating | https://leetcode. | sliding window, if we see same char twice within curr window, shift start position; |
| https://youtu.be/gqXU1UyA8pk | String | Longest Repeating Character Replace | https://leetcode. | PAY ATTENTION: limited to chars A-Z; for each capital char, check if it could create the longest repeating substr, use sliding window to optimize; check if windowlen=1 works, if yes, increment len, if not, shift window right; |
| https://youtu.be/jSto004AJbM | String | Minimum Window Substring | https://leetcode. | need is num of unique char in T, HAVE is num of char we have valid count for, sliding window, move right until valid, if valid, increment left until invalid, to check validity keep track if the count of each unique char is satisfied; |
| https://youtu.be/9UtInBqnCgA | String | Valid Anagram | https://leetcode. | hashmap to count each char in str1, decrement for str2; |
| https://youtu.be/vzdNOK2oB2E | String | Group Anagrams | https://leetcode. | for each of 26 chars, use count of each char in each word as tuple for key in dict, value is the list of anagrams; |
| https://youtu.be/WTzjTskDFMg | String | Valid Parentheses | https://leetcode. | push opening brace on stack, pop if matching close brace, at end if stack empty, return true; |
| https://youtu.be/jXJ16kPFWg | String | Valid Palindrome | https://leetcode. | left, right pointers, update left and right until each points at alphanum, compare left and right, continue until left >= right, don't distinguish between upper/lowercase; |
| https://youtu.be/XYQecbcd6_c | String | Longest Palindromic Substring | https://leetcode. | foreach char in str, consider it were the middle, consider if pali was odd or even; |
| https://youtu.be/4RACzI5-du8 | String | Palindromic Substrings | https://leetcode. | same as longest palindromic string, each char in str as middle and expand outwards, do same for pali of even len; maybe read up on manachers alg |
| https://youtu.be/B1k_sxOSgv8 | String | Encode and Decode Strings (Leetcode | https://leetcode. | store length of str before each string and delimiter like '#'; |
| https://youtu.be/hTM3phVI6YQ | Tree | Maximum Depth of Binary Tree | https://leetcode. | recursive dfs to find max-depth of subtrees; iterative bfs to count number of levels in tree |
| https://youtu.be/vRbbcKXCxOw | Tree | Same Tree | https://leetcode. | recursive dfs on both trees at the same time; iterative bfs compare each level of both trees |
| https://youtu.be/OnSn2XEQ4MY | Tree | Invert/Flip Binary Tree | https://leetcode. | recursive dfs to invert subtrees; bfs to invert levels, use collections.deque; iterative dfs is easy with stack if doing pre-order traversal |
| https://youtu.be/Hr5cWUld4vU | Tree | Binary Tree Maximum Path Sum | https://leetcode. | helper returns maxpathsum without splitting branches, inside helper we also update maxSum by computing maxpathsum WITH a split; |
| https://youtu.be/6ZnyEApgFYg | Tree | Binary Tree Level Order Traversal | https://leetcode. | iterative bfs, add prev level which doesn't have any nulls to the result; |
| https://youtu.be/u4JAi2JJhI8 | Tree | Serialize and Deserialize Binary Tree | https://leetcode. | bfs every single non-null node is added to string, and it's children are added too, even if they're null, deserialize by adding each non-null node to queue, deque node, it's children are next two nodes in string; |
| https://youtu.be/E36O5SWp-LE | Tree | Subtree of Another Tree | https://leetcode. | traverse s to check if any subtree in s equals t; merkle hashing? |
| https://youtu.be/ihj4IQGZ2zc | Tree | Construct Binary Tree from Preorder a | https://leetcode. | first element in pre-order is root, elements left of root in in-order are left subtree, right of root are right subtree, recursively build subtrees; |
| https://youtu.be/s6ATEkipzow | Tree | Validate Binary Search Tree | https://leetcode. | trick is use built in python min/max values float("inf"), "-inf", as parameters; iterative in-order traversal, check each val is greater than prev; |
| https://youtu.be/5LUXSvjmGCw | Tree | Kth Smallest Element in a BST | https://leetcode. | non-optimal store tree in sorted array; iterative dfs in-order and return the kth element processed, go left until null, pop, go right once; |
| https://youtu.be/gs2LMfuOR9k | Tree | Lowest Common Ancestor of BST | https://leetcode. | compare p, q values to curr node, base case: one is in left, other in right subtree, then curr is lca; |
| https://youtu.be/oobqoCJlHA0 | Tree | Implement Trie (Prefix Tree) | https://leetcode. | node has children characters, and bool if its an ending character, node DOESN'T have or need char, since root node doesn't have a char, only children; |
| https://youtu.be/BTf05gs_8iU | Tree | Add and Search Word | https://leetcode. | if char = "." run search for remaining portion of word on all of curr nodes children; |
| https://youtu.be/asbcE9mZz_U | Tree | Word Search II | https://leetcode. | trick: I though use trie to store the grid, reverse thinking, instead store dictionary words, dfs on each cell, check if cell's char exists as child of root node in trie, if it does, update currNode, and check neighbors, a word could exist multiple times in grid, so don't add duplicates; |
| https://youtu.be/q5a5OiGbT6Q | Heap | Merge K Sorted Lists | https://leetcode. | we always want the min of the current frontier, we can store frontier in heap of size k for efficient pop/push; divide and conquer merging lists; |
| https://youtu.be/YPTqKIgVk-k | Heap | Top K Frequent Elements | https://leetcode. | minheap that's kept at size k, if its bigger than k pop the min, by the end it should be left with k largest; |
| https://youtu.be/itmhHWaHupI | Heap | Find Median from Data Stream | https://leetcode. | maintain curr median, and all num greater than med in a minHeap, and all num less than med in a maxHeap, after every insertion update median depending on odd/even num of elements; |