A BETTER VERSION	https://neetcode	e.io/			
	Category	Name	Link	Notes	
	Arrays	Two Sum		Notes The standard of the sta	
	Arrays	Best Time to Buy and Sell Stock	https://leetcode.com/problems/best-time-to-buy-and-sell-		
	Arrays	Contains Duplicate		hashset to get unique values in array, to check for duplicates easily	
	Arrays	Product of Array Except Self		make two passes, first in-order, second in-reverse, to compute products	
https://youtu.be/5WZI3MMT0Eg	Arrays	Maximum Subarray	https://leetcode.com/problems/maximum-subarray/	pattern: prev subarray can't be negative, dynamic programming: compute max sum for each prefix	
https://youtu.be/IXVy6YWFcRM	Arrays	Maximum Product Subarray	https://leetcode.com/problems/maximum-product-subarra	a dp: compute max and max-abs-val for each prefix subarr;	
https://youtu.be/nIVW4P8b1VA	Arrays	Find Minimum in Rotated Sorted Array	https://leetcode.com/problems/find-minimum-in-rotated-s	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.com/problems/search-in-rotated-sorted-a	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		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;	
	Arrays	Container With Most Water		shrinking window, left/right initially at endpoints, shift the pointer with min height;	
	Binary	Sum of Two Integers	https://leetcode.com/problems/sum-of-two-integers/	add bit by bit, be mindful of carry, after adding, if carry is still 1, then add it as well;	
	Binary	Number of 1 Bits		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;	
	Binary	Counting Bits		write out result for num=16 to figure out pattern; res[i] = res[i - offset], where offset is the biggest power of 2 <= 1;	
	Binary	Missing Number		compute expected sum - real sum; xor n with each index and value;	
	Binary	Reverse Bits		reverse each of 32 bits;	
	Dynamic Programming Dynamic Programming	Climbing Stairs Coin Change	https://leetcode.com/problems/climbing-stairs/ https://leetcode.com/problems/coin-change/	subproblem find [n-1] and [n-2], sum = n; the problem find [n-1] and [n-2], sum = n; the problem find [n-1] and [n-2], sum = n; the problem find [n-1] and [n-2], sum = n; the problem find [n-1] and [n-2], sum = n; the problem find [n-1] and [n-2]	
				top-cown: recursive exp., for announc, urant not each count, cache to sure previous count or each announce and the previous count or each announce and the previous count or announce anno	
	Dynamic Programming	Longest Increasing Subsequence		recursive. Interest main, get studest with main and window from mind or main many mindow from mind from mind from mindow from	
	Dynamic Programming	Longest Common Subsequence Word Break Problem		recursive. It into curso are equal min is or remaining or each, esser max or, is or in its after remain or zind arise to continue remain or inst, cache result, nested for loop to compute the cache without recursion, for each prefix, if prefix is in dict and wordbreak/remaining styll-free, then return True, cache result of wordbreak;	
	Dynamic Programming			to each print, it print is in the call who to the call control print, it print is in the call who to the call call and in the call call	
	Dynamic Programming			visualizate the decision tree, uses case to a cut and in	
	Dynamic Programming			subarr = arr without first & last, get max of subarr, then pick which of first/last should be added to it	
	Dynamic Programming	Decode Ways	https://leetcode.com/problems/decode-ways/	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	
	Dynamic Programming	Unique Paths		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;	
	Dynamic Programming	Jump Game		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,	
	Graph	Clone Graph	https://leetcode.com/problems/clone-graph/	recursive dfs, hashmap for visited nodes	
	Graph	Course Schedule		build adjacentry_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, visited, still visiting	
	Graph	Pacific Atlantic Water Flow		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;	
	Graph	Number of Islands		foreach cell, if cell is 1 and unvisited run dfs, increment cound and marking each contigous 1 as visited	
https://youtu.be/P6RZZMu_maU	Graph	Longest Consecutive Sequence		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		Alien Dictionary (Leetcode Premium)	https://leetcode.com/problems/alien-dictionary/	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/bXsUuownnoQ		Graph Valid Tree (Leetcode Premium)		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;	
https://youtu.be/8f1XPm4WOUc	Graph			ds on each node that hasn't been visited, increment component count, adjacency list; bfs and union find are possible;	
	Interval	Insert Interval		insert new interval in order, then merge intervals; newinterval could only merge with one interval that comes before it, then add remaining intervals; sort each interval, overlapping intervals should be adjacent, iterate and build solution; also graph method, less efficient, more complicated of the control of the control of the complex of the control of the contro	
	Interval	Merge Intervals Non-overlapping Intervals		Soft each microst, overlapping intervas should be adjusted, it leads and soft soft microst, and graph microst, essentially compared to the com	
https://youtu.be/PaJxqZVPhbg	Interval	Meeting Rooms (Leetcode Premium)		Instead or Femous, good new low maximum or meets by our an include, soft intervals, but to compute maxime return meets as, you to compute maxime return meets as, you confidence to compute maxime return meets as, you confidence to confidence and confidence to confidenc	
https://youtu.be/FdzJmTCVyJU	Interval	Meeting Rooms II (Leetcode Premium)		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 I	before curr started, using min hean:
	Linked List	Reverse a Linked List		iterate through maintaining cur and prev, recursively reverse, return new head of list	, , , , , , , , , , , , , , , , , , , ,
	Linked List	Detect Cycle in a Linked List	https://leetcode.com/problems/linked-list-cycle/	dict to remember visited nodes; two pointers at different speeds, if they meet there is loop	
	Linked List	Merge Two Sorted Lists	https://leetcode.com/problems/merge-two-sorted-lists/		
	Linked List	Merge K Sorted Lists	https://leetcode.com/problems/merge-k-sorted-lists/	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.com/problems/remove-nth-node-from-en	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.com/problems/reorder-list/	reverse second half of list, then easily reorder it; non-optimal way is to store list in array;	
	Matrix	Set Matrix Zeroes	https://leetcode.com/problems/set-matrix-zeroes/	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;	
	Matrix	Spiral Matrix		keep track of visited cells; keep track of boundaries, layer-by-layer;	
	Matrix	Rotate Image		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;	
	Matrix	Word Search	https://leetcode.com/problems/word-search/		
	String			dfs on each cell, for each search remember visited cells, and remove cur visited cell right before you return from dfs;	
			https://leetcode.com/problems/longest-substring-without-	sliding window, if we see same char twice within curr window, shift start position;	
	String	Longest Repeating Character Replacement	https://leetcode.com/problems/longest-substring-without- https://leetcode.com/problems/longest-repeating-characte	sliding window, if we see same char twice within curr window, shift start position; pMY ATTENTION: limited to chars A-2; 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 String	Longest Repeating Character Replacement Minimum Window Substring	https://leetcode.com/problems/longest-substring-without- https://leetcode.com/problems/longest-repeating-characte https://leetcode.com/problems/minimum-window-substring	sliding window, if we see same char butice within curr window, shift start position; BY 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=2 works, if yes, increment len, if not, shift window right; need is num of unique char in T, HAVE is num of char we have valid count for, move right until valid, if valid, increment left until invalid, to check validity keep track if the count of each unique char is statisfied;	
https://youtu.be/jSto0O4AJbM https://youtu.be/9UtInBqnCgA	String String String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram	https://leetcode.com/problems/longest-substring-without- https://leetcode.com/problems/longest-repeating-characte https://leetcode.com/problems/minimum-window-substrin https://leetcode.com/problems/valid-anagram/	sliding window, if we see same char twice within curr window, shift start position; @PM ATTENTION: limited to chars A-2; 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; need is num of unique char in T, HAVE is num of char we have valid count for, sliding window, move right until valid, increment left until invalid, to check validity keep track if the count of each unique char is satisfied; hashmap to count each char in str.j, decrement for str.2;	
https://youtu.be/jSto004AJbM https://youtu.be/9UtinBqnCgA https://youtu.be/vzdNOK2oB2E	String String String String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams	https://leetcode.com/problems/longest-substring-without- https://leetcode.com/problems/longest-repeating-character https://leetcode.com/problems/minimum-window-substrin https://leetcode.com/problems/valid-anagram/ https://leetcode.com/problems/group-anagrams/	slding window, if we see same char brice within curr window, shift start position; 90 M TENTION: limited to chars A-2; 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; need is num of unique char in T, 14ME 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; hashmap to count each char in str.), decrement for str.2; (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/jSto004AJbM https://youtu.be/9UtinBqnCgA https://youtu.be/vzdNOK2oB2E https://youtu.be/WTzjTskDFMg	String String String String String String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses	https://leetcode.com/problems/longest-substring-without- https://leetcode.com/problems/longest-repealing-character https://leetcode.com/problems/minimum-window-substrin https://leetcode.com/problems/valid-anagram/ https://leetcode.com/problems/group-anagrams/ https://leetcode.com/problems/group-anagrams/	sliding window, if we see same char twice within curr window, shift start position; #WATENTION: limited to chars A-2; 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; meed is num of nique char in F, 14Me; is num of char we have valid count for, sliding window, move right until valid, increment left until invalid, to check validity keep track if the count of each unique char is satisfied; hashmap to count each char in str1, decrement for str2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true;	
https://youtu.be/jStofO4A.JbM https://youtu.be/9UtinRgnCgA https://youtu.be/vzdNOK2oB2E https://youtu.be/WTziTskDFMg https://youtu.be/jJXJ16kPFWg	String String String String String String String String String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindrome	https://leetcode.com/aroblems/longest-substrine.without- thits://leetcode.com/aroblems/longest-repeating-strint- https://leetcode.com/aroblems/minimum-window-substri- https://leetcode.com/aroblems/valid-anagram/ https://leetcode.com/aroblems/valid-anagram/ https://leetcode.com/aroblems/group-anagrams/ https://leetcode.com/aroblems/group-anagrams/ https://leetcode.com/aroblems/yalid-paindrome/	slding window, if we see same char brice within curr window, shift start position; 90 M TENTION: limited to chars A-2; 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; need is num of unique char in T, 14ME 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; hashmap to count each char in str.), decrement for str.2; (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/jStofO4A.JbM https://youtu.be/9UtinRgnCgA https://youtu.be/vzdNOK2oB2E https://youtu.be/WTziTskDFMg https://youtu.be/jJXJ16kPFWg	String String String String String String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses	https://leetcode.com/problems/longest-substrine.without- https://leetcode.com/problems/longest-repeating-charact- https://leetcode.com/problems/longest-repeating-charact- https://leetcode.com/problems/valid-anagram/ https://leetcode.com/problems/valid-aparam/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/	sliding window, if we see same char twice within curr window, shift start position; @WATENTION: limited to char A-2; 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; need is num of unique char in T, HaVe is num of char we have valid count for, sliding window, move right until valid, increment left until invalid, to check validity keep track if the count of each unique char is satisfied; hashmap to count each char in str1, decrement for str2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; 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/jStoiO4AJbM https://youtu.be/9UtinBqnCgA https://youtu.be/wzdNOK2o82E https://youtu.be/WziTskDFMg https://youtu.be/JIXIf&PFWg https://youtu.be/JIXIf&PFWg https://youtu.be/XYQecbcd6_c	String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindrome Longest Palindromic Substring	https://leetcode.com/problems/longest-substrine.without- https://leetcode.com/problems/longest-repeating-charact- https://leetcode.com/problems/longest-repeating-charact- https://leetcode.com/problems/valid-anagram/ https://leetcode.com/problems/valid-aparam/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/ https://leetcode.com/problems/valid-parentheses/	sliding window, if we see same char twice within curr window, shift start position; #W ATENTION: limited to chars A-2; 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; 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; hashmap to count each char in str1, decrement for str2; 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; push opening brace on stack, pop if maching close brace, at end if stack empty, return true; 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; foreach char in str, consider it were the middle, consider if pall was odd or even; ame as longest pallnormic string, each char in str as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg	
https://youtu.be//St0004AJbM https://youtu.be/9UtinBanGgA https://youtu.be/varlOKZoBZE https://youtu.be/VTITSkDFMg https://youtu.be/IXIIGkPFMg https://youtu.be/IXIGkPFMg https://youtu.be/IXIGkPFMg https://youtu.be/IXIGkPFMg https://youtu.be/IXIGkPFMg https://youtu.be/IXIGkPFMg https://youtu.be/BIk sxOSgv8	String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parientheses Valid Palindrome Longest Palindromic Substring Palindromic Substring	https://leetcode.com/aroblems/longest-substrine.without- https://leetcode.com/aroblems/longest-repeatine.charact- https://leetcode.com/aroblems/minimum-window-substri- https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-parentheses/ https://leetcode.com/aroblems/walid-parentheses/ https://leetcode.com/aroblems/subit-palindromic-substrinus/ https://leetcode.com/aroblems/palindromic-substrinus/ https://leetcode.com/aroblems/palindromic-substrinus/ https://leetcode.com/aroblems/palindromic-substrinus/ https://leetcode.com/aroblems/palindromic-substrinus/	sliding window, if we see same char twice within curr window, shift start position; #W ATENTION: limited to chars A-2; 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; 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; hashmap to count each char in str1, decrement for str2; 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; push opening brace on stack, pop if maching close brace, at end if stack empty, return true; 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; foreach char in str, consider it were the middle, consider if pall was odd or even; ame as longest pallnormic string, each char in str as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg	
https://youtu.be/jSto004A.lbM https://youtu.be/JSUInBanGcA https://youtu.be/JSUInBanGcA https://youtu.be/WTiTsIOEMg https://youtu.be/WTiTsIOEMg https://youtu.be/SYGecbcd6 c https://youtu.be/SRCits-du8 https://youtu.be/SRCits-du8 https://youtu.be/SIL ssoSgv8 https://youtu.be/SIL ssoSgv8 https://youtu.be/MINBahWiSQ	String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parlintheses Valid Palindrome Longest Palindromic Substring Palindromic Substring Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree	https://leetcode.com/aroblems/longest-substrine.without https://leetcode.com/aroblems/longest-spealine.in-charact https://leetcode.com/aroblems/minimum-windows-substri https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-palindromie- https://leetcode.com/aroblems/walid-palindromie- https://leetcode.com/aroblems/longest-palindromie-substrings/ https://leetcode.com/aroblems/faindromie-substrings/ https://leetcode.com/aroblems/faindromie-substrings/ https://leetcode.com/aroblems/faindromie-substrings/ https://leetcode.com/aroblems/faindromie-substrings/ https://leetcode.com/aroblems/faindromie-substrings/ https://leetcode.com/aroblems/maximum-depth-of-binary https://leetc	sliding window, if we see same char brice within curr window, shift start position; 9W ATTENTION: limited to chars A-2; 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; 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; hashmap to count each char in 1 str.], decrement for str.2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, update left and right until each points at alphanum, companie left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach char in str, consider if were the middle, consider if pail was odd or even; same as longest palindromic string, each char in str as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg store length of str before each string and delimiter like W'; recursive dis to find max-depth of subtrees; iterative bits to count number of levels in tree recursive dis to find max depth of subtrees; iterative bits on count number of level both trees	
https://youtu.be/jSto004A.lbM https://youtu.be/jUtinBanGA https://youtu.be/jUtinBanGA https://youtu.be/wTijTs0FMg https://youtu.be/wTijTs0FMg https://youtu.be/jXijCs0FMg https://youtu.be/sXijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/mg https://yo	String Tree Tree	Longest Repeating Character Replacement Minimum Window Substring Valid Anagrams Valid Parentheses Valid Palindrome Longest Palindromic Substring Palindromic Substrings Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree	https://leetcode.com/aroblems/longest-substrine.without. https://leetcode.com/aroblems/longest-repeating-charact https://leetcode.com/aroblems/longest-repeating-charact https://leetcode.com/aroblems/minimum-window-substri https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-apaintmorne/ https://leetcode.com/aroblems/valid-paintmorne/ https://leetcode.com/aroblems/valid-paintmorne/ https://leetcode.com/aroblems/longest-paintmorne/ https://leetcode.com/aroblems/longest-paintmorne/ https://leetcode.com/aroblems/longest-paintmorne/ https://leetcode.com/aroblems/longest-paintmorne/ https://leetcode.com/aroblems/longest-paintmorne/ https://leetcode.com/aroblems/longest-paintmorne/ https://leetcode.com/aroblems/longest-paintmorne/ https://leetcode.com/aroblems/minimum-depth-of-binary- https://leetcode.com/aroblems/minimum-depth-of-binary- https://leetcode.com/aroblems/minimum-tree/	skding window, if we see same char twice within curr window, shift start position; 8W ATENTION: Imited to Chara A-2; 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; need is num of unique char in T, 14We is num of char we have valid count for, skilling window, move right until valid, increment left until invalid, to check validity keep track if the count of each unique char is satisfied; hashmap to count each char in str.l, decrement for str.2; for each of 26 chars, use count of each har in each word as tuple for key in dict, value is the list of anagrams; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; 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; foreach char in str., consider it were the middle, consider if pall was odd or even; same as longest palindromic string, each char in str as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg store length of str. before each string and delimiter like "; recursive dis to find max-depth of subtrees; iterative bis to count number of levels in tree recursive dis to find max-depth of subtrees; iterative bis compare each level of both trees recursive dis to invert subtrees; but somet levels, use collections. deeque, terestive dis is easy with stack if doing pre-order traversal	
https://youtu.be/jSto004A.lbM https://youtu.be/jUtinBanGA https://youtu.be/jUtinBanGA https://youtu.be/wTijTs0FMg https://youtu.be/wTijTs0FMg https://youtu.be/jXijCs0FMg https://youtu.be/sXijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/sijCs0FMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/htMg https://youtu.be/mg https://yo	String Tring String String String	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parlintheses Valid Palindrome Longest Palindromic Substring Palindromic Substring Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree	https://leetcode.com/aroblems/longest-substrine.without- thitss://leetcode.com/aroblems/longest-repeatine-fact- https://leetcode.com/aroblems/minimum-window-substri- https://leetcode.com/aroblems/galid-anagram/ https://leetcode.com/aroblems/galid-anagram/ https://leetcode.com/aroblems/galid-palindromis- https://leetcode.com/aroblems/galid-palindromis- https://leetcode.com/aroblems/galid-palindromis-lubstrings/ https://leetcode.com/aroblems/longest-palindromis-lubstrings/ https://leetcode.com/aroblems/encode-and-recode-string https://leetcode.com/aroblems/mainum-depth-of-binar- https://leetcode.com/aroblems/mainum-depth-of-binar- https://leetcode.com/aroblems/mainum-depth-of-binar- https://leetcode.com/aroblems/mainum-depth-of-binar- https://leetcode.com/aroblems/mainum-depth-of-binar- https://leetcode.com/aroblems/mainum-depth-of-binar-vtree/ https://leetcode.com/aroblems/mainum-depth-of-binar-vtree/	sliding window, if we see same char brice within curr window, shift start position; 90% ATENTION: limited to chars A-2; 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; need is num of orinque char in T, 140KE 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; hashmap to count each char in str.), decrement for str.?; for each of 26 shars, use count of each or in each word as tuple for key in dict, value is the list of anagrams; push opening brace on stack, opp if matching close brace, at end if stack empty, return true; left, right pointers, update left and right until each points at alphanum, compane left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach that in str, consider if were the middle, consider if pail was odd or even; same as longest pulliformic string, each char in rist as middled and expand outwards, do same for pail of even len; maybe read up on manachers alg store length of str before each string and delimiter like '#'; recursive dis to inoth trees at the same time, iterative bits conpane each level of both trees recursive dis to inboth trees at the same time, iterative bits conpane each level of both trees recursive dis to inwert subres; bits to invert levels, use collections deque; iterative dis is easy with stack if doing pre-order traversal helper returns mapathsum without splitting branches, include helper we also undeste massium by computing manapsthsum WITH a split;	
https://youtu.be/jSto004A.lbM https://youtu.be/SUIRIBAGGA https://youtu.be/SUIRIBAGGA https://youtu.be/WISTISOFMg https://youtu.be/WISTISOFMg https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/YXQecbcd6_c https://youtu.be/ISTISOFMGA_XXQecbcd6_c https://youtu.be/ISTI	String Tree Tree Tree Tree	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindrome Longest Palindrome Longest Palindrome Longest Palindrome Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Invert/Filip Binary Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traversal	https://leetcode.com/aroblems/longest-substrine.without- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-parentheses/ https://leetcode.com/aroblems/valid-paintheses/ https://leetcode.com/aroblems/valid-paintheses/ https://leetcode.com/aroblems/longest-paintheses/ https://leetcode.com/aroblems/longest-paintheses/ https://leetcode.com/aroblems/longest-paintheses/ https://leetcode.com/aroblems/maintheses/ https://leetcode.com/aroblems/maintheses/ https://leetcode.com/aroblems/maintheses/ https://leetcode.com/aroblems/maintheses/ https://leetcode.com/aroblems/maintheses/ https://leetcode.com/aroblems/maintheses/ https://leetcode.com/aroblems/hanat-tree-leet-leet-fat- https://leetcode.com/aroblems/hanat-tree-leet-leet-order-tra	stding window, if we see same char brice within curr window, shift start position; #W ATENTION: limited to clark 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; need is num of unique char in T, 1MeV is mun of char we have valid count for; sliding window, move right until valid, increment left until invalid, to check validity keep track if the count of each unique char is satisfied; hashmap to count each char in str.l, decrement for str.2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; 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; foreach char in str, consider it were the middle, consider if pail was odd or even; same as longest palindromic string, each char in str as middle and eapand outwards, do same for pail of even len; maybe read up on manachers alg store length of str. before each string and delimiter like "; recursive dis to find max depth of subtress; iterative bits to count number of levels in tree recursive dis to invert subtress; bits to invert levels, use collections.deque; iterative dis is easy with stack if doing pre-order traversal helper returns maxpathsum without splitting branches, inside helper we also update maxSum by computing maxpathsum WITH a split; iterative bits, add previously the count of the pass of the count of the result; literative bits, add previously the count of the coun	
https://youtu.be/Stx0004A.bM https://youtu.be/sufNoRcoB2 https://youtu.be/sufNoRcoB2 https://youtu.be/wThT8x0FMg https://youtu.be/WThT8x0FMg https://youtu.be/WThT8x0FMg https://youtu.be/NYGeb6d5 c https://youtu.be/StRCoEd6d5 https://youtu.be/StRCoEd6d5 https://youtu.be/StRCoEd6d5 https://youtu.be/StRCoEd6d5 https://youtu.be/StRCoEd6d6d5 https://youtu.be/StRCoEd6d6d6d6d6d6d6d6d6d6d6d6d6d6d6d6d6d6d6	String Tree Tree Tree Tree Tree Tree	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindromic Substring Palindromic Substring Palindromic Substring Palindromic Substrings Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traversal Servatize and Descalaze Binary Tree	https://leetcode.com/aroblems/longest-substrine.without- https://leetcode.com/aroblems/longest-repeatine-strine- https://leetcode.com/aroblems/minimum-window-substri- https://leetcode.com/aroblems/willid-aromanisms/ https://leetcode.com/aroblems/valid-aromanisms/ https://leetcode.com/aroblems/valid-patentheses/ https://leetcode.com/aroblems/longest-palindromic-substrings/ https://leetcode.com/aroblems/longest-palindromic-substrings/ https://leetcode.com/aroblems/longest-palindromic-substrings/ https://leetcode.com/aroblems/longest-palindromic-substrings/ https://leetcode.com/aroblems/mamium-depth-d-binary- https://leetcode.com/aroblems/mamium-depth-d-binary- https://leetcode.com/aroblems/mamium-depth-d-binary-tree/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/hinary-tree-mamium-palintss/ https://leetcode.com/aroblems/ https://leetcode.com/aroblems/ https://leetcode.com/aroblems/ https://leetcode.com/aroblems/ https://leetcode.com/aroblems/ https://leetcode.com/aroblems/ https://leetcode.com/aroblems/ https://leetcode.com/aroblems/ https://lee	sliding window, if we see same char brice within curr window, shift start position; 90% ATTENTION: limited to chars A-2; 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; are dis num of orinque chair in 7, 140KE is num of chair 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; hashmap to count each chair in str), decrement for str2; for each of 26 chairs, use count of each orin each word as tuple for key in dict, value is the list of anagrams; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, update left and right until each points at alphanum, compane left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach chair in str, consider it were the middle, consider if pail was odd or even; same as longest paillardomic string, each chair in str as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg store length of str before each string and delimiter like 'B'; cursived its for lima as depth of substress; letterable bits count number of levels in tree recursive dits to lima as depth of substress; letterable bits count number of levels in tree recursive dits to invert substress; bits to invert levels, use collections.deque; terrative difs is easy with stack if doing pre-order traversal helper returns magnathsum without splitting branches, inside helper we also update masSum by computing maspathsum WTH1 a split; iterative bits, add prev level which doesn't have any nulls to the result; bit every single non-null node is added to string, and it's children are added too, even if they're null, desertailize by adding each non-null node to queue, deque node, it's children are next two nodes in string;	
https://youtu.be/Sto004A.bM https://youtu.be/SUIRBAGGA https://youtu.be/SUIRBAGGA https://youtu.be/WTSTAGFMg https://youtu.be/WTSTAGFMg https://youtu.be/ENGC66066, https://youtu.be/ENGC66066, https://youtu.be/ENGC66066, https://youtu.be/ENGC606066, https://youtu.be/ENGC6060606, https://youtu.be/ENGC6060606, https://youtu.be/ENGC6060606, https://youtu.be/SUIRG6060606, ht	String Tring String String String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindromic Longest Palindromic Substring Palindromic Substrings Palindromic Substrings Paccode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Invert/Flip Binary Tree Silnary Tree Maximum Path Sum Binary Tree Level Order Traversal Serfalize and Desertalize Binary Tree Subtree of Another Tree	https://leetcode.com/aroblems/longest-substring-without- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/minimum-window-substri- https://leetcode.com/aroblems/walid-anagram/. https://leetcode.com/aroblems/walid-anagram/. https://leetcode.com/aroblems/walid-palindromic-subst- https://leetcode.com/aroblems/walid-palindromic-subst- https://leetcode.com/aroblems/bing-talindromic-subst- https://leetcode.com/aroblems/bing-tode-and-decode-string- https://leetcode.com/aroblems/maximum-depth-of-binary- https://leetcode.com/aroblems/binare-tree/ https://leetcode.com/aroblems/binare-tree/ https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-maximum-pat- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra- https://leetcode.com/aroblems/binare-tree-leet-order-tra-https://leetcode.com/aroblems/binare-tree-leet-order-tra-https://leetcode.com/aroblems/binare-tree-leet-order-tra-https://leetcode.com/aroblems/binare-tree-leet-order-tra-https://leetcode.com/aroblems/binare-tree-leet-order-tra-https://leetcode.com/aroblems/binare-dra-https://leetcode.com/aroblems/binare-dra-https://leetco	sliding window, if we see same char butice within curr window, shift start position; #W ATTENTON: limited to chars A-2; 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; meed is num of unique char in T, HAVE is num of char we have valid count for, skilding 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; hashmap to count each char in str., decrement for str.2; for each of 26 chairs, use count of each char in each word as tuple for key in dict, value is the list of anagrams; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, update left and right until each points at aphanum, compane left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach thar in str, consider it were the middle, consider if pall was odd or even; same as longest palindromic string, each char in str as middle and eapand outwards, do same for pall of even len; maybe read up on manachers alg store length of str. before each string and delimiter like ""; recursive dis to noth trees at the same time, iterative bits on compare each level of both trees recursive dis to noth trees at the same time, iterative bits on compare each level of both trees recursive dis to noth trees at the same time, iterative bits on compare each level of both trees recursive dis to noth trees at the same time, iterative bits on compare each level of both trees recursive dis to noth trees at the same time, iterative bits on compare each level of both trees recursive dis to noth trees at the same time, iterative bits, on compare each level of both trees recursive dis to noth trees at the same time, iterative bits, and generally the same time to the result; bits every single non-rull node to queue, deque node, it's children ar	
https://youtu.be/StoOO4A.bM https://youtu.be/StoOO4A.bM https://youtu.be/vdNOc2oB2E https://youtu.be/vdNOc2oB2E https://youtu.be/vdNOc2oB2E https://youtu.be/VMTSTSOPMg https://youtu.be/WTSTSOPMg https://youtu.be/MTSTSOPMg https://youtu.be/NYGeb665.c https://youtu.be/NYGeb665.c https://youtu.be/STBC.sOSyG https://youtu.be/MTSGOMHOSYO https://youtu.be/STGOMSSWO-LE https://youtu.	String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Bepeating Character Replacement Minimum Window Substring Valid Anagrams Valid Parentheses Valid Palindrome Longest Pali	https://leetcode.com/aroblems/longest-substrine.without- https://leetcode.com/aroblems/longest-repeating-strine- https://leetcode.com/aroblems/longest-repeating-strin- https://leetcode.com/aroblems/valid-anagram/. https://leetcode.com/aroblems/valid-anagram/. https://leetcode.com/aroblems/valid-anagram/. https://leetcode.com/aroblems/longest-palindromic-substrines/. https://leetcode.com/aroblems/longest-palindromic-substrines/. https://leetcode.com/aroblems/longest-palindromic-substrines/. https://leetcode.com/aroblems/longest-palindromic-substrines/. https://leetcode.com/aroblems/longest-palindromic-substrines/. https://leetcode.com/aroblems/mamium-depth-to-binay- https://leetcode.com/aroblems/mamium-depth-to-binay- lthps://leetcode.com/aroblems/mamium-depth-to-binay-longest-palindromic-substrines/. https://leetcode.com/aroblems/palinan-tree-maximum-palintps://leetcode.com/aroblems/palinan-tree-level-order-tra- https://leetcode.com/aroblems/palinan-tree-level-order-tra- https://leetcode.com/aroblems/palinan-tree-level-order-tra- https://leetcode.com/aroblems/palinan-tree-franther-tree-frant	slding window, if we see same char brice within curr window, shift start position; 90% ATTENTION: limited to char A-S, or each capital char, check if it could create the longest repeating substr, use sliding window to optimize; check if window/end window for the count of each unique char in S, HAVE is num of char we have valid count for, slding 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; hashmap to count each char in str), decrement for str2; for each of 26 chars, use count of each or in each word as tuple for key in dict, value is the list of anagrams; push opening brace on stack, upor if matching close brace, at end if stack empty, return true; left, right pointers, update left an diright until each points at alphanum, compare left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach char in str, consider it were the middle, consider if pail was odd or even; same as longest pailleromic string, each char in str as modelled and expand outwards, do same for pail of even len; maybe read up on manachers alg store length of str before each string and delimiter like 'it'; encursive dits for lima ace depth of substress; terative bits coonn number of levels in tree recursive dits on both trees at the same time, iterative bits compare each level of both trees recursive dits to lima ace leght of substress; terative bits compare each level of both trees recursive dits to invert subtress; bits to invert levels, use collections deque; iterative difs is easy with stack if doing pre-order traversal helper returns managrathsum without splitting branches, inside helper we also update mas/cum by computing maspathsum WTH1 a split; literative bits, add prev level which doesn't have any nulls to the result; bit every single non-null node as ded each string and collections deque; iterative difs is easy with stack if doing pre-order traversal bits every single non-null node as ded each strin	
https://youtu.be/StoOoAA.bM https://youtu.be/StoOoAA.bM https://youtu.be/stMOGCoBZ https://youtu.be/wtMOGCoBZ https://youtu.be/WT3TsAOFMg https://youtu.be/WT3TsAOFMg https://youtu.be/BTACSFOW https://youtu.be/BTACSFOW https://youtu.be/BTACSFOW https://youtu.be/BTACSFOW https://youtu.be/STACSFOW https://	String Tren Tree Tree Tree Tree Tree Tree Tree	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindrome Longest Palindrome Longest Palindrome Longest Palindrome Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Invert/Filip Binary Tree Binary Tree Maximum eath Sum Binary Tree Maximum eath Sum Binary Tree Longest Palindrome Sertalize and Desertalize Binary Tree Southere of Another Tree Construct Binary Tree from Preorder and Inorder Validate Binary Tree Broarch Tree	https://lectode.com/aroblems/longest-substrine.without https://lectode.com/aroblems/longest-spealine.ichauch https://lectode.com/aroblems/minimum-window-substri https://lectode.com/aroblems/minimum-window-substri https://lectode.com/aroblems/valid-anargam/ https://lectode.com/aroblems/walid-palindromis- https://lectode.com/aroblems/longest-salindromis-ubstrines/ https://lectode.com/aroblems/longest-salindromis-ubstrines/ https://lectode.com/aroblems/sinidromis-ubstrines/ https://lectode.com/aroblems/sinidromis-ubstrines/ https://lectode.com/aroblems/sinidromis-ubstrines/ https://lectode.com/aroblems/maximum-depth-of-binary https://lectode.com/aroblems/maximum-depth-of-binary https://lectode.com/aroblems/longe-tree-maximum-salints/ https://lectode.com/aroblems/sinidra-tree-maximum-salints/ https://lectode.com/aroblems/sinidra-tree-level-orde-tre- https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-binary-tree-final-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/sinidra-de-sinidra-https://lectode.com/aroblems/	sliding window, if we see same char butice within curr window, shift start position; 90 X TTENTION: limited to chars A-2; 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; need is num of unique char in T, 14ME 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; hashmap to count each char in rist., decrement for stt2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, used used for and interest at alphanum, compane left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach daw in sit, consider if were the middle, consider if pail was odd or even; same as longest palindromic string, each char in str as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg store length of sit before each string, each char in str as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg store length of sit before each string, and delimiter like '#') recursive dis to infort max-depth of subtrees; leterative bits compare each level of both trees recursive dis to into three at the same time, leterative bits compare each level of both trees recursive dis to into three sit the same time, leterative bits compare each level of both trees recursive dis to into three sit the same time, leterative bits compare each level of both trees recursive dis to into three sit the same time, leterative bits of compare each level of both trees recursive dis to into three sit the same time, leterative bits of compare each level of both trees recursive dis to into three sit the same time, leterat	
https://youtu.be/Sto00AA.lbM https://youtu.be/SUIRBAGGA https://youtu.be/SUIRBAGGA https://youtu.be/WTSTADEPMg https://youtu.be/WTSTADEPMg https://youtu.be/WTSTADEPMg https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/XXQecbcd6_c https://youtu.be/TMBABDWBYQ https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/STADEPMg https://youtu.be/SUNSymGG	String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Bepeating Character Replacement Minimum Window Substring Valid Anagrams Group Anagrams Valid Parentheses Valid Palindromic Jobstring Palindromic Substring Palindromic Substrings Faced and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traveral Serialize and Desertalize Binary Tree Subtree of Another Tree Construct Binary Tree from Preorder and Inorder Validate Binary Search Tree Validate Binary Tree From Preorder and Inorder Validate Binary Search Tree Validate Binary Free From Preorder and Inorder Validate Binary Search Tree Kith Smallest Element in a BST	https://leetcode.com/aroblems/longest-substrine.without- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/minimum-window-substri- https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-anagram/ https://leetcode.com/aroblems/walid-painertheses/ https://leetcode.com/aroblems/sulid-painertheses/ https://leetcode.com/aroblems/sulid-painertheses/ https://leetcode.com/aroblems/sulid-painertheses/ https://leetcode.com/aroblems/mainertheses/ https://leetcode.com/aroblems/mainertheses/ https://leetcode.com/aroblems/mainertheses/ https://leetcode.com/aroblems/mainertheses/ https://leetcode.com/aroblems/mainertheses/ https://leetcode.com/aroblems/mainertheses/ https://leetcode.com/aroblems/sulnar-tree-leet-order-tra- https://leetcode.com/aroblems/sulnar-leet-order-tra- https://leetcode.com/aroblems/sulna-	stding window, if we see same char brice within curr window, shift start position; #W ATENTION: limited to chark A2; for each capital char, check if it could create the longest repeating substr, use sliding window to optimize; check if windowlens! works, if yes, increment len, if not, shift window right; need is num of unique char in T, 1MeS is mun of char we have valid count for; sliding window, move right until valid, increment left until invalid, to check validity keep track if the count of each unique char is satisfied; hashmap to count each char in str.l, decrement for str.2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; 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; foreach char in str, consider it were the middle, consider if pail was odd or even; same as longest palindromic string, each char in str as middle and eapand outwards, do same for pail of even len; maybe read up on manachers alg store length of str. before each string and delimiter like "Y; recursive dis to find max depth of subtress; iterative bits to count number of levels in tree recursive dis to invert subtress; bits to invert levels, use collections.deque; iterative dis is easy with stack if doing pre-order traversal heigher returns maxpathsum without splitting branches, inside heigher we also update maxisum by computing maxpathsum WITH a split; iterative bits, add previously have a paid to the result; bits every single non-null node is added to string, and it's children are added too, even if they're null, desertalize by adding each non-null node to queue, deque node, it's children are next two nodes in string; traverse sto to technic flam systems in secalast. I meth banks like lement processed, pile furth lunly, pp., go right none; first e	
https://youtu.be/Stx00A4.bM https://youtu.be/sutNoGc8282E https://youtu.be/wdNOGc8282E https://youtu.be/wdNOGc8282E https://youtu.be/WTSTx8DFMg https://youtu.be/WTSTx8DFMg https://youtu.be/STXGeSPG8 https://youtu.be/STXGeSPG8 https://youtu.be/STXGESPG9 https://you	String Tring String String String String String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindrome Longest Palindromic Substring Palindromic Substring Palindromic Substring Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Invert/Filip Binary Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traversal Serialize and Decodersitize Binary Tree Subtree of Another Tree Construct Binary Tree from Preorder and Inorder Validate Binary Search Tree Kth Smallest Element in a BST Lowest Comman Ancestor of BST	https://lectode.com/aroblems/longest-substrine.without- https://lectode.com/aroblems/longest-spealine.in-charuct- https://lectode.com/aroblems/minimum-window-substri- https://lectode.com/aroblems/minimum-window-substri- https://lectode.com/aroblems/wall-darentheses/ https://lectode.com/aroblems/wall-darentheses/ https://lectode.com/aroblems/mall-darentheses/ https://lectode.com/aroblems/mall-darentheses/ https://lectode.com/aroblems/mall-darentheses/ https://lectode.com/aroblems/mall-darentheses/ https://lectode.com/aroblems/mall-darentheses/ https://lectode.com/aroblems/mall-mall-mall-decode-string- https://lectode.com/aroblems/mall-mall-mall-decode-string- https://lectode.com/aroblems/mall-mall-tes- https://lectode.com/aroblems/mall-tes-mall-mall-mall- https://lectode.com/aroblems/scal-tize-and-desertal-tes- https://lectode.com/aroblems/scal-tize-and-desertal-tes- https://lectode.com/aroblems/scal-tize-and-desertal-tes- https://lectode.com/aroblems/scal-tize-and-desertal-tes- https://lectode.com/aroblems/scal-tize-and-desertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-and-sertal-tes-https://lectode.com/aroblems/scal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-and-sertal-tes-	sliding window, if we see same char brice within curr window, shift start position; 90 XTENTION: limited to chars A-2; 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; need is num of unique char in T, 14ME's num of char we have valid count for, siding 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; hashmap to count each char in rist. decrement for stiZ; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, used used for adjust until each points at alphanum, compane left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach char in str, consider it were the middle, consider if pail was odd or even; same as longest polluriormic string, each char in rist as middled and expand outwards, do same for pail of even len; maybe read up on manachers alg store length of str before each string and delimiter like '#'; recursive dis to nith trees at the same time, iterative bits conunt number of levels in tree recursive dis to nith trees at the same time, iterative bits compane each level of both trees recursive dis to inthe treas at the same time, iterative bits compane each level of both trees recursive dis to indome there at the same time, iterative bits compane each level of both trees recursive dis to inthe treas at the same time, iterative bits compane each level of both trees recursive dis to indome three sit the same time, iterative bits compane each level of both trees recursive dis to indome three sit the same time, iterative bits compane each level of both trees recursive dis to indome three sit the same time, iterative bits compane each level of both trees recursive d	
https://youtu.be/Sto004A.lbM https://youtu.be/SUIRBAGCA https://youtu.be/SUIRBAGCA https://youtu.be/WhiTistOFMg https://youtu.be/WhiTistOFMg https://youtu.be/WhiTistOFMg https://youtu.be/WhiTistOFMg https://youtu.be/KNGeb066. https://youtu.be/KNGeb0666. https://youtu.be/ITMSAMWSTQ https://youtu.be/ITMSAMWSTQ https://youtu.be/ITMSAMWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/ITMSAWWSTQ https://youtu.be/SUIXSWMGCW https://youtu.be/SUIXSWMGCW https://youtu.be/SUIXSWMGCW https://youtu.be/SUIXSWMGCW https://youtu.be/SUIXSWMGCW https://youtu.be/SUIXSWMGCW	String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Bepeating Character Replacement Minimum Window Substring Valid Anagrams Group Anagrams Valid Parentheses Valid Palindromic Jobstring Palindromic Substring Palindromic Substrings Faced and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traveral Serialize and Desertalize Binary Tree Subtree of Another Tree Construct Binary Tree from Preorder and Inorder Validate Binary Search Tree Validate Binary Tree From Preorder and Inorder Validate Binary Search Tree Validate Binary Free From Preorder and Inorder Validate Binary Search Tree Kith Smallest Element in a BST	https://leetcode.com/aroblems/longest-substring-without- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/minimum-window-substri- https://leetcode.com/aroblems/walld-aparents/ https://leetcode.com/aroblems/walld-aparents- https://leetcode.com/aroblems/walld-paintdnomic-substrings/ https://leetcode.com/aroblems/paintdnomic-substrings/ https://leetcode.com/aroblems/maintdnomic-substrings/ https://leetcode.com/aroblems/maintdnomic-substrings/ https://leetcode.com/aroblems/maintdnomic-substrings/ https://leetcode.com/aroblems/maintdnomic-substrings/ https://leetcode.com/aroblems/maintdnomic-substrings/ https://leetcode.com/aroblems/hinar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-order-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- https://leetcode.com/aroblems/binar-tree-leet-pree-tra- ter-pree-tra	sliding window, if we see same char brice within curr window, shift start position; #W ATTENTON: limited to chars A-2; for each capital char, check if it could create the longest repeating substr, use sliding window to optimize; check if windowien=2 works, if yes, increment len, if not, shift window right; meed 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 statisfied; hashmap to count each char in str.l, decrement for str.2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, update left and right until each points at aphanum, company rel and right, continue until left >= right, don't distinguish between upper/lowercase; foreach char in str, consider it were the middle, consider if pall was odd or even; same as longest palindromic string, each char in str as middle and eapand outwards, do same for pall of even len; maybe read up on manachers alg store length of str. before each string and delimiter like "! recursive dis to noth treas at the same time, lettake leb for compare each level of both trees recursive dis to noth treas at the same time, lettake leb for compare each level of both trees recursive dis to noth treas at the same time, lettake leb for compare each level of both trees recursive dis to noth treas at the same time, lettake leb for compare each level of both trees recursive dis to noth treas at the same time, lettake leb for compare each level of both trees recursive dis to noth treas at the same time, lettake leb for compare each level of both trees recursive dis to noth treas at the same time, lettake leb for compare each level of both trees recursive dis to noth treas at the same time, lettake leb for compare each level of both trees	
https://youtu.be/St0004A.bM https://youtu.be/St0004A.bM https://youtu.be/vdNOIc263E https://youtu.be/vdNOIc263E https://youtu.be/vdNOIc263E https://youtu.be/WTST86PMg https://youtu.be/WTST86PMg https://youtu.be/WTST86PMg https://youtu.be/ST8C5E05Mg https://youtu.be/SBC5E05Mg https://youtu.be/	String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Bepeating Character Replacement Minimum Window Substring Valid Anagrams Valid Parentheses Valid Palindromic Substring Palindromic Substring Palindromic Substring Palindromic Substring Palindromic Substrings Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traversal Serfalize and Describating Binary Tree Subtree of Another Tree Construct Binary Tree From Preorder and Inorder Validate Binary Search Tree Validate Maximum Ancestor of 85T Implement Trie (Prefix Tree) Add and Search Word	https://leetcode.com/aroblems/longest-substrine.without- https://leetcode.com/aroblems/longest-repeatine-tith- https://leetcode.com/aroblems/longest-repeatine-tith- https://leetcode.com/aroblems/longest-peatine-tith- https://leetcode.com/aroblems/longest-peatine-tith- https://leetcode.com/aroblems/longest-peatine-tien- https://leetcode.com/aroblems/longest-peatine-tien- https://leetcode.com/aroblems/longest-peatine-tien- https://leetcode.com/aroblems/longest-peatine-tien- https://leetcode.com/aroblems/longest-peatine-tien- https://leetcode.com/aroblems/mainter-mis-ubstrings/ https://leetcode.com/aroblems/mainter-mis- https://leetcode.com/aroblems/mainter-free/ https://leetcode.com/aroblems/mainter-free/ https://leetcode.com/aroblems/longer-tien-mainter- https://leetcode.com/aroblems/bana-tree-leet-order-tree/ https://leetcode.com/aroblems/bana-tree-leet-order-tree/ https://leetcode.com/aroblems/sort-tiel-mainter-ser-leet-order-tree/ https://leetcode.com/aroblems/sort-tuch-lina-y-tee-leet-order-tree/ https://leetcode.com/aroblems/sort-tuch-lina-y-tee-leet-in- https://leetcode.com/aroblems/sort-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor- https://leetcode.com/aroblems/longest-common-ancestor-	sliding window, if we see same char brice within curr window, shift start position; 90% ATENTION: limited to chars A-2; 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; need is num of unique char in T, 140KE 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, hashmap to count each char in str1, decrement for str2; for each of 26 chars, use count of each or in each word as tuple for key in dict, value is the list of anagrams; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, used count of each char in each word as taphanum, compane left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach char in str, consider if were the middle, consider if pail was odd or even; same as longest pullationnic string, each char in rist as middle and expand outwards, do same for pall of even len; maybe read up on manachers alg store length of str before each string and delimiter like '#'; recursive dis to inoth trees at the same time, iterative bits concumnt number of levels in tree recursive dis to inoth trees at the same time, iterative bits compane each level of both trees recursive dis to initer substrees, bits on invert levels, use collections, deque, leterative dis is easy with stack if doing pre-order traversal helper returns managhstrum without splitting branches, include helper we also update masSum by computing managhstrum WITH1 a split; iterative bits, add prevel level which doesn't have any nulls to the result; bits every single non-null node is added to string, and it's children are leaft southere, right of not are right subtree; stock is use built in python min/max values float** in in-order are left subtree, right of not are	
https://youtu.be/Sto0004.hbM https://youtu.be/St01008.ncgA https://youtu.be/wh/NC2682E https://youtu.be/wh/NC2682E https://youtu.be/Wh/Ts180FMg https://youtu.be/Wh/Ts180FMg https://youtu.be/RX1686FMg https://youtu.be/RX1686FMg https://youtu.be/RX1686FMg https://youtu.be/E1R ss05g/g https://youtu.be/E1R ss05g/g https://youtu.be/E1R ss05g/g https://youtu.be/E1R ss05g/g https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMgAUSHB https://youtu.be/StMGAUSHB https://youtu.be/StMGAUSHB https://youtu.be/StMGAUSHB https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO https://youtu.be/sobsoc/ilHAO	String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Repeating Character Replacement Minimum Window Substring Valid Anagram Group Anagrams Valid Parentheses Valid Palindromic Longest Palindromic Substring Palindromic Substrings Paccode and Decode Strings (Leetcode Premium) Maximum Degth of Binary Tree Same Tree Invert/Flip Binary Tree Binary Tree Level Order Traversal Serialize and Desertalize Binary Tree Substree of Another Tree Construct Binary Tree from Preorder and Inorder Validate Binary Search Tree Kih Smallest Element in a BST Lowest Common Ancestor of BST Implement Trie (Verbit Tree) Add and Search Word Word Search III	https://leetcode.com/aroblems/longest-substring-without- https://leetcode.com/aroblems/longest-repeating-charact- https://leetcode.com/aroblems/gminimum-window-substri- https://leetcode.com/aroblems/gminimum-window-substri- https://leetcode.com/aroblems/gmilc-parentheses/ https://leetcode.com/aroblems/gmilc-parentheses/ https://leetcode.com/aroblems/longest-paindur/pmic-subst- https://leetcode.com/aroblems/longest-paindur/pmic-subst- https://leetcode.com/aroblems/longest-paindur/pmic-subst- https://leetcode.com/aroblems/longest-paindur/pmic-substrings/ https://leetcode.com/aroblems/longest-paindur/pmic-substrings/ https://leetcode.com/aroblems/maintum-depth-of-binary https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain- https://leetcode.com/aroblems/longe-tree-maintum-pain-pain-pain-pain-pain-pain-pain-pain	sliding window, if we see same char brice within curr window, shift start position; #W ATTENTON: limited to chars A-2; 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; meed 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; hashmap to count each char in rist, decrement for str2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, used count of each unique char is satisfied; ## In the count of each unique char is satis	so don't add duplicates;
https://youtu.be/Stx00A4.bM https://youtu.be/Stx00A2.bM https://youtu.be/vdNOR2082E https://youtu.be/vdNOR2082E https://youtu.be/vdNOR2082E https://youtu.be/vdNOR2082E https://youtu.be/vdNOR2082E https://youtu.be/vdNOR2082E https://youtu.be/VM218180PMg https://youtu.be/SM2C82GMg https://youtu.be/SM2C82GMg https://youtu.be/SM2C82GMg https://youtu.be/SM2C82GMg https://youtu.be/sM2C82GMg https://youtu.be/sM2C82GMg https://youtu.be/sM2C82GMg https://youtu.be/SM2C82GMg https://youtu.be/SM2C8GMg https://youtu.be/SM2C8GMg https://youtu.be/SM2C8GMg https://youtu.be/SM2C8GMg https://youtu.be/SM2C8GMg https://youtu.be/SM2C8GMg https://youtu.be/SM2C8GMg https://youtu.be/sM2CMG htt	String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Bepeating Character Replacement Minimum Window Substring Valid Anagrams Valid Parentheses Valid Palindromic Substring Palindromic Substring Palindromic Substring Palindromic Substring Palindromic Substrings Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traversal Serfalize and Describating Binary Tree Subtree of Another Tree Construct Binary Tree From Preorder and Inorder Validate Binary Search Tree Validate Maximum Ancestor of 85T Implement Trie (Prefix Tree) Add and Search Word	https://lectcode.com/aroblems/longest-substrine.without- https://lectcode.com/aroblems/longest-repeatine-function- https://lectcode.com/aroblems/longest-repeatine-function- https://lectcode.com/aroblems/wininum-window-substri- ntps://lectcode.com/aroblems/longest-patients-repeating- https://lectcode.com/aroblems/longest-patients-suf- https://lectcode.com/aroblems/longest-patients-suf- https://lectcode.com/aroblems/longest-patients-patients- https://lectcode.com/aroblems/longest-patients-patients- https://lectcode.com/aroblems/longest-patients-patients- https://lectcode.com/aroblems/longest-patients-patients- https://lectcode.com/aroblems/longest-patients-patients-patients- https://lectcode.com/aroblems/longest-patients-pat	slding window, if we see same char brice within curr window, shift start position; WA TENTION: limited to chars A-2; 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; need is num of orinque chair in T, 140/Es num of chair we have valid count for, slding 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; hashmap to count each chair in str), decrement for str2; for each of 26 chairs, use count of each chair in each word as tuple for key in dict, value is the list of anagrams; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, used self end an eight until each points at alphanum, compane left and right, continue until left >= right, don't distinguish between upper/lowercase; foreach thair in str, consider it were the middle, consider if pail was odd or even; same as longest pailindromic string, each chair in str as middle and expand outwands, do same for pall of even len; maybe read up on manachers alg store length of str before each string and delimiter like 'it'; excursive dis to infort subtrees; the state bits compane each level of both trees recursive dis to infort subtrees; the state bits compane each level of both trees recursive dis to infort subtrees; that subtrees, the state bits compane each level of both trees recursive dis to infort subtrees; that subtrees, subtrees the subtrees, results be fix count number of events in tree recursive dis to infort subtrees; that subtrees, subtrees, subtrees, subtrees, subtrees, subtrees, subtrees, results be fixed until subtrees; the subtrees at the same time, tenative bits count number of events in tree length of subtrees; the subtrees, subtrees, results be fixed on the subtrees, recursive by the subtrees; this is even single non-null node is added to str	so don't add duplicates;
https://youtu.be/Stx00A4.bM https://youtu.be/Stx00A4.bM https://youtu.be/Stx00A2082 https://youtu.be/wtX00C2082 https://youtu.be/wtX00C2082 https://youtu.be/WT3T40PMg https://youtu.be/WT3T40PMg https://youtu.be/WT3T40PMg https://youtu.be/RXCG50465. https://youtu.be/RXCG504665. https://youtu.be/RXCG505Wg https://youtu.be/RXCG505Wg https://youtu.be/Stx00Wg https://youtu.	String Tree Tree Tree Tree Tree Tree Tree Tre	Longest Bepeating Character Replacement Minimum Window Substring Valid Anagrams Group Anagrams Valid Parentheses Valid Palindrome Longest Palindrome Substring Palindrome Substring Palindrome Substrings Encode and Decode Strings (Leetcode Premium) Maximum Depth of Binary Tree Same Tree Invert/Filip Binary Tree Binary Tree Maximum Path Sum Binary Tree Level Order Traversal Serialize and Deserialize Binary Tree Subtree of Another Tree Construct Binary Tree from Preorder and Inorder Validate Binary Search Tree Validate Binary Search Tree Validate Common Ancestor of BST Implement Tile (Prefix Tree) Add and Search Word Word Search III Merge K Sorted Lists	https://lectode.com/aroblems/ongest-substrine.without https://lectode.com/aroblems/ongest-spealine.icharact https://lectode.com/aroblems/minimum-window-substri https://lectode.com/aroblems/minimum-window-substri https://lectode.com/aroblems/walid-anagram/ https://lectode.com/aroblems/walid-painernheses/ https://lectode.com/aroblems/walid-painernheses/ https://lectode.com/aroblems/milid-painernheses/ https://lectode.com/aroblems/miniernheses/ https://lectode.com/aroblems/miniernheses/ https://lectode.com/aroblems/miniernheses/ https://lectode.com/aroblems/miniernheses/ https://lectode.com/aroblems/miniernheses/ https://lectode.com/aroblems/miniernheses/ https://lectode.com/aroblems/miniertheses/ https://lectode.com/aroblems/siniertheses/ https://lectode.com/aroblems/siniertheses/ https://lectode.com/aroblems/sinier-tree-miniertheses/ https://lectode.com/aroblems/sinier-tree-level-order-tree- https://lectode.com/aroblems/sinier-and-deserialuse-hittps://lectode.com/aroblems/sinier-tree-level-order-tree- https://lectode.com/aroblems/sinier-hitms-rise-refus-tree- https://lectode.com/aroblems/sinier-hitms-rise-refus-tree- https://lectode.com/aroblems/minier-hitms-rise-refus-tree- https://lectode.com/aroblems/minier-hitms-rise-refus-tree- https://lectode.com/aroblems/docst-acom/aroblems/ https://lectode.com/aroblems/docst-acom/aroblems/ https://lectode.com/aroblems/docst-acom/aroblems/ https://lectode.com/aroblems/docst-acom/aroblems/ https://lectode.com/aroblems/docst-acom/aroblems/ https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto- https://lectode.com/aroblems/box-formon-anaesto-	sliding window, if we see same char brice within curr window, shift start position; #W ATTENTON: limited to chars A-2; 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; meed 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; hashmap to count each char in rist, decrement for str2; 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; push opening brace on stack, pop if matching close brace, at end if stack empty, return true; left, right pointers, used count of each unique char is satisfied; ## In the count of each unique char is satis	so don't add duplicates;