Transformer AI

The Leet-Code Engineer

A Book

Brian Abbott, October 2024

Table of Contents

[1 Overview 27](#_Toc181146359)

[1.1 Leet-Code Topic Areas 27](#_Toc181146360)

[1.2 Leet-Code Problem Sets 29](#_Toc181146361)

[2 1, Two-Sum 35](#_Toc181146362)

[2.1 Description 35](#_Toc181146363)

[2.2 Test Examples and Constraints 35](#_Toc181146364)

[2.3 Solution 36](#_Toc181146365)

[3 3, Longest Substring without Repeating Chars 36](#_Toc181146366)

[3.1 Description 36](#_Toc181146367)

[3.2 Test Examples and Constraints 37](#_Toc181146368)

[3.3 Solution 38](#_Toc181146369)

[3.4 Analysis 38](#_Toc181146370)

[4 8, String to Integer 38](#_Toc181146371)

[4.1 Description 38](#_Toc181146372)

[4.2 Test Examples and Constraints 39](#_Toc181146373)

[4.3 Solution 40](#_Toc181146374)

[5 11, Container with the Most Water 41](#_Toc181146375)

[5.1 Description 41](#_Toc181146376)

[5.2 Test Examples and Constraints 41](#_Toc181146377)

[5.3 Solution 42](#_Toc181146378)

[6 12, Integer to Roman 43](#_Toc181146379)

[6.1 Description 43](#_Toc181146380)

[6.2 Test Examples and Constraints 44](#_Toc181146381)

[6.3 Solution 45](#_Toc181146382)

[7 13, Roman to Integer 46](#_Toc181146383)

[7.1 Description 46](#_Toc181146384)

[7.2 Test Examples and Constraints 47](#_Toc181146385)

[7.3 Solution 47](#_Toc181146386)

[8 15, 3-Sum 48](#_Toc181146387)

[8.1 Description 48](#_Toc181146388)

[8.2 Test Examples and Constraints 48](#_Toc181146389)

[8.3 Solution 49](#_Toc181146390)

[9 16, 3-Sum Closest 49](#_Toc181146391)

[9.1 Description 49](#_Toc181146392)

[9.2 Test Examples and Constraints 49](#_Toc181146393)

[9.3 Solution 50](#_Toc181146394)

[10 , Implement StrStr 50](#_Toc181146395)

[10.1 Description 50](#_Toc181146396)

[10.2 Test Examples and Constraints 50](#_Toc181146397)

[10.3 Solution 50](#_Toc181146398)

[11 48, Rotate Image 51](#_Toc181146399)

[11.1 Description 51](#_Toc181146400)

[11.2 Test Examples and Constraints 52](#_Toc181146401)

[11.3 Solution 53](#_Toc181146402)

[12 49, Group Anagrams 53](#_Toc181146403)

[12.1 Description 53](#_Toc181146404)

[12.2 Test Examples and Constraints 53](#_Toc181146405)

[12.3 Solution 53](#_Toc181146406)

[13 76, Minimum Window Substring 54](#_Toc181146407)

[13.1 Description 54](#_Toc181146408)

[13.2 Test Examples and Constraints 54](#_Toc181146409)

[13.3 Solution 54](#_Toc181146410)

[14 165, Compare Version Numbers 55](#_Toc181146411)

[14.1 Description 55](#_Toc181146412)

[14.2 Test Examples and Constraints 55](#_Toc181146413)

[14.3 Solution 55](#_Toc181146414)

[15 238, Product of Array Except Self 56](#_Toc181146415)

[15.1 Description 56](#_Toc181146416)

[15.2 Test Examples and Constraints 56](#_Toc181146417)

[15.3 Solution 56](#_Toc181146418)

[16 268, Missing Number 56](#_Toc181146419)

[16.1 Description 56](#_Toc181146420)

[16.2 Test Examples and Constraints 57](#_Toc181146421)

[16.3 Solution 57](#_Toc181146422)

[17 273, Integer to English Words 57](#_Toc181146423)

[17.1 Description 57](#_Toc181146424)

[17.2 Test Examples and Constraints 57](#_Toc181146425)

[17.3 Solution 58](#_Toc181146426)

[18 387, First Unique Character in a String 58](#_Toc181146427)

[18.1 Description 58](#_Toc181146428)

[18.2 Test Examples and Constraints 58](#_Toc181146429)

[18.3 Solution 58](#_Toc181146430)

[19 678, Valid Parenthesis 59](#_Toc181146431)

[19.1 Description 59](#_Toc181146432)

[19.2 Test Examples and Constraints 59](#_Toc181146433)

[19.3 Solution 59](#_Toc181146434)

from typing import List

[20 819, Most Common Word 60](#_Toc181146435)

[20.1 Description 60](#_Toc181146436)

[20.2 Test Examples and Constraints 60](#_Toc181146437)

[20.3 Solution 60](#_Toc181146438)

[21 937, Re-Order Log Files 61](#_Toc181146439)

[21.1 Description 61](#_Toc181146440)

[21.2 Test Examples and Constraints 61](#_Toc181146441)

[21.3 Solution 61](#_Toc181146442)

from typing import List

[22 42, Trapping Rain Water 62](#_Toc181146443)

[22.1 Description 62](#_Toc181146444)

[22.2 Test Examples and Constraints 62](#_Toc181146445)

[22.3 Solution 62](#_Toc181146446)

[23 2, Add Two Numbers 62](#_Toc181146447)

[23.1 Description 62](#_Toc181146448)

[23.2 Test Examples and Constraints 63](#_Toc181146449)

[23.3 Solution 63](#_Toc181146450)

[24 21, Merge Two Sorted Lists 63](#_Toc181146451)

[24.1 Description 63](#_Toc181146452)

[24.2 Test Examples and Constraints 63](#_Toc181146453)

[24.3 Solution 64](#_Toc181146454)

[25 25, Reverse Nodes in K-Group 64](#_Toc181146455)

[25.1 Description 64](#_Toc181146456)

[25.2 Test Examples and Constraints 64](#_Toc181146457)

[25.3 Solution 64](#_Toc181146458)

[26 138, Copy List with Random Pointer 65](#_Toc181146459)

[26.1 Description 65](#_Toc181146460)

[26.2 Test Examples and Constraints 65](#_Toc181146461)

[26.3 Solution 66](#_Toc181146462)

[27 206, Reverse Linked Lists 66](#_Toc181146463)

[27.1 Description 66](#_Toc181146464)

[27.2 Test Examples and Constraints 66](#_Toc181146465)

[27.3 Solution 66](#_Toc181146466)

[28 23, Merge K-Sorted Lists 67](#_Toc181146467)

[28.1 Description 67](#_Toc181146468)

[28.2 Test Examples and Constraints 67](#_Toc181146469)

[28.3 Solution 67](#_Toc181146470)

[29 98, Validate Binary Search Tree 68](#_Toc181146471)

[29.1 Description 68](#_Toc181146472)

[29.2 Test Examples and Constraints 68](#_Toc181146473)

[29.3 Solution 68](#_Toc181146474)

[30 101, Symmetric Tree 68](#_Toc181146475)

[30.1 Description 68](#_Toc181146476)

[30.2 Test Examples and Constraints 69](#_Toc181146477)

[30.3 Solution 69](#_Toc181146478)

[31 102, Binary Tree, Level Order Traversal 69](#_Toc181146479)

[31.1 Description 69](#_Toc181146480)

[31.2 Test Examples and Constraints 69](#_Toc181146481)

[31.3 Solution 70](#_Toc181146482)

[32 103, Binary Tree, ZigZag Order Traversal 70](#_Toc181146483)

[32.1 Description 70](#_Toc181146484)

[32.2 Test Examples and Constraints 70](#_Toc181146485)

[32.3 Solution 70](#_Toc181146486)

[33 124, Binary Tree, Maximum Path Sum 71](#_Toc181146487)

[33.1 Description 71](#_Toc181146488)

[33.2 Test Examples and Constraints 71](#_Toc181146489)

[33.3 Solution 71](#_Toc181146490)

[34 126, Word Ladder II 72](#_Toc181146491)

[34.1 Description 72](#_Toc181146492)

[34.2 Test Examples and Constraints 72](#_Toc181146493)

[34.3 Solution 72](#_Toc181146494)

[35 127, Word Ladder 73](#_Toc181146495)

[35.1 Description 73](#_Toc181146496)

[35.2 Test Examples and Constraints 73](#_Toc181146497)

[35.3 Solution 73](#_Toc181146498)

[36 200, Number of Islands 74](#_Toc181146499)

[36.1 Description 74](#_Toc181146500)

[36.2 Test Examples and Constraints 74](#_Toc181146501)

[36.3 Solution 74](#_Toc181146502)

[37 207, Course Schedule 75](#_Toc181146503)

[37.1 Description 75](#_Toc181146504)

[37.2 Test Examples and Constraints 75](#_Toc181146505)

[37.3 Solution 75](#_Toc181146506)

[38 236, Lowest Common Ancestor of Binary Tree 75](#_Toc181146507)

[38.1 Description 75](#_Toc181146508)

[38.2 Test Examples and Constraints 76](#_Toc181146509)

[38.3 Solution 76](#_Toc181146510)

[39 543, Diameter of Binary Tree 76](#_Toc181146511)

[39.1 Description 76](#_Toc181146512)

[39.2 Test Examples and Constraints 76](#_Toc181146513)

[39.3 Solution 77](#_Toc181146514)

[40 675, Cut-Off Trees for Golf Event 77](#_Toc181146515)

[40.1 Description 77](#_Toc181146516)

[40.2 Test Examples and Constraints 78](#_Toc181146517)

[40.3 Solution 78](#_Toc181146518)

[41 733, Flood Fill 78](#_Toc181146519)

[41.1 Description 78](#_Toc181146520)

[41.2 Test Examples and Constraints 79](#_Toc181146521)

[41.3 Solution 79](#_Toc181146522)

[42 17, Letter Combinations of a Phone Number 79](#_Toc181146523)

[42.1 Description 80](#_Toc181146524)

[42.2 Test Examples and Constraints 80](#_Toc181146525)

[42.3 Solution 80](#_Toc181146526)

[43 22, Generate Parenthesis 81](#_Toc181146527)

[43.1 Description 81](#_Toc181146528)

[43.2 Test Examples and Constraints 81](#_Toc181146529)

[43.3 Solution 81](#_Toc181146530)

[44 79, Word Search 81](#_Toc181146531)

[44.1 Description 81](#_Toc181146532)

[44.2 Test Examples and Constraints 81](#_Toc181146533)

[44.3 Solution 82](#_Toc181146534)

[45 212, Word Search II 82](#_Toc181146535)

[45.1 Description 82](#_Toc181146536)

[45.2 Test Examples and Constraints 82](#_Toc181146537)

[45.3 Solution 82](#_Toc181146538)

[46 4, Median of Two Sorted Arrays 83](#_Toc181146539)

[46.1 Description 83](#_Toc181146540)

[46.2 Test Examples and Constraints 83](#_Toc181146541)

[46.3 Solution 83](#_Toc181146542)

[47 33, Search in Rotated Sorted Array 83](#_Toc181146543)

[47.1 Description 83](#_Toc181146544)

[47.2 Test Examples and Constraints 83](#_Toc181146545)

[47.3 Solution 84](#_Toc181146546)

[48 56, Merge Intervals 84](#_Toc181146547)

[48.1 Description 84](#_Toc181146548)

[48.2 Test Examples and Constraints 84](#_Toc181146549)

[48.3 Solution 85](#_Toc181146550)

[49 167, Two Sum II, input array is sorted 85](#_Toc181146551)

[49.1 Description 85](#_Toc181146552)

[49.2 Test Examples and Constraints 85](#_Toc181146553)

[49.3 Solution 85](#_Toc181146554)

[50 215, Kth largest element in an array 86](#_Toc181146555)

[50.1 Description 86](#_Toc181146556)

[50.2 Test Examples and Constraints 86](#_Toc181146557)

[50.3 Solution 86](#_Toc181146558)

[51 253, Meeting Rooms II 86](#_Toc181146559)

[51.1 Description 86](#_Toc181146560)

[51.2 Test Examples and Constraints 86](#_Toc181146561)

[51.3 Solution 87](#_Toc181146562)

[52 347, Top K Frequent Elements 87](#_Toc181146563)

[52.1 Description 87](#_Toc181146564)

[52.2 Test Examples and Constraints 87](#_Toc181146565)

[52.3 Solution 87](#_Toc181146566)

[53 973, K closest points to origin 88](#_Toc181146567)

[53.1 Description 88](#_Toc181146568)

[53.2 Test Examples and Constraints 88](#_Toc181146569)

[53.3 Solution 88](#_Toc181146570)

[54 5, Longest Palindrome Substring 88](#_Toc181146571)

[54.1 Description 88](#_Toc181146572)

[54.2 Test Examples and Constraints 88](#_Toc181146573)

[54.3 Solution 89](#_Toc181146574)

[55 53, Maximum Subarray 89](#_Toc181146575)

[55.1 Description 89](#_Toc181146576)

[55.2 Test Examples and Constraints 89](#_Toc181146577)

[55.3 Solution 89](#_Toc181146578)

[56 121, Best Time to Buy and Sell a Stock 90](#_Toc181146579)

[56.1 Description 90](#_Toc181146580)

[56.2 Test Examples and Constraints 90](#_Toc181146581)

[56.3 Solution 90](#_Toc181146582)

[57 139, Word Break 90](#_Toc181146583)

[57.1 Description 90](#_Toc181146584)

[57.2 Test Examples and Constraints 90](#_Toc181146585)

[57.3 Solution 91](#_Toc181146586)

[58 322, Coin Change 91](#_Toc181146587)

[58.1 Description 91](#_Toc181146588)

[58.2 Test Examples and Constraints 91](#_Toc181146589)

[58.3 Solution 91](#_Toc181146590)

[59 146, LRU Cache 92](#_Toc181146591)

[59.1 Description 92](#_Toc181146592)

[59.2 Test Examples and Constraints 92](#_Toc181146593)

[59.3 Solution 92](#_Toc181146594)

[60 155, Min Stack 92](#_Toc181146595)

[60.1 Description 92](#_Toc181146596)

[60.2 Test Examples and Constraints 92](#_Toc181146597)

[60.3 Solution 93](#_Toc181146598)

[61 295, Find Median from Data Stream 93](#_Toc181146599)

[61.1 Description 93](#_Toc181146600)

[61.2 Test Examples and Constraints 93](#_Toc181146601)

[61.3 Solution 93](#_Toc181146602)

[62 297, Serialize and Deserialize a Binary Tree 94](#_Toc181146603)

[62.1 Description 94](#_Toc181146604)

[62.2 Test Examples and Constraints 94](#_Toc181146605)

[62.3 Solution 94](#_Toc181146606)

[63 348, Design Tic-Tac-Toe 94](#_Toc181146607)

[63.1 Description 94](#_Toc181146608)

[63.2 Test Examples and Constraints 94](#_Toc181146609)

[63.3 Solution 95](#_Toc181146610)

[64 642, Design Search Auto-Complete System 95](#_Toc181146611)

[64.1 Description 95](#_Toc181146612)

[64.2 Test Examples and Constraints 95](#_Toc181146613)

[64.3 Solution 95](#_Toc181146614)

[65 895, Maximum Frequency Stack 96](#_Toc181146615)

[65.1 Description 96](#_Toc181146616)

[65.2 Test Examples and Constraints 96](#_Toc181146617)

[65.3 Solution 96](#_Toc181146618)

[66 7, Reverse Integer 96](#_Toc181146619)

[66.1 Description 96](#_Toc181146620)

[66.2 Test Examples and Constraints 96](#_Toc181146621)

[66.3 Solution 97](#_Toc181146622)

[67 176, Second Highest Salary 97](#_Toc181146623)

[67.1 Description 97](#_Toc181146624)

[67.2 Test Examples and Constraints 97](#_Toc181146625)

[67.3 Solution 97](#_Toc181146626)

[68 763, Partition Labels 98](#_Toc181146627)

[68.1 Description 98](#_Toc181146628)

[68.2 Test Examples and Constraints 98](#_Toc181146629)

[68.3 Solution 98](#_Toc181146630)

[69 957, Prison Cells after N Days 98](#_Toc181146631)

[69.1 Description 98](#_Toc181146632)

[69.2 Test Examples and Constraints 98](#_Toc181146633)

[69.3 Solution 99](#_Toc181146634)

[70 1140, Stone Game II 100](#_Toc181146635)

[70.1 Description 100](#_Toc181146636)

[70.2 Test Examples and Constraints 100](#_Toc181146637)

[70.3 Solution 100](#_Toc181146638)

[71 54, Spiral Matrix 100](#_Toc181146639)

[71.1 Description 100](#_Toc181146640)

[71.2 Test Examples and Constraints 101](#_Toc181146641)

[71.3 Solution 101](#_Toc181146642)

[72 542, 01 Matrix 101](#_Toc181146643)

[72.1 Description 101](#_Toc181146644)

[72.2 Test Examples and Constraints 101](#_Toc181146645)

[72.3 Solution 102](#_Toc181146646)

[73 766, Toeplitz Matrix 102](#_Toc181146647)

[73.1 Description 102](#_Toc181146648)

[73.2 Test Examples and Constraints 102](#_Toc181146649)

[73.3 Solution 102](#_Toc181146650)

[74 867, Transpose Matrix 103](#_Toc181146651)

[74.1 Description 103](#_Toc181146652)

[74.2 Test Examples and Constraints 103](#_Toc181146653)

[74.3 Solution 103](#_Toc181146654)

[75 59, Spiral Matrix II 103](#_Toc181146655)

[75.1 Description 103](#_Toc181146656)

[75.2 Test Examples and Constraints 103](#_Toc181146657)

[75.3 Solution 104](#_Toc181146658)

[76 73, Set Matrix Zeroes 104](#_Toc181146659)

[76.1 Description 104](#_Toc181146660)

[76.2 Test Examples and Constraints 104](#_Toc181146661)

[76.3 Solution 104](#_Toc181146662)

[77 311, Sparse Matrix Multiplication 104](#_Toc181146663)

[77.1 Description 104](#_Toc181146664)

[77.2 Test Examples and Constraints 105](#_Toc181146665)

[77.3 Solution 105](#_Toc181146666)

[78 566, Reshape the Matrix 105](#_Toc181146667)

[78.1 Description 105](#_Toc181146668)

[78.2 Test Examples and Constraints 105](#_Toc181146669)

[78.3 Solution 106](#_Toc181146670)

[79 756, Pyramid Transition Matrix 106](#_Toc181146671)

[79.1 Description 106](#_Toc181146672)

[79.2 Test Examples and Constraints 106](#_Toc181146673)

[79.3 Solution 106](#_Toc181146674)

[80 519, Random Flip Matrix 107](#_Toc181146675)

[80.1 Description 107](#_Toc181146676)

[80.2 Test Examples and Constraints 107](#_Toc181146677)

[80.3 Solution 107](#_Toc181146678)

[81 885, Spiral Matrix III 107](#_Toc181146679)

[81.1 Description 107](#_Toc181146680)

[81.2 Test Examples and Constraints 107](#_Toc181146681)

[81.3 Solution 108](#_Toc181146682)

[82 1314, Matrix Block Sum 108](#_Toc181146683)

[82.1 Description 108](#_Toc181146684)

[82.2 Test Examples and Constraints 108](#_Toc181146685)

[82.3 Solution 108](#_Toc181146686)

[83 1572, Matrix Diagonal Sum 109](#_Toc181146687)

[83.1 Description 109](#_Toc181146688)

[83.2 Test Examples and Constraints 109](#_Toc181146689)

[83.3 Solution 109](#_Toc181146690)

[84 1975, Maximum Matrix Sum 109](#_Toc181146691)

[84.1 Description 109](#_Toc181146692)

[84.2 Test Examples and Constraints 109](#_Toc181146693)

[84.3 Solution 110](#_Toc181146694)

[85 2326, Spiral Matrix IV 110](#_Toc181146695)

[85.1 Description 110](#_Toc181146696)

[85.2 Test Examples and Constraints 110](#_Toc181146697)

[85.3 Solution 110](#_Toc181146698)

[86 2906, Construct Product Matrix 111](#_Toc181146699)

[86.1 Description 111](#_Toc181146700)

[86.2 Test Examples and Constraints 111](#_Toc181146701)

[86.3 Solution 111](#_Toc181146702)

[87 3033, Modify the Matrix 111](#_Toc181146703)

[87.1 Description 111](#_Toc181146704)

[87.2 Test Examples and Constraints 111](#_Toc181146705)

[87.3 Solution 112](#_Toc181146706)

[88 3248, Snake in Matrix 112](#_Toc181146707)

[88.1 Description 112](#_Toc181146708)

[88.2 Test Examples and Constraints 112](#_Toc181146709)

[88.3 Solution 112](#_Toc181146710)

[89 74, Search a 2D Matrix 113](#_Toc181146711)

[89.1 Description 113](#_Toc181146712)

[89.2 Test Examples and Constraints 113](#_Toc181146713)

[89.3 Solution 113](#_Toc181146714)

[90 861, Score after Flipping Matrix 113](#_Toc181146715)

[90.1 Description 113](#_Toc181146716)

[90.2 Test Examples and Constraints 113](#_Toc181146717)

[90.3 Solution 114](#_Toc181146718)

[91 1329, Sort the Matrix Diagonally 114](#_Toc181146719)

[91.1 Description 114](#_Toc181146720)

[91.2 Test Examples and Constraints 114](#_Toc181146721)

[91.3 Solution 114](#_Toc181146722)

[92 2679, Sum in a Matrix 115](#_Toc181146723)

[92.1 Description 115](#_Toc181146724)

[92.2 Test Examples and Constraints 115](#_Toc181146725)

[92.3 Solution 115](#_Toc181146726)

[93 240, Search a 2D Matrix II 115](#_Toc181146727)

[93.1 Description 115](#_Toc181146728)

[93.2 Test Examples and Constraints 115](#_Toc181146729)

[93.3 Solution 116](#_Toc181146730)

[94 1030, Matrix Cells in Distance Order 116](#_Toc181146731)

[94.1 Description 116](#_Toc181146732)

[94.2 Test Examples and Constraints 116](#_Toc181146733)

[94.3 Solution 116](#_Toc181146734)

[95 1091, Shortest Path in Binary Matrix 117](#_Toc181146735)

[95.1 Description 117](#_Toc181146736)

[95.2 Test Examples and Constraints 117](#_Toc181146737)

[95.3 Solution 117](#_Toc181146738)

from typing import List

[96 1632, Rank transform of a Matrix 117](#_Toc181146739)

[96.1 Description 117](#_Toc181146740)

[96.2 Test Examples and Constraints 117](#_Toc181146741)

[96.3 Solution 118](#_Toc181146742)

[97 1380, Lucky Numbers in a Matrix 118](#_Toc181146743)

[97.1 Description 118](#_Toc181146744)

[97.2 Test Examples and Constraints 118](#_Toc181146745)

[97.3 Solution 118](#_Toc181146746)

[98 2319, Check if Matrix is X-Matrix 119](#_Toc181146747)

[98.1 Description 119](#_Toc181146748)

[98.2 Test Examples and Constraints 119](#_Toc181146749)

[98.3 Solution 119](#_Toc181146750)

[99 2392, Build a Matrix with Conditions 119](#_Toc181146751)

[99.1 Description 119](#_Toc181146752)

[99.2 Test Examples and Constraints 119](#_Toc181146753)

[99.3 Solution 120](#_Toc181146754)

[100 2675, Array of Objects to Matrix 120](#_Toc181146755)

[100.1 Description 120](#_Toc181146756)

[100.2 Test Examples and Constraints 120](#_Toc181146757)

[100.3 Solution 120](#_Toc181146758)

[101 2718, Sum of Matrix after Queries 121](#_Toc181146759)

[101.1 Description 121](#_Toc181146760)

[101.2 Test Examples and Constraints 121](#_Toc181146761)

[101.3 Solution 121](#_Toc181146762)

[102 2946, Matrix Similarity after Cycle Shifts 121](#_Toc181146763)

[102.1 Description 121](#_Toc181146764)

[102.2 Test Examples and Constraints 121](#_Toc181146765)

[102.3 Solution 122](#_Toc181146766)

[103 3078, Match Alpha-Numeric Pattern in Matrix I 122](#_Toc181146767)

[103.1 Description 122](#_Toc181146768)

[103.2 Test Examples and Constraints 122](#_Toc181146769)

[103.3 Solution 122](#_Toc181146770)

[104 329, Longest increasing path in a Matrix 123](#_Toc181146771)

[104.1 Description 123](#_Toc181146772)

[104.2 Test Examples and Constraints 123](#_Toc181146773)

[104.3 Solution 123](#_Toc181146774)

[105 1253, Reconstruct a 2-row binary matrix 123](#_Toc181146775)

[105.1 Description 123](#_Toc181146776)

[105.2 Test Examples and Constraints 123](#_Toc181146777)

[105.3 Solution 124](#_Toc181146778)

[106 1582, Special positions in a binary matrix 124](#_Toc181146779)

[106.1 Description 124](#_Toc181146780)

[106.2 Test Examples and Constraints 124](#_Toc181146781)

[106.3 Solution 124](#_Toc181146782)

[107 2373, Largest local values in a Matrix 125](#_Toc181146783)

[107.1 Description 125](#_Toc181146784)

[107.2 Test Examples and Constraints 125](#_Toc181146785)

[107.3 Solution 125](#_Toc181146786)

[108 378, Kth Smallest element in a sorted Matrix 125](#_Toc181146787)

[108.1 Description 125](#_Toc181146788)

[108.2 Test Examples and Constraints 125](#_Toc181146789)

[108.3 Solution 126](#_Toc181146790)

[109 562, Longest Line of Consecutive one in Matrix 126](#_Toc181146791)

[109.1 Description 126](#_Toc181146792)

[109.2 Test Examples and Constraints 126](#_Toc181146793)

[109.3 Solution 126](#_Toc181146794)

[110 1252, Cells with odd values in a Matrix 127](#_Toc181146795)

[110.1 Description 127](#_Toc181146796)

[110.2 Test Examples and Constraints 127](#_Toc181146797)

[110.3 Solution 127](#_Toc181146798)

[111 1337, The K weakest rows in a Matrix 127](#_Toc181146799)

[111.1 Description 127](#_Toc181146800)

[111.2 Test Examples and Constraints 127](#_Toc181146801)

[111.3 Solution 128](#_Toc181146802)

[112 1351, Count Negative numbers in a Sorted Matrix 128](#_Toc181146803)

[112.1 Description 128](#_Toc181146804)

[112.2 Test Examples and Constraints 128](#_Toc181146805)

[112.3 Solution 128](#_Toc181146806)

[113 1594, Maximum non-negative product in a Matrix 129](#_Toc181146807)

[113.1 Description 129](#_Toc181146808)

[113.2 Test Examples and Constraints 129](#_Toc181146809)

[113.3 Solution 129](#_Toc181146810)

[114 2387, Median of a row-wise sorted Matrix 129](#_Toc181146811)

[114.1 Description 129](#_Toc181146812)

[114.2 Test Examples and Constraints 129](#_Toc181146813)

[114.3 Solution 130](#_Toc181146814)

[115 2713, Maximum Strictly Increasing cells in a Matrix 130](#_Toc181146815)

[115.1 Description 130](#_Toc181146816)

[115.2 Test Examples and Constraints 130](#_Toc181146817)

[115.3 Solution 130](#_Toc181146818)

[116 2732, Find a good subset of the Matrix 131](#_Toc181146819)

[116.1 Description 131](#_Toc181146820)

[116.2 Test Examples and Constraints 131](#_Toc181146821)

[116.3 Solution 131](#_Toc181146822)

[117 1605, Find valid Matrix given row and column sums 131](#_Toc181146823)

[117.1 Description 131](#_Toc181146824)

[117.2 Test Examples and Constraints 131](#_Toc181146825)

[117.3 Solution 132](#_Toc181146826)

[118 1886, Determine weather Matrix can be obtained by Rotation 132](#_Toc181146827)

[118.1 Description 132](#_Toc181146828)

[118.2 Test Examples and Constraints 132](#_Toc181146829)

[118.3 Solution 132](#_Toc181146830)

[119 2123, Minimum operations to remove Adjacent Ones in Matrix 133](#_Toc181146831)

[119.1 Description 133](#_Toc181146832)

[119.2 Test Examples and Constraints 133](#_Toc181146833)

[119.3 Solution 133](#_Toc181146834)

[120 1284, Minimum number of Flips to Convert Binary Matrix to Zero Matrix 133](#_Toc181146835)

[120.1 Description 133](#_Toc181146836)

[120.2 Test Examples and Constraints 133](#_Toc181146837)

[120.3 Solution 134](#_Toc181146838)

[121 2435, Paths in Matrix who’s sum is divisible by K 134](#_Toc181146839)

[121.1 Description 134](#_Toc181146840)

[121.2 Test Examples and Constraints 134](#_Toc181146841)

[121.3 Solution 134](#_Toc181146842)

[122 1439, Find the Kth smallest sum of a matrix with sorted rows 135](#_Toc181146843)

[122.1 Description 135](#_Toc181146844)

[122.2 Test Examples and Constraints 135](#_Toc181146845)

[122.3 Solution 135](#_Toc181146846)

[123 2556, Disconnect Path in a Binary Matrix by at most one flip 135](#_Toc181146847)

[123.1 Description 135](#_Toc181146848)

[123.2 Test Examples and Constraints 135](#_Toc181146849)

[123.3 Solution 136](#_Toc181146850)

[124 44, Wildcard Matching 136](#_Toc181146851)

[124.1 Description 136](#_Toc181146852)

[124.2 Test Examples and Constraints 136](#_Toc181146853)

[124.3 Solution 136](#_Toc181146854)

[125 1023, Camelcase Matching 137](#_Toc181146855)

[125.1 Description 137](#_Toc181146856)

[125.2 Test Examples and Constraints 137](#_Toc181146857)

[125.3 Solution 137](#_Toc181146858)

[126 10, Regular Expression Matching 137](#_Toc181146859)

[126.1 Description 137](#_Toc181146860)

[126.2 Test Examples and Constraints 138](#_Toc181146861)

[126.3 Solution 138](#_Toc181146862)

[127 473, Matchsticks to Square 138](#_Toc181146863)

[127.1 Description 138](#_Toc181146864)

[127.2 Test Examples and Constraints 138](#_Toc181146865)

[127.3 Solution 139](#_Toc181146866)

[128 544, Output Contest Matches 139](#_Toc181146867)

[128.1 Description 139](#_Toc181146868)

[128.2 Test Examples and Constraints 139](#_Toc181146869)

[128.3 Solution 139](#_Toc181146870)

[129 686, Repeated String Match 140](#_Toc181146871)

[129.1 Description 140](#_Toc181146872)

[129.2 Test Examples and Constraints 140](#_Toc181146873)

[129.3 Solution 140](#_Toc181146874)

[130 942, DI String Match 140](#_Toc181146875)

[130.1 Description 140](#_Toc181146876)

[130.2 Test Examples and Constraints 141](#_Toc181146877)

[130.3 Solution 141](#_Toc181146878)

[131 792, Number of Matching Sub-Sequences 141](#_Toc181146879)

[131.1 Description 141](#_Toc181146880)

[131.2 Test Examples and Constraints 141](#_Toc181146881)

[131.3 Solution 142](#_Toc181146882)

[132 2301, Match Substring after Replacement 142](#_Toc181146883)

[132.1 Description 142](#_Toc181146884)

[132.2 Test Examples and Constraints 142](#_Toc181146885)

[132.3 Solution 142](#_Toc181146886)

[133 1408, String Matching in an Array 143](#_Toc181146887)

[133.1 Description 143](#_Toc181146888)

[133.2 Test Examples and Constraints 143](#_Toc181146889)

[133.3 Solution 143](#_Toc181146890)

[134 1688, Count of Matches in a Tournament 143](#_Toc181146891)

[134.1 Description 143](#_Toc181146892)

[134.2 Test Examples and Constraints 143](#_Toc181146893)

[134.3 Solution 144](#_Toc181146894)

[135 1773, Count items matching a rule 144](#_Toc181146895)

[135.1 Description 144](#_Toc181146896)

[135.2 Test Examples and Constraints 144](#_Toc181146897)

[135.3 Solution 144](#_Toc181146898)

[136 2339, All the matches of the league 145](#_Toc181146899)

[136.1 Description 145](#_Toc181146900)

[136.2 Test Examples and Constraints 145](#_Toc181146901)

[136.3 Solution 145](#_Toc181146902)

[137 2410, Maximum matching of Players with Trainers 145](#_Toc181146903)

[137.1 Description 145](#_Toc181146904)

[137.2 Test Examples and Constraints 145](#_Toc181146905)

[137.3 Solution 146](#_Toc181146906)

[138 971, Flip binary tree to match pre-order traversal 146](#_Toc181146907)

[138.1 Description 146](#_Toc181146908)

[138.2 Test Examples and Constraints 146](#_Toc181146909)

[138.3 Solution 146](#_Toc181146910)

[139 2019, The Score of Students Solving Math Expression 147](#_Toc181146911)

[139.1 Description 147](#_Toc181146912)

[139.2 Test Examples and Constraints 147](#_Toc181146913)

[139.3 Solution 147](#_Toc181146914)

[140 3034, Numbers of Sub-arrays that match a pattern I 147](#_Toc181146915)

[140.1 Description 147](#_Toc181146916)

[140.2 Test Examples and Constraints 148](#_Toc181146917)

[140.3 Solution 148](#_Toc181146918)

[141 3036, Numbers of Sub-arrays that match a pattern II 148](#_Toc181146919)

[141.1 Description 148](#_Toc181146920)

[141.2 Test Examples and Constraints 148](#_Toc181146921)

[141.3 Solution 149](#_Toc181146922)

# Overview

Leet-Code has changed the coding landscape. The level of skill and quality of engineering delivered to organizations has revolutionized the Technology Industry Landscape. What the leet-code practiced engineer can render to any problem-domain, the speed at which they can do it and the quality of code that they deliver is the reason and motivation behind this book.

## Leet-Code Topic Areas

* Array
* String
* Hash Table
* Dynamic Programming
* Math
* Sorting
* Greedy
* Depth-First Search
* Database
* Binary Search
* Matrix
* Tree
* Breadth-First Search
* Bit Manipulation
* Two Pointers
* Heap (Priority Queue)
* Binary Tree
* Prefix Sum
* Simulation
* Stack
* Counting
* Graph
* Sliding Window
* Design
* Backtracking
* Enumeration
* Union Find
* Linked List
* Ordered Set
* Monotonic Stack
* Number Theory
* Trie
* Segment Tree
* Bitmask
* Recursion
* Divide and Conquer
* Queue
* Combinatorics
* Binary Search Tree
* Hash Function
* Binary Indexed Tree
* Geometry
* Memoization
* String Matching
* Topological Sort
* Rolling Hash
* Shortest Path
* Game Theory
* Interactive
* Data Stream
* Monotonic Queue
* Brainteaser
* Randomized
* Merge Sort
* Doubly-Linked List
* Counting Sort
* Iterator
* Concurrency
* Probability and Statistics
* Quickselect
* Suffix Array
* Bucket Sort
* Minimum Spanning Tree
* Shell
* Line Sweep
* Reservoir Sampling
* Strongly Connected Component
* Eulerian Circuit
* Radix Sort
* Rejection Sampling
* Biconnected Component

## Leet-Code Problem Sets

* Amazon Problems
  + 1, Two-Sum
  + 3, Longest Substring without Repeating Chars
  + 8, String to Integer
  + 11, Container with the Most Water
  + 12, Integer to Roman
  + 13, Roman to Integer
  + 15, 3-Sum
  + 16, 3-Sum Closest
  + , Implement StrStr
  + 48, Rotate Image
  + 49, Group Anagrams
  + 76, Minimum Window Substring
  + 165, Compare Version Numbers
  + 238, Product of Array Except Self
  + 268, Missing Number
  + 273, Integer to English Words
  + 387, First Unique Character in a String
  + 678, Valid Parenthesis
  + 819, Most Common Word
  + 937, Re-Order Log Files
  + 42, Trapping Rain Water
  + 2, Add Two Numbers
  + 21, Merge Two Sorted Lists
  + 25, Reverse Nodes in K-Group
  + 138, Copy List with Random Pointer
  + 206, Reverse Linked Lists
  + 23, Merge K-Sorted Lists
  + 98, Validate Binary Search Tree
  + 101, Symmetric Tree
  + 102, Binary Tree, Level Order Traversal
  + 103, Binary Tree, ZigZag Order Traversal
  + 124, Binary Tree, Maximum Path Sum
  + 126, Word Ladder II
  + 127, Word Ladder
  + 200, Number of Islands
  + 207, Course Schedule
  + 236, Lowest Common Ancestor of Binary Tree
  + 543, Diameter of Binary Tree
  + 675, Cut-Off Trees for Golf Event
  + 733, Flood Fill
  + 17, Letter Combinations of a Phone Number
  + 22, Generate Parenthesis
  + 79, Word Search
  + 212, Word Search II
  + 4, Median of Two Sorted Arrays
  + 33, Search in Rotated Sorted Array
  + 56, Merge Intervals
  + 167, Two Sum II, input array is sorted
  + 215, Kth largest element in an array
  + 253, Meeting Rooms II
  + 347, Top K Frequent Elements
  + 973, K closest points to origin
  + 5, Longest Palindrome Substring
  + 53, Maximum Subarray
  + 121, Best Time to Buy and Sell a Stock
  + 139, Word Break
  + 322, Coin Change
  + 146, LRU Cache
  + 155, Min Stack
  + 295, Find Median from Data Stream
  + 297, Serialize and Deserialize a Binary Tree
  + 348, Design Tic-Tac-Toe
  + 642, Design Search Auto-Complete System
  + 895, Maximum Frequency Stack
  + 7, Reverse Integer
  + 176, Second Highest Salary
  + 763, Partition Labels
  + 957, Prison Cells after N Days
* Meta Problems
  + 1140, Stone Game II
  + 54, Spiral Matrix
  + 542, 01 Matrix
  + 766, Toeplitz Matrix
  + 867, Transpose Matrix
  + 59, Spiral Matrix II
  + 73, Set Matrix Zeroes
  + 311, Sparse Matrix Multiplication
  + 566, Reshape the Matrix
  + 756, Pyramid Transition Matrix
  + 519, Random Flip Matrix
  + 885, Spiral Matrix III
  + 1314, Matrix Block Sum
  + 1572, Matrix Diagonal Sum
  + 1975, Maximum Matrix Sum
  + 2326, Spiral Matrix IV
  + 2906, Construct Product Matrix
  + 3033, Modify the Matrix
  + 3248, Snake in Matrix
  + 74, Search a 2D Matrix
  + 861, Score after Flipping Matrix
  + 1329, Sort the Matrix Diagonally
  + 2679, Sum in a Matrix
  + 240, Search a 2D Matrix II
  + 1030, Matrix Cells in Distance Order
  + 1091, Shortest Path in Binary Matrix
  + 1632, Rank transform of a Matrix
  + 1380, Lucky Numbers in a Matrix
  + 2319, Check if Matrix is X-Matrix
  + 2392, Build a Matrix with Conditions
  + 2675, Array of Objects to Matrix
  + 2718, Sum of Matrix after Queries
  + 2946, Matrix Similarity after Cycle Shifts
  + 3078, Match Alpha-Numeric Pattern in Matrix I
  + 329, Longest increasing path in a Matrix
  + 1253, Reconstruct a 2-row binary matrix
  + 1582, Special positions in a binary matrix
  + 2373, Largest local values in a Matrix
  + 378, Kth Smallest element in a sorted Matrix
  + 562, Longest Line of Consecutive one in Matrix
  + 1252, Cells with odd values in a Matrix
  + 1337, The K weakest rows in a Matrix
  + 1351, Count Negative numbers in a Sorted Matrix
  + 1594, Maximum non-negative product in a Matrix
  + 2387, Median of a row-wise sorted Matrix
  + 2713, Maximum Strictly Increasing cells in a Matrix
  + 2732, Find a good subset of the Matrix
  + 1605, Find valid Matrix given row and column sums
  + 1886, Determine weather Matrix can be obtained by Rotation
  + 2123, Minimum operations to remove Adjacent Ones in Matrix
  + 1284, Minimum number of Flips to Convert Binary Matrix to Zero Matrix
  + 2435, Paths in Matrix who’s sum is divisible by K
  + 1439, Find the Kth smallest sum of a matrix with sorted rows
  + 2556, Disconnect Path in a Binary Matrix by at most one flip
  + 44, Wildcard Matching
  + 1023, Camelcase Matching
  + 10, Regular Expression Matching
  + 473, Matchsticks to Square
  + 544, Output Contest Matches
  + 686, Repeated String Match
  + 942, DI String Match
  + 792, Number of Matching Sub-Sequences
  + 2301, Match Substring after Replacement
  + 1408, String Matching in an Array
  + 1688, Count of Matches in a Tournament
  + 1773, Count items matching a rule
  + 2339, All the matches of the league
  + 2410, Maximum matching of Players with Trainers
  + 971, Flip binary tree to match pre-order traversal
  + 2019, The Score of Students Solving Math Expression
  + 3034, Numbers of Sub-arrays that match a pattern I
  + 3036, Numbers of Sub-arrays that match a pattern II

# 1, Two-Sum

## Description

Given an array of integers nums and an integer target, return *indices of the two numbers such that they add up to target*. You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice. You can return the answer in any order.

## Test Examples and Constraints

**Example 1:**

**Input:** nums = [2,7,11,15], target = 9

**Output:** [0,1]

**Explanation:** Because nums[0] + nums[1] == 9, we return [0, 1].

**Example 2:**

**Input:** nums = [3,2,4], target = 6

**Output:** [1,2]

**Example 3:**

**Input:** nums = [3,3], target = 6

**Output:** [0,1]

**Constraints:**

* 2 <= nums.length <= 104
* -109 <= nums[i] <= 109
* -109 <= target <= 109
* **Only one valid answer exists.**

**Follow-up:**Can you come up with an algorithm that is less than O(n2) time complexity?

## Solution

from typing import List

class Solution:

    def twoSum(self, nums: List[int], target: int) -> List[int]:

        prevMap = {}

        for i, n in enumerate(nums):

            diff = target - n

            if diff in prevMap:

                return [prevMap[diff], i]

            prevMap[n] = i

        return

# 3, Longest Substring without Repeating Chars

## Description

Given a string s, find the length of the **longest** **substring** without repeating characters.

## Test Examples and Constraints

**Example 1:**

**Input:** s = "abcabcbb"

**Output:** 3

**Explanation:** The answer is "abc", with the length of 3.

**Example 2:**

**Input:** s = "bbbbb"

**Output:** 1

**Explanation:** The answer is "b", with the length of 1.

**Example 3:**

**Input:** s = "pwwkew"

**Output:** 3

**Explanation:** The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

**Constraints:**

* 0 <= s.length <= 5 \* 104
* s consists of English letters, digits, symbols and spaces.

## Solution

class Solution:

    def lengthOfLongestSubstring(self, s: str) -> int:

        charSet = set()

        l = 0

        res = 0

        for r in range(len(s)):

            while s[r] in charSet:

                charSet.remove(s[l])

                l += 1

            charSet.add(s[r])

            res = max(res, r - l + 1)

        return res

## Analysis

TBD

# 8, String to Integer

## Description

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer.

The algorithm for myAtoi(string s) is as follows:

1. **Whitespace**: Ignore any leading whitespace (" ").
2. **Signedness**: Determine the sign by checking if the next character is '-' or '+', assuming positivity is neither present.
3. **Conversion**: Read the integer by skipping leading zeros until a non-digit character is encountered or the end of the string is reached. If no digits were read, then the result is 0.
4. **Rounding**: If the integer is out of the 32-bit signed integer range [-231, 231 - 1], then round the integer to remain in the range. Specifically, integers less than -231 should be rounded to -231, and integers greater than 231 - 1 should be rounded to 231 - 1.

Return the integer as the final result.

## Test Examples and Constraints

**Example 1:**

**Input:** s = "42"

**Output:** 42

**Explanation:**

The underlined characters are what is read in and the caret is the current reader position.

Step 1: "42" (no characters read because there is no leading whitespace)

^

Step 2: "42" (no characters read because there is neither a '-' nor '+')

^

Step 3: "42" ("42" is read in)

^

**Example 2:**

**Input:** s = " -042"

**Output:** -42

**Explanation:**

Step 1: " -042" (leading whitespace is read and ignored)

^

Step 2: " -042" ('-' is read, so the result should be negative)

^

Step 3: " -042" ("042" is read in, leading zeros ignored in the result)

^

**Example 3:**

**Input:** s = "1337c0d3"

**Output:** 1337

**Explanation:**

Step 1: "1337c0d3" (no characters read because there is no leading whitespace)

^

Step 2: "1337c0d3" (no characters read because there is neither a '-' nor '+')

^

Step 3: "1337c0d3" ("1337" is read in; reading stops because the next character is a non-digit)

^

**Example 4:**

**Input:** s = "0-1"

**Output:** 0

**Explanation:**

Step 1: "0-1" (no characters read because there is no leading whitespace)

^

Step 2: "0-1" (no characters read because there is neither a '-' nor '+')

^

Step 3: "0-1" ("0" is read in; reading stops because the next character is a non-digit)

^

**Example 5:**

**Input:** s = "words and 987"

**Output:** 0

**Explanation:**

Reading stops at the first non-digit character 'w'.

**Constraints:**

* 0 <= s.length <= 200
* s consists of English letters (lower-case and upper-case), digits (0-9), ' ', '+', '-', and '.'.

## Solution

from typing import List

# 11, Container with the Most Water

## Description

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return *the maximum amount of water a container can store*. **Notice** that you may not slant the container.

## Test Examples and Constraints

**Example 1:**

A blue and black graph

Description automatically generated

**Input:** height = [1,8,6,2,5,4,8,3,7]

**Output:** 49

**Explanation:** The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

**Example 2:**

**Input:** height = [1,1]

**Output:** 1

**Constraints:**

* n == height.length
* 2 <= n <= 105
* 0 <= height[i] <= 104

## Solution

from typing import List

class Solution:

    def maxArea(self, height: List[int]) -> int:

        l, r = 0, len(height) - 1

        res = 0

        while l < r:

            res = max(res, (r - l) \* min(height[l], height[r]))

            if height[l] < height[r]:

                l += 1

            else:

                r -= 1

        return res

# 12, Integer to Roman

## Description

Seven different symbols represent Roman numerals with the following values:

| Symbol | Value |
| --- | --- |
| I | 1 |
| V | 5 |
| X | 10 |
| L | 50 |
| C | 100 |
| D | 500 |
| M | 1000 |

Roman numerals are formed by appending the conversions of decimal place values from highest to lowest. Converting a decimal place value into a Roman numeral has the following rules:

* If the value does not start with 4 or 9, select the symbol of the maximal value that can be subtracted from the input, append that symbol to the result, subtract its value, and convert the remainder to a Roman numeral.
* If the value starts with 4 or 9 use the **subtractive form** representing one symbol subtracted from the following symbol, for example, 4 is 1 (I) less than 5 (V): IV and 9 is 1 (I) less than 10 (X): IX. Only the following subtractive forms are used: 4 (IV), 9 (IX), 40 (XL), 90 (XC), 400 (CD) and 900 (CM).
* Only powers of 10 (I, X, C, M) can be appended consecutively at most 3 times to represent multiples of 10. You cannot append 5 (V), 50 (L), or 500 (D) multiple times. If you need to append a symbol 4 times use the **subtractive form**.

Given an integer, convert it to a Roman numeral.

## Test Examples and Constraints

**Example 1:**

**Input:** num = 3749

**Output:** "MMMDCCXLIX"

**Explanation:**

3000 = MMM as 1000 (M) + 1000 (M) + 1000 (M)

700 = DCC as 500 (D) + 100 (C) + 100 (C)

40 = XL as 10 (X) less of 50 (L)

9 = IX as 1 (I) less of 10 (X)

Note: 49 is not 1 (I) less of 50 (L) because the conversion is based on decimal places

**Example 2:**

**Input:** num = 58

**Output:** "LVIII"

**Explanation:**

50 = L

8 = VIII

**Example 3:**

**Input:** num = 1994

**Output:** "MCMXCIV"

**Explanation:**

1000 = M

900 = CM

90 = XC

4 = IV

**Constraints:**

* 1 <= num <= 3999

## Solution

class Solution:

    def intToRoman(self, num: int) -> str:

        symList = [

            ["I", 1], ["IV", 4], ["V", 5], ["IX", 9], ["X", 10],

            ["XL", 40], ["L", 50], ["XC", 90], ["C", 100],

            ["CD", 400], ["D", 500], ["CM", 900], ["M", 1000]]

        res = ""

        for sym, val in reversed(symList):

            if num // val:

                count = num // val

                res += (sym \* count)

                num = num % val

        return res

# 13, Roman to Integer

## Description

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

**Symbol** **Value**

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

* I can be placed before V (5) and X (10) to make 4 and 9.
* X can be placed before L (50) and C (100) to make 40 and 90.
* C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer.

## Test Examples and Constraints

**Example 1:**

**Input:** s = "III"

**Output:** 3

**Explanation:** III = 3.

**Example 2:**

**Input:** s = "LVIII"

**Output:** 58

**Explanation:** L = 50, V= 5, III = 3.

**Example 3:**

**Input:** s = "MCMXCIV"

**Output:** 1994

**Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

**Constraints:**

* 1 <= s.length <= 15
* s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').
* It is **guaranteed** that s is a valid roman numeral in the range [1, 3999].

## Solution

class Solution:

    def romanToInt(self, s: str) -> int:

        roman = { 'I': 1, 'V': 5, 'X': 10, 'L': 50,

                  'C': 100, 'D': 500, 'M': 1000 }

        res = 0

        length = len(s)

        for i in range(length):

            si = roman[s[i]]

            sip = roman[s[i + 1]] if i + 1 < length else 0

            if i + 1 < length and si < sip:

                res -= si

            else:

                res += si

        return res

# 15, 3-Sum

## Description

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

Notice that the solution set must not contain duplicate triplets.

## Test Examples and Constraints

**Example 1:**

**Input:** nums = [-1,0,1,2,-1,-4]

**Output:** [[-1,-1,2],[-1,0,1]]

**Explanation:**

nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.

nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.

nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.

The distinct triplets are [-1,0,1] and [-1,-1,2].

Notice that the order of the output and the order of the triplets does not matter.

**Example 2:**

**Input:** nums = [0,1,1]

**Output:** []

**Explanation:** The only possible triplet does not sum up to 0.

**Example 3:**

**Input:** nums = [0,0,0]

**Output:** [[0,0,0]]

**Explanation:** The only possible triplet sums up to 0.

**Constraints:**

* 3 <= nums.length <= 3000
* -105 <= nums[i] <= 105

## Solution

from typing import List

# 16, 3-Sum Closest

## Description

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target.

Return *the sum of the three integers*.

You may assume that each input would have exactly one solution.

## Test Examples and Constraints

**Example 1:**

**Input:** nums = [-1,2,1,-4], target = 1

**Output:** 2

**Explanation:** The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

**Example 2:**

**Input:** nums = [0,0,0], target = 1

**Output:** 0

**Explanation:** The sum that is closest to the target is 0. (0 + 0 + 0 = 0).

**Constraints:**

* 3 <= nums.length <= 500
* -1000 <= nums[i] <= 1000
* -104 <= target <= 104

## Solution

from typing import List

# , Implement StrStr

## Description

Fill me

## Test Examples and Constraints

You

## Solution

from typing import List

# 48, Rotate Image

## Description

You are given an n x n 2D matrix representing an image, rotate the image by **90** degrees (clockwise).

You have to rotate the image [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm), which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

## Test Examples and Constraints

**Example 1:**

A black arrow pointing to a white rectangular object

Description automatically generated

**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [[7,4,1],[8,5,2],[9,6,3]]

**Example 2:**

A number with a arrow pointing to the right

Description automatically generated with medium confidence

**Input:** matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]

**Output:** [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]

**Constraints:**

* n == matrix.length == matrix[i].length
* 1 <= n <= 20
* -1000 <= matrix[i][j] <= 1000

## Solution

from typing import List

# 49, Group Anagrams

## Description

Given an array of strings strs, group the anagrams together. You can return the answer in **any order**.

## Test Examples and Constraints

You

## Solution

from typing import List

# 76, Minimum Window Substring

## Description

Given two strings s and t of lengths m and n respectively, return *the****minimum window*** ***substring***

*of*s*such that every character in*t*(****including duplicates****) is included in the window*. If there is no such substring, return *the empty string*"".

The testcases will be generated such that the answer is **unique**.

## Test Examples and Constraints

You

## Solution

from typing import List

# 165, Compare Version Numbers

## Description

Given two **version strings**, version1 and version2, compare them. A version string consists of **revisions** separated by dots '.'. The **value of the revision** is its **integer conversion** ignoring leading zeros.

To compare version strings, compare their revision values in **left-to-right order**. If one of the version strings has fewer revisions, treat the missing revision values as 0.

Return the following:

* If version1 < version2, return -1.
* If version1 > version2, return 1.
* Otherwise, return 0.

## Test Examples and Constraints

You

## Solution

from typing import List

# 238, Product of Array Except Self

## Description

Given an integer array nums, return *an array* answer *such that* answer[i] *is equal to the product of all the elements of* nums *except* nums[i].

The product of any prefix or suffix of nums is **guaranteed** to fit in a **32-bit** integer.

You must write an algorithm that runs in O(n) time and without using the division operation.

## Test Examples and Constraints

You

## Solution

from typing import List

# 268, Missing Number

## Description

Given an array nums containing n distinct numbers in the range [0, n], return *the only number in the range that is missing from the array.*

## Test Examples and Constraints

You

## Solution

from typing import List

# 273, Integer to English Words

## Description

Convert a non-negative integer num to its English words representation.

## Test Examples and Constraints

You

## Solution

from typing import List

# 387, First Unique Character in a String

## Description

Given a string s, find the **first** non-repeating character in it and return its index.

If it **does not** exist, return -1.

## Test Examples and Constraints

You

## Solution

from typing import List

# 678, Valid Parenthesis

## Description

Given a string s containing only three types of characters: '(', ')' and '\*', return true *if* s *is****valid***.

The following rules define a **valid** string:

* Any left parenthesis '(' must have a corresponding right parenthesis ')'.
* Any right parenthesis ')' must have a corresponding left parenthesis '('.
* Left parenthesis '(' must go before the corresponding right parenthesis ')'.
* '\*' could be treated as a single right parenthesis ')' or a single left parenthesis '(' or an empty string "".

## Test Examples and Constraints

You

## Solution

from typing import List

# 819, Most Common Word

## Description

Given a string paragraph and a string array of the banned words banned, return *the most frequent word that is not banned*. It is **guaranteed** there is **at least one word** that is not banned, and that the answer is **unique**.

The words in paragraph are **case-insensitive** and the answer should be returned in **lowercase**.

## Test Examples and Constraints

You

## Solution

from typing import List

# 937, Re-Order Log Files

## Description

You are given an array of logs. Each log is a space-delimited string of words, where the first word is the **identifier**.

There are two types of logs:

* **Letter-logs**: All words (except the identifier) consist of lowercase English letters.
* **Digit-logs**: All words (except the identifier) consist of digits.

Reorder these logs so that:

1. The **letter-logs** come before all **digit-logs**.
2. The **letter-logs** are sorted lexicographically by their contents. If their contents are the same, then sort them lexicographically by their identifiers.
3. The **digit-logs** maintain their relative ordering.

Return *the final order of the logs*.

## Test Examples and Constraints

You

## Solution

from typing import List

# 42, Trapping Rain Water

## Description

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it can trap after raining.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2, Add Two Numbers

## Description

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order**, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

## Test Examples and Constraints

You

## Solution

from typing import List

# 21, Merge Two Sorted Lists

## Description

You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list*.

## Test Examples and Constraints

You

## Solution

from typing import List

# 25, Reverse Nodes in K-Group

## Description

Given the head of a linked list, reverse the nodes of the list k at a time, and return *the modified list*.

k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.

You may not alter the values in the list's nodes, only nodes themselves may be changed.

## Test Examples and Constraints

You

## Solution

from typing import List

# 138, Copy List with Random Pointer

## Description

A linked list of length n is given such that each node contains an additional random pointer, which could point to any node in the list, or null.

Construct a [**deep copy**](https://en.wikipedia.org/wiki/Object_copying#Deep_copy) of the list. The deep copy should consist of exactly n **brand new** nodes, where each new node has its value set to the value of its corresponding original node. Both the next and random pointer of the new nodes should point to new nodes in the copied list such that the pointers in the original list and copied list represent the same list state. **None of the pointers in the new list should point to nodes in the original list**.

For example, if there are two nodes X and Y in the original list, where X.random --> Y, then for the corresponding two nodes x and y in the copied list, x.random --> y.

Return *the head of the copied linked list*.

The linked list is represented in the input/output as a list of n nodes. Each node is represented as a pair of [val, random\_index] where:

* val: an integer representing Node.val
* random\_index: the index of the node (range from 0 to n-1) that the random pointer points to, or null if it does not point to any node.

Your code will **only** be given the head of the original linked list.

## Test Examples and Constraints

You

## Solution

from typing import List

# 206, Reverse Linked Lists

## Description

Given the head of a singly linked list, reverse the list, and return *the reversed list*.

## Test Examples and Constraints

You

## Solution

from typing import List

# 23, Merge K-Sorted Lists

## Description

You are given an array of k linked-lists lists, each linked-list is sorted in ascending order.

*Merge all the linked-lists into one sorted linked-list and return it.*

## Test Examples and Constraints

You

## Solution

from typing import List

# 98, Validate Binary Search Tree

## Description

Given the root of a binary tree, *determine if it is a valid binary search tree (BST)*.

A **valid BST** is defined as follows:

* The left subtree of a node contains only nodes with keys **less than** the node's key.
* The right subtree of a node contains only nodes with keys **greater than** the node's key.
* Both the left and right subtrees must also be binary search trees.

## Test Examples and Constraints

You

## Solution

from typing import List

# 101, Symmetric Tree

## Description

Given the root of a binary tree, *check whether it is a mirror of itself* (i.e., symmetric around its center).

## Test Examples and Constraints

You

## Solution

from typing import List

# 102, Binary Tree, Level Order Traversal

## Description

Given the root of a binary tree, return *the level order traversal of its nodes' values*. (i.e., from left to right, level by level).

## Test Examples and Constraints

You

## Solution

from typing import List

# 103, Binary Tree, ZigZag Order Traversal

## Description

Given the root of a binary tree, return *the zigzag level order traversal of its nodes' values*. (i.e., from left to right, then right to left for the next level and alternate between).

## Test Examples and Constraints

You

## Solution

from typing import List

# 124, Binary Tree, Maximum Path Sum

## Description

A **path** in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connecting them. A node can only appear in the sequence **at most once**. Note that the path does not need to pass through the root.

The **path sum** of a path is the sum of the node's values in the path.

Given the root of a binary tree, return *the maximum****path sum****of any****non-empty****path*.

## Test Examples and Constraints

You

## Solution

from typing import List

# 126, Word Ladder II

## Description

A **transformation sequence** from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

* Every adjacent pair of words differs by a single letter.
* Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
* sk == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return *all the****shortest transformation sequences****from* beginWord *to* endWord*, or an empty list if no such sequence exists. Each sequence should be returned as a list of the words*[beginWord, s1, s2, ..., sk].

## Test Examples and Constraints

You

## Solution

from typing import List

# 127, Word Ladder

## Description

A **transformation sequence** from word beginWord to word endWord using a dictionary wordList is a sequence of words beginWord -> s1 -> s2 -> ... -> sk such that:

* Every adjacent pair of words differs by a single letter.
* Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
* sk == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return *the****number of words****in the****shortest transformation sequence****from* beginWord *to* endWord*, or*0*if no such sequence exists.*

## Test Examples and Constraints

You

## Solution

from typing import List

# 200, Number of Islands

## Description

Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return *the number of islands*.

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

## Test Examples and Constraints

You

## Solution

from typing import List

# 207, Course Schedule

## Description

There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you **must** take course bi first if you want to take course ai.

* For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.

Return true if you can finish all courses. Otherwise, return false.

## Test Examples and Constraints

You

## Solution

from typing import List

# 236, Lowest Common Ancestor of Binary Tree

## Description

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the [definition of LCA on Wikipedia](https://en.wikipedia.org/wiki/Lowest_common_ancestor): “The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**).”

## Test Examples and Constraints

You

## Solution

from typing import List

# 543, Diameter of Binary Tree

## Description

Given the root of a binary tree, return *the length of the****diameter****of the tree*.

The **diameter** of a binary tree is the **length** of the longest path between any two nodes in a tree. This path may or may not pass through the root.

The **length** of a path between two nodes is represented by the number of edges between them.

## Test Examples and Constraints

You

## Solution

from typing import List

# 675, Cut-Off Trees for Golf Event

## Description

You are asked to cut off all the trees in a forest for a golf event. The forest is represented as an m x n matrix. In this matrix:

* 0 means the cell cannot be walked through.
* 1 represents an empty cell that can be walked through.
* A number greater than 1 represents a tree in a cell that can be walked through, and this number is the tree's height.

In one step, you can walk in any of the four directions: north, east, south, and west. If you are standing in a cell with a tree, you can choose whether to cut it off.

You must cut off the trees in order from shortest to tallest. When you cut off a tree, the value at its cell becomes 1 (an empty cell).

Starting from the point (0, 0), return *the minimum steps you need to walk to cut off all the trees*. If you cannot cut off all the trees, return -1.

**Note:** The input is generated such that no two trees have the same height, and there is at least one tree needs to be cut off.

## Test Examples and Constraints

You

## Solution

from typing import List

# 733, Flood Fill

## Description

You are given an image represented by an m x n grid of integers image, where image[i][j] represents the pixel value of the image. You are also given three integers sr, sc, and color. Your task is to perform a **flood fill** on the image starting from the pixel image[sr][sc].

To perform a **flood fill**:

1. Begin with the starting pixel and change its color to color.
2. Perform the same process for each pixel that is **directly adjacent** (pixels that share a side with the original pixel, either horizontally or vertically) and shares the **same color** as the starting pixel.
3. Keep **repeating** this process by checking neighboring pixels of the *updated* pixels and modifying their color if it matches the original color of the starting pixel.
4. The process **stops** when there are **no more** adjacent pixels of the original color to update.

Return the **modified** image after performing the flood fill.

## Test Examples and Constraints

You

## Solution

from typing import List

# 17, Letter Combinations of a Phone Number

## Description

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

A close-up of a phone keypad

Description automatically generated

## Test Examples and Constraints

You

## Solution

from typing import List

# 22, Generate Parenthesis

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 79, Word Search

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 212, Word Search II

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 4, Median of Two Sorted Arrays

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 33, Search in Rotated Sorted Array

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 56, Merge Intervals

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 167, Two Sum II, input array is sorted

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 215, Kth largest element in an array

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 253, Meeting Rooms II

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 347, Top K Frequent Elements

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 973, K closest points to origin

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 5, Longest Palindrome Substring

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 53, Maximum Subarray

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 121, Best Time to Buy and Sell a Stock

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 139, Word Break

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 322, Coin Change

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 146, LRU Cache

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 155, Min Stack

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 295, Find Median from Data Stream

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 297, Serialize and Deserialize a Binary Tree

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 348, Design Tic-Tac-Toe

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 642, Design Search Auto-Complete System

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 895, Maximum Frequency Stack

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 7, Reverse Integer

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 176, Second Highest Salary

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 763, Partition Labels

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 957, Prison Cells after N Days

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

Part II, META QUESTIONS

# 1140, Stone Game II

## Description

Fill me.

## Test Examples and Constraints

Fill me.

## Solution

from typing import List

# 54, Spiral Matrix

## Description

Fill me.

## Test Examples and Constraints

Fill me.

## Solution

from typing import List

# 542, 01 Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 766, Toeplitz Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 867, Transpose Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 59, Spiral Matrix II

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

# 73, Set Matrix Zeroes

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 311, Sparse Matrix Multiplication

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 566, Reshape the Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 756, Pyramid Transition Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 519, Random Flip Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 885, Spiral Matrix III

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1314, Matrix Block Sum

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1572, Matrix Diagonal Sum

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1975, Maximum Matrix Sum

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2326, Spiral Matrix IV

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2906, Construct Product Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 3033, Modify the Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 3248, Snake in Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 74, Search a 2D Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 861, Score after Flipping Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1329, Sort the Matrix Diagonally

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2679, Sum in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 240, Search a 2D Matrix II

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1030, Matrix Cells in Distance Order

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1091, Shortest Path in Binary Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1632, Rank transform of a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1380, Lucky Numbers in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2319, Check if Matrix is X-Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2392, Build a Matrix with Conditions

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2675, Array of Objects to Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2718, Sum of Matrix after Queries

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2946, Matrix Similarity after Cycle Shifts

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 3078, Match Alpha-Numeric Pattern in Matrix I

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 329, Longest increasing path in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1253, Reconstruct a 2-row binary matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1582, Special positions in a binary matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2373, Largest local values in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 378, Kth Smallest element in a sorted Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 562, Longest Line of Consecutive one in Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1252, Cells with odd values in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1337, The K weakest rows in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1351, Count Negative numbers in a Sorted Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1594, Maximum non-negative product in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2387, Median of a row-wise sorted Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2713, Maximum Strictly Increasing cells in a Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2732, Find a good subset of the Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1605, Find valid Matrix given row and column sums

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1886, Determine weather Matrix can be obtained by Rotation

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2123, Minimum operations to remove Adjacent Ones in Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1284, Minimum number of Flips to Convert Binary Matrix to Zero Matrix

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2435, Paths in Matrix who’s sum is divisible by K

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1439, Find the Kth smallest sum of a matrix with sorted rows

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2556, Disconnect Path in a Binary Matrix by at most one flip

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 44, Wildcard Matching

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1023, Camelcase Matching

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 10, Regular Expression Matching

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 473, Matchsticks to Square

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 544, Output Contest Matches

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 686, Repeated String Match

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 942, DI String Match

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 792, Number of Matching Sub-Sequences

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2301, Match Substring after Replacement

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1408, String Matching in an Array

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1688, Count of Matches in a Tournament

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 1773, Count items matching a rule

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2339, All the matches of the league

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2410, Maximum matching of Players with Trainers

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 971, Flip binary tree to match pre-order traversal

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 2019, The Score of Students Solving Math Expression

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 3034, Numbers of Sub-arrays that match a pattern I

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List

# 3036, Numbers of Sub-arrays that match a pattern II

## Description

Fill me.

## Test Examples and Constraints

You

## Solution

from typing import List