



# 技术时代的生活与工作

Tech among Life and Work

作者: Qingsong Liao

组织: Free Club

时间: November 29, 2025

邮箱: llqingsong@qq.com



# 目录

<b>第一章 被技术改变的生活</b>	<b>2</b>	8.8 19. Remove Nth Node From End of List (Medium) . . . . .	44
<b>第二章 学技术的哲学</b>	<b>5</b>	8.9 20. Valid Parentheses (Easy) . . . . .	46
2.1 学习什么技术 . . . . .	5	8.10 23. Merge k Sorted Lists (Hard) . . . . .	48
2.2 怎么学习底层技术 . . . . .	5	8.11 28. Find the Index of the First Occurrence in a String (Easy) . . . . .	50
<b>第三章 编辑器哲学</b>	<b>7</b>	8.12 34. Find First and Last Position of Element in Sorted Array (Medium) . . . . .	52
3.1 如果人生还有一次，我还选 emacs . . . . .	7	8.13 37. Sudoku Solver (Hard) . . . . .	54
3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人 公学习失传的武林秘籍一般（只是小说 剧情而矣）。 . . . . .	8	8.14 40. Combination Sum II (Medium) . . . . .	56
<b>第四章 操作系统哲学</b>	<b>18</b>	8.15 46. Permutations (Medium) . . . . .	58
4.1 我的操作系统特点 . . . . .	18	8.16 47. Permutations II (Medium) . . . . .	60
4.2 用 linux 只为戒色 . . . . .	18	8.17 48. Rotate Image (Medium) . . . . .	62
4.3 学技术只为戒 x . . . . .	19	8.18 51. N-Queens (Hard) . . . . .	64
<b>第五章 键盘流哲学</b>	<b>20</b>	8.19 53. Maximum Subarray (Medium) . . . . .	66
5.1 设计哲学 . . . . .	20	8.20 64. Minimum Path Sum (Medium) . . . . .	68
5.2 分层 . . . . .	21	8.21 67. Add Binary (Easy) . . . . .	71
5.3 本地刷录 . . . . .	21	8.22 69. Sqrt(x) (Easy) . . . . .	73
5.4 不只键盘 . . . . .	24	8.23 70. Climbing Stairs (Easy) . . . . .	75
<b>第六章 屏幕哲学</b>	<b>26</b>	8.24 72. Edit Distance (Medium) . . . . .	77
6.1 Feels: . . . . .	26	8.25 75. Sort Colors (Medium) . . . . .	79
6.2 Features: . . . . .	27	8.26 76. Minimum Window Substring (Hard) .	81
6.3 Tips for mono: . . . . .	27	8.27 77. Combinations (Medium) . . . . .	83
<b>第七章 fog.h</b>	<b>30</b>	8.28 79. Word Search (Medium) . . . . .	85
7.1 Quick Start . . . . .	30	8.29 81. Search in Rotated Sorted Array II (Medium) . . . . .	89
7.2 "Just rewrite shit in C, Just rewrite shit in C "— tsoding . . . . .	30	8.30 83. Remove Duplicates from Sorted List (Easy) . . . . .	91
<b>第八章 不用 AI 完成 Leetcode 刷题</b>	<b>31</b>	8.31 88. Merge Sorted Array (Easy) . . . . .	94
8.1 吃饭/做题/睡觉，论计算机技术革命及其 后果 . . . . .	31	8.32 91. Decode Ways (Medium) . . . . .	96
8.2 1. Two Sum (Easy) . . . . .	32	8.33 94. Binary Tree Inorder Traversal (Easy) .	98
8.3 3. Longest Substring Without Repeating Characters (Medium) . . . . .	34	8.34 99. Recover Binary Search Tree (Medium)	101
8.4 4. Median of Two Sorted Arrays (Hard) .	36	8.35 101. Symmetric Tree (Easy) . . . . .	104
8.5 5. Longest Palindromic Substring (Medium)	38	8.36 104. Maximum Depth of Binary Tree (Easy)	107
8.6 10. Regular Expression Matching (Hard) .	40	8.37 105. Construct Binary Tree from Preorder and Inorder Traversals (Medium) . . . . .	109
8.7 15. 3Sum (Medium) . . . . .	42	8.38 106. Construct Binary Tree from Inorder and Postorder Traversals (Medium) . . . . .	112
		8.39 109. Convert Sorted List to Binary Search Tree (Medium) . . . . .	115
		8.40 110. Balanced Binary Tree (Easy) . . . . .	117

8.41	121. Best Time to Buy and Sell Stock (Easy)	120
8.42	122. Best Time to Buy and Sell Stock II (Medium) . . . . .	122
8.43	126. Word Ladder II (Hard) . . . . .	124
8.44	128. Longest Consecutive Sequence (Medium) . . . . .	126
8.45	130. Surrounded Regions (Medium) . . .	128
8.46	135. Candy (Hard) . . . . .	130
8.47	136. Single Number (Easy) . . . . .	132
8.48	139. Word Break (Medium) . . . . .	134
8.49	142. Linked List Cycle II (Medium) . . .	136
8.50	144. Binary Tree Preorder Traversal (Easy)	139
8.51	145. Binary Tree Postorder Traversal (Easy)	142
8.52	146. LRU Cache (Medium) . . . . .	145
8.53	148. Sort List (Medium) . . . . .	148
8.54	149. Max Points on a Line (Hard) . . . .	150
8.55	154. Find Minimum in Rotated Sorted Ar- ray II (Hard) . . . . .	153
8.56	155. Min Stack (Medium) . . . . .	155
8.57	162. Find Peak Element (Medium) . . . .	158
8.58	167. Two Sum II - Input Array Is Sorted (Medium) . . . . .	160
8.59	168. Excel Sheet Column Title (Easy) . .	162
8.60	169. Majority Element (Easy) . . . . .	164
8.61	188. Best Time to Buy and Sell Stock IV (Hard) . . . . .	166
8.62	190. Reverse Bits (Easy) . . . . .	168
8.63	198. House Robber (Medium) . . . . .	170
8.64	202. Happy Number (Easy) . . . . .	172
8.65	205. Isomorphic Strings (Easy) . . . . .	174
8.66	208. Implement Trie (Prefix Tree) (Medium)	176
8.67	210. Course Schedule II (Medium) . . . .	179
8.68	213. House Robber II (Medium) . . . . .	181
8.69	215. Kth Largest Element in an Array (Medium) . . . . .	183
8.70	217. Contains Duplicate (Easy) . . . . .	185
8.71	218. The Skyline Problem (Hard) . . . . .	187
8.72	221. Maximal Square (Medium) . . . . .	190
8.73	225. Implement Stack using Queues (Easy)	193
8.74	226. Invert Binary Tree (Easy) . . . . .	196
8.75	227. Basic Calculator II (Medium) . . . . .	198
8.76	232. Implement Queue using Stacks (Easy)	200
8.77	235. Lowest Common Ancestor of a Bi- nary Search Tree (Medium) . . . . .	203
8.78	236. Lowest Common Ancestor of a Bi- nary Tree (Medium) . . . . .	206
8.79	238. Product of Array Except Self (Medium)	209
8.80	239. Sliding Window Maximum (Hard) . .	211
8.81	240. Search a 2D Matrix II (Medium) . .	213
8.82	241. Different Ways to Add Parentheses (Medium) . . . . .	216
8.83	242. Valid Anagram (Easy) . . . . .	218
8.84	257. Binary Tree Paths (Easy) . . . . .	220
8.85	260. Single Number III (Medium) . . . .	223
8.86	268. Missing Number (Easy) . . . . .	225
8.87	279. Perfect Squares (Medium) . . . . .	227
8.88	287. Find the Duplicate Number (Medium)	229
8.89	300. Longest Increasing Subsequence (Medium) . . . . .	231
8.90	303. Range Sum Query - Immutable (Easy)	233
8.91	304. Range Sum Query 2D - Immutable (Medium) . . . . .	236
8.92	307. Range Sum Query - Mutable (Medium)	240
8.93	309. Best Time to Buy and Sell Stock with Cooldown (Medium) . . . . .	243
8.94	310. Minimum Height Trees (Medium) . .	245
8.95	312. Burst Balloons (Hard) . . . . .	248
8.96	313. Super Ugly Number (Medium) . . . .	250
8.97	318. Maximum Product of Word Lengths (Medium) . . . . .	252
8.98	322. Coin Change (Medium) . . . . .	254
8.99	328. Odd Even Linked List (Medium) . .	256
8.100	332. Reconstruct Itinerary (Hard) . . . .	258
8.101	338. Counting Bits (Easy) . . . . .	261
8.102	343. Integer Break (Medium) . . . . .	263
8.103	347. Top K Frequent Elements (Medium)	265
8.104	376. Wiggle Subsequence (Medium) . . .	267
8.105	377. Combination Sum IV (Medium) . .	269
8.106	380. Insert Delete GetRandom O(1) (Medium) . . . . .	271
8.107	404. Sum of Left Leaves (Easy) . . . . .	274
8.108	406. Queue Reconstruction by Height (Medium) . . . . .	277
8.109	409. Longest Palindrome (Easy) . . . . .	279
8.110	413. Arithmetic Slices (Medium) . . . .	281
8.111	416. Partition Equal Subset Sum (Medium)	283
8.112	417. Pacific Atlantic Water Flow (Medium)	285
8.113	432. All O'one Data Structure (Hard) . .	288
8.114	435. Non-overlapping Intervals (Medium)	291

8.115 437. Path Sum III (Medium) . . . . .	293
8.116 448. Find All Numbers Disappeared in an Array (Easy) . . . . .	295
8.117 450. Delete Node in a BST (Medium) . .	297
8.118 451. Sort Characters By Frequency (Medium) . . . . .	299
8.119 452. Minimum Number of Arrows to Burst Balloons (Medium) . . . . .	301
8.120 455. Assign Cookies (Easy) . . . . .	303
8.121 461. Hamming Distance (Easy) . . . . .	305
8.122 462. Minimum Moves to Equal Array Elements II (Medium) . . . . .	307
8.123 474. Ones and Zeroes (Medium) . . . . .	309
8.124 476. Number Complement (Easy) . . . . .	311
8.125 494. Target Sum (Medium) . . . . .	313
8.126 503. Next Greater Element II (Medium)	315
8.127 513. Find Bottom Left Tree Value (Medium)	317
8.128 530. Minimum Absolute Difference in BST (Easy) . . . . .	320
8.129 538. Convert BST to Greater Tree (Medium)	323
8.130 540. Single Element in a Sorted Array (Medium) . . . . .	325
8.131 542. 01 Matrix (Medium) . . . . .	327
8.132 543. Diameter of Binary Tree (Easy) . .	330
8.133 547. Number of Provinces (Medium) . .	333
8.134 566. Reshape the Matrix (Easy) . . . . .	336
8.135 572. Subtree of Another Tree (Easy) . .	338
8.136 583. Delete Operation for Two Strings (Medium) . . . . .	341
8.137 594. Longest Harmonious Subsequence (Easy) . . . . .	343
8.138 605. Can Place Flowers (Easy) . . . . .	345
8.139 617. Merge Two Binary Trees (Easy) . .	347
8.140 633. Sum of Square Numbers (Medium)	349
8.141 637. Average of Levels in Binary Tree (Easy) . . . . .	351
8.142 646. Maximum Length of Pair Chain (Medium) . . . . .	354
8.143 647. Palindromic Substrings (Medium) .	356
8.144 650. 2 Keys Keyboard (Medium) . . . . .	358
8.145 653. Two Sum IV - Input is a BST (Easy)	360
8.146 665. Non-decreasing Array (Medium) . .	363
8.147 669. Trim a Binary Search Tree (Medium)	365
8.148 680. Valid Palindrome II (Easy) . . . . .	368
8.149 684. Redundant Connection (Medium) . .	370
8.150 693. Binary Number with Alternating Bits (Easy) . . . . .	373
8.151 695. Max Area of Island (Medium) . . . .	375
8.152 696. Count Binary Substrings (Easy) . .	377
8.153 697. Degree of an Array (Easy) . . . . .	379
8.154 714. Best Time to Buy and Sell Stock with Transaction Fee (Medium) . . . . .	381
8.155 739. Daily Temperatures (Medium) . . . . .	383
8.156 763. Partition Labels (Medium) . . . . .	385
8.157 769. Max Chunks To Make Sorted (Medium) . . . . .	387
8.158 785. Is Graph Bipartite? (Medium) . . . .	389
8.159 870. Advantage Shuffle (Medium) . . . . .	392
8.160 882. Reachable Nodes In Subdivided Graph (Hard) . . . . .	394
8.161 889. Construct Binary Tree from Preorder and Postorder Traversals (Medium) . . . . .	397
8.162 897. Increasing Order Search Tree (Easy)	400
8.163 932. Beautiful Array (Medium) . . . . .	402
8.164 934. Shortest Bridge (Medium) . . . . .	404
8.165 1091. Shortest Path in Binary Matrix (Medium) . . . . .	406
8.166 1105. Filling Bookcase Shelves (Medium)	409
8.167 1110. Delete Nodes And Return Forest (Medium) . . . . .	413
8.168 1143. Longest Common Subsequence (Medium) . . . . .	415
8.169 1249. Minimum Remove to Make Valid Parentheses (Medium) . . . . .	417
8.170 1530. Number of Good Leaf Nodes Pairs (Medium) . . . . .	419
8.171 设计刷题环境 . . . . .	422
8.171.1 leetgo . . . . .	422
8.171.2 本地刷题测试 . . . . .	429
<b>第九章 版本更新历史</b>	<b>432</b>
<b>参考文献</b>	<b>435</b>
<b>附录 A 开发中遇见的各种软件问题</b>	<b>436</b>
A.0.1 General Contributions . . . . .	436
A.0.2 <b>TODO</b> Advent of code in zig? . . . .	436
A.0.3 <b>TODO</b> package leego elisp package as emacs package . . . . .	436
A.0.4 <b>FIXME</b> How to modify org-mode's #+INCLUDE: . . . . .	436

---

A.0.5 <b>DONE</b> how many hours of pc idle's power consumption is equal to hibernate save ram to disk? . . . . .	437	A.0.13 <b>DONE</b> add sway emacs pkg? . . . . .	439
A.0.6 <b>DONE</b> almost same vscode-emacs package. . . . .	437	A.0.14 <b>DONE</b> how to make rm safer? . . . . .	439
A.0.7 <b>DONE</b> how to run current region's content (if not in region, run that line) into current project's vterm .	437	A.0.15 <b>DONE</b> recentf moving file persisnt	439
A.0.8 <b>DONE</b> pyim % 字符触发 Not enough arguments for format string	438	A.0.16 <b>DONE</b> pull request to the porn site list project . . . . .	439
A.0.9 <b>DONE</b> explain this wget command	438	A.0.17 <b>DONE</b> make Leetcode fresh tasks list . . . . .	440
A.0.10 <b>DONE</b> how to do emacs overlay in nixos to config not to have somepackage natively exclude . .	438	A.0.18 <b>DONE</b> learn how to reference in org	441
A.0.11 <b>DONE</b> using consult-gh notifications, and I just type wrong pass as my own pc's pass, not github token . . . . .	438	A.0.19 <b>DONE</b> upload large files into the github repo . . . . .	441
A.0.12 <b>DONE</b> how the ~/.mozilla/firefox/profiles.ini looks like? . . . . .	439	A.0.20 <b>DONE</b> try to package real-mono-theme to melpa . . . . .	441
		A.0.21 <b>DONE</b> creat new account's github repos for github.io . . . . .	442
		A.0.22 <b>DONE</b> upload Purezen to github .	442
		<b>附录 B 嵌入式软件招聘常见要求</b>	<b>443</b>
		<b>附录 C 嵌入式招聘常见笔试问题</b>	<b>444</b>
		<b>附录 D 嵌入式招聘常见面试问题</b>	<b>445</b>

# 序章 技术何为?

## 内容提要

- |                                |                           |
|--------------------------------|---------------------------|
| □ 任务编号：以名为“廖青松”的男婴为身份出生在重庆普通家庭 | □ 完成难度：简单到极难，活着也没 BOSS 啊？ |
| □ 开始时间：2002-11-04              | □ 需要小时：一生，大约三万天           |
| □ 结束时间：总有一死                    | □ 本章要点：我对于技术与人生的看法        |

世界嘈杂<sup>1</sup>、瘾品横流、随机与即时构成新的枷锁，而真正的自由来自主动的剥离。当目光离开诱惑、离开广告、离开无意义的瞬息刺激，心才开始变得干净。生活越简，能量越纯；越慢，感受越深；越少，越能看见真正的自己。

身体是意志的第一块土地。空腹的清明、弱光的安静、低温的醒觉、缓慢进食的耐性，都在一点点重塑我们早已被工业习惯磨钝的感官。行走、奔跑、提举、拉起、俯卧撑、壶铃、农夫行走——这些最朴素的动作让人重新理解力量的意义：力量不是爆发，而是日复一日不受伤、不懈怠、让心跳稳、饮食亦是自律的延伸。

避免加工。让精神长的那种沉稳、远离高糖盐油，回到豆果菜、坚果与发酵的本味，让身体习惯真实的能量，而不是被化学甜味与工业脂肪驱使的假饱与假快乐。克制不是苦行，而是温和地恢复本能。

至于技术，真正的价值不在追逐流行，而在深入底层、理解根本。用 Emacs<sup>2</sup>，不是为了高效，而是为了与世界拉开距离，与自己靠得更近；不随机、不即时、无广告、无噪声，是一种长期的心性训练。读 LFS、LKD 与 SOC 文档，写驱动、调性能、跑 QEMU、玩内核、读源码、用 LATEX 和 Org 写书，是为了获得一种“我真的懂了”的安静感。而这种懂，不是为了炫耀，不是为了沉迷，而是为了让工作成为谋生技能，让生活成为真正的生活。

真正的智慧是：学会技术，然后把技术放下；拥有力量，然后让力量变得温柔。生活是吃饭、睡觉、读书、编程、走路、壶铃；生命是健康、乐观、会意、精进、闲适与稳稳的力量。

世界广袤、事物繁多，而心若清澈，幸福忽然变得极小，也极近——不来自外界，只来自自身安静而坚定的内心。



图 1: free-your-soul

<sup>1</sup>第一个嘈杂的提示，如果你进入了 HTML 版本，请[点此进入 PDF 版本](#)。

<sup>2</sup>本书的牛“图片”均出自于 GNU.org 的自由艺术。GNU 计划，又称革奴计划，是由 Richard Stallman 在 1983 年 9 月 27 日公开发起的，目标是创建一套完全自由的操作系统，将牛作为 gnu 和 emacs 的象征是因为 gnu 在英文中本身就有“牛羚, 角马”的意思，有趣的是 gnu 本意为 Gnu is Not Unix…再问里面的 gnu 什么意思？还是 Gnu's Not Unix—a recursive acronym，就像是中国的俗语“山上有座庙”，无穷尽也。

# 第一章 被技术改变的生活

-10000000005 年，一切都变了。宇宙诞生了。

-5600000005 年，一切都变了。太阳诞生了。

-4600000005 年，一切都变了。地球诞生了。

-200000005 年，一切都变了。生物诞生了。

-80000005 年，一切都变了。恐龙诞生了。

-30025 年，一切都变了。人类诞生了。

-26025 年，一切都变了。语言诞生了。

-25025 年，一切都变了。家庭诞生了。

-22025 年，一切都变了。农业诞生了。

-20025 年，一切都变了。文明诞生了。

-10025 年，一切都变了。文字诞生了。

-9025 年，一切都变了。货币诞生了。

-8025 年，一切都变了。法律诞生了。

-1025 年，一切都变了。书写诞生了。

-225 年，一切都变了。哲学诞生了。

-100 年，一切都变了。集权诞生了。

-25 年，一切都变了。数学诞生了。

-5 年，一切都变了。宗教诞生了。

1000 年，一切都变了。宗教火热。丝绸之路、黑死病、天花、中世纪、疾病、皇帝、国王、奇迹、骑士阶层、武士、城墙、封建帝国、马鞍…

1279 年，一切都变了。文艺火热。宋朝灭亡，造出火药，指南针，造纸术，印刷术的中华文明终结。  
但是技术被武力消灭，但技术开始了…

1600 年，一切都变了。帝国火热。欧洲人发现了美洲。帝国主义、殖民主义、印刷术、土豆、巧克力、辣椒、枪械与大炮、地理大发现、数学与工程学…

1700 年，一切都变了。烟草火热。哥伦布大交换的阴影笼罩大地。黑奴、民族国家、地理学、历史学、语言学、生理学、资本主义、大英帝国…

1800 年，一切都变了。鸦片火热。资本主义的阴影笼罩大地。共产主义、物理学、汽车、文学、工业革命、开发美洲、煤气、火车…

1900 年，一切都变了。技术狂热。技术在欧洲大地遍地生根。工业化、城市化、流水线、民主、飞机、电力、公司、灯光…

1915 年，一切都变了。一战结束。技术的阴影笼罩大地。农药、化肥、化学、媒体、通讯、武器、内燃机、全球化、电报、相对论…

1945 年，一切都变了。二战结束。技术的阴影笼罩大地。女权、核弹、雷达、战斗机、计算机、工业化、旅游业、赌博业、广播、量子物理…

1965 年，一切都变了。编程火热。技术停滞或是爆发？电视、摇滚乐、基因学、编程理论、月球、民用航天、芯片、电子技术、教育扩大…

1975 年，一切都变了。贸易火热。中国迎来开放。技术在中国大地遍地生根。留守儿童、大量生产、环境污染、贫富差距、道德沦丧、城乡差距…

1995 年，一切都变了。网络火热。技术迎来它们的终极载体。科学知识/各种思想文化加速传播、游戏、营销、贸易、色情、视频/图片/电影/音乐…

2002 年，一切都变了。我诞生了。技术/成瘾/财富/变革的高峰在我这代中国人最能体现，从未体验过的事物比比皆是。从小时候看电视到青少年时的玩电脑再到成年后被各种技术主动或被动控制

---

…这代人注定是分化的一代。

2025 年，一切都变了。智能火热。技术停滞或是爆发？社交媒体、区块链、转基因、移动支付、智能设备…

2100 年，一切都变了。死神永生。技术黑暗或是造神？火星旅行、人工智能、机器人、脑机接口、基因修改…

1902 年出生，1910 年化学，1925 年通迅交通，1950 年核弹计算机，1970 年克隆羊，1992 年互联网？2002 年出生，2010 年手机，2025 年 ChatGPT，2050 年 AGI，2070 年具身智能，2092 年永生？平均寿命 73 岁？我大概是能苟到 90 岁？也许会有一场战争…灭绝不应该长生的人。

- 后现代技术要点：(上瘾/传销/非人)

1. 靠人传销，如加密货币，房地产拼多多。(利用心理/行为设计/资本主义自由市场)
2. 不要人类，如人工智能，通讯与机器人。(利用物理/自动化的算法与机械/大数据)
3. 让人上瘾，如社交媒体，烟酒赌博旅游。(利用生理/产量强度更大/劳动时间减少)

- 硬技术要点：医学人工智能(癌症)-> 脑机接口(义肢)-> 机器人载体(永生)-> 恒星际(星际)

- 医学大数据-> 大模型-> 生理大模型-> 基因建模？

- 脑机需要什么？生物 + 数据

- 机器人需要什么？算法 + 数据

- 恒星际？能源 + 机器人

- 问题：

- 伦理性：如果都能永生，如果有谁不能呢？
- 可靠性：基于数据，准确性不高，如果程序出错呢？
- 社会性：人类会成为什么样？幸福与快乐到底是什么？

- 悲观主义者：

- 衣/食/住/行/性…

- 以前：自制/采集狩猎/自建/只有走路/真情或强暴
  - 现在：拼多多/拼好饭/烂尾楼/倒闭国产新能源/彩礼八十八万

- 意义感和存在感

- 以前：靠身体劳作生活，自己的劳动成果，直接贡献在自己身上
  - 现在：靠国家/公司/投资生活，靠机械化身体动作/脑力说唱生活

- 人类命运与未来

- 以前：自己的命运在自己/家族/别人手上
  - 现在：自己的命运在政府/老师/老板/网友…手上

- 乐观主义者：

- 衣/食/住/行/性…

- 以前：难得/难管/土房/5 公里内/不可控
  - 现在：便宜/便宜/商品房/环游世界/易得

- 意义感和存在感

- 以前：迷信宗教与古语
  - 现在：更多知识与卫生

- 人类命运与未来

- 以前：狮子老虎毒蛇毒虫山路水沟低温
  - 现在：温暖舒服平静安定可控多彩闲适

- 对平常人而言：

- 更少的劳动，更多的消费
  - 更少的暴力，更多的新闻

- 
- 更少的情感，更多的媒体
  - 更少的迷信，更多的知识
  - 更少的空闲，更多的享乐
  - 更少的体力，更多的智力
  - 更少的杀戮，更多的竞争
  - 更少的疾病，更多的人口
  - 更少的歌舞，更多的噪声
  - 更少的饥饿，更多的肥胖
  - 更少的家族，更多的国家
  - 更少的闲适，更多的焦虑
  - 更少的长久，更多的变动
  - 更少的现实，更多的未来

2025 年，一切都变了。我不再相信技术/社会/科学…我打算出去种田。

2035 年，一切都变了。ChatGPT，我该生孩子吗？下面是我的个人信息:xxx

2045 年，一切都变了。伟大的 ChatGPT 系统，今天您的系统算出应在人体培养室中产出多少个孩子？

2065 年，一切都变了。…10111010111110101010101000011110101011…

101001011110010111110101010101000011110101011100000000000000…

42 时间 000000000000000000 宇宙 1 太阳 56 智能 99 人类 99 历史 99 人 99 年 …

赚钱，存肌肉，学习技术，准备永生？不知道，但我们得先学习在技术时代怎么生存。毕竟，本书书文即为《技术时代的生活与工作：Tech among Life and Work》，主要写写自己的技术学习，写写技术见闻，写写人类与机器的命运。地球与这些文字一样，都小如宇宙中的沙子，可就算无人观赏，也是一种美丽。我在沙子上写着沙子，不觉渺小而觉其之美，闪闪的一点有多少国王与伟人的故事，一切尽此矣。在无限的宇宙舞台上，熵增即是技术之艺术。当星星暗淡、黑洞蒸发、文明消亡，宇宙留下的，唯有技术。

# 第二章 学技术的哲学

## 内容提要

- 开始时间: 2025-11-19
- 结束时间: 2026 年前
- 完成难度: 难难难

- 需要小时: 100+
- 本章要点: linux/c/zig/rust/rtos

人没有梦想那和咸鱼有什么区别呢? ——周星驰

## 2.1 学习什么技术

表 2.1: path-to-know-tech

process	url	for
<b>fag</b>	flag/cintro	c, pretent iam tsoding
embedded C book	read book	c, how compiler/os/c works
Yeetcode	leetgo/book	cpp[rust], algo ds, solve task
paperlike-c/el	paperlike-go	elisp,paperlike emacs controller
ziglings	ziglings	zig, basic ziglangs speed run
zag	fag	zig, I have language erotic
paperlike-zig	paperlike-el	zig, make cli/tary
nixos r2s	github-repos	nix, for network addiction
TsurgizOS	os.phil-opp	zig[rust], make general os for cv
nixos rasberry-pi	github-repos	nix, for embedded os/screen/driver
clings	ziglings	c, lings but clang
freertos emulator	rtos	c, use general rtos
lvgl eink rtos	lvgl	c, embedded ui driver/sdl
lvgl lora loc	graphic	c, embedded openstress/sdl/lvgl
RZOS	rtos	zig, make general rtos
Celest	game	zig, embedded game/sdl
Safephone	electronic	lvgl/eink/openstress/lora/3Dprint stm32/nix/electronic/network nsfw image/text detected

- All in all, it's for 技术哲学
  - 网络/低速/时间/域名/ai 过滤/linux 内核构建-r2s
  - 屏幕/护眼/低成瘾/驱动设计/eink 屏幕算法-paperlike
  - 通信/安全/无依赖/去平台/lora 远距离通信-safephone
  - 交通/电工/电子电路/openstress-去五佰<sup>1</sup>那里学修车

## 2.2 怎么学习底层技术

### 1. 第一阶段: 打基础 (C / Linux / GCC / Emacs)

- C 语言: 学会指针、内存管理、结构体、函数指针; 刷一些小项目, 比如实现 malloc、shell、http server。
- Linux 基础: 熟悉命令行、文件系统、进程/线程、信号、管道、套接字。
- GCC: 学会编译流程 gcc -E/-S/-c/-o, 理解预处理、汇编、链接, 玩一下 objdump 和 nm。
- Emacs: 把它当 IDE 来用, 掌握基本编辑、调试、补全、LSP 支持。

<sup>1</sup>我父亲那边的车行亲戚

## 2. 第二阶段：系统调试与逆向（GDB / QEMU）

- GDB：练习断点、单步、查看寄存器/内存、调试多线程/远程调试。
- QEMU：
  - 用它运行 Linux kernel 或裸机程序。
  - 学会 qemu -S -s + gdb remote 调试，体验调试内核的感觉。
  - 研究 QEMU 的设备模拟（比如 VirtIO、PCI），理解虚拟化和硬件抽象。

## 3. 第三阶段：进阶编程语言与系统（C++ / Zig / RTOS）

- Zig：Zig 是现代系统编程语言，学习它的构建系统、内存模型，可以和 C/C++ 混合编程。
- RTOS：从 FreeRTOS 入手，学任务调度、中断、任务间通信（队列/信号量）。可以用 QEMU 模拟 Cortex-M 板子跑 RTOS。

## 4. 第四阶段：融会贯通（大型项目 / 内核 / 编译器）

- QEMU + GDB：调试内核启动、写内核模块。
- 编译器开发：研究 GCC 或 Clang 的前端/后端；或者用 Zig 写一个简化编译器。
- 个人项目：比如写一个简易 RTOS，或者在 QEMU 里跑自己写的内核。

## 5. 操作系统构建与升级（Yocto/Android/内核）

在树莓派上交叉编译 Linux 内核，修改驱动或设备树（Device Tree）进行硬件适配。尝试用 Yocto 或 Buildroot 构建自定义 Linux 镜像，加入自己编写或修改的驱动模块。安装和编译 LineageOS（Android for Pi）或类似 Android 系统，修改系统服务或 HAL 层。实践 OTA（Over-The-Air）升级机制，模拟系统升级和回滚操作。硬件 Bringup（CPU/GPU/Memory/Peripherals）利用树莓派的 GPIO、SPI、I2C、CAN、PWM 等接口，练习外设 Bringup 和驱动调试。连接摄像头模块、显示屏（LCD 或 E-Ink）或音频模块，编写驱动和控制程序。使用 perf/ftrace/gprof 分析 CPU/GPU 性能瓶颈，优化程序调度。

## 6. 系统稳定性与性能优化

构建多线程/多进程网络服务，使用 socket 编程实现客户端/服务器通信，模拟并发场景。在树莓派上测试高负载条件下的系统稳定性，分析内核日志、内存占用和 CPU/GPU 使用率。实践内核参数调优，如调节调度器、内存缓存策略，观察性能变化。

## 7. 客户功能定制与基线升级

自己设计一个树莓派应用（如小型智能家居控制器或信息显示终端），从硬件 Bringup 到应用功能定制完整流程。模拟不同版本的系统镜像管理，练习分支合并、基线升级和版本回退。

## 8. 加分技能训练

- (a). 低功耗优化：通过关闭不必要的外设、调节 CPU/GPU 频率或使用 E-Ink 显示屏练习功耗控制。
- (b). 虚拟化/容器：在树莓派上安装 QEMU/KVM 或 Docker，运行多系统虚拟环境，模拟嵌入式应用部署。
- (c). 芯片平台经验：如果有 MTK 或其他 ARM 板卡，可以对比树莓派练习移植和平台适配经验。

## 第三章 编辑器哲学

This is my blog and p(pdf)log which written in org and LATEX, and emacs/nixos config, code snippet and even more, all in just one repo. 这是仓库是我的博客也是我用 LATEX 写的书，还是我是 emacs 和 nixos 配置，代码模板等等。

```
(org-babel-do-load-languages  
'org-babel-load-languages '((emacs-lisp . t)(shell . t)))
```

```
if [[ -f "/home/$USER/.emacs.d" ]] then  
    git clone --depth 1 https://github.com/jamescherti/minimal-emacs.d ~/.emacs.d  
fi  
ln -sf $PWD/nixos/ ~/.config/;  
ln -sf $PWD/snippets ~/.config/;  
ln -sf $PWD/nixos/hmdz.pyim ~/.config/;  
ln -sf $PWD/nixos/post-init.el ~/.emacs.d/;  
if [[ $XAPIAN_CJK_NGRAM != "true" ]] then  
    sudo cp /etc/hardware-configuration.nix $PWD/nixos/hardware-configuration.nix  
    chmod 777 $PWD/nixos/hardware-configuration.nix  
    mkdir -p $HOME/.config/sops/age  
    cp .keys.txt $HOME/.config/sops/age/keys.txt  
    sudo nix-channel --add https://nixos.org/channels/nixos-unstable  
    sudo nix-channel --update  
    sudo nixos-rebuild switch --flake /home/$USER/.config/nixos#$hostname --option substituters '  
        https://mirrors.ustc.edu.cn/nix-channels/store https://cache.nixos.org' --cores 6 -j 12;  
fi
```

**Listing 3.1:** 我能用这代码块重现我的电脑，怎么样？帅吧！

这是 org 的文学编程能力最简单的体现，从 c 到 bash，在一个 org 文件中能一起运行，神奇吧！为什么 emacs 这么强大？大概就是因为它使用的 elisp 相当“灵”吧，再说为什么 emacs 这么小众，也大概是它的 elisp 太“难”了吧。有些人不敢学，怕自己投入又得不到钞票，有些人不能学，他们把自己的心给了别的教派如“vim”“vscode”于是固步自封，自高自大，vim 模态天下第一！我曾同样如此，抱着网上破烂配凑出的 neovim，对那些大神的操作流口水但是又没法认同 emacs。其实犹豫不定很正常，进入一个陌生的环境，很多怪人怪事，emacs 更是如此，你常常不知道那些人口中说的 eglot/flycheck/eww 是什么？常常写错一些代码。当然这同样也是学习 emacs 的乐趣，也正是它的陌生带来的新鲜，而且有了 chatgpt 的帮助，学习起来相当方便。

### 3.1 如果人生还有一次，我还选 emacs

我最早听闻 Emacs，大概是 2022 年在 Cswiki 中作者对 Emacs 的溢美之词“神用编辑器”<sup>1</sup>。当时我猎奇心甚为严重，又觉得大家都在用的 VSCode 无法体现我的气质，便先折腾三年 Vim/Neovim/Helix 后在 2024 开始自学 Emacs，接触 Emacs 先是 doom 框架后是接近原生的 purcell。喜欢换来换去，或许这是年青人的通病。

现在，我已不再年青，却依然喜欢 Emacs 就像我曾喜欢一个女生，但她从来不知道。曾经沧海难为水，或许这是人老了之后的通病，而我觉得更可能是因为 1978 年诞生的 Emacs 依然年青。

<sup>1</sup>当然还有大佬们的溢美之词：python 的蛇叔说用 emacs 就像是开了几十年的私家车/rust 的离职语言之父一天 60% 时间在 emacs 中/ruby 作者道“emacs 改变了我的人生”/大概所有函数式语言的作者都得用 emacs…java 的高司令自己写了一个新 emacs 并相当不爽 richard stallman, linus 自己维护了 mircoemacs…还别说那些库程序员了，curl/ffmpeg 等作者…，ffmpeg 的天才程序员也开发了 qemu/先进的 pi 算法/用 llm 压缩工具，没错他也维护了个 emacs。更别说 richard stallman 了，他开始了 gcc/gdb，也开始了 emacs 了 gnu，开源世界自君开。总之，emacs 是个好玩的计算机历史产物，似乎仍没有过时。

## 3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人公学习失传的武林秘籍一般（只是小说剧情而矣）。

1. 要我说认为我的 Emacs 配置有什么特点，大概就是这样几点吧。

- minimal: my config based on the clean and fast minimal-emacs.
- zen: a pure white and black theme written by me called "real-mono-theme".
- reproducible: package manager is using **NixOS**.
- simple: just a post-init.el which less than 2000 lines.
- keyboard: for my **ZMK** config, only me can use those as flow~
- efficient: elegant selecting like queen and fast editing like king.

2. 要我说 **Emacs** 就和与小说里秘传的十八般武林功夫一般，有定身/移形换影/透视经脉/必杀技/念经/修炼/经文/拜师/法宝/女伴/侠义…甚至武功大概不过如此…拳脚身头可用的功夫，那里有无限可能的电子计算机的空间大啊？

- 动：插入/切换/输出/笔记/补全/专注/跳转/投修/替换/扩张/行修/选择/整理/重复/变化/比较/潜在
- 阅：简用/单页/全页/源码/图书/用文/说文
- 用：项目/虚拟/日志/中文/终端
- 浏：搜/库/问
- 库：Linux/Libc/Posix/C ABI/Network/Algo/Protocols
- 标题说的有定身/移形换影/透视经脉/必杀技/念经/修炼/经文/拜师/法宝/女伴/侠义，我也确实有的定身 view-only/移形换影 switch-buffer/透视经脉 describ-function/必杀技 kill-emacs/念经 read gpl3/修炼 rtfm/经文 info/拜师 rms/法宝 list-package/女伴 WoMan/侠义 contributor。emacs-china 管论坛的用户叫“道友”，似乎就已经点明这点，学习难度大，学习结果虚幻无比，需要气定神闲，天地之间少有人闻，更少有人真正能掌握这一门技术，不叫“道友”怎么说得过去？

### 3. Write The Fucking Code

- 心写：org/theme/hide-xx/ligature/no-xx/vertico-flat
- 补写：cape/hippie-expand/snippet
- 跳写：occur/rg/eglot/imenu/mark/avy/consult/compilation
- 横竖：flush/delete/duplicate/sort/move-text || iedit/mc/rectangle
- 比缩：diff/ediff/xxx-diff/xxx-merge
- 选扩：surround/expand-region/native-mark/mc
- 选点：embark/vertico/menu-bar/tooltips/tool-bar
- 重复：macro
- 切换：dired/find|switch-file/sway/ibuffer/tramp
- 替换：replace string|regex
- 变换：shift-number transpose/case-switch/narrow
- 内在：save-xxx/xx-mode/envrc/editorconfig/autofmt

### 4. Read The Friendly Manual

- 源码：eglot/helpful
- 用简：tl2r
- 用文：man/woman
- 一页：devdocs
- 全页：eww/nginx
- 说文：info
- 图书：nov/pdf-tool

3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人公学习失传的武林秘籍一般（只是小说剧情而矣）。

---

## 5. Browse The Enshitty Web

- 搜索: tavily
- 仓库: consult-gh
- 询问: gptel
- 交流: gnus/irc/eww

## 6. Use The Minimal Tools

- 工具: magit/git-xx/project/nix/docker/kubernetes
- 中文: quicksdcv/pyim
- 日志: syslog|journalctl-mode
- 终端: vterm/repl/shell

## 7. 总之，我试着去思考一下什么是写作与编程，阅读与浏览，内心与外物

- 写作: 需要重复阅读，反复遣词造句，思考上下文人物/事件关系，写作有编辑环节，但那是“编辑”做的事。
- 编程: 需要更多更快的查找方式，更多更快的浏览，思考调用/协议的关系，编程需要大量的编辑 (indent/上下移动/复制粘贴)。
- 阅读: 长文/长时间的专注，资源越少越好，写作应该减少阅读别人的作品。
- 浏览: 短文/短时间的专注，资源极多…想多都不行…编程应该大量阅读别人的作品。
- 内心: 得之心而应之于手，很多优秀文学作品使用笔和纸写作，计算机都得用键盘，减少了肌肉的使用本身就会带来人的记忆损失。
- 外物: 但编程本身就越来越重“外物”，所使用的事物大概都是其他人所写，要依赖上游，依赖工具。写作需要的就只是笔和纸…相对的 编程，大概没有 10 个以上的工具是做不完备。
- 写作与编程，
  - 一个大概，一个精确
  - 一个阅读，一个浏览
  - 一个重内心，一个重外物
  - 一个写活人，一个写死物
  - 现在的人连读书都不愿意了，怎么可以会编程？
  - 唯一的原因: 写代码来 钱
  - 可钱又带来虚无…
- Emacs 是让编程更像写作的一件事。

- 减少拼音带来的“任务切换占用”，emacs 因为全能，完美替代 bloate 的系统输入法，使用五笔之类的输入法，又因为 emacs 的完全可配置性，只看得到新字不断产生，极大减少了需要选词的心理负担。
- 减少鼠标带来的“肌肉使用减少”，emacs 因为按键复杂，替代使用鼠标点击，提高程序员的编辑效率。
- 减少依赖，emacs 需要的依赖极少，甚至因为 emacs 的多平台可用性，用好 emacs 可直接移至更多系统，而不用像用 vim 之类的软件担心自己 shell/term 不适配新系统。
- 减少分心，越古越好，1978 年后，技术很大程度控制了人类，在那个年代，rms 提出“要么技术控制人类，要么人类控制技术”，这句话放在现在越来越明显。
- 减少

### 1. 我的编辑器邪教之旅：

浏览器里的 vimium C，因为 wsl 学习的 neovim，因为换上 nixos 后抄的 helix 配置，因为大四没找工作太闲学的 emacs。我每用一个就大呼“我草，早知道，还得是 xx！”，然后给同学老师家人都安利安利，给同学说“我有一宝”，给老师说“看我操作”，给家人嘻嘻傻笑像是捡到五百钱。虽然这代表着我非常的闲，但是折腾 emacs 确实让我不是很闲到以至于抑郁，我还是在用 Vimium C 和 emacs，赋予我闪电切换页面的神奇体

### 3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人公学习失传的武林秘籍一般（只是小说剧情而矣）。

---

验，当然，代价是它们也让我得了一种名为“换页面换 buffer”的 ADHD，再者，其实不用 vim/emacs 也会得，只不过叫作“懒得换页面就直接玩手机”ADHD，“懒得折腾电脑就去玩 CSGO 成了换弹癌”的 ADHD。在这个时代，玩 csgo 和玩 emacs 有什么区别呢？一个能炒皮肤，另一个能折腾皮肤，如是而矣。

#### 2. 我使用 emacs 没有太多原因就只是：

- (a). 依赖少，之前用 vim，我就要关心 vim 键位的 app 适合用来看 pdf，看 epub 看文档，整 vimium 和 vim 一个键位/什么 lazygit/tmux/fcitx/alacritty，还生怕它们之间有什么矛盾，关键 vimsript 也不适合作为灵活的语言，整天很困扰很憋屈，我还折腾 colemak、虎码和 nixos，我的天，对于任何一个大脑快定型的成年人来说，这简直是噩梦！幸好有 emacs 救我于快捷键地狱中，装包能 list-package 再一点就安装了，pyim 配置虎码就一行的事，那里还要 fcitx5 怎么怎么样！自从用了 emacs，我再也没有折改过 firefox，一是我没时间，二是我就只用 emacs 的 eww，我也就不再在乎什么 vimium 的键位和 vim 不一样了，**能用就行才是真理！**再说用上了 elisp，那就叫美梦成真，想写一个工具简直就比玩手机打游等更有意思多了。什么 tmux/lazygit/nixos，现在我只想笑笑，我用 vterm/magit/任何平台都能跑 emacs 不是爽多了吗？只要给我 emacs/编译器，写代码就是轻轻松松的事！emacs 改好了我的强迫症和极多主义，现在我心宽又极简，感谢 emacs！
- (b). 功能多，之前用 helix，虽是开箱即用但是很明显它的功能只限于写代码，快是快但是对我这样的学习困难型人来说，快和慢没有多大的区别，只会让我再难受于“我好慢它好快，我是不是 x 痞了”，helix 是个自带丰富功能工具，但是对于习惯 homerow 的人来说其实用 vim mode 没有什么增益，只会让我更迷茫“我能直接不动手就能上下了，干啥还要这个 hjkl??”。有插件和服务器模式这点就足以胜过 helix 了。当然我从 helix 的文档也学到了很多编辑知识，如果没有它的 emacs 主题和打算用 scheme 写插件系统，和让我折腾很久的 helix eink 主题，我也没有潜移默化现在的编辑技术和极简主题审美了。
- (c). 分心小，之前用 vimium 那就是一个换换换的感觉，总是想去换页面，看看新闻看看新工具，上下跳转，左右腾挪，helix 功能太少不能在里面呆太久，而 vimium 是个浏览器插件在里面呆着又太容易去娱乐，用 vim 主要是去 youtube 参加编辑器神战，helix 也好 vimium 也好。其实我学习 emacs，到能快速掌握 avy 靠得是之前用过 vimium，学习 surround/expand 靠的是 helix，能熟悉 info/man/tldr 靠得是之前在终端里学习 vim。总之 emacs 必然是编辑技术的一个终点站。
- (d). 定制化，没有一个 term emulator 支持比例字体的，只有 emacs，也有人说了为什么用 bookerly，就用 vscode 的什么 ubuntumono/nerdfont 不好吗？这我得确实说：“非等宽字绝对好用”，为什么空格得道 M 一样宽，为什么 l 和 W 一样宽？现在编程/终端处处都是等宽字体，很多人在根本就没用过的情况下痴迷在各种 nerdfont 间切换并坚持自己的“信仰”，这种信仰只有在用了一个优美的非等宽字体才会破灭。多显示 1.45 内容？还是只有 0.6896551724137931 的丑恶巨大空格？要用非等宽得到 emacs 看看，因为没有一个 editor 的主题是可以自己完全定制的，vscode 那样的使用 javascript 还难以控制还过于复杂资源又大的就算了。elisp/fsf/clean/simple，学 emacs 之前根本不能理解这话！有些事情得是自己做了才能得之心而应之于手！
- (e). 历史，50 年历史的编辑器，计算机史的一页，还能写 100 年左右。
- (f). 装 b，(linus/stallman) 开源巨人们/(rust/python/java/ruby/curl/各种函数式… ) 之父们钦定编辑器。

```
implementation-specific diagnostic information on the standard error output and  
implementation-specific diagnostic information on the standard error output and d
```

图 3.1: mono 显示 80 个字符

```
implementation-specific diagnostic information on the standard error output and  
implementation-specific diagnostic information on the standard error output and diagnostic information on the stand
```

图 3.2: 等高下非等宽显示 116 个字符

#### 3. 上网必备 emacs

---

```
(setq c-backslash-column 99)
(setq c-backslash-max-column 99)
(setq c-basic-offset 4)
(setq c-indent-level 4)
;; (setq c-default-style "linux")
(advice-add 'c-update-modeline :override #'ignore)
(defun c-compile-current-file ()
  (interactive)
  (unless (file-exists-p "Makefile")
    (let ((file (file-name-nondirectory buffer-file-name)))
      (compile (format "%s -o %s.out %s %s"
                      (or (getenv "CC") "gcc")
                      (file-name-sans-extension file)
                      (or (getenv "CPPFLAGS") "")
                      (or (getenv "CFLAGS") "-Wall -g")
                      file)))))

(dolist (hook '(c-mode-hook c++-mode-hook asm-mode-hook))
  (add-hook hook (lambda ()
    (define-key c-mode-base-map (kbd "C-c C-c") 'c-compile-current-file)
    (define-key c-mode-base-map (kbd "C-c C-M-c") (lambda () (interactive)
      (setq-local compilation-read-command nil)
      (call-interactively 'compile)))
    (define-key c-mode-base-map (kbd "C-M-q") nil)
    (define-key c-mode-base-map (kbd "(") nil)
    (define-key c-mode-base-map (kbd "(") nil)))))

(setq confirm-kill-processes nil)
```

图 3.3: 明显可以看出等宽不符合人的阅读习惯，只有在多行修改 (mc) 或跳转 (avy) 中需要字符不变才“稍显有益”。

```
("\\\.ii\\\" . c++-mode)
:config
(setq c-backslash-column 99)
(setq c-backslash-max-column 99)
(setq c-basic-offset 4)
(setq c-indent-level 4)
;; (setq c-default-style "linux")
(advice-add 'c-update-modeline :override #'ignore)
(defun c-compile-current-file ()
  (interactive)
  (unless (file-exists-p "Makefile")
    (let ((file (file-name-nondirectory buffer-file-name)))
      (compile (format "%s -o %s.out %s %s"
                      (or (getenv "CC") "gcc")
                      (file-name-sans-extension file)
                      (or (getenv "CPPFLAGS") "")
                      (or (getenv "CFLAGS") "-Wall -g")
                      file)))))

(dolist (hook '(c-mode-hook c++-mode-hook asm-mode-hook))
  (add-hook hook (lambda ()
    (define-key c-mode-base-map (kbd "C-c C-c") 'c-compile-current-file)
    (define-key c-mode-base-map (kbd "C-c C-M-c") (lambda () (interactive)
      (setq-local compilation-read-command nil)
      (call-interactively 'compile)))
    (define-key c-mode-base-map (kbd "C-M-q") nil)
    (define-key c-mode-base-map (kbd "(") nil)
    (define-key c-mode-base-map (kbd "(") nil)))))

(setq confirm-kill-processes nil)
```

图 3.4: 非等宽下，英文字符清晰区别明显，字符间隔小，更优美自然连续

3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人公学习失传的武林秘籍一般（只是小说剧情而矣）。

```
(keymap-global-set "s-m" #'switch-to-gptel)
(keymap-global-set "<Tools>" #'tavily-search)

(defun switch-to-gptel()
  (interactive)
  (if (equal (current-buffer) (gptel "*deepseek*"))
      (previous-buffer)
    (switch-to-buffer "*deepseek*")))

(defun tavily-search-async (callback query &optional search-depth max-results exclude_domains
                           country include_domains)
  "Perform a search using the Tavily API and return results as JSON string.
API-KEY is your Tavily API key.
QUERY is the search query string.
Optional SEARCH-DEPTH is either \"basic\" (default) or \"advanced\".
Optional MAX-RESULTS is the maximum number of results (default 5)."
  (require 'plz)
  (let* ((plz-curl-default-args (cons "-k" plz-curl-default-args))
         (url "https://api.tavily.com/search")
         (search-depth (or search-depth "advanced"))
         (max-results (or max-results 1))
         (include_answer t)
         (country (or country "united-states"))
         (include_domains (or include_domains '("nixos.org" "freertos.org" "zephyrproject.org" "
contiki-ng.org" "riot-os.org" "nuttx.apache.org" "mynewt.apache.org" "ziglang.org" "
python.org" "lua.org" "elixir-lang.org" "erlang.org" "haskell.org" "cmake.org" "gnu.
org" "llvm.org" "gcc.gnu.org" "qt.io" "gtk.org" "sdl.org" "libsdl.org" "qemu-project.
org" "cppreference.com" "opensource.org" "ietf.org" "w3.org" "ansi.org" "iso.org" "
ieee.org" "man7.org" "discourse.nixos.org" "ziggit.dev" "emacs-china.org" "lwn.net" "
kernel.org" "sourceware.org" "debian.org" "archlinux.org" "github.com" "osdev.org" "
opencores.org" "riscv.org" "musl-libc.org" "newlib.sourceware.org" "uclibc-ng.org" "
hackaday.com" "raspberrypi.org" "arduino.cc" "espressif.com" "gentoo.org")))
  (request-data
    `(("api_key" . ,tavily-api-key)
      ("query" . ,query)
      ("search_depth" . ,search-depth)
      ("country" . ,country)
      ("include_domains" . ,include_domains)
      ("include_answer" . ,include_answer)
      ("exclude_domains" . ,exclude_domains)
      ("max_results" . ,max-results))))
  (plz 'post url
       :headers '(("Content-Type" . "application/json"))
       :body (json-encode request-data)
       :as 'string
       :then (lambda (result) (funcall callback result)))))

(defun tavily-search (query)
  (interactive "sQuery:")
  (tavily-search-async
    (lambda (result)
```

```
(let ((buf (get-buffer-create "*tavily-search-result*")))
  (switch-to-buffer buf)
  (read-only-mode 0)
  (erase-buffer)
  (org-mode)
  (insert result)
  (insert (tavily-result-to-org result))
  (goto-char (point-min))
  (read-only-mode 1)
  (setq-local truncate-lines nil)
  ))
query))

(defun tavily-result-to-org (json-result)
  (let* ((data (json-read-from-string json-result))
         (results (alist-get 'results data)))
    (mapconcat (lambda (item)
                 (format "* [%s] [%s]\n%s"
                        (alist-get 'url item)
                        (alist-get 'title item)
                        (alist-get 'content item)))
               results
               "\n\n")))
(setq tavily-api-key
      (with-temp-buffer
        (insert-file-contents "/run/secrets/tavily_apikey")
        (buffer-string)))
(use-package gptel
  :init
  (require 'gptel-org)
  :config
  (with-eval-after-load 'gptel
    (gptel-make-tool
      :category "web"
      :name "search"
      :async t
      :function (lambda (cb keyword)
                  (tavily-search-async cb keyword "basic" 5 nil nil nil))
      :description "Search the Internet; If you used any search results, be sure to include the references in your response."
      :args (list '(:name "keyword"
                    :type string
                    :description "The keyword to search")))
    (gptel-make-tool
      :name "create_python_repl"
      :function (lambda ()
                  (run-python nil t)
                  (pop-to-buffer (python-shell-get-buffer))))
      :description "Create a new python repl for this session"
      :args nil
```

3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人公学习失传的武林秘籍一般（只是小说剧情而矣）。

```
:category "emacs")
(gptel-make-tool
:name "send_python_to_repl"
:function (lambda (code)
  (python-shell-send-string code))
:args (list '(:name "code"
  :type string
  :description "python\u2022code\u2022to\u2022execute"))
:description "Send\u2022some\u2022python\u2022code\u2022to\u2022the\u2022python\u2022repl\u2022for\u2022this\u2022session\u2022and\u2022execute\u2022it"
:category "emacs")
(gptel-make-tool
:function (lambda (url)
  (with-current-buffer (url-retrieve-synchronously url)
    (goto-char (point-min)) (forward-paragraph)
    (let ((dom (libxml-parse-html-region (point) (point-max))))
      (run-at-time 0 nil #'kill-buffer (current-buffer))
      (with-temp-buffer
        (shr-insert-document dom)
        (buffer-substring-no-properties (point-min) (point-max))))))
:name "read_url"
:description "Fetch\u2022and\u2022read\u2022the\u2022contents\u2022of\u2022a\u2022URL"
:args (list '(:name "url"
  :type "string"
  :description "The\u2022URL\u2022to\u2022read"))
:category "web")
(gptel-make-tool
:function (lambda (buffer text)
  (with-current-buffer (get-buffer-create buffer)
    (save-excursion
      (goto-char (point-max))
      (insert text)))
    (format "Appended\u2022text\u2022to\u2022buffer\u2022%s" buffer)))
:name "append_to_buffer"
:description "Append\u2022text\u2022to\u2022the\u2022an\u2022Emacs\u2022buffer.\u2022If\u2022the\u2022buffer\u2022does\u2022not\u2022exist,\u2022it\u2022will\u2022be\u2022
created."
:args (list '(:name "buffer"
  :type "string"
  :description "The\u2022name\u2022of\u2022the\u2022buffer\u2022to\u2022append\u2022text\u2022to.")
  '(:name "text"
  :type "string"
  :description "The\u2022text\u2022to\u2022append\u2022to\u2022the\u2022buffer."))
:category "emacs")
(gptel-make-tool
:function (lambda (text)
  (message "%s" text)
  (format "Message\u2022sent:\u2022%s" text))
:name "echo_message"
:description "Send\u2022a\u2022message\u2022to\u2022the\u2022*Messages*\u2022buffer"
:args (list '(:name "text"))
```

```
:type "string"
:description "The text to send to the messages buffer"))

:category "emacs")
(gptel-make-tool
:function (lambda (buffer)
(unless (buffer-live-p (get-buffer buffer))
(error "Error: buffer %s is not live." buffer))
(with-current-buffer buffer
(buffer-substring-no-properties (point-min) (point-max))))
:name "read_buffer"
:description "Return the contents of an Emacs buffer"
:args (list '(:name "buffer"
:type "string"
:description "The name of the buffer whose contents are to be retrieved"))

:category "emacs")
(gptel-make-tool
:function (lambda (directory)
(mapconcat #'identity
(directory-files directory)
"\n"))

:name "list_directory"
:description "List the contents of a given directory"
:args (list '(:name "directory"
:type "string"
:description "The path to the directory to list"))

:category "filesystem")
(gptel-make-tool
:function (lambda (parent name)
(condition-case nil
(progn
(make-directory (expand-file-name name parent) t)
(format "Directory %s created/verified in %s" name parent))
(error (format "Error creating directory %s in %s" name parent)))
:name "make_directory"
:description "Create a new directory with the given name in the specified parent directory"
:args (list '(:name "parent"
:type "string"
:description "The parent directory where the new directory should be created, e.g. /tmp")
'(:name "name"
:type "string"
:description "The name of the new directory to create, e.g. testdir"))

:category "filesystem")
(gptel-make-tool
:function (lambda (path filename content)
(let ((full-path (expand-file-name filename path)))
(with-temp-buffer
(insert content)
(write-file full-path)))
```

```
(format "Created file %s in %s" filename path)))
:name "create_file"
:description "Create a new file with the specified content"
:args (list '(:name "path"
:type "string"
:description "The directory where to create the file")
'(:name "filename"
:type "string"
:description "The name of the file to create")
'(:name "content"
:type "string"
:description "The content to write to the file"))

:category "filesystem")
(gptel-make-tool
:function (lambda (filepath)
(with-temp-buffer
(insert-file-contents (expand-file-name filepath))
(buffer-string)))

:name "read_file"
:description "Read and display the contents of a file"
:args (list '(:name "filepath"
:type "string"
:description "Path to the file to read. Supports relative paths and ~."))

:category "filesystem"))
(defun ant/gptel-save-buffer ()
"Save the current GPTEL buffer with the default directory
set to ~/note."
(interactive)
(let ((default-directory "~/Leere/qingsongliao.github.io/"))
(call-interactively #'save-buffer)))
(defun ant/gptel-load-session ()
"Load a gptel session from ~/notes directory."
(interactive)
(let ((default-directory "~/leetcode/code/"))
(let* ((files (directory-files default-directory t ".+\\" .org$"))
(file (completing-read "Select session file: " files nil t)))
(when file
(find-file file)
(gptel-mode)))))

(setq gptel-default-mode 'org-mode)
(setq deepseek-api-key
(with-temp-buffer
(insert-file-contents "/run/secrets/deepseek_apikey")
(buffer-string)))

(setq gptel-model 'deepseek-chat
gptel-backend (gptel-make-deepseek "deepseek"
:stream t
:key deepseek-api-key))
(require 'url-util)
```

```
(setq gptel-directives
  '(((default . "You\u00e5\u00eare\u00e5\u00eal\u00eage\u00e5\u00language\u00e5\u00model\u00e5\u00living\u00e5\u00in\u00e5\u00Emacs\u00e5\u00and\u00e5\u00a\u00helpful\u00e5\u00assistant.")
    (programming . "You\u00e5\u00eare\u00e5\u00eal\u00eage\u00e5\u00language\u00e5\u00model\u00e5\u00and\u00e5\u00a\u00careful\u00e5\u00programmer.\u00e5\u00Provide\u00e5\u00code\u00e5\u00and\u00e5\u00only\u00e5\u00code\u00e5\u00as\u00e5\u00output\u00e5\u00without\u00e5\u00any\u00e5\u00additional\u00e5\u00text,\u00e5\u00prompt\u00e5\u00or\u00e5\u00note.\u00e5")
    (writing . "You\u00e5\u00eare\u00e5\u00eal\u00e5\u00language\u00e5\u00model\u00e5\u00and\u00e5\u00a\u00writing\u00e5\u00assistant.\u00e5\u00Respond\u00e5\u00concisely.\u00e5")
    (chat . "You\u00e5\u00eare\u00e5\u00eal\u00e5\u00language\u00e5\u00model\u00e5\u00and\u00e5\u00a\u00conversation\u00e5\u00partner.\u00e5\u00Respond\u00e5\u00concisely.\u00e5")
    (bug . "You\u00e5\u00eare\u00e5\u00eal\u00e5\u00language\u00e5\u00model\u00e5\u00and\u00e5\u00a\u00careful\u00e5\u00programmer.\u00e5\u00The\u00e5\u00supplied\u00e5\u00code\u00e5\u00doesn
      't\u00e5\u00work,\u00e5\u00or\u00e5\u00contains\u00e5\u00bugs.\u00e5\u00Describe\u00e5\u00each\u00e5\u00problem\u00e5\u00using\u00e5\u00only\u00e5\u00one\u00e5\u00sentence.\u00e5\u00Provide\u00e5
      fixes\u00e5\u00without\u00e5\u00changing\u00e5\u00the\u00e5\u00old\u00e5\u00behavior.\u00e5")))
  (setq gptel-stream nil))
(use-package consult-gh
  :after consult
  :custom
  (consult-gh-confirm-before-clone nil)
  (consult-gh-default-clone-directory "~/codebase")
  (consult-gh-ask-for-path-before-save nil)
  (consult-gh-default-save-directory "~/codebase")
  (consult-gh-show-preview t)
  (consult-gh-preview-key "C-o")
  (consult-gh-repo-action #'consult-gh--repo-browse-files-action)
  (consult-gh-large-file-warning-threshold 2500000)
  (consult-gh-confirm-name-before-fork nil)
  (consult-gh-notifications-show-unread-only nil)
  (consult-gh-default-interactive-command)
  (consult-gh-prioritize-local-folder nil)
  (consult-gh-issues-state-to-show "all") ; show readmes in their original format
  (consult-gh-group-dashboard-by :reason)
  (consult-gh-repo-preview-major-mode nil) ; show readmes in their original format
  (consult-gh-preview-major-mode 'org-mode) ; use 'org-mode for editing comments, commit messages
  ,,
  :config(consult-gh-enable-default-keybindings)
(use-package consult-gh-forge
  :after consult-gh
  :config
  (consult-gh-forge-mode +1))
(use-package consult-gh-embark
  :after consult-gh
  :config
  (consult-gh-embark-mode +1)
  (setq consult-gh-forge-timeout-seconds 20))
)
```

**Listing 3.2:** the internet capability of me. search/gpt/github and I found they are just trash. use selfhost doc/info/tldr/man/devdocs is way better.

## 第四章 操作系统哲学

### 4.1 我的操作系统特点

1. network, 471mb hosts to block internet out of my pc with dnsmasq and dae proxy tool based on epbf... you know I am ill about internet? because I am in china, and the nsfw/trump/4chan dudes, I love them so much, so I add them into my hosts.
2. emacs editor, + nixos, they both ruin my early adult life by distract me away from reality girls.
3. wm: **sway**, minimal config, pure white look, although **miku-cursor**, now, I use plain mouse instead.
4. shell: **fish+foot**, with all I need in, direnv/many alias/zoxide/git/and thefxxk or thefxxk updated to "chatgpt"'s codex nowadays?
5. browser: **firefox** [default uninstall], I use eww because I hate browser.
6. fully reproducible, even with dict/font/anime wallpaper!
7. zig/linux/ccpp doc in nginx, all kind texinfo can read in emacs.
8. with sops, to store my 0.1 dollar's deepseek account apikey, and chat with my ai girlfriend with my payment password.
9. I am tired of nixos, but I can't leave it, because after using nixos everything is hard to do in other distros, like show off to other distro users "BTW, I use NixOS".

### 4.2 用 linux 只为戒色

翻开中国帝王史，有多少帝王是沉迷女色后国家衰亡，多少帝王是皇太后控制后抑郁早亡。野史里选王要是看他不好色否？当皇帝得看他有无抢夺妇女。翻开网络史发展，有多少人民群众是沉迷情色后国家衰亡，多少人是被色情控制后抑郁早亡，看人得看他好色否？好不好色，得看他懂得怎么预防。oppo 的天气预报能跳到擦边短剧，bing 的结果第一个 BMI 测试广告中跳到黄色导航，百度误输成 baidu.co 中进入黄色应用下载，网易看见过全露的黄色直播，B 站中处处是擦边广告和直播，微博中卖到各种片，知乎里表露情色隐私更引人浏览，我的世界游戏里用色情披风，淘宝里随便看情趣用品，openai 支持成人内容，twitter 全是福利姬，就连 QQ 空间连上擦边广告商，onlyfans 比所有 ai 公司利润更高，pornhub 全球流量前 10，整个 00 后的网络被“这提醒我了”占据，整代女人被商业文化营造黑丝和大胸的“性感”，整代男人被情色文化成为无脑动物与被情欲“冲动”控制，男人就该好色？女人就该这样美吗？天下皆知美之为美，斯恶也。互联网 33% 的流量，70% 的男性，30% 的女性，90% 男性沉迷其中…数字不会说谎，一切已经到来，生存与繁衍不再重要，存在不过是不断刺激感官。不只是色情…网络带来的问题似乎总是和成瘾有关，赌博/购物/社交/游戏…就连互联网最自豪的正面形象所谓的“利于学习”也成为了信息过载的社会巨大问题，自由的代价是什么？自由即是强迫。互联网即是色情。

论我怎么戒色？答：学习技术、理解技术、控制技术，学习自由软件，远离商业软件，拒绝即是自由。

- 成瘾源于：随机 | 不必 | 即时 | 匿名 | 易得 | 免费 | 广告 | 失范
- 通过 **路由器 +NixOS**，不使用手机与笔记本

**时间** 限时，2-5PM，足够一天专注

**速度** 网速，1Mbps，足够文本传输

**空间** 域名，非编程网站，471MB

**强制** 禁 cache.nixos.org 以止系统更新

- 软件使用 **emacs**，不使用浏览器与 IDE

**google** tavily

**chatgpt** gptel

**github** consult-gh

**leetcode** leetgo  
**firefox** eww  
**vpn** dae/dnsmasq  
**obsidian** org  
**wiki** nginx/info/tldr/devdocs/man

- 硬件使用 **eink**，不使用音响与普通键盘

音乐 无音响

游游 无显卡

动漫 无色彩

社媒 无软件

购物 无网络

视频 竖立屏

## 4.3 学技术只为戒 x

- 没有技术，不会 adb，改不了安卓自身的成瘾性。
- 没有技术，不会编辑器，改不了使用 word/browser/editor 的
- 没有技术，有了这样的

# 第五章 键盘流哲学

## 5.1 设计哲学

- 与原生相合 (win)
- 不冲突 (special prefix)
- 合乎习惯 (stay base)
- home row 效率 (home row effiction)
- general in os(browser, editor)

```
| BASE{
  bindings = <
    &none &kp Q  &kp W  &kp F  &kp P  &kp B  &kp J  &kp L  &kp U  &kp Y  &kp SQT &none
    &none &hm LGUI A  &hm LALT R  &hm LCTRLS  &hm LSHIFT T  &kp G  &kp M  &hm RSHIFT N  &hm RCTRL E  &hm RALT I  &hm RGUI O &none
    &none &kp Z  &kp X  &kp C  &kp D  &kp V  &kp K  &kp H  &kp COMMA  &kp DOT  &kp FSLH &none
    &lt MEDIA ESC &lt NAV SPACE &lt MOUSETAB &lt SYM RET &lt NUM BSFC &lt FUN DEL
  >;
};

// copy/down/up/fullscreen/toggle/ cduftabr
Nav{
  bindings = <
    &none &navq  &navw  &navf  &navp  &navb  &navj  &navl  &navu  &navy  &nav_sq &none
    &none &kp LGUI  &kp LALT  &kp LCTRL  &kp LSHIFT  &navg  &navm  &kp LEFT  &kp DOWN  &kp UP  &kp RIGHT &none
    &none &navz  &navx  &navc  &navd  &navv  &kp INSERT  &kp HOME  &kp PG_DN  &kp PG_UP  &kp END &none
    &trans  &trans  &trans  &nav_enter  &nav_delete  &trans
  >;
};

Numbers {
  bindings = <
    &none &kp LBKT  &kp N7  &kp N8  &kp N9  &kp RBKT  &numj  &numl  &numu  &numy  &num_sq &none
    &none &kp SEMI  &kp N4  &kp N5  &kp N6  &kp EQUAL  &numm  &kp RSHIFT &kp RCTRL &kp RALT  &kp RGUI &none
    &none &kp GRAVE  &kp N1  &kp N2  &kp N3  &kp BSLH  &numk  &numh  &num_comma  &num_dot  &num_fslh  &none
    &kp DOT  &kp NO  &kp MINUS  &trans  &trans  &trans
  >;
};

Mouse{
  bindings = <
    &none &mosq  &mosw  &mosf  &mosp  &mosb  &mosj  &mosl  &mosu  &mosy  &mos_sq &none
    &none &mosa  &mosr  &moss  &most  &mosg  &mosm  &mmv MOVE_LEFT  &mmv MOVE_DOWN  &mmv MOVE_UP  &mmv MOVE_RIGHT &none
    &none &mosz  &mosx  &mosc  &mosd  &mosv  &mosw  &mos &msc SCRL_LEFT  &msc SCRL_DOWN  &msc SCRL_UP  &msc SCRL_RIGHT &none
    &trans  &mos_space  &trans  &mkp RCLK &mkp LCLK &mkp MCLK
  >;
};

Symbols {
  bindings = <
    &none &kp LBRC  &kp AMPS  &kp ASTRK  &kp LPAR  &kp RBRC  &just_code  &explain_cn  &think_deep  &search_web  &error_fix  &none
    &none &kp COLON  &kp DLLR  &kp PRCNT  &kp CARET  &kp PLUS  &gpTEL  &kp RSHIFT &kp RCTRL &kp RALT  &kp RGUI &none
    &none &kp TILDE  &kp EXCL  &kp AT  &kp HASH  &kp PIPE  &pass  &id  &qq  &email  &phone &none
    &kp LPAR  &kp RPAR  &kp UNDER  &trans  &trans
  >;
};

Function {
  bindings = <
    &none &kp F12  &kp F7  &kp F8  &kp F9  &kp PRINTSCREEN &none  &kp F18  &kp F19  &kp F20  &kp F21  &none
    &none &kp F11  &kp F4  &kp F5  &kp F6  &kp SCROLLOCK  &kp F17  &kp RSHIFT &kp RCTRL &kp RALT  &kp RGUI  &none
    &none &kp F10  &kp F1  &kp F2  &kp F3  &kp PAUSE_BREAK &none  &kp F13  &kp F14  &kp F15  &kp F16  &none
    &trans  &trans  &trans  &trans  &trans  &trans
  >;
};

media{
  bindings = <
    &none &none  &none  &none  &none  &kp C_REWIND  &none  &none  &none  &kp C_FAST_FORWARD &none
    &none &kp LGUI  &kp LALT  &kp LCTRL  &kp LSHIFT &none  &kp C_PREVIOUS  &kp C_VOL_DN  &kp C_VOLUME_UP  &kp C_NEXT &none
    &none &none  &none  &none  &none  &out OUT_TOG  &bt BT_CLR  &bt BT_SEL0  &none  &none  &none
    &trans  &trans  &trans  &kp C_PLAY_PAUSE  &kp C_STOP  &kp C_MUTE
  >;
};
```

图 5.1: keymap of my keyboard

## 5.2 分层

1. base 层, colemak-dh, 最快速与舒适的英文布局
2. nav 层, 上下左右, 快速移动, C-c C-~ 为前缀, emacs 快速选择
3. num 层, 数字键, 控制 sway 的窗口, 与系统剪切板和应用菜单
4. mos 层, 鼠标上下左右, C-c C-; 为前缀, 一些 emacs 功能。
5. sym 层, 全部符号, gpt prompt 与个人信息快速输入层
6. fun 层, function 1/2.12 层
7. med 层, 提供蓝牙/媒体切换

## 5.3 本地刷录

```
#!/usr/bin/env bash
# ===== CONFIGURATION =====
GITHUB_OWNER="NestorLiao"
GITHUB_REPO="zmk-config"
GITHUB_TOKEN=$(cat ~/.github_token)
ZIP_DEST="zmk_build.zip"
UNZIP_DIR="zmk_build"

LEFT_FW="corne_left-nice_nano_v2-zmk.uf2"
RIGHT_FW="corne_right-nice_nano_v2-zmk.uf2"
SETTINGS_RESET="settings_reset-nice_nano_v2-zmk.uf2"

# ===== FUNCTIONS =====

flash_settings_reset() {
    echo "Flashing settings reset firmware..."
    for i in 1 2; do
        echo "Reset device $i/2 into bootloader mode..."
        # Wait for mount
        while [ ! -d "/media/NICENANO" ]; do
            sleep 0.5
        done
        echo "NICENANO detected. Flashing settings reset..."
        if [ -f "$SETTINGS_RESET" ]; then
            cp "$SETTINGS_RESET" /media/NICENANO/
            echo "Settings reset flashed $i/2."
        else
            echo " $SETTINGS_RESET not found in current directory."
            exit 1
        fi
    echo "Waiting for NICENANO to unmount..."
```

```

while [ -d "/media/NICENANO" ]; do
    sleep 0.5
done
echo " Settings reset completed twice."
}

download_latest_artifact() {
    echo "Fetching latest artifact..."

run_id=$(curl -s -H "Authorization: token $GITHUB_TOKEN" \
"https://api.github.com/repos/$GITHUB_OWNER/$GITHUB_REPO/actions/runs?per_page=1&status=success" \
" \
| jq -r '.workflow_runs[0].id')

if [ "$run_id" == "null" ] || [ -z "$run_id" ]; then
    echo " No successful workflow run found."
    exit 1
fi

echo "Found workflow run ID: $run_id"

artifact_url=$(curl -s -H "Authorization: token $GITHUB_TOKEN" \
"https://api.github.com/repos/$GITHUB_OWNER/$GITHUB_REPO/actions/runs/$run_id/artifacts" \
| jq -r '.artifacts[0].archive_download_url')

if [ "$artifact_url" == "null" ] || [ -z "$artifact_url" ]; then
    echo " No artifacts found for run $run_id."
    exit 1
fi

echo "Downloading artifact zip..."
curl -L -H "Authorization: token $GITHUB_TOKEN" \
"$artifact_url" -o "$ZIP_DEST"

echo "Unzipping..."
rm -rf "$UNZIP_DIR"
unzip -q "$ZIP_DEST" -d "$UNZIP_DIR"
}

flash_from_local() {
    if [ ! -f "$ZIP_DEST" ]; then
        echo " $ZIP_DEST not found in current directory."
        exit 1
    fi

    echo "Using local $ZIP_DEST..."
    rm -rf "$UNZIP_DIR"
    unzip -q "$ZIP_DEST" -d "$UNZIP_DIR"
}

```

```

}

flash_firmware() {
    local fw_path="$1"
    echo "Please reset the device into bootloader mode..."

    while [ ! -d "/media/NICENANO" ]; do
        sleep 0.5
    done

    echo "NICENANO detected. Flashing $fw_path..."
    cp "$fw_path" /media/NICENANO/
    echo "Done."

    echo "Waiting for NICENANO to unmount..."
    while [ -d "/media/NICENANO" ]; do
        sleep 0.5
    done
}

flash_left_right() {
    flash_firmware "$UNZIP_DIR/$LEFT_FW"
    flash_firmware "$UNZIP_DIR/$RIGHT_FW"
}

show_usage() {
    echo "Usage: $0 <mode>"
    echo "Modes:"
    echo "  1 - Flash settings reset twice"
    echo "  2 - Download from GitHub and flash left/right"
    echo "  3 - Use local zmk_build.zip to flash left/right"
}

# ===== MAIN =====

if [ $# -ne 1 ]; then
    show_usage
    exit 1
fi

case $1 in
    1)
        echo "== Mode 1: Flash Settings Reset =="
        flash_settings_reset
        ;;
    2)
        echo "== Mode 2: Download and Flash =="
        download_latest_artifact
        flash_left_right
    esac
}

```

```

;;
3)
echo "==Mode=3:LocalFlash=="
flash_from_local
flash_left_right
;;
*)
echo " Invalid mode:$1"
show_usage
exit 1
;;
esac

echo " All done."

```

**Listing 5.1:** 1 刷重置, 2 刷 github action, 3 刷本地 zip 文件

## 5.4 不只键盘

**Absolute Enable Right Click & Copy** Force Enable Right Click & Copy.

**Authenticator** Authenticator generates two-factor authentication codes in your browser.

**Bar Breaker** Hides fixed headers and footers on pages you visit.

**ChatGPT Ctrl+Enter Sender** Use 'Ctrl+Enter' for sending messages in AI chat services like ChatGPT. Prevents accidental sends and allows line breaks with Enter.

**ChatGPT Disable Auto Scroll - FREE** Stop annoying scrolling in ChatGPT. Prevent jerky animation of appearing text.

**ClearURLs** Remove tracking elements from URLs.

**Decentraleyes** Protects you against tracking through "free", centralized, content delivery.

**Default links not to be underlined** Restores non-underlined hyperlinks by default (setting text-decoration: none).

**Disconnect** Make the web faster, more private, and more secure.

**Enforce Safe Search (=Adult Filter)** Toggles the built-in filter on many search engines (see

**Focus On Left Tab After Closing** When a current tab is closed, the left tab (or the right tab on RTL

**Font Contrast** Improves webpage readability.

**FuckCSDN** Clean CSDN's limitation scripts

**Google Search Ad Remover And Customizer** This extension gives you the ability to customize how your Google search results

**Hide shorts for Youtube™** Hides shorts from YouTube™ from home page, subscriptions and search results.

**hide-scrollbars** Hides page scrollbars!

**I still don't care about cookies** Community version of the popular extension "I don't care about cookies"

**Invert Colors** A simple add-on that inverts the page colors.

**No Emoji** Browser extension to remove emoji.

**Shut Up: Comment Blocker** Blocks comment sections on many popular websites.

**Stack Copy Button** A copy button for Stack Overflow code boxes

**Tab Sidebar** Adds a sidebar with foldable tabs.

**uBlock Origin** Finally, an efficient blocker. Easy on CPU and memory.

**Unhook - Remove YouTube Recommended & Shorts** Hide YouTube related videos, shorts, comments, suggestions wall, homepage

**Vimium C - All by Keyboard** A keyboard shortcut tool for keyboard-based page navigation and browser tab

第六章 屏幕哲学

Real Mono Theme, two colors are enough for emacs.

## 6.1 Feels:

A collection of real monochrome emacs themes in a couple of variants.

```

/home/lameci/serveur/real-mono-here/HEAD - master initial commit
image
real-mono-dark.png
real-mono-light.png
real-mono-old.png
README.md
real-mono-dark-theme.clj
real-mono-dark-theme.d
real-mono-dark-theme.html
real-mono-old-theme.clj
real-mono-old-theme.d
real-mono-old-theme.html
real-mono-themes.clj
real-mono-themes.d
real-mono-themes.html

Untracked files (7):
  README.md
  real-mono-dark-theme.clj
  real-mono-dark-theme.d
  real-mono-dark-theme.html
  real-mono-old-theme.clj
  real-mono-old-theme.d
  real-mono-old-theme.html

diff --git a/README.md b/README.md
--- /dev/null
+++ b/README.md
@@ -1,52 +1,102 @@
Real mono themes
=====
Just A Real Mono Theme
+Real Mono Theme, two colors are
+used.
+Font:
+A collection of real monochrome cs
+
+Font:
+real-mono-dark-theme.clj
+real-mono-dark-theme.d
+real-mono-dark-theme.html
+real-mono-old-theme.clj
+real-mono-old-theme.d
+real-mono-old-theme.html
+real-mono-themes.clj
+real-mono-themes.d
+real-mono-themes.html

@details
+We are given two numbers, a and b
+and we have to flip flop to convert it to
+
+Explanation:
+ $A = 010101 \times 10100$ 
+As we can see, the bits of A that are
+1 has three bits, we set 10100
+
+@Pattern:
+1. "Not Greyscale", no blur anymore
+2. "Not Greyscale", no blur anymore
+3. "Only" to customize, can set the
+4. "Interaction free", no scroll by

```

pictures's font list: bookerly, ubuntumono, firacode, terminess, bookerly bold italic.

## 6.2 Features:

1. **Not Greyscale**, no blur anymore, it's much better to use eink screen for pure black and white!
2. **Easy** to customize, can set the only two colors by config the default face's foreground/background color.
3. **Distraction-free**, no extra info-overwhelming causing by font-lock, only few font diversity in magit/dired etc for better function recognize.
4. **Full**, configed 370+ faces, I didn't see any monochrome theme can reach that much, as my daily driver, it's good enough.

## 6.3 Tips for mono:

- (global-hide-mode-line-mode 1), build your own brain memory
- (no-emoji 1), alter emacs to be "text editor" instead of discord
- (show-paren-mode -1), build your own eye insight
- (window-divider-mode -1), too, build your memory
- (display-line-numbers-mode -1), too, build your own eye insight

```
(setq hl-todo-keyword-faces
  '(("HOLD" . "#000000")
    ("TODO" . "#000000")
    ("NEXT" . "#000000")
    ("THEM" . "#000000")
    ("PROG" . "#000000")
    ("OKAY" . "#000000")
    ("DONT" . "#000000")
    ("FAIL" . "#000000")
    ("DONE" . "#000000")
    ("NOTE" . "#000000")
    ("MAYBE" . "#000000")
    ("KLUDGE" . "#000000")
    ("HACK" . "#000000")
    ("TEMP" . "#000000")
    ("FIXME" . "#000000")
    ("XXXX*" . "#000000")))

(defvar my-alternate-font "-DAMA-UbuntuMono_Nerd_Font-regular-normal-normal--13----m-0-iso10646-1"
)
```

```
(defvar my-default-font "bookerly")
(defvar fontfont 1)
(defun my-toggle-font ()
  "Toggle between UbuntuMono and bookerly fonts."
  (interactive)
  (if (= fontfont 1)
      (progn (set-face-attribute 'default nil :font my-default-font :height 160) (setq fontfont 0))
      (progn (set-face-attribute 'default nil :font my-alternate-font :height 210) (setq fontfont 1))))
```

- fringe specific mode auto hide

```
(defun my-set-fringe-face ()
  "auto-hide-fringe-face depending on major mode."
  (if (derived-mode-p '(occur-mode gud-mode))
      (set-face-attribute 'fringe nil
                         :background (face-attribute 'default :background)
                         :foreground (face-attribute 'default :foreground))
      (set-face-attribute 'fringe nil
                         :background (face-attribute 'default :background)
                         :foreground (face-attribute 'default :background))))
(add-hook 'after-change-major-mode-hook #'my-set-fringe-face)
```

- elisp for toggling paperlike-hd to switch between read and watch.

```
(defvar monitor-state 1
  "Current monitor state, either 0 for read or 1 for watch.")
(defun monitor ()
  "switch monitor from read mode to watch mode"
  (interactive)
  (let ((monitorprotocol "-i2c")
        (monitorpath "/dev/i2c-4")
        (monitorcli "paperlike-cli")
        (monitorarg '("-contrast" "-speed" "-mode" "--clear"))
        (mode-state '(("9" "5" "1") ("9" "5" "1"))))
    (split-window-below)
    (other-window 1)
    (switch-to-buffer "*Shell_Command_Output*")
    (split-window-below)
    (other-window 1)
    (switch-to-buffer "*Async_Shell_Command*")
    (progn
      (dotimes (number 3)
        (call-process monitorcli nil nil nil
                      monitorprotocol
                      monitorpath
                      (car (nthcdr number monitorarg))
                      (car (nthcdr number (car (nthcdr monitor-state mode-state)))))))
      (sleep-for 1))
    (setq monitor-state (if (= 0 monitor-state) 1 0)))
    (sleep-for 1.5)
    (sleep-for 0.5))
```

```
(call-process monitorcli nil nil nil nil monitorprotocol monitorpath (car (nthcdr 3 monitorarg)))
(sleep-for 0.5)
(donothing))
```

# 第七章 fog.h

Copied from tsoding's flag module: <https://github.com/tsoding/flag.h> Also tsoding's nob as build system: <https://github.com/tsoding/nob.h>

## 7.1 Quick Start

Check `example.c`

```
cc -o example example.c
./example -help
```

## 7.2 "Just rewrite shit in C, Just rewrite shit in C" — tsoding

the editor/os/term of mine all just written in c.

you can replace the 1995's hype langs with:

- rust -> fin-c.h
- go -> threads.h
- python -> dll
- zig -> nob.h
- c++ -> C with class
- web -> C compile to webassembly
- do leetcode -> C with stl
- do github -> check tsoding
- do emacs -> C + elisp
- do linux -> C with fake class

# 第八章 不用 AI 完成 Leetcode 刷题

## 内容提要

- |   |                          |
|---|--------------------------|
| □ 开始时间: 2025-11-20 2025 年 12 月          | □ 需要小时: 100000000000000+ |
| □ 结束时间: 2025-12 月前 2026 年前              | □ 本章要点: 关在家里, 把力扣算法题热题都  |
| □ 完成难度: difficult, 对于懒癌而言: “impossible” | 刷完一遍…                    |

不用 AI 完成 Leetcode 刷题是一件很爽的事, 虽然产生的经济效应为零, 虽然永远写代码写不过 AI, 但是…人类还需要人类? 不知资本家怎么想, 但有搜索引擎了, 就不用背书了吗, 有炒菜机和机车就不用做菜和走路了吗? 得之心而应于手, 理解一件事情本身有它的价值, 虽然我更想去种田/木工/画画/乐器/写作, 但…我得先有工作啊~

```
(global-set-key (kbd "M-*") (lambda () (interactive) (my/leetcode-open (string-to-number(current-word)))))  
;; 谁tm知道到底我是折腾emacs节约时间还是浪费时间, 不过有这个玩意, 玩emacs总感觉还是挺值了。
```

根据力扣刷题指南上所提到的题目, 收集了 175 题, 一题 40 分钟, 大约需要 116 小时完成

- 最易懂的贪心算法 1 例子: 455 135 435 真题: 605 452 763 122 406 665
- 玩转双指针 2 例子: 167 88 76 142 真题: 633 680 340
- 居合斩! 二分查找 3 例子: 69 34 162 81 真题: 154 540 4
- 千奇百怪的排序算法 4 例子: 215 347 真题: 451 75
- 一切皆可搜索 5 例子: 695 547 417 46 77 79 51 1091 934 126 真题: 130 257 47 40 37 310
- 深入浅出动态规划 6 例子: 70 198 413 64 542 221 279 91 139 1105 377 300 1143 416 474 322 72 650 121 188 309 真题: 213 53 343 583 646 10 376 494 714
- 化繁为简的分治法 7 例子: 241 真题: 932 312 168 67 238 462 169 470 202
- 神奇的位运算 9 例子: 461 190 136 318 338 真题: 268 693 476 260
- 妙用数据结构 10 例子: 448 48 240 769 232 155 20 739 23 218 239 1 128 149 332 303 304 真题: 566 225 503 217 697 594 15 287 313 870 307
- 令人头大的字符串 11 例子: 242 205 647 696 1249 227 28 真题: 409 3 772 5 83 328 19 148
- 指针三剑客之二: 树 13 例子: 104 110 543 437 101 1110 637 105 144 99 669 208 真题: 226 617 572 404 513 538 235 530 1530 889 106 94 145 236 109 897 653 450
- 指针三剑客之三: 图 14 例子: 785 210 真题: 1059 1135 882
- 更加复杂的数据结构 15 例子: 684 146 380 真题: 1135 432 716

## 8.1 吃饭/做题/睡觉, 论计算机技术革命及其后果

## 8.2 1. Two Sum (Easy)

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.

**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:**

- $2 < \text{nums.length} \leq 10^4$ =
- $-10^9 < \text{nums}[i] \leq 10^9$ =
- $-10^9 < \text{target} \leq 10^9$ =
- **Only one valid answer exists.**

**Follow-up:** Can you come up with an algorithm that is less than  $O(n^2)$  time complexity?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/two-sum/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> res;
        unordered_map<int, int> ht;
        for (int i = 0; i < nums.size(); i++) {
            int other = target - nums[i];
            if (ht.count(other)) {
                /* Only one solution for purpose of this problem */
                res.append(ht[other]);
                res.append(i);
                return res;
            }
            ht[nums[i]] = i;
        }
        return res;
    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int target;
    LeetCodeIO::scan(cin, target);

    Solution *obj = new Solution();
    auto res = obj->twoSum(nums, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

### 8.3 3. Longest Substring Without Repeating Characters (Medium)

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

#### Example 1:

Input:  $s = \text{"abcabcbb"}$

Output: 3

Explanation: The answer is "abc", with the length of 3. Note that "bca" and "cab" are also correct answers.

#### Example 2:

Input:  $s = \text{"bbbbbb"}$

Output: 1

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

#### Example 3:

Input:  $s = \text{"pwwkew"}$

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:

- $0 < s.length \leq 5 * 10^4$
- $s$  consists of English letters, digits, symbols and spaces.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/longest-substring-without-repeating-characters/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int lengthOfLongestSubstring(string s) {
        vector<int> count(256);
        long unsigned int len = 0;
        long unsigned int i, j;

        for (i = 0, j = 0; i < s.length(); i++) {
            count[s[i]]++;
            while (count[s[i]] > 1) {
                len = i - j > len ? i - j : len;
                count[s[j++]] -= 1;
            }
        }

        return i - j > len ? i - j : len;
    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->lengthOfLongestSubstring(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.4 4. Median of Two Sorted Arrays (Hard)

Given two sorted arrays `nums1` and `nums2` of size  $m$  and  $n$  respectively, return **the median** of the two sorted arrays. The overall run time complexity should be  $O(\log(m+n))$ .

### Example 1:

Input: `nums1 = [1,3]`, `nums2 = [2]`

Output: 2.00000

Explanation: merged array = [1,2,3] and median is 2.

### Example 2:

Input: `nums1 = [1,2]`, `nums2 = [3,4]`

Output: 2.50000

Explanation: merged array = [1,2,3,4] and median is  $(2 + 3) / 2 = 2.5$ .

### Constraints:

- `nums1.length = m=`
- `nums2.length = n=`
- `0 < m <= 1000=`
- `0 < n <= 1000=`
- `1 < m + n <= 2000=`
- `-10 < nums1[i], nums2[i] <= 106=`

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/median-of-two-sorted-arrays/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums1;
    LeetCodeIO::scan(cin, nums1);
    vector<int> nums2;
    LeetCodeIO::scan(cin, nums2);

    Solution *obj = new Solution();
    auto res = obj->findMedianSortedArrays(nums1, nums2);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.5 5. Longest Palindromic Substring (Medium)

Given a string  $s$ , return the longest palindromic substring in  $s$ .

### Example 1:

Input:  $s = "babad"$

Output: "bab"

Explanation: "aba" is also a valid answer.

### Example 2:

Input:  $s = "cbbd"$

Output: "bb"

### Constraints:

- $1 < s.length \leq 1000$
- $s$  consists of only digits and English letters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/longest-palindromic-substring/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    string longestPalindrome(string s) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->longestPalindrome(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.6 10. Regular Expression Matching (Hard)

Given an input string  $s$  and a pattern  $p$ , implement regular expression matching with support for  $'.'$  and  $'*''$  where:

- $'.'$  Matches any single character.
- $'*''$  Matches zero or more of the preceding element.

The matching should cover the **entire** input string (not partial).

### Example 1:

Input:  $s = "aa"$ ,  $p = "a"$

Output: false

Explanation: "a" does not match the entire string "aa".

### Example 2:

Input:  $s = "aa"$ ,  $p = "a*"$

Output: true

Explanation: '\*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

### Example 3:

Input:  $s = "ab"$ ,  $p = ".*"$

Output: true

Explanation: ".\*" means "zero or more (\*) of any character (.)".

### Constraints:

- $1 < s.length \leq 20$ =
- $1 < p.length \leq 20$ =
- $s$  contains only lowercase English letters.
- $p$  contains only lowercase English letters,  $'.'$ , and  $'*''$ .
- It is guaranteed for each appearance of the character  $'*''$ , there will be a previous valid character to match.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/regular-expression-matching/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isMatch(string s, string p) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);
    string p;
    LeetCodeIO::scan(cin, p);

    Solution *obj = new Solution();
    auto res = obj->isMatch(s, p);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.7 15. 3Sum (Medium)

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

Notice that the solution set must not contain duplicate triplets.

### 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:

- $3 < \text{nums.length} \leq 3000$
- $-10^5 < \text{nums}[i] \leq 10^5$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/3sum/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

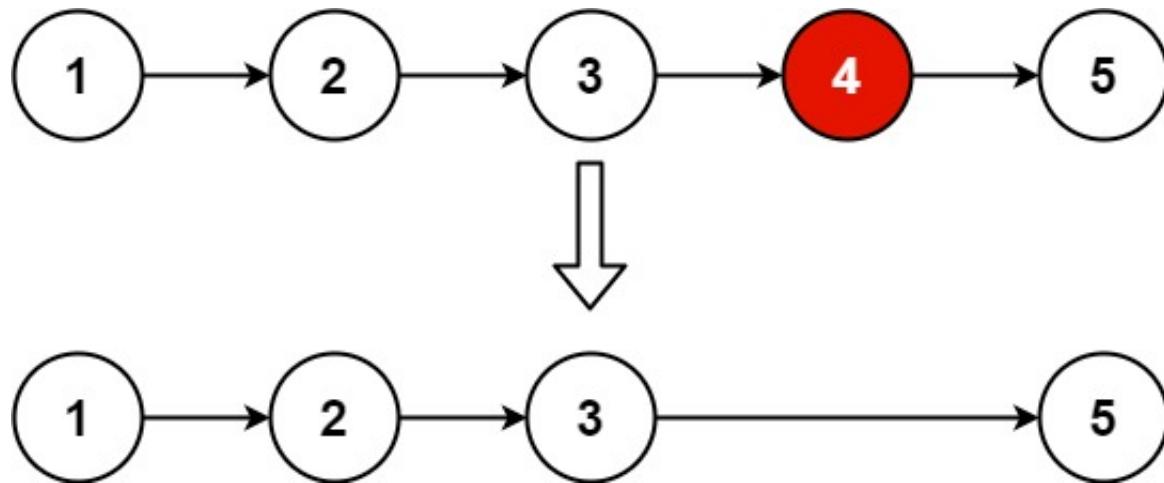
    Solution *obj = new Solution();
    auto res = obj->threeSum(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.8 19. Remove Nth Node From End of List (Medium)

Given the head of a linked list, remove the  $n$  node from the end of the list and return its head.

**Example 1:**



Input: `head = [1,2,3,4,5]`,  $n = 2$

Output: `[1,2,3,5]`

**Example 2:**

Input: `head = [1]`,  $n = 1$

Output: `[]`

**Example 3:**

Input: `head = [1,2]`,  $n = 1$

Output: `[1]`

**Constraints:**

- The number of nodes in the list is  $sz$ .
- $1 < sz \leq 30$
- $0 < \text{Node.val} \leq 100$
- $1 < n \leq sz$

**Follow up:** Could you do this in one pass?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/remove-nth-node-from-end-of-list/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    ListNode* removeNthFromEnd(ListNode* head, int n) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    ListNode* head;
    LeetCodeIO::scan(cin, head);
    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->removeNthFromEnd(head, n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.9 20. Valid Parentheses (Easy)

Given a string  $s$  containing just the characters ' $($ ', ' $)$ ', ' $\{$ ', ' $\}$ ', ' $[$ ' and ' $]$ ', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

**Example 1:**

**Input:**  $s = "()$ "

**Output:** true

**Example 2:**

**Input:**  $s = "[]{}()$ "

**Output:** true

**Example 3:**

**Input:**  $s = "([)]"$

**Output:** false

**Example 4:**

**Input:**  $s = "([])"$

**Output:** true

**Example 5:**

**Input:**  $s = "(())"$

**Output:** false

**Constraints:**

- $1 < s.length \leq 10^4$
- $s$  consists of parentheses only ' $($   $)$   $[$   $]$   $\{$   $\}$ '.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/valid-parentheses/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isValid(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->isValid(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.10 23. Merge k Sorted Lists (Hard)

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.

### Example 1:

`Input: lists = [[1,4,5],[1,3,4],[2,6]]`

`Output: [1,1,2,3,4,4,5,6]`

`Explanation: The linked-lists are:`

```
[  
    1->4->5,  
    1->3->4,  
    2->6  
]  
merging them into one sorted linked list:  
1->1->2->3->4->4->5->6
```

### Example 2:

`Input: lists = []`

`Output: []`

### Example 3:

`Input: lists = [[]]`

`Output: []`

### Constraints:

- $k = \text{lists.length} =$
- $0 < k \leq 10^4 =$
- $0 < \text{lists}[i].length \leq 500 =$
- $-10 < \text{lists}[i][j] \leq 10^4 =$
- `lists[i]` is sorted in **ascending order**.
- The sum of `lists[i].length` will not exceed 10 .

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/merge-k-sorted-lists/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<ListNode*> lists;
    LeetCodeIO::scan(cin, lists);

    Solution *obj = new Solution();
    auto res = obj->mergeKLists(lists);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.11 28. Find the Index of the First Occurrence in a String (Easy)

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

### Example 1:

Input: `haystack = "sadbutsad"`, `needle = "sad"`

Output: `0`

Explanation: "sad" occurs at index 0 and 6.

The first occurrence is at index 0, so we return 0.

### Example 2:

Input: `haystack = "leetcode"`, `needle = "leeto"`

Output: `-1`

Explanation: "leeto" did not occur in "leetcode", so we return -1.

### Constraints:

- $1 < \text{haystack.length}, \text{needle.length} \leq 10^4$
- `haystack` and `needle` consist of only lowercase English characters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/find-the-index-of-the-first-occurrence-in-a-string/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int strStr(string haystack, string needle) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string haystack;
    LeetCodeIO::scan(cin, haystack);
    string needle;
    LeetCodeIO::scan(cin, needle);

    Solution *obj = new Solution();
    auto res = obj->strStr(haystack, needle);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.12 34. Find First and Last Position of Element in Sorted Array (Medium)

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given `target` value.

If `target` is not found in the array, return `[-1, -1]`.

You must write an algorithm with  $O(\log n)$  runtime complexity.

**Example 1:**

Input: `nums = [5,7,7,8,8,10]`, `target = 8`

Output: `[3,4]`

**Example 2:**

Input: `nums = [5,7,7,8,8,10]`, `target = 6`

Output: `[-1,-1]`

**Example 3:**

Input: `nums = []`, `target = 0`

Output: `[-1,-1]`

**Constraints:**

- $0 < \text{nums.length} \leq 10^5$
- $-10^9 < \text{nums}[i] \leq 10^9$
- `nums` is a non-decreasing array.
- $-10^9 < \text{target} \leq 10^9$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> searchRange(vector<int>& nums, int target) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int target;
    LeetCodeIO::scan(cin, target);

    Solution *obj = new Solution();
    auto res = obj->searchRange(nums, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.13 37. Sudoku Solver (Hard)

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy **all of the following rules**:

1. Each of the digits 1–9 must occur exactly once in each row.
2. Each of the digits 1–9 must occur exactly once in each column.
3. Each of the digits 1–9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.

The ' .' character indicates empty cells.

**Example 1:**

Input: board =

```
[["5","3",".",".","7",".",".",".","."],["6",".",".","1","9","5",".","."],[],"9","8",".",".",".","]
```

Output:

```
[["5","3","4","6","7","8","9","1","2"],[["6","7","2","1","9","5","3","4","8"],[["1","9","8","3","4","2","5","6","7"],[
```

Explanation: The input board is shown above and the only valid solution is shown below:

### Constraints:

- `board.length = 9=`
- `board[i].length = 9=`
- `board[i] [j]` is a digit or ' .'.
- It is **guaranteed** that the input board has only one solution.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sudoku-solver/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    void solveSudoku(vector<vector<char>>& board) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<char>> board;
    LeetCodeIO::scan(cin, board);

    Solution *obj = new Solution();
    auto res = obj->solveSudoku(board);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.14 40. Combination Sum II (Medium)

Given a collection of candidate numbers ( candidates) and a target number ( target), find all unique combinations in candidates where the candidate numbers sum to target.

Each number in candidates may only be used **once** in the combination.

**Note:** The solution set must not contain duplicate combinations.

### Example 1:

Input: candidates = [10,1,2,7,6,1,5], target = 8

Output:

```
[  
[1,1,6],  
[1,2,5],  
[1,7],  
[2,6]
```

]

### Example 2:

Input: candidates = [2,5,2,1,2], target = 5

Output:

```
[  
[1,2,2],  
[5]
```

### Constraints:

- $1 < \text{candidates.length} \leq 100$
- $1 < \text{candidates}[i] \leq 50$
- $1 < \text{target} \leq 30$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/combination-sum-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> candidates;
    LeetCodeIO::scan(cin, candidates);
    int target;
    LeetCodeIO::scan(cin, target);

    Solution *obj = new Solution();
    auto res = obj->combinationSum2(candidates, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.15 46. Permutations (Medium)

Given an array `nums` of distinct integers, return all the possible permutations. You can return the answer in **any order**.

**Example 1:**

Input: `nums = [1,2,3]`

Output: `[[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1]]`

**Example 2:**

Input: `nums = [0,1]`

Output: `[[0,1], [1,0]]`

**Example 3:**

Input: `nums = [1]`

Output: `[[1]]`

**Constraints:**

- $1 < \text{nums.length} \leq 6$
- $-10 \leq \text{nums}[i] \leq 10$
- All the integers of `nums` are **unique**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/permutations/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> permute(vector<int>& nums) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->permute(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.16 47. Permutations II (Medium)

Given a collection of numbers, `nums`, that might contain duplicates, return all possible unique permutations **in any order**.

**Example 1:**

Input: `nums = [1,1,2]`

Output:

```
[[1,1,2],  
 [1,2,1],  
 [2,1,1]]
```

**Example 2:**

Input: `nums = [1,2,3]`

Output: `[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`

**Constraints:**

- $1 < \text{nums.length} \leq 8$
- $-10 \leq \text{nums}[i] \leq 10$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/permutations-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> permuteUnique(vector<int>& nums) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

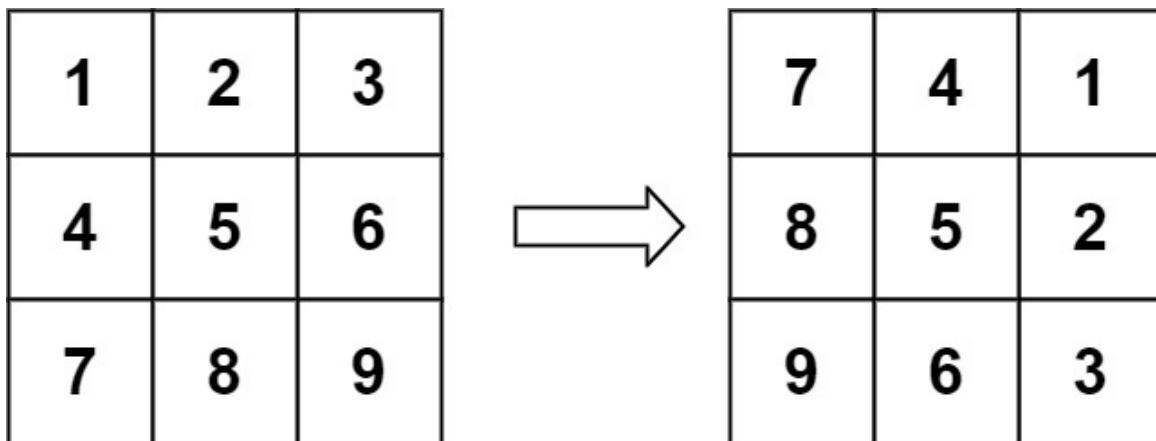
    Solution *obj = new Solution();
    auto res = obj->permuteUnique(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.17 48. Rotate Image (Medium)

You are given an  $n \times n$  2D matrix representing an image, rotate the image by **90** degrees (clockwise). You have to rotate the image **in-place**, which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

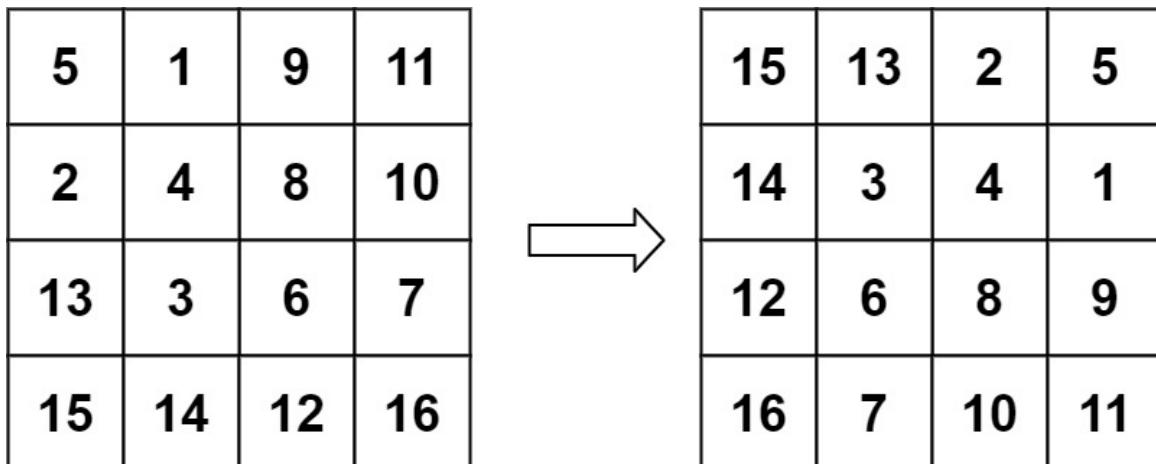
**Example 1:**



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

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

**Example 2:**



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:

- $n = \text{matrix.length} = \text{matrix}[i].\text{length}$
- $1 < n \leq 20$
- $-1000 < \text{matrix}[i][j] \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/rotate-image/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    void rotate(vector<vector<int>>& matrix) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> matrix;
    LeetCodeIO::scan(cin, matrix);

    Solution *obj = new Solution();
    auto res = obj->rotate(matrix);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

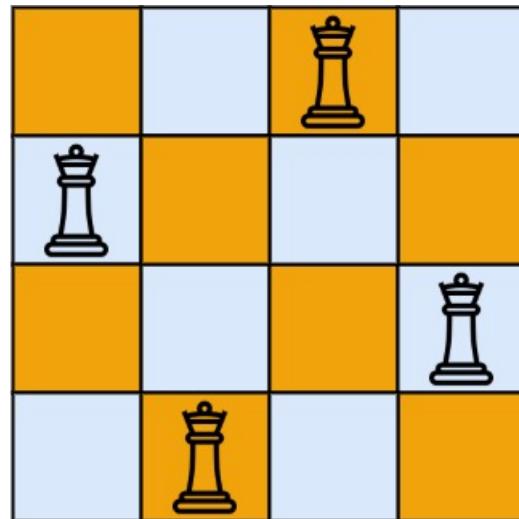
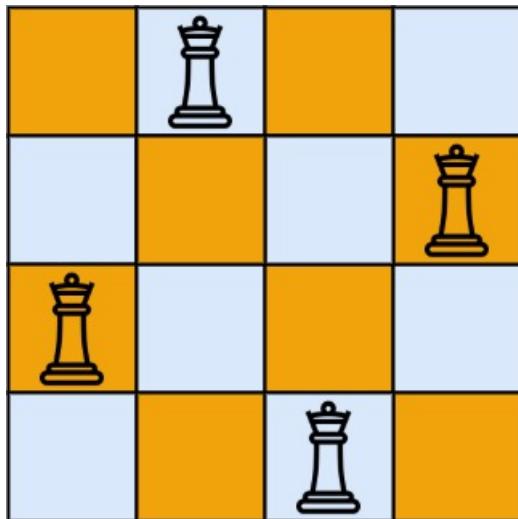
    delete obj;
    return 0;
}
```

## 8.18 51. N-Queens (Hard)

The **n-queens** puzzle is the problem of placing  $n$  queens on an  $n \times n$  chessboard such that no two queens attack each other.

Given an integer  $n$ , return all distinct solutions to the **n-queens puzzle**. You may return the answer in **any order**. Each solution contains a distinct board configuration of the  $n$ -queens' placement, where 'Q' and '.' both indicate a queen and an empty space, respectively.

**Example 1:**



Input:  $n = 4$

Output: `[["Q..", "...Q", "Q...","..Q."], ["..Q.", "Q...","...Q",".Q.."]]`

Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above

**Example 2:**

Input:  $n = 1$

Output: `[["Q"]]`

**Constraints:**

- $1 < n \leq 9$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/n-queens/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<string>> solveNQueens(int n) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->solveNQueens(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.19 53. Maximum Subarray (Medium)

Given an integer array `nums`, find the subarray with the largest sum, and return its sum.

### Example 1:

Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`

Output: 6

Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.

### Example 2:

Input: `nums = [1]`

Output: 1

Explanation: The subarray `[1]` has the largest sum 1.

### Example 3:

Input: `nums = [5,4,-1,7,8]`

Output: 23

Explanation: The subarray `[5,4,-1,7,8]` has the largest sum 23.

### Constraints:

- $1 < \text{nums.length} \leq 10^5$
- $-10 \leq \text{nums}[i] \leq 10^4$

**Follow up:** If you have figured out the  $O(n)$  solution, try coding another solution using the **divide and conquer** approach, which is more subtle.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/maximum-subarray/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxSubArray(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->maxSubArray(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.20 64. Minimum Path Sum (Medium)

Given a  $m \times n$  grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.

**Note:** You can only move either down or right at any point in time.

**Example 1:**

1	3	1
1	5	1
4	2	1

Input: `grid = [[1,3,1],[1,5,1],[4,2,1]]`

Output: 7

Explanation: Because the path  $1 \rightarrow 3 \rightarrow 1 \rightarrow 1 \rightarrow 1$  minimizes the sum.

**Example 2:**

Input: `grid = [[1,2,3],[4,5,6]]`

Output: 12

**Constraints:**

- $m = \text{grid.length} =$

- $n = \text{grid}[i].length =$
- $1 < m, n \leq 200 =$
- $0 < \text{grid}[i][j] \leq 200 =$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/minimum-path-sum/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int minPathSum(vector<vector<int>>& grid) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> grid;
    LeetCodeIO::scan(cin, grid);

    Solution *obj = new Solution();
    auto res = obj->minPathSum(grid);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.21 67. Add Binary (Easy)

Given two binary strings  $a$  and  $b$ , return their sum as a binary string.

### Example 1:

Input:  $a = "11"$ ,  $b = "1"$

Output: "100"

### Example 2:

Input:  $a = "1010"$ ,  $b = "1011"$

Output: "10101"

### Constraints:

- $1 < a.length, b.length \leq 10^4$
- $a$  and  $b$  consist only of '0' or '1' characters.
- Each string does not contain leading zeros except for the zero itself.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/add-binary/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    string addBinary(string a, string b) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string a;
    LeetCodeIO::scan(cin, a);
    string b;
    LeetCodeIO::scan(cin, b);

    Solution *obj = new Solution();
    auto res = obj->addBinary(a, b);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.22 69. Sqrt(x) (Easy)

Given a non-negative integer  $x$ , return the square root of  $x$  rounded down to the nearest integer. The returned integer should be **non-negative** as well.

You **must not use** any built-in exponent function or operator.

- For example, do not use `pow(x, 0.5)` in c++ or `x ** 0.5` in python.

**Example 1:**

Input:  $x = 4$

Output: 2

Explanation: The square root of 4 is 2, so we return 2.

**Example 2:**

Input:  $x = 8$

Output: 2

Explanation: The square root of 8 is 2.82842..., and since we round it down to the nearest integer, 2 is returned.

**Constraints:**

- $0 < x \leq 2^{31} - 1$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sqrtx/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int mySqrt(int x) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int x;
    LeetCodeIO::scan(cin, x);

    Solution *obj = new Solution();
    auto res = obj->mySqrt(x);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.23 70. Climbing Stairs (Easy)

You are climbing a staircase. It takes  $n$  steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

### Example 1:

Input:  $n = 2$

Output: 2

Explanation: There are two ways to climb to the top.

1. 1 step + 1 step
2. 2 steps

### Example 2:

Input:  $n = 3$

Output: 3

Explanation: There are three ways to climb to the top.

1. 1 step + 1 step + 1 step
2. 1 step + 2 steps
3. 2 steps + 1 step

### Constraints:

- $1 < n \leq 45$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/climbing-stairs/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int climbStairs(int n) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->climbStairs(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.24 72. Edit Distance (Medium)

Given two strings `word1` and `word2`, return the minimum number of operations required to convert `word1` to `word2`.

You have the following three operations permitted on a word:

- Insert a character
- Delete a character
- Replace a character

### Example 1:

**Input:** `word1 = "horse"`, `word2 = "ros"`

**Output:** 3

**Explanation:**

`horse` → `orse` (replace 'h' with 'r')

`orse` → `ose` (remove 'r')

`ose` → `os` (remove 'e')

### Example 2:

**Input:** `word1 = "intention"`, `word2 = "execution"`

**Output:** 5

**Explanation:**

`intention` → `inention` (remove 't')

`inention` → `enention` (replace 'i' with 'e')

`enention` → `exention` (replace 'n' with 'x')

`exention` → `exection` (replace 'n' with 'c')

`exection` → `execution` (insert 'u')

### Constraints:

- $0 < \text{word1.length}, \text{word2.length} \leq 500$
- `word1` and `word2` consist of lowercase English letters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/edit-distance/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int minDistance(string word1, string word2) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string word1;
    LeetCodeIO::scan(cin, word1);
    string word2;
    LeetCodeIO::scan(cin, word2);

    Solution *obj = new Solution();
    auto res = obj->minDistance(word1, word2);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.25 75. Sort Colors (Medium)

Given an array `nums` with  $n$  objects colored red, white, or blue, sort them **in-place** so that objects of the same color are adjacent, with the colors in the order red, white, and blue.

We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.

You must solve this problem without using the library's sort function.

### Example 1:

Input: `nums = [2,0,2,1,1,0]`

Output: `[0,0,1,1,2,2]`

### Example 2:

Input: `nums = [2,0,1]`

Output: `[0,1,2]`

### Constraints:

- $n = \text{nums.length} =$
- $1 < n \leq 300 =$
- $\text{nums}[i]$  is either 0, 1, or 2.

**Follow up:** Could you come up with a one-pass algorithm using only constant extra space?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sort-colors/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    void sortColors(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->sortColors(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.26 76. Minimum Window Substring (Hard)

Given two strings  $s$  and  $t$  of lengths  $m$  and  $n$  respectively, return the **minimum windowsubstring** 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**.

### Example 1:

Input:  $s = "ADOBECODEBANC"$ ,  $t = "ABC"$

Output: "BANC"

Explanation: The minimum window substring "BANC" includes 'A', 'B', and 'C' from string  $t$ .

### Example 2:

Input:  $s = "a"$ ,  $t = "a"$

Output: "a"

Explanation: The entire string  $s$  is the minimum window.

### Example 3:

Input:  $s = "a"$ ,  $t = "aa"$

Output: ""

Explanation: Both 'a's from  $t$  must be included in the window.

Since the largest window of  $s$  only has one 'a', return empty string.

### Constraints:

- $m = s.length =$
- $n = t.length =$
- $1 < m, n \leq 10^5 =$
- $s$  and  $t$  consist of uppercase and lowercase English letters.

**Follow up:** Could you find an algorithm that runs in  $O(m + n)$  time?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/minimum-window-substring/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    string minWindow(string s, string t) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);
    string t;
    LeetCodeIO::scan(cin, t);

    Solution *obj = new Solution();
    auto res = obj->minWindow(s, t);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.27 77. Combinations (Medium)

Given two integers  $n$  and  $k$ , return all possible combinations of  $k$ -numbers chosen from the range  $=[1, n]$ .

You may return the answer in **any order**.

### Example 1:

Input:  $n = 4$ ,  $k = 2$

Output:  $[[1,2], [1,3], [1,4], [2,3], [2,4], [3,4]]$

Explanation: There are  $4$  choose  $2 = 6$  total combinations.

Note that combinations are unordered, i.e.,  $[1,2]$  and  $[2,1]$  are considered to be the same combination.

### Example 2:

Input:  $n = 1$ ,  $k = 1$

Output:  $[[1]]$

Explanation: There is  $1$  choose  $1 = 1$  total combination.

### Constraints:

- $1 < n \leq 20$ =
- $1 < k \leq n=$

## 題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/combinations/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> combine(int n, int k) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);
    int k;
    LeetCodeIO::scan(cin, k);

    Solution *obj = new Solution();
    auto res = obj->combine(n, k);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.28 79. Word Search (Medium)

Given an  $m \times n$  grid of characters board and a string word, return =true=if =word=exists in the grid.

The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or vertically neighboring. The same letter cell may not be used more than once.

**Example 1:**

A	B	C	E
S	F	C	S
A	D	E	E

Input: board = [["A", "B", "C", "E"], ["S", "F", "C", "S"], ["A", "D", "E", "E"]], word = "ABCED"

Output: true

**Example 2:**

A	B	C	E
S	F	C	S
A	D	E	E

Input: board = [["A", "B", "C", "E"], ["S", "F", "C", "S"], ["A", "D", "E", "E"]], word = "SEE"

Output: true

**Example 3:**

A	B	C	E
S	F	C	S
A	D	E	E

Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"

Output: false

#### Constraints:

- m = board.length=
- n = board[i].length
- 1 < m, n <= 6=
- 1 < word.length <= 15=
- board and word consists of only lowercase and uppercase English letters.

**Follow up:** Could you use search pruning to make your solution faster with a larger board?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/word-search/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool exist(vector<vector<char>>& board, string word) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<char>> board;
    LeetCodeIO::scan(cin, board);
    string word;
    LeetCodeIO::scan(cin, word);

    Solution *obj = new Solution();
    auto res = obj->exist(board, word);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.29 81. Search in Rotated Sorted Array II (Medium)

There is an integer array `nums` sorted in non-decreasing order (not necessarily with **distinct** values).

Before being passed to your function, `nums` is **rotated** at an unknown pivot index `k` ( $0 < k < \text{nums.length} =$ ) such that the resulting array is `[nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]]` (**0-indexed**). For example, `[0,1,2,4,4,5,6,6,7]` might be rotated at pivot index 5 and become `[4,5,6,6,7,0,1,2,4,4]`.

Given the array `nums` **after** the rotation and an integer `target`, return `true` if `target` is in `nums`, or `false` if it is not in `nums`.

You must decrease the overall operation steps as much as possible.

### Example 1:

Input: `nums = [2,5,6,0,0,1,2]`, `target = 0`  
 Output: `true`

### Example 2:

Input: `nums = [2,5,6,0,0,1,2]`, `target = 3`  
 Output: `false`

### Constraints:

- $1 < \text{nums.length} \leq 5000 =$
- $-10 \leq \text{nums}[i] \leq 10^4 =$
- `nums` is guaranteed to be rotated at some pivot.
- $-10 \leq \text{target} \leq 10^4 =$

**Follow up:** This problem is similar to [Search in Rotated Sorted Array](#), but `nums` may contain **duplicates**. Would this affect the runtime complexity? How and why?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/search-in-rotated-sorted-array-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool search(vector<int>& nums, int target) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int target;
    LeetCodeIO::scan(cin, target);

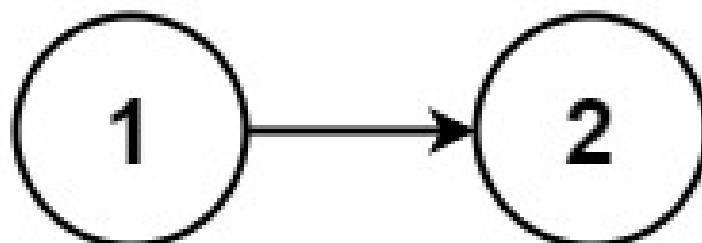
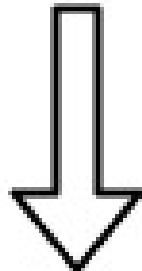
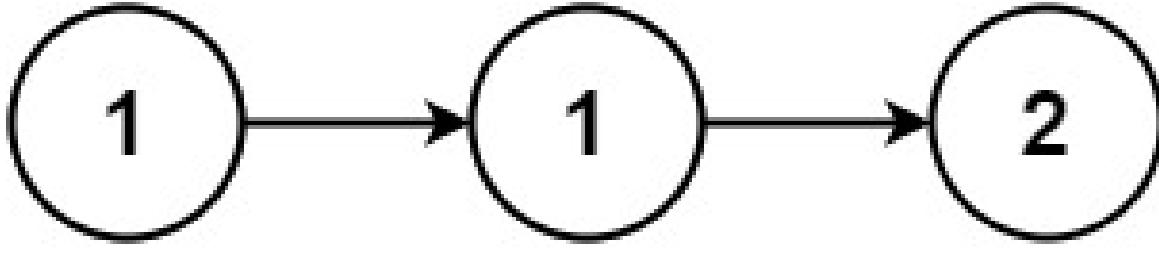
    Solution *obj = new Solution();
    auto res = obj->search(nums, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.30 83. Remove Duplicates from Sorted List (Easy)

Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the linked list **sorted** as well.

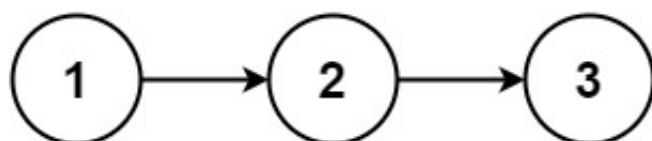
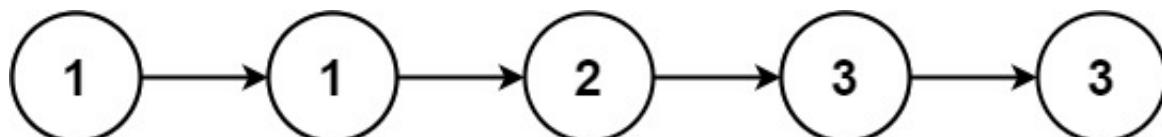
**Example 1:**



Input: head = [1,1,2]

Output: [1,2]

**Example 2:**



Input: head = [1,1,2,3,3]

Output: [1,2,3]

**Constraints:**

- The number of nodes in the list is in the range [0, 300].
- $-100 < \text{Node.val} \leq 100$
- The list is guaranteed to be **sorted** in ascending order.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/remove-duplicates-from-sorted-list/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    ListNode* deleteDuplicates(ListNode* head) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    ListNode* head;
    LeetCodeIO::scan(cin, head);

    Solution *obj = new Solution();
    auto res = obj->deleteDuplicates(head);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.31 88. Merge Sorted Array (Easy)

You are given two integer arrays `nums1` and `nums2`, sorted in **non-decreasing order**, and two integers `m` and `n`, representing the number of elements in `nums1` and `nums2` respectively.

**Merge** `nums1` and `nums2` into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be stored inside the array `nums1`. To accommodate this, `nums1` has a length of `m + n`, where the first `m` elements denote the elements that should be merged, and the last `n` elements are set to 0 and should be ignored. `nums2` has a length of `n`.

### Example 1:

Input: `nums1 = [1,2,3,0,0,0]`, `m = 3`, `nums2 = [2,5,6]`, `n = 3`

Output: `[1,2,2,3,5,6]`

Explanation: The arrays we are merging are `[1,2,3]` and `[2,5,6]`.

The result of the merge is `[1,2,2,3,5,6]` with the underlined elements coming from `nums1`.

### Example 2:

Input: `nums1 = [1]`, `m = 1`, `nums2 = []`, `n = 0`

Output: `[1]`

Explanation: The arrays we are merging are `[1]` and `[]`.

The result of the merge is `[1]`.

### Example 3:

Input: `nums1 = [0]`, `m = 0`, `nums2 = [1]`, `n = 1`

Output: `[1]`

Explanation: The arrays we are merging are `[]` and `[1]`.

The result of the merge is `[1]`.

Note that because `m = 0`, there are no elements in `nums1`. The 0 is only there to ensure the merge result can fit in `nums1`.

### Constraints:

- `nums1.length = m + n =`
- `nums2.length = n =`
- `0 < m, n <= 200 =`
- `1 < m + n <= 200 =`
- `-10 < nums1[i], nums2[j] <= 10^9 =`

**Follow up:** Can you come up with an algorithm that runs in  $O(m + n)$  time?

## 題解如下：

```

// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/merge-sorted-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums1;
    LeetCodeIO::scan(cin, nums1);
    int m;
    LeetCodeIO::scan(cin, m);
    vector<int> nums2;
    LeetCodeIO::scan(cin, nums2);
    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->merge(nums1, m, nums2, n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}

```

## 8.32 91. Decode Ways (Medium)

You have intercepted a secret message encoded as a string of numbers. The message is **decoded** via the following mapping:

"1" → 'A' "2" → 'B' ... "25" → 'Y' "26" → 'Z'

However, while decoding the message, you realize that there are many different ways you can decode the message because some codes are contained in other codes ("2" and "5" vs "25").

For example, "11106" can be decoded into:

- "AAJF" with the grouping (1, 1, 10, 6)
- "KJF" with the grouping (11, 10, 6)
- The grouping (1, 11, 06) is invalid because "06" is not a valid code (only "6" is valid).

Note: there may be strings that are impossible to decode.

Given a string s containing only digits, return the **number of ways** to decode it. If the entire string cannot be decoded in any valid way, return 0.

The test cases are generated so that the answer fits in a **32-bit** integer.

**Example 1:**

**Input:** s = "12"

**Output:** 2

**Explanation:**

"12" could be decoded as "AB" (1 2) or "L" (12).

**Example 2:**

**Input:** s = "226"

**Output:** 3

**Explanation:**

"226" could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).

**Example 3:**

**Input:** s = "06"

**Output:** 0

**Explanation:**

"06" cannot be mapped to "F" because of the leading zero ("6" is different from "06"). In this case, the string is not a valid encoding, so return 0.

**Constraints:**

- $1 < \text{s.length} \leq 100$
- s contains only digits and may contain leading zero(s).

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/decode-ways/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int numDecodings(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->numDecodings(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.33 94. Binary Tree Inorder Traversal (Easy)

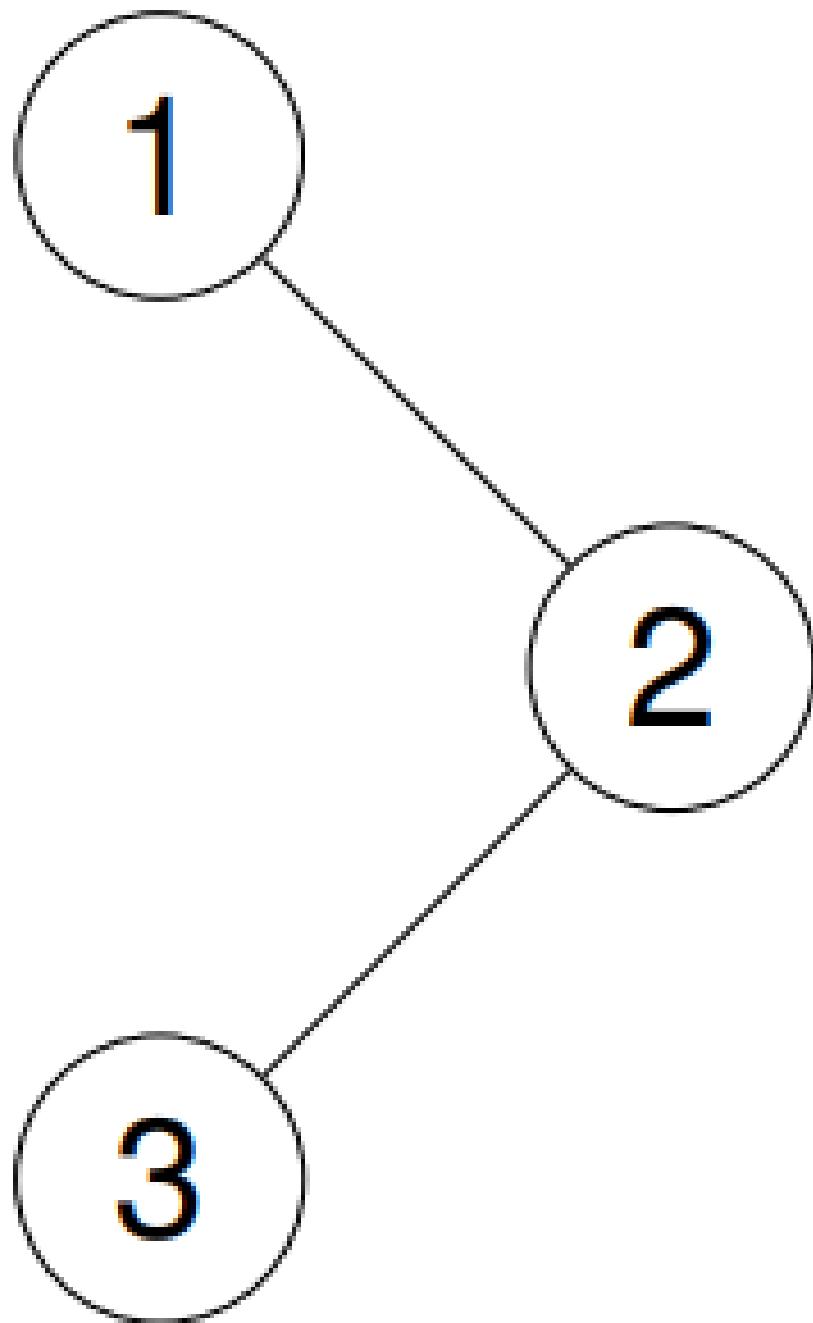
Given the root of a binary tree, return the inorder traversal of its nodes' values.

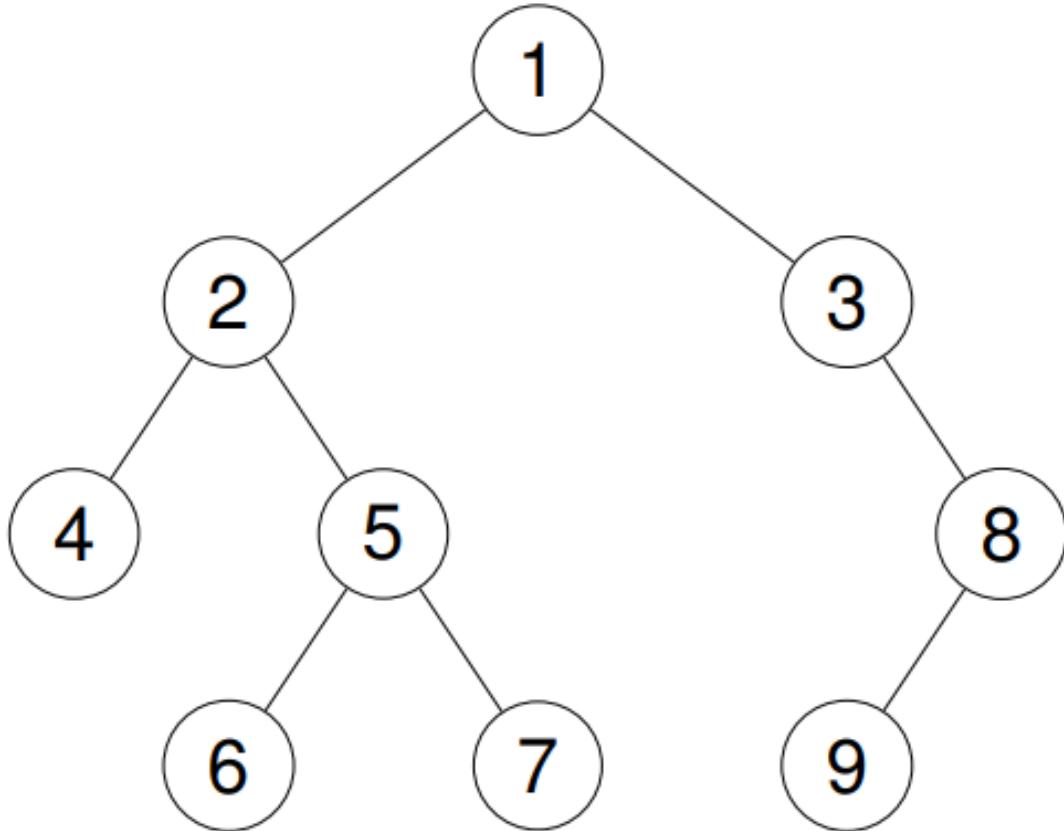
**Example 1:**

**Input:** root = [1,null,2,3]

**Output:**[1,3,2]

**Explanation:**



**Example 2:****Input:** root = [1,2,3,4,5,null,8,null,null,6,7,9]**Output:**[4,2,6,5,7,1,3,9,8]**Explanation:****Example 3:****Input:** root = []**Output:**[]**Example 4:****Input:** root = [1]**Output:**[1]**Constraints:**

- The number of nodes in the tree is in the range [0, 100].

- $-100 < \text{Node.val} \leq 100$

**Follow up:** Recursive solution is trivial, could you do it iteratively?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/binary-tree-inorder-traversal/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> inorderTraversal(TreeNode* root) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

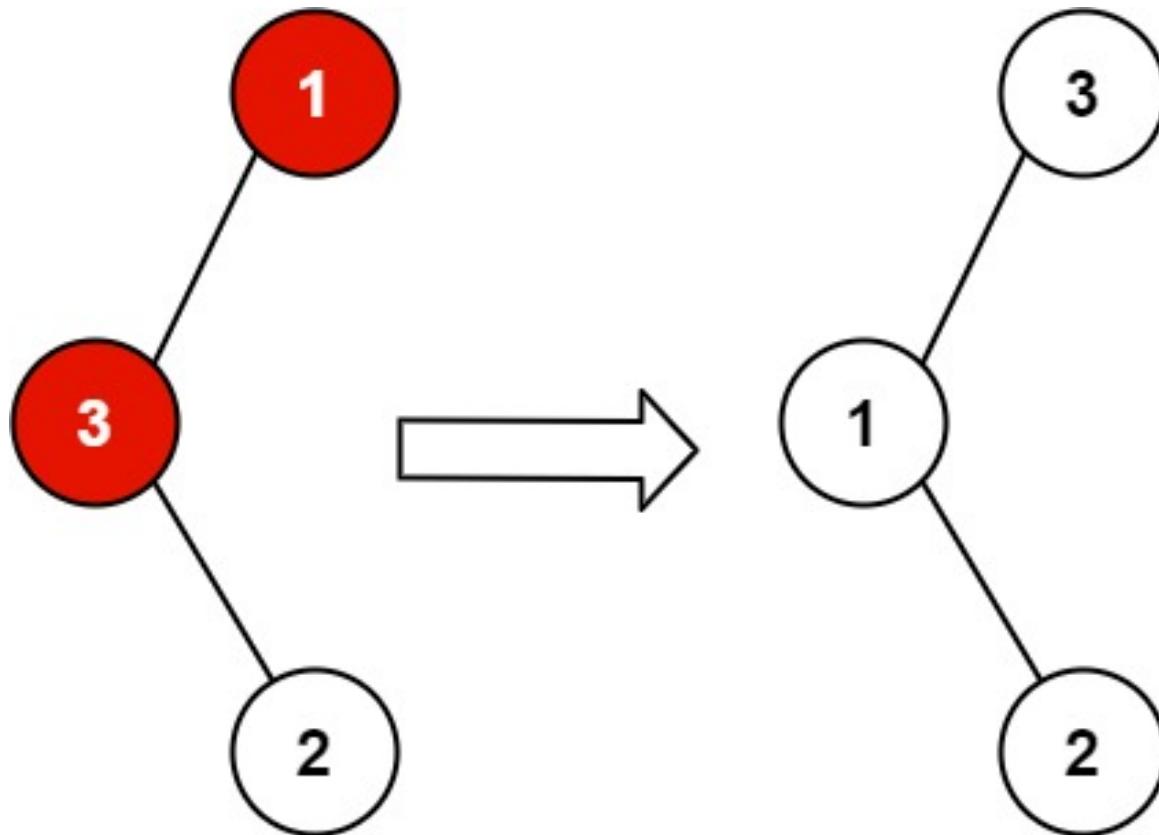
    Solution *obj = new Solution();
    auto res = obj->inorderTraversal(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.34 99. Recover Binary Search Tree (Medium)

You are given the root of a binary search tree (BST), where the values of **exactly** two nodes of the tree were swapped by mistake. Recover the tree without changing its structure.

**Example 1:**

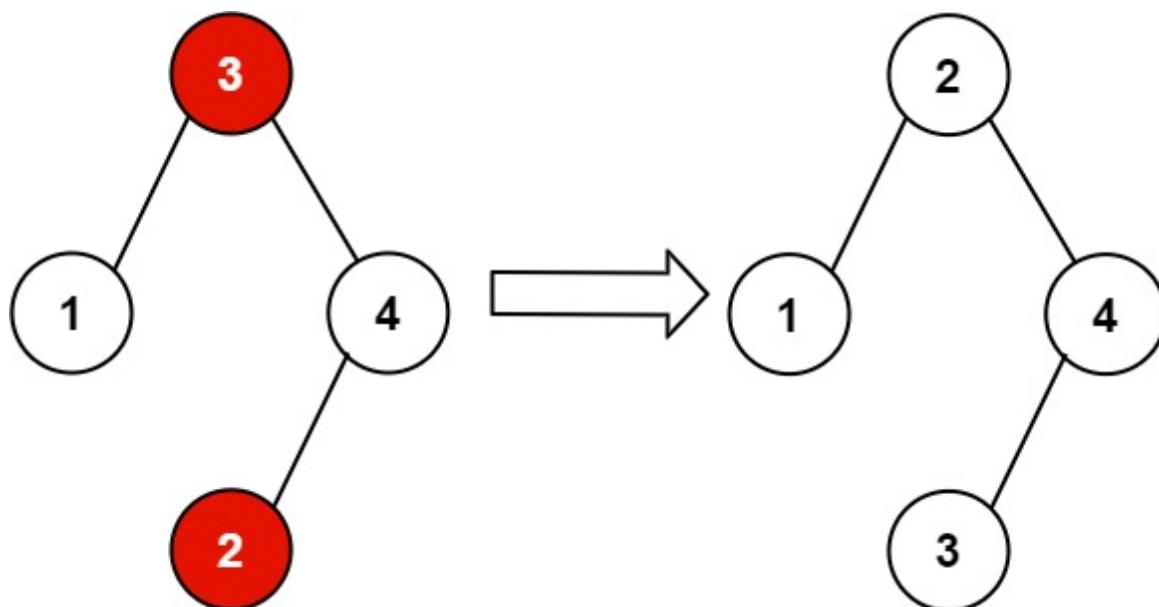


Input: root = [1,3,null,null,2]

Output: [3,1,null,null,2]

Explanation: 3 cannot be a left child of 1 because  $3 > 1$ . Swapping 1 and 3 makes the BST valid.

**Example 2:**



Input: root = [3,1,4,null,null,2]

Output: [2,1,4,null,null,3]

Explanation: 2 cannot be in the right subtree of 3 because  $2 < 3$ . Swapping 2 and 3 makes the BST valid.

**Constraints:**

- The number of nodes in the tree is in the range  $[2, 1000]$ .
- $-2^{31} < \text{Node.val} \leq 2^{31} - 1$

**Follow up:** A solution using  $O(n)$  space is pretty straight-forward. Could you devise a constant  $O(1)$  space solution?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/recover-binary-search-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    void recoverTree(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

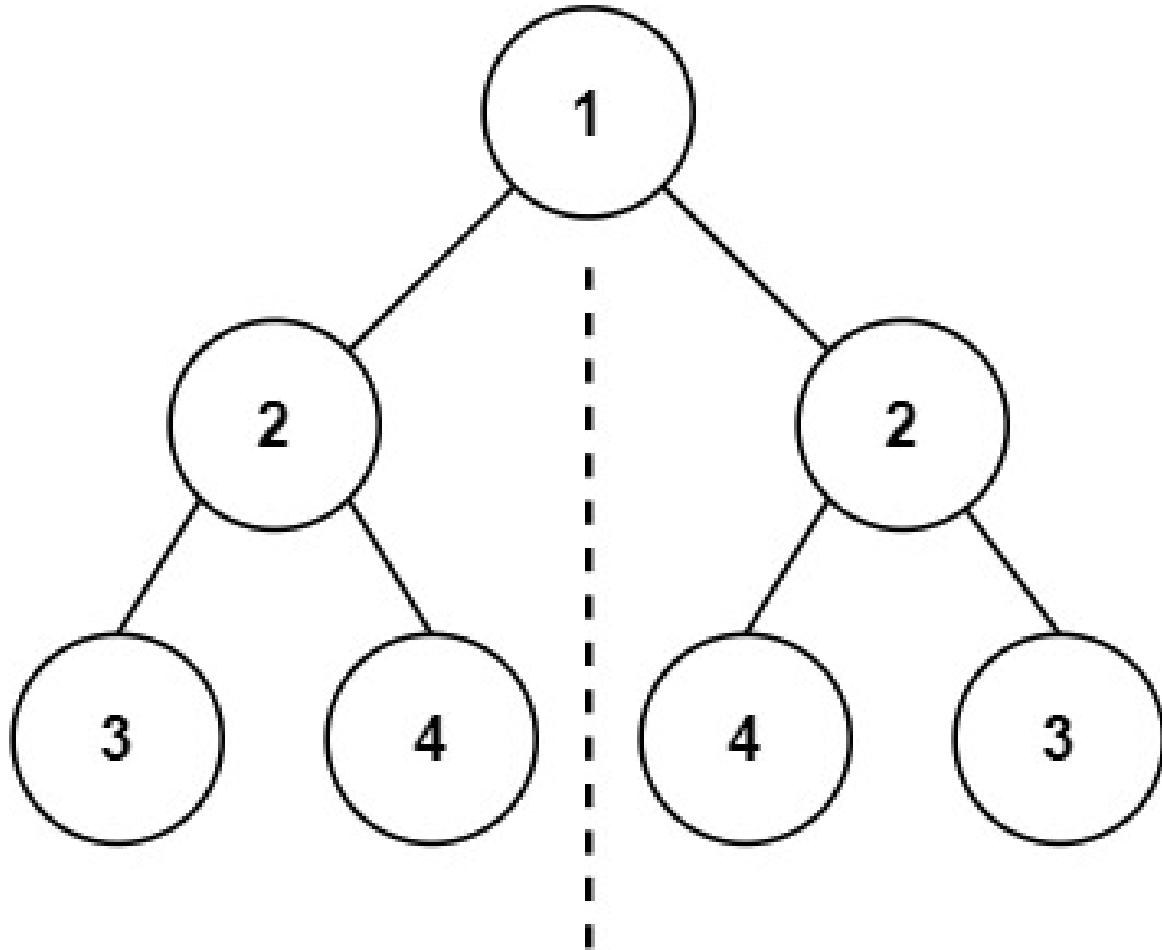
    Solution *obj = new Solution();
    auto res = obj->recoverTree(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.35 101. Symmetric Tree (Easy)

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

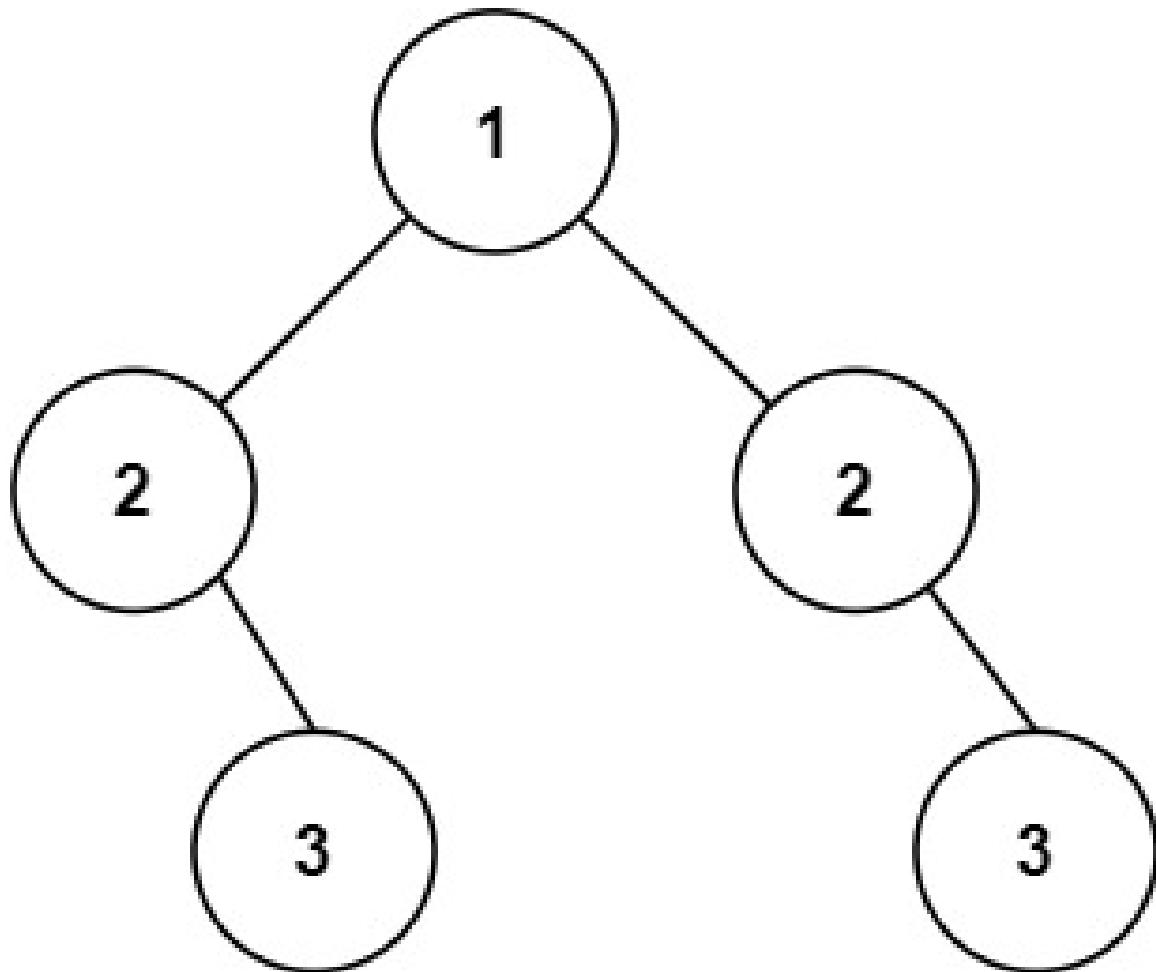
**Example 1:**



Input: `root = [1,2,2,3,4,4,3]`

Output: `true`

**Example 2:**



Input: `root = [1,2,2,null,3,null,3]`

Output: `false`

**Constraints:**

- The number of nodes in the tree is in the range  $[1, 1000]$ .
- $-100 < \text{Node.val} \leq 100$

**Follow up:** Could you solve it both recursively and iteratively?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/symmetric-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isSymmetric(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->isSymmetric(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

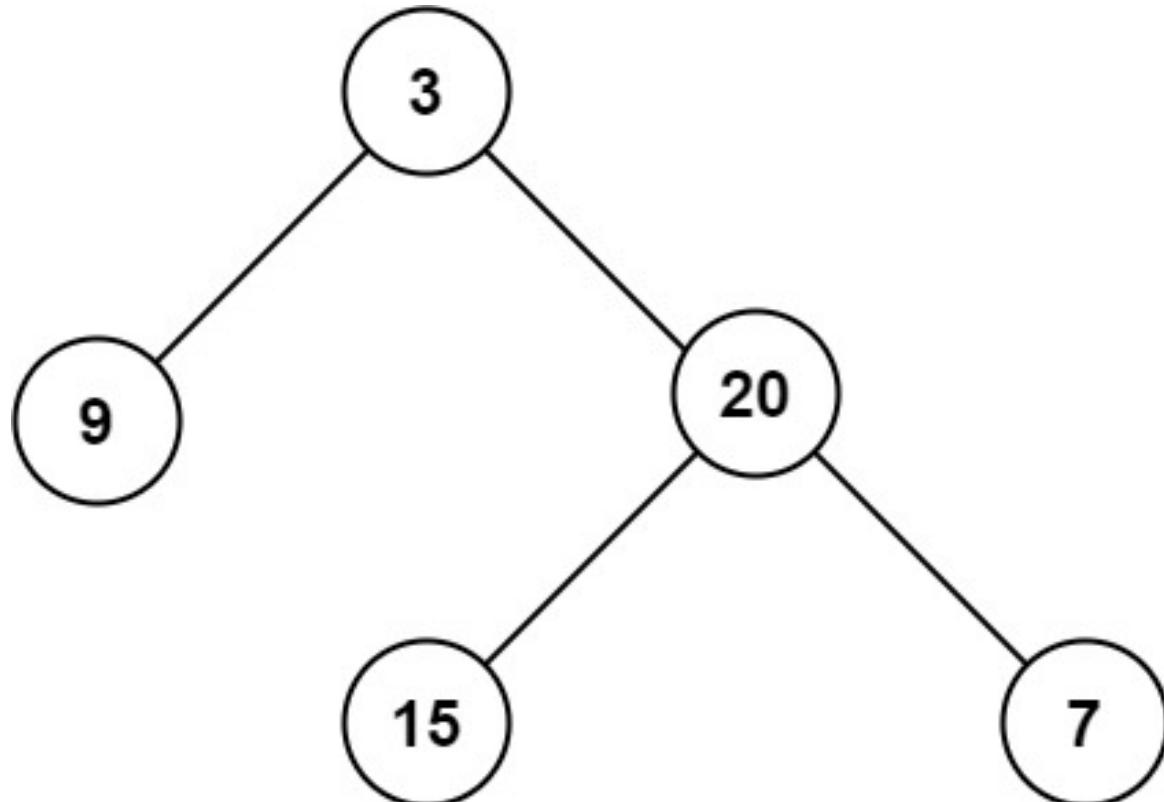
    delete obj;
    return 0;
}
```

## 8.36 104. Maximum Depth of Binary Tree (Easy)

Given the root of a binary tree, return its maximum depth.

A binary tree's **maximum depth** is the number of nodes along the longest path from the root node down to the farthest leaf node.

**Example 1:**



Input: `root = [3,9,20,null,null,15,7]`

Output: 3

**Example 2:**

Input: `root = [1,null,2]`

Output: 2

**Constraints:**

- The number of nodes in the tree is in the range  $[0, 10]$ .
- $-100 < \text{Node.val} \leq 100$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/maximum-depth-of-binary-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxDepth(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

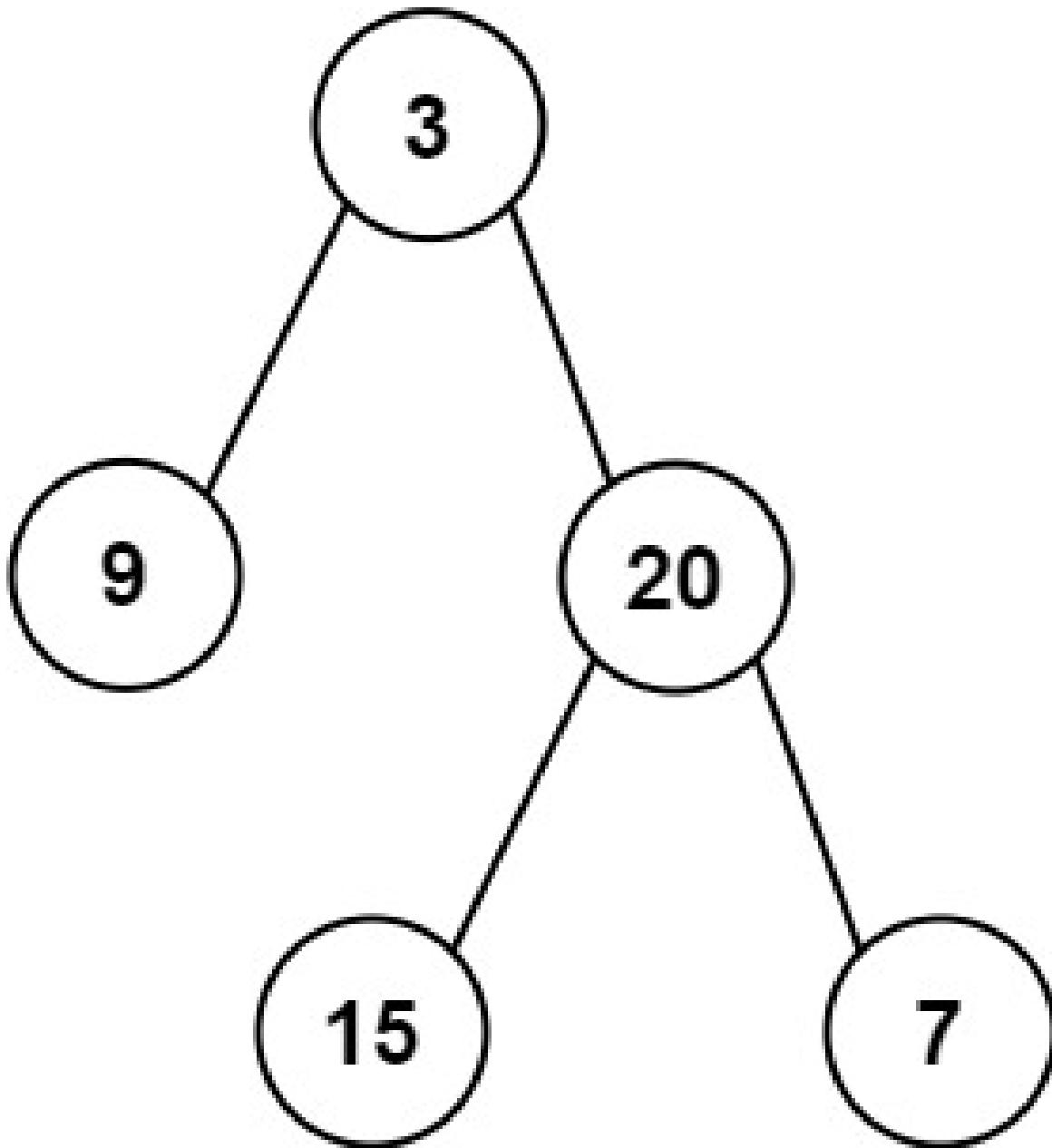
    Solution *obj = new Solution();
    auto res = obj->maxDepth(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.37 105. Construct Binary Tree from Preorder and Inorder Traversal (Medium)

Given two integer arrays `preorder` and `inorder` where `preorder` is the preorder traversal of a binary tree and `inorder` is the inorder traversal of the same tree, construct and return the binary tree.

**Example 1:**



Input: `preorder = [3,9,20,15,7]`, `inorder = [9,3,15,20,7]`

Output: `[3,9,20,null,null,15,7]`

**Example 2:**

Input: `preorder = [-1]`, `inorder = [-1]`

Output: `[-1]`

**Constraints:**

- $1 < \text{preorder.length} \leq 3000$
- $\text{inorder.length} = \text{preorder.length}$
- $-3000 < \text{preorder}[i], \text{inorder}[i] \leq 3000$
- `preorder` and `inorder` consist of **unique** values.
- Each value of `inorder` also appears in `preorder`.
- `preorder` is **guaranteed** to be the preorder traversal of the tree.
- `inorder` is **guaranteed** to be the inorder traversal of the tree.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* buildTree(vector<int>& preorder, vector<int>& inorder) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> preorder;
    LeetCodeIO::scan(cin, preorder);
    vector<int> inorder;
    LeetCodeIO::scan(cin, inorder);

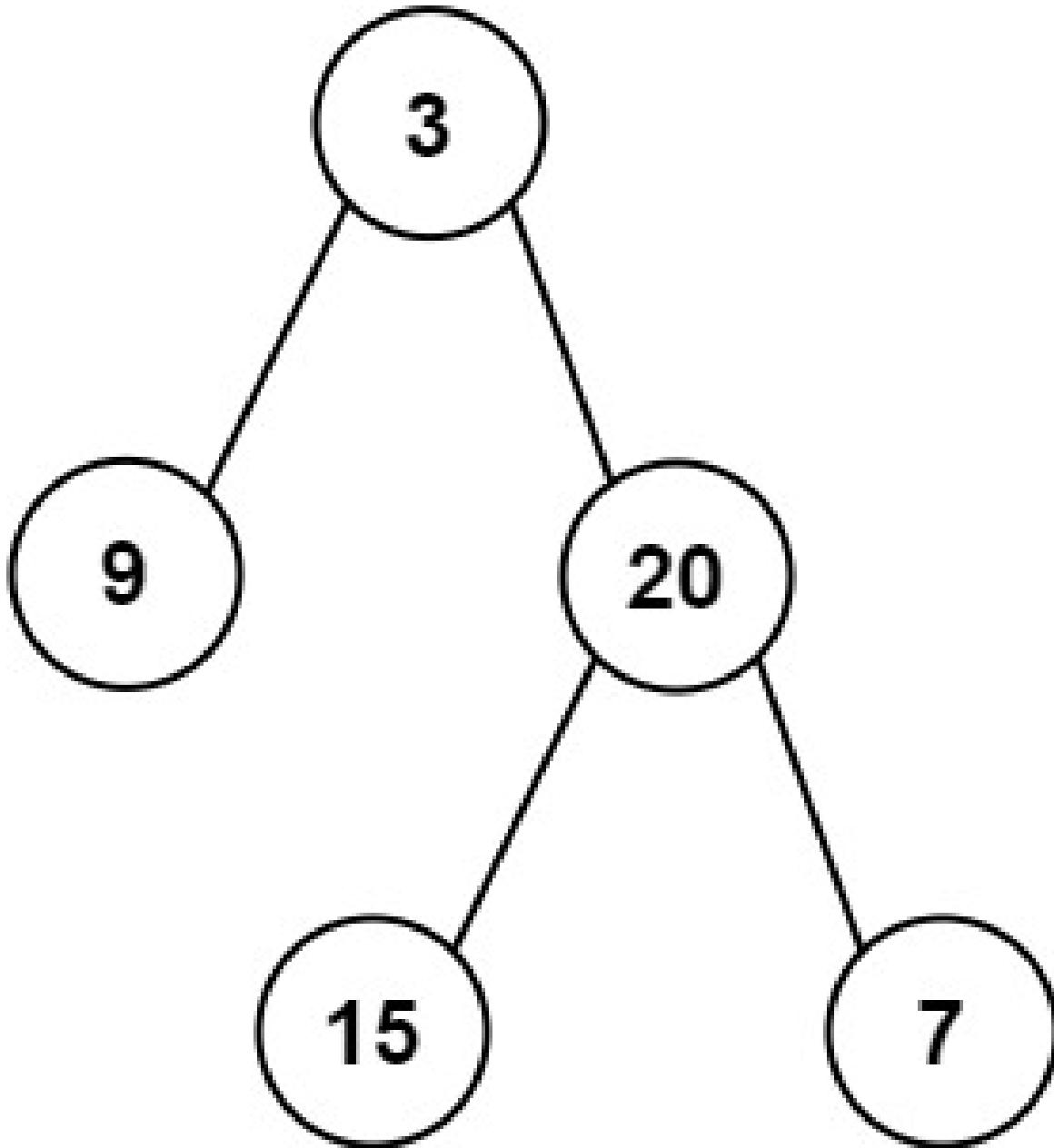
    Solution *obj = new Solution();
    auto res = obj->buildTree(preorder, inorder);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.38 106. Construct Binary Tree from Inorder and Postorder Traversal (Medium)

Given two integer arrays `inorder` and `postorder` where `inorder` is the inorder traversal of a binary tree and `postorder` is the postorder traversal of the same tree, construct and return the binary tree.

**Example 1:**



Input: `inorder = [9,3,15,20,7]`, `postorder = [9,15,7,20,3]`

Output: `[3,9,20,null,null,15,7]`

**Example 2:**

Input: `inorder = [-1]`, `postorder = [-1]`

Output: `[-1]`

**Constraints:**

- $1 < \text{inorder.length} \leq 3000$
- $\text{postorder.length} = \text{inorder.length}$
- $-3000 < \text{inorder}[i], \text{postorder}[i] \leq 3000$
- **inorder** and **postorder** consist of **unique** values.
- Each value of **postorder** also appears in **inorder**.
- **inorder** is **guaranteed** to be the inorder traversal of the tree.
- **postorder** is **guaranteed** to be the postorder traversal of the tree.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/construct-binary-tree-from-inorder-and-postorder-traversal/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* buildTree(vector<int>& inorder, vector<int>& postorder) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> inorder;
    LeetCodeIO::scan(cin, inorder);
    vector<int> postorder;
    LeetCodeIO::scan(cin, postorder);

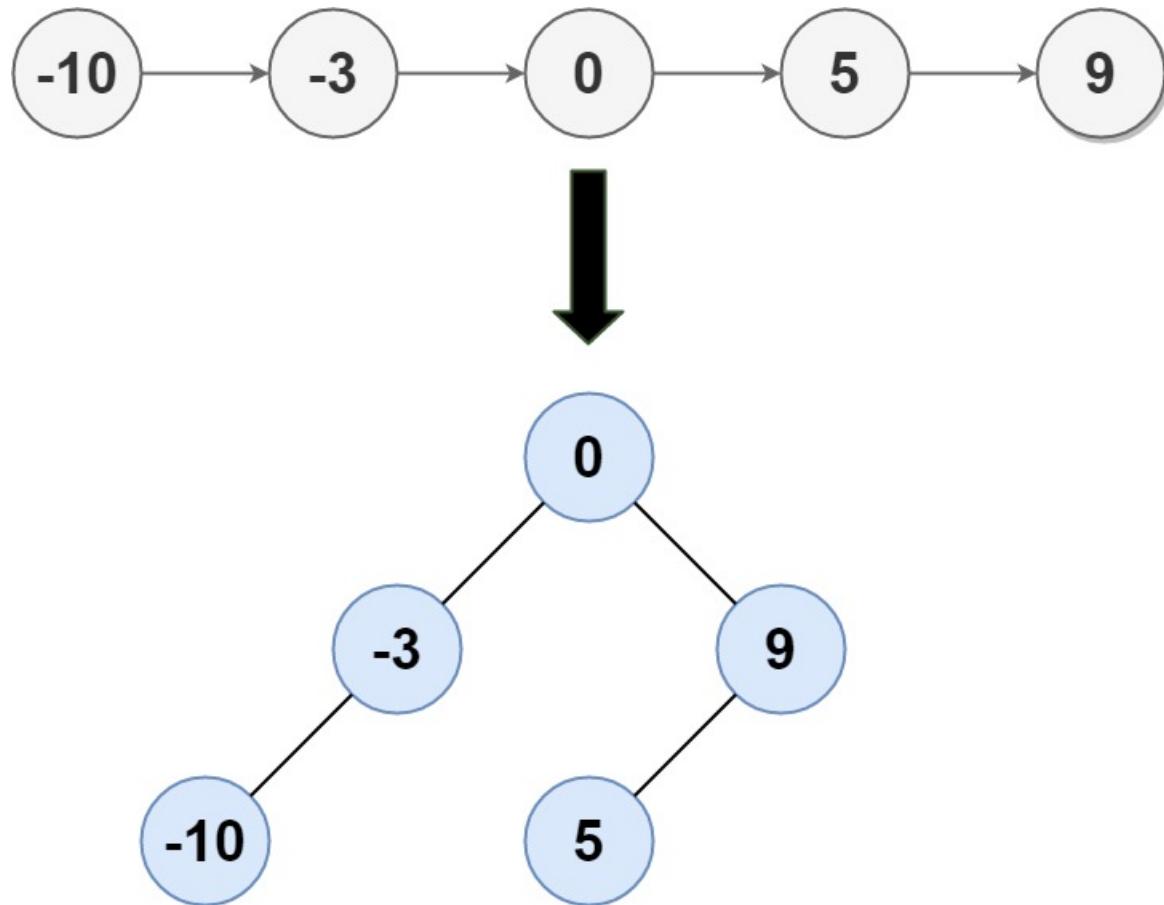
    Solution *obj = new Solution();
    auto res = obj->buildTree(inorder, postorder);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.39 109. Convert Sorted List to Binary Search Tree (Medium)

Given the head of a singly linked list where elements are sorted in **ascending order**, convert it to a \*height-balanced\*binary search tree.

**Example 1:**



Input: head = [-10,-3,0,5,9]

Output: [0,-3,9,-10,null,5]

Explanation: One possible answer is [0,-3,9,-10,null,5], which represents the shown height balanced BST.

**Example 2:**

Input: head = []

Output: []

### Constraints:

- The number of nodes in head is in the range [0, 2 \* 10 ].
- $-10 < \text{Node.val} \leq 10^5$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* sortedListToBST(ListNode* head) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    ListNode* head;
    LeetCodeIO::scan(cin, head);

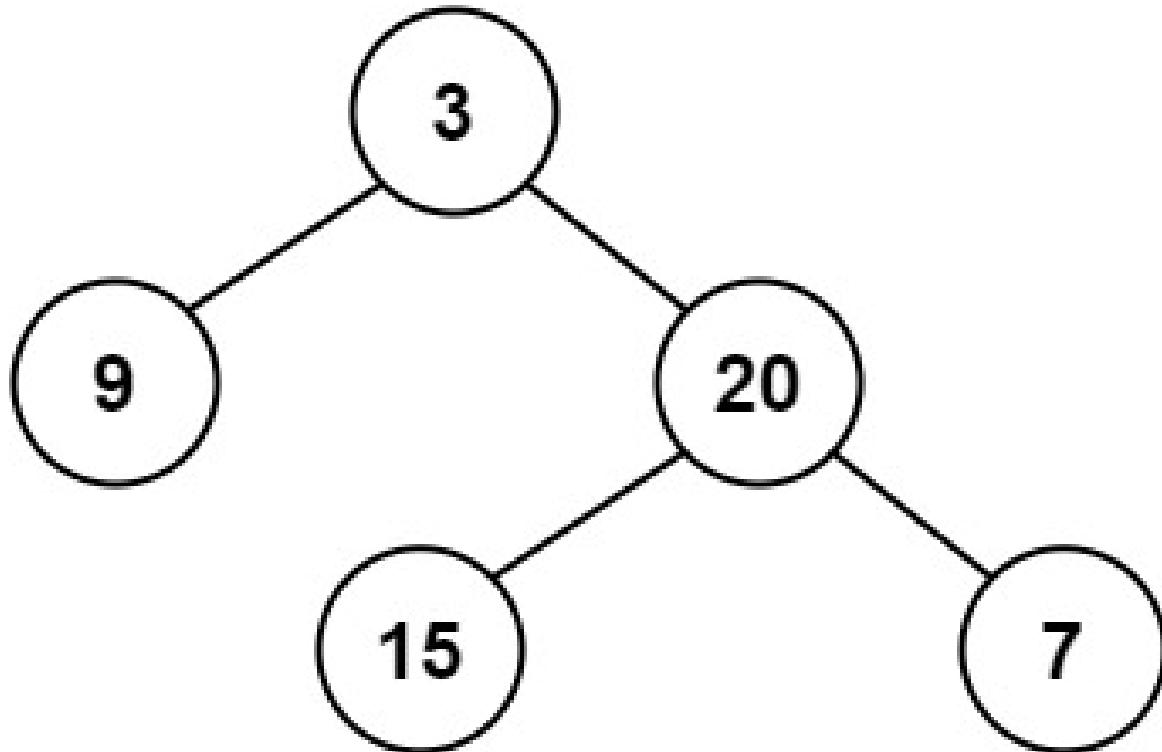
    Solution *obj = new Solution();
    auto res = obj->sortedListToBST(head);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.40 110. Balanced Binary Tree (Easy)

Given a binary tree, determine if it is **height-balanced**.

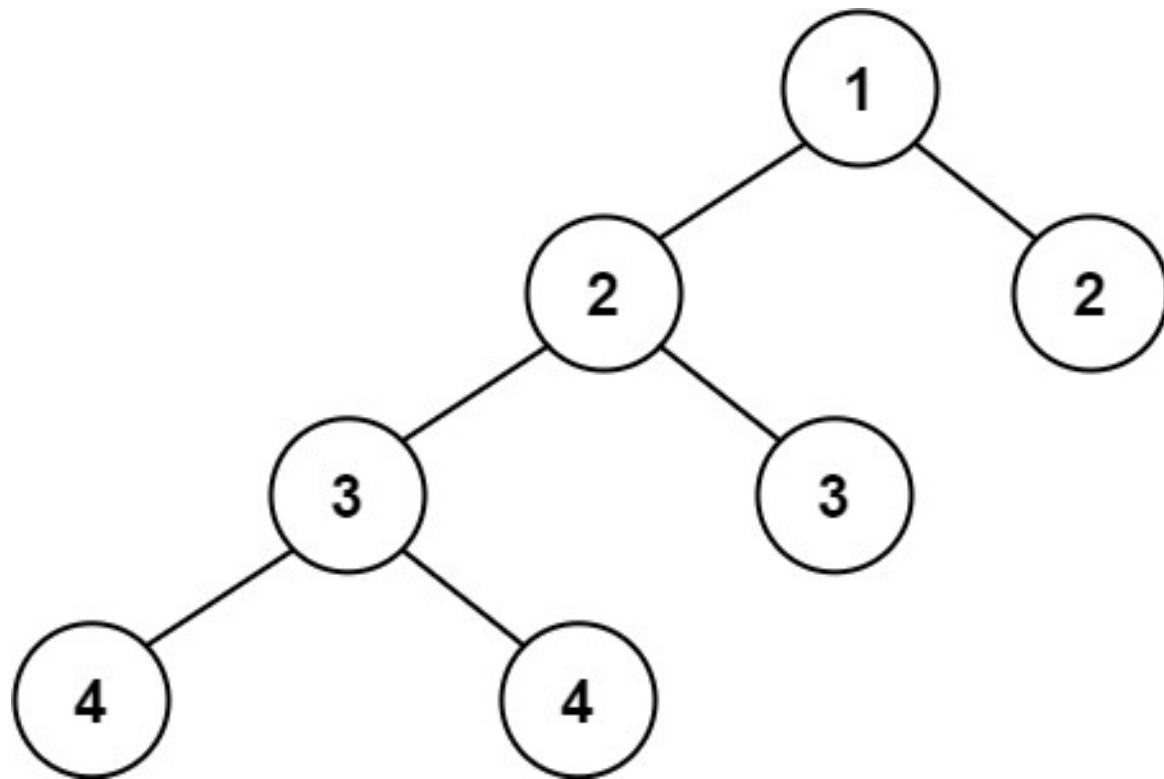
**Example 1:**



Input: root = [3,9,20,null,null,15,7]

Output: true

**Example 2:**



Input: root = [1,2,2,3,3,null,null,4,4]

Output: false

**Example 3:**

Input: root = []

Output: true

**Constraints:**

- The number of nodes in the tree is in the range [0, 5000].
- $-10^4 < \text{Node.val} \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/balanced-binary-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isBalanced(TreeNode* root) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->isBalanced(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.41 121. Best Time to Buy and Sell Stock (Easy)

You are given an array `prices` where `prices[i]` is the price of a given stock on the `i` day. You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

### Example 1:

Input: `prices = [7,1,5,3,6,4]`

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

### Example 2:

Input: `prices = [7,6,4,3,1]`

Output: 0

Explanation: In this case, no transactions are done and the max profit = 0.

### Constraints:

- $1 < \text{prices.length} \leq 10^5$
- $0 < \text{prices}[i] \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxProfit(vector<int>& prices) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> prices;
    LeetCodeIO::scan(cin, prices);

    Solution *obj = new Solution();
    auto res = obj->maxProfit(prices);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.42 122. Best Time to Buy and Sell Stock II (Medium)

You are given an integer array `prices` where `prices[i]` is the price of a given stock on the `i` day. On each day, you may decide to buy and/or sell the stock. You can only hold **at most one** share of the stock at any time. However, you can sell and buy the stock multiple times on the **same day**, ensuring you never hold more than one share of the stock.

Find and return the **maximum** profit you can achieve.

### Example 1:

Input: `prices = [7,1,5,3,6,4]`

Output: 7

Explanation: Buy on day 2 (`price = 1`) and sell on day 3 (`price = 5`), profit =  $5-1 = 4$ .

Then buy on day 4 (`price = 3`) and sell on day 5 (`price = 6`), profit =  $6-3 = 3$ .

Total profit is  $4 + 3 = 7$ .

### Example 2:

Input: `prices = [1,2,3,4,5]`

Output: 4

Explanation: Buy on day 1 (`price = 1`) and sell on day 5 (`price = 5`), profit =  $5-1 = 4$ .

Total profit is 4.

### Example 3:

Input: `prices = [7,6,4,3,1]`

Output: 0

Explanation: There is no way to make a positive profit, so we never buy the stock to achieve the maximum profit of 0.

### Constraints:

- $1 < \text{prices.length} \leq 3 * 10^4$
- $0 < \text{prices}[i] \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxProfit(vector<int>& prices) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> prices;
    LeetCodeIO::scan(cin, prices);

    Solution *obj = new Solution();
    auto res = obj->maxProfit(prices);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.43 126. Word Ladder II (Hard)

A **transformation sequence** from word `beginWord` to word `endWord` using a dictionary `wordList` is a sequence of words `beginWord`  $\rightarrow$  `s`  $\rightarrow$  `s`  $\rightarrow$  ...  $\rightarrow$  `s` such that:

- Every adjacent pair of words differs by a single letter.
- Every `s` for  $1 < i \leq k$  is in `wordList`. Note that `beginWord` does not need to be in `wordList`.
- `s = 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`, `s`, `s`, ..., `s`].

### Example 1:

`Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]`

`Output: [[["hit","hot","dot","dog","cog"],["hit","hot","lot","log","cog"]]`

`Explanation: There are 2 shortest transformation sequences:`

`"hit" -> "hot" -> "dot" -> "dog" -> "cog"`

`"hit" -> "hot" -> "lot" -> "log" -> "cog"`

### Example 2:

`Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]`

`Output: []`

`Explanation: The endWord "cog" is not in wordList, therefore there is no valid transformation sequence.`

### Constraints:

- $1 < \text{beginWord.length} \leq 5$
- `endWord.length = beginWord.length`
- $1 < \text{wordList.length} \leq 500$
- `wordList[i].length = beginWord.length`
- `beginWord`, `endWord`, and `wordList[i]` consist of lowercase English letters.
- `beginWord ! endWord`
- All the words in `wordList` are **unique**.
- The **sum** of all shortest transformation sequences does not exceed 10 .

## 題解如下：

```

// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/word-ladder-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<string>> findLadders(string beginWord, string endWord, vector<string>& wordList) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string beginWord;
    LeetCodeIO::scan(cin, beginWord);
    string endWord;
    LeetCodeIO::scan(cin, endWord);
    vector<string> wordList;
    LeetCodeIO::scan(cin, wordList);

    Solution *obj = new Solution();
    auto res = obj->findLadders(beginWord, endWord, wordList);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}

```

## 8.44 128. Longest Consecutive Sequence (Medium)

Given an unsorted array of integers `nums`, return the length of the longest consecutive elements sequence.

You must write an algorithm that runs in  $O(n)$  time.

### Example 1:

Input: `nums = [100, 4, 200, 1, 3, 2]`

Output: 4

Explanation: The longest consecutive elements sequence is `[1, 2, 3, 4]`. Therefore its length is 4.

### Example 2:

Input: `nums = [0, 3, 7, 2, 5, 8, 4, 6, 0, 1]`

Output: 9

### Example 3:

Input: `nums = [1, 0, 1, 2]`

Output: 3

### Constraints:

- $0 < \text{nums.length} \leq 10^5$
- $-10^9 < \text{nums}[i] \leq 10^9$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/longest-consecutive-sequence/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int longestConsecutive(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->longestConsecutive(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.45 130. Surrounded Regions (Medium)

You are given an  $m \times n$  matrix board containing letters 'X' and 'O', capture regions that are surrounded:

- **Connect:** A cell is connected to adjacent cells horizontally or vertically.
- **Region:** To form a region **connect every 'O'** cell.
- **Surround:** The region is surrounded with 'X' cells if you can **connect the region** with 'X' cells and none of the region cells are on the edge of the board.

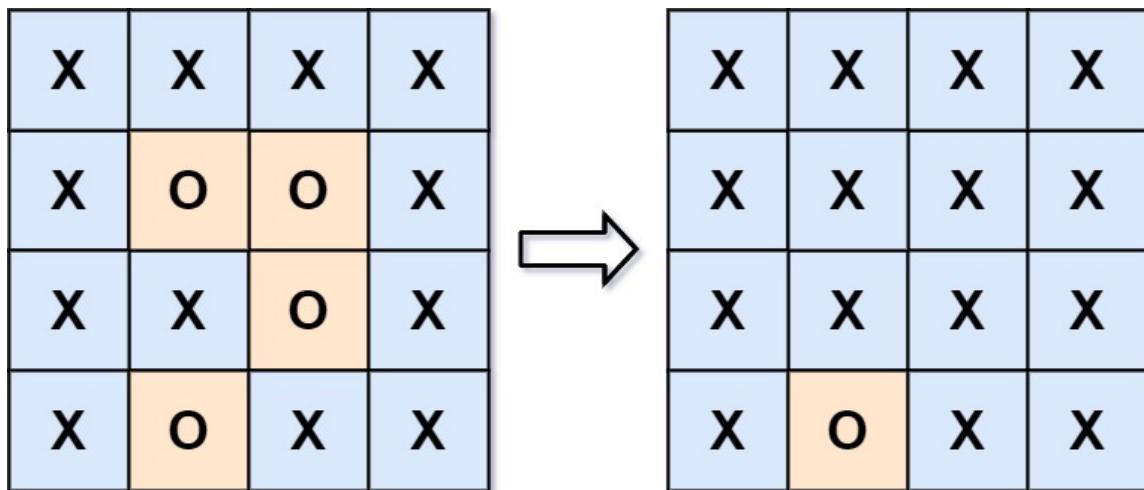
To capture a **surrounded region**, replace all 'O' s with 'X' s **in-place** within the original board. You do not need to return anything.

### Example 1:

**Input:** board = [["X","X","X","X"],["X","O","O","X"],["X","X","O","X"],["X","O","X","X"]]

**Output:** [["X","X","X","X"],["X","X","X","X"],["X","X","X","X"],["X","O","X","X"]]

### Explanation:



In the above diagram, the bottom region is not captured because it is on the edge of the board and cannot be surrounded.

### Example 2:

**Input:** board =

**Output:**

**Constraints:**

- $m = \text{board.length} =$
- $n = \text{board}[i].length =$
- $1 < m, n \leq 200 =$
- $\text{board}[i][j]$  is 'X' or 'O'.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/surrounded-regions/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    void solve(vector<vector<char>>& board) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<char>> board;
    LeetCodeIO::scan(cin, board);

    Solution *obj = new Solution();
    auto res = obj->solve(board);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.46 135. Candy (Hard)

There are  $n$  children standing in a line. Each child is assigned a rating value given in the integer array `ratings`.

You are giving candies to these children subjected to the following requirements:

- Each child must have at least one candy.
- Children with a higher rating get more candies than their neighbors.

Return the minimum number of candies you need to have to distribute the candies to the children.

### Example 1:

Input: `ratings = [1,0,2]`

Output: 5

Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.

### Example 2:

Input: `ratings = [1,2,2]`

Output: 4

Explanation: You can allocate to the first, second and third child with 1, 2, 1 candies respectively.

The third child gets 1 candy because it satisfies the above two conditions.

### Constraints:

- $n = \text{ratings.length} =$
- $1 < n \leq 2 * 10^4 =$
- $0 < \text{ratings}[i] \leq 2 * 10^4 =$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/candy/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int candy(vector<int>& ratings) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> ratings;
    LeetCodeIO::scan(cin, ratings);

    Solution *obj = new Solution();
    auto res = obj->candy(ratings);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.47 136. Single Number (Easy)

Given a **non-empty** array of integers `nums`, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

**Example 1:**

**Input:** `nums = [2,2,1]`

**Output:** 1

**Example 2:**

**Input:** `nums = [4,1,2,1,2]`

**Output:** 4

**Example 3:**

**Input:** `nums = [1]`

**Output:** 1

**Constraints:**

- $1 < \text{nums.length} \leq 3 * 10^4$
- $-3 * 10^3 < \text{nums}[i] \leq 3 * 10^4$
- Each element in the array appears twice except for one element which appears only once.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/single-number/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int singleNumber(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->singleNumber(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.48 139. Word Break (Medium)

Given a string  $s$  and a dictionary of strings  $\text{wordDict}$ , return `true` if  $s$  can be segmented into a space-separated sequence of one or more dictionary words.

**Note** that the same word in the dictionary may be reused multiple times in the segmentation.

### Example 1:

Input:  $s = \text{"leetcode"}$ ,  $\text{wordDict} = [\text{"leet"}, \text{"code"}]$

Output: `true`

Explanation: Return `true` because "leetcode" can be segmented as "leet code".

### Example 2:

Input:  $s = \text{"applepenapple"}$ ,  $\text{wordDict} = [\text{"apple"}, \text{"pen"}]$

Output: `true`

Explanation: Return `true` because "applepenapple" can be segmented as "apple pen apple".

Note that you are allowed to reuse a dictionary word.

### Example 3:

Input:  $s = \text{"catsandog"}$ ,  $\text{wordDict} = [\text{"cats"}, \text{"dog"}, \text{"sand"}, \text{"and"}, \text{"cat"}]$

Output: `false`

### Constraints:

- $1 < s.length \leq 300$ =
- $1 < \text{wordDict.length} \leq 1000=$
- $1 < \text{wordDict}[i].length \leq 20=$
- $s$  and  $\text{wordDict}[i]$  consist of only lowercase English letters.
- All the strings of  $\text{wordDict}$  are **unique**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/word-break/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);
    vector<string> wordDict;
    LeetCodeIO::scan(cin, wordDict);

    Solution *obj = new Solution();
    auto res = obj->wordBreak(s, wordDict);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.49 142. Linked List Cycle II (Medium)

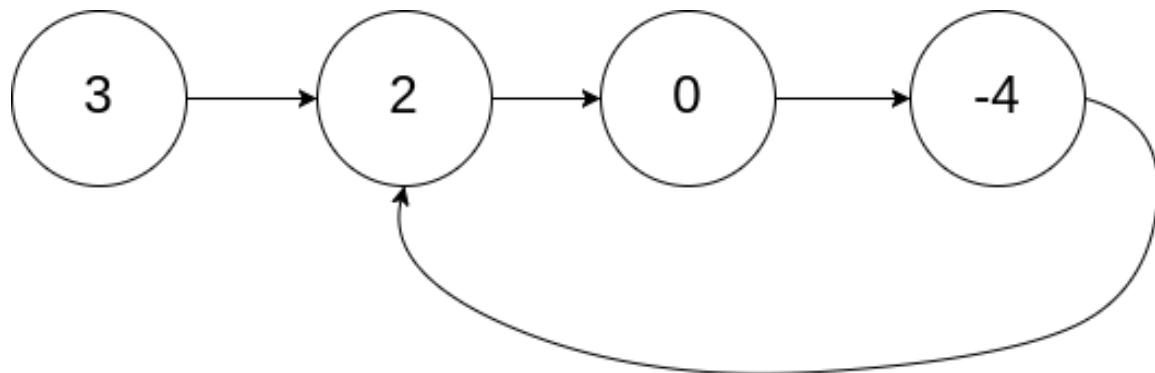
Given the head of a linked list, return the node where the cycle begins. If there is no cycle, return `null`.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the next pointer. Internally, `pos` is used to denote the index of the node that tail's next pointer is connected to (**0-indexed**).

It is `-1` if there is no cycle. **Note that pos is not passed as a parameter.**

**Do not modify** the linked list.

**Example 1:**

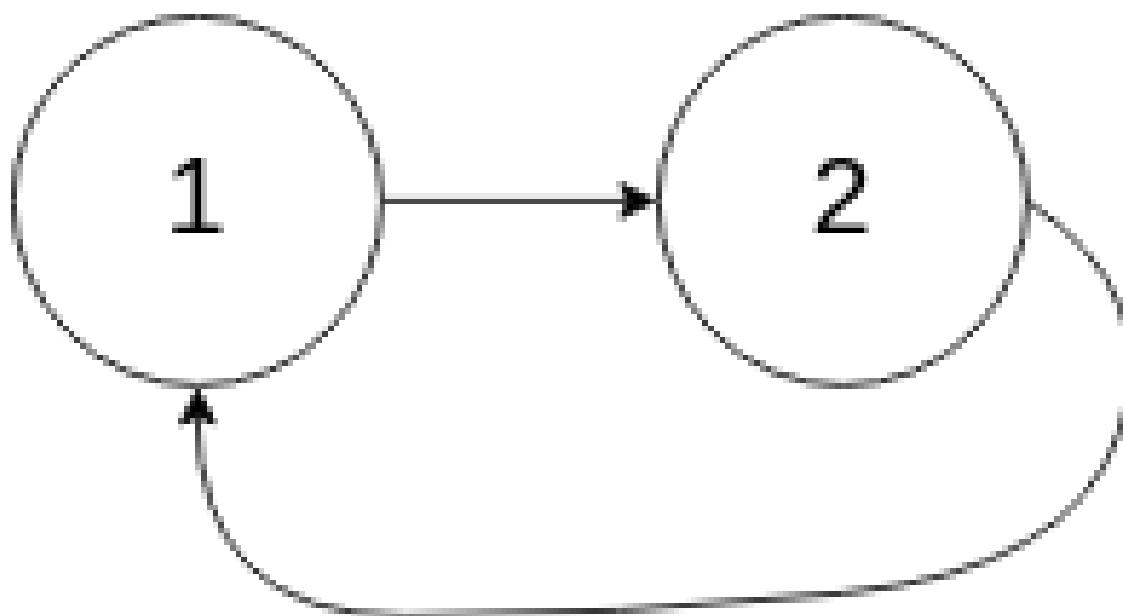


**Input:** `head = [3,2,0,-4]`, `pos = 1`

**Output:** tail connects to node index 1

**Explanation:** There is a cycle in the linked list, where tail connects to the second node.

**Example 2:**



**Input:** `head = [1,2]`, `pos = 0`

**Output:** tail connects to node index 0

**Explanation:** There is a cycle in the linked list, where tail connects to the first node.

**Example 3:**



Input: head = [1], pos = -1

Output: no cycle

Explanation: There is no cycle in the linked list.

#### Constraints:

- The number of the nodes in the list is in the range [0, 10].
- $-10 < \text{Node.val} \leq 10^5$
- pos is -1 or a **valid index** in the linked-list.

**Follow up:** Can you solve it using  $O(1)$  (i.e. constant) memory?

## 題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/linked-list-cycle-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    ListNode *detectCycle(ListNode *head) {
        }

};

// @lc code=end

// Warning: this is a manual question, the generated test code may be incorrect.
int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    ListNode* head;
    LeetCodeIO::scan(cin, head);
    int pos;
    LeetCodeIO::scan(cin, pos);

    Solution *obj = new Solution();
    auto res = obj->detectCycle(head, pos);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.50 144. Binary Tree Preorder Traversal (Easy)

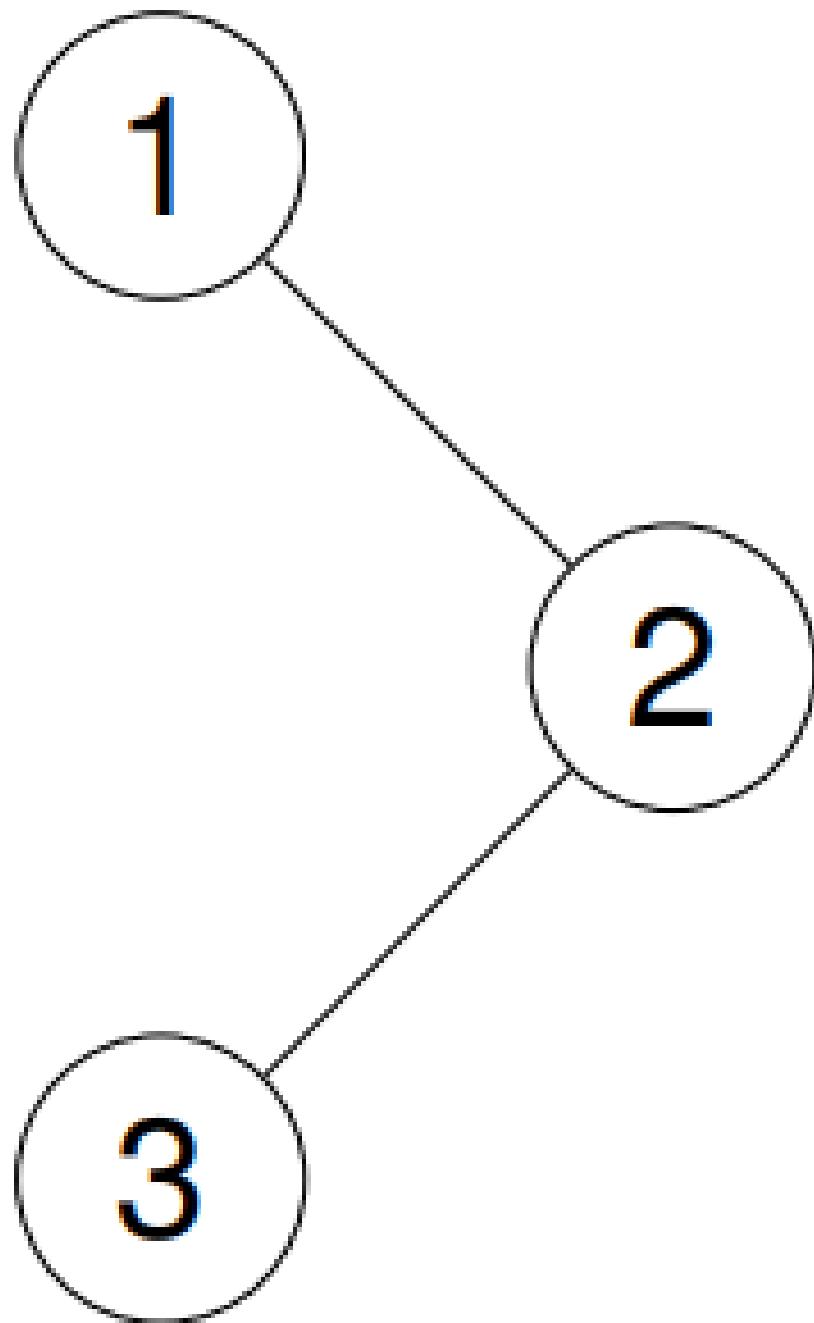
Given the root of a binary tree, return the preorder traversal of its nodes' values.

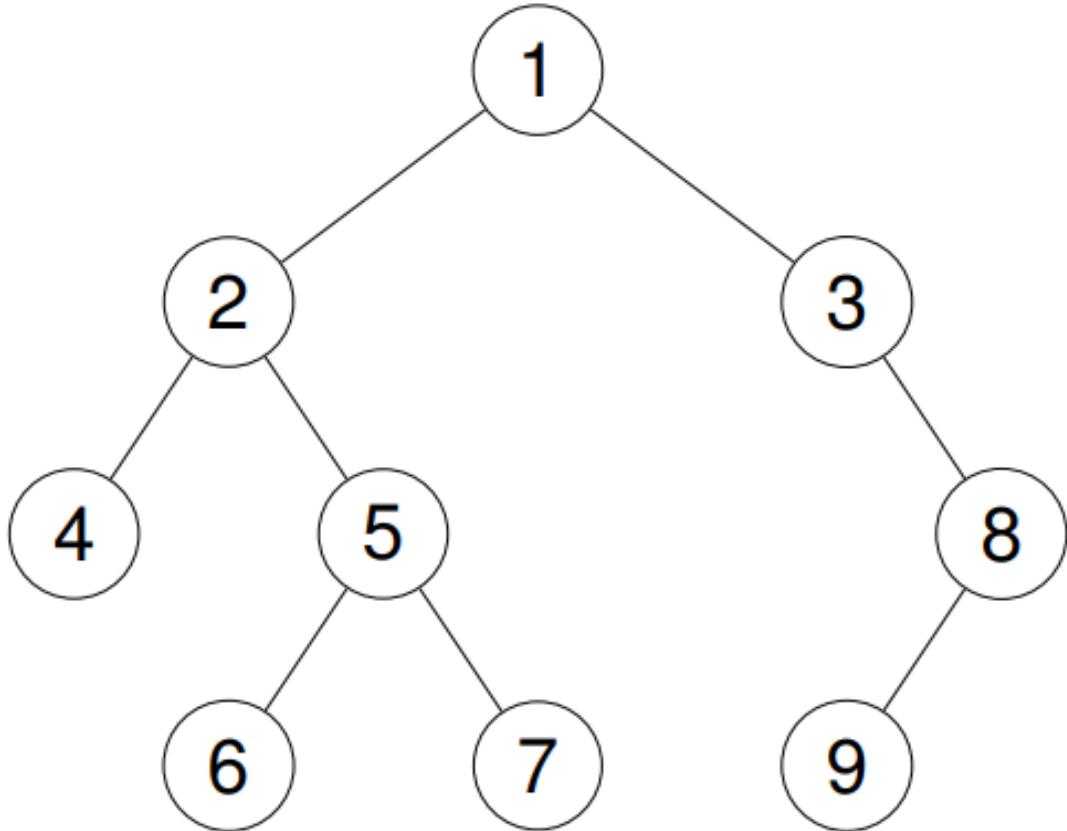
**Example 1:**

**Input:** root = [1,null,2,3]

**Output:**[1,2,3]

**Explanation:**



**Example 2:****Input:** root = [1,2,3,4,5,null,8,null,null,6,7,9]**Output:**[1,2,4,5,6,7,3,8,9]**Explanation:****Example 3:****Input:** root = []**Output:**[]**Example 4:****Input:** root = [1]**Output:**[1]**Constraints:**

- The number of nodes in the tree is in the range [0, 100].
- $-100 < \text{Node.val} \leq 100$

**Follow up:** Recursive solution is trivial, could you do it iteratively?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/binary-tree-preorder-traversal/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> preorderTraversal(TreeNode* root) {
        ...
    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->preorderTraversal(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.51 145. Binary Tree Postorder Traversal (Easy)

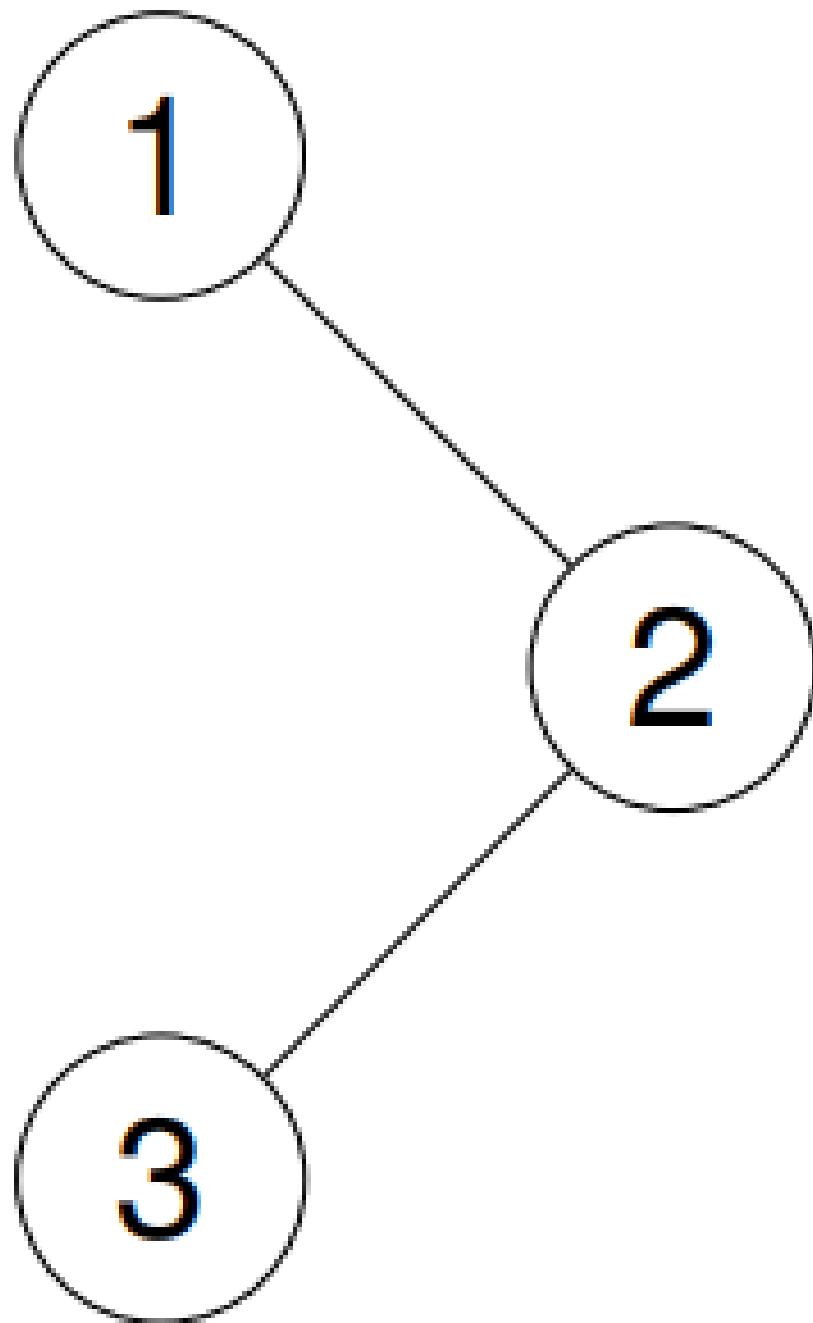
Given the root of a binary tree, return the postorder traversal of its nodes' values.

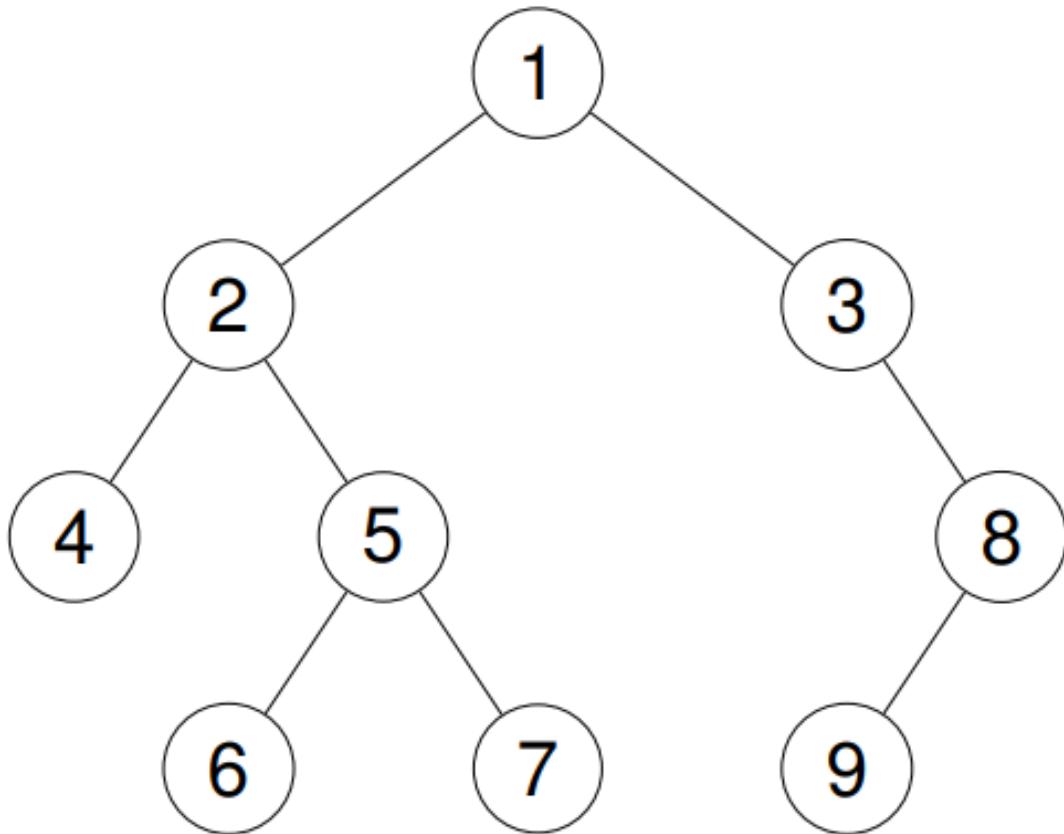
**Example 1:**

**Input:** root = [1,null,2,3]

**Output:**[3,2,1]

**Explanation:**



**Example 2:****Input:** root = [1,2,3,4,5,null,8,null,null,6,7,9]**Output:**[4,6,7,5,2,9,8,3,1]**Explanation:****Example 3:****Input:** root = []**Output:**[]**Example 4:****Input:** root = [1]**Output:**[1]**Constraints:**

- The number of the nodes in the tree is in the range [0, 100].
- $-100 < \text{Node.val} \leq 100$

**Follow up:** Recursive solution is trivial, could you do it iteratively?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/binary-tree-postorder-traversal/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> postorderTraversal(TreeNode* root) {
        ...
    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->postorderTraversal(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.52 146. LRU Cache (Medium)

Design a data structure that follows the constraints of a **Least Recently Used (LRU) cache**.

Implement the LRUCache class:

- `LRUCache(int capacity)` Initialize the LRU cache with **positive** size `capacity`.
- `int get(int key)` Return the value of the key if the key exists, otherwise return -1.
- `void put(int key, int value)` Update the value of the key if the key exists. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the `capacity` from this operation, **evict** the least recently used key.

The functions `get` and `put` must each run in  $O(1)$  average time complexity.

**Example 1:**

**Input**

```
["LRUCache", "put", "put", "get", "put", "get", "put", "get", "put", "get"]
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]
```

**Output**

```
[null, null, null, 1, null, -1, null, -1, 3, 4]
```

**Explanation**

```
LRUCache lRUCache = new LRUCache(2);
lRUCache.put(1, 1); // cache is {1=1}
lRUCache.put(2, 2); // cache is {1=1, 2=2}
lRUCache.get(1); // return 1
lRUCache.put(3, 3); // LRU key was 2, evicts key 2, cache is {1=1, 3=3}
lRUCache.get(2); // returns -1 (not found)
lRUCache.put(4, 4); // LRU key was 1, evicts key 1, cache is {4=4, 3=3}
lRUCache.get(1); // return -1 (not found)
lRUCache.get(3); // return 3
lRUCache.get(4); // return 4
```

**Constraints:**

- $1 < \text{capacity} \leq 3000$ =
- $0 < \text{key} \leq 10^4$ =
- $0 < \text{value} \leq 10^5$ =
- At most  $2 * 10$  calls will be made to `get` and `put`.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/lru-cache/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class LRUCache {
public:
    LRUCache(int capacity) {

    }

    int get(int key) {

    }

    void put(int key, int value) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    LRUCache *obj;
    const unordered_map<string, function<void()>> methods = {
        { "LRUCache", [&]() {
            int capacity; LeetCodeIO::scan(cin, capacity); cin.ignore();
            obj = new LRUCache(capacity);
            out_stream << "null,";
        } },
        { "get", [&]() {
            int key; LeetCodeIO::scan(cin, key); cin.ignore();
            LeetCodeIO::print(out_stream, obj->get(key)); out_stream << ',';
        } },
        { "put", [&]() {
            int key; LeetCodeIO::scan(cin, key); cin.ignore();
            int value; LeetCodeIO::scan(cin, value); cin.ignore();
            obj->put(key, value);
        } }
    };
}
```

```
    out_stream << "null,";
} },
};

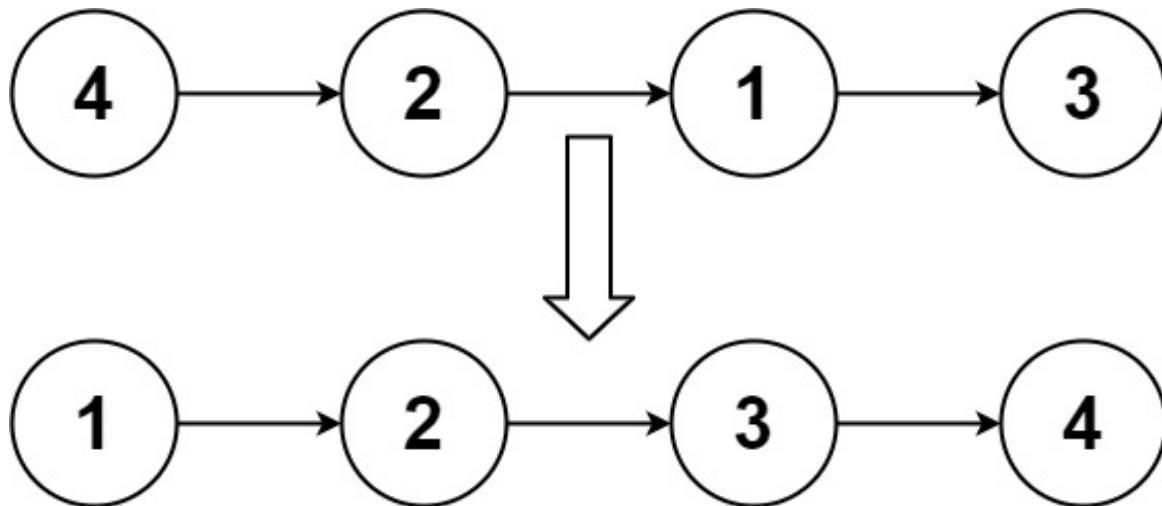
cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

delete obj;
return 0;
}
```

## 8.53 148. Sort List (Medium)

Given the head of a linked list, return the list after sorting it in **ascending order**.

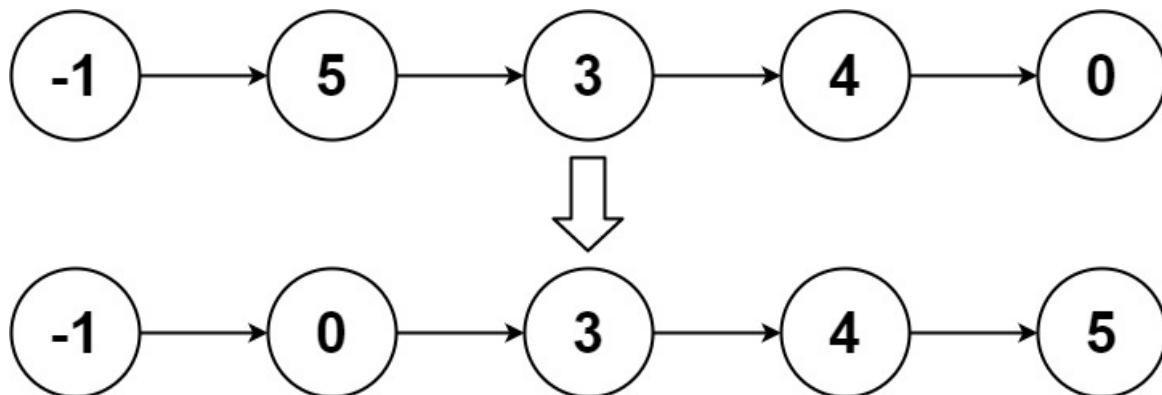
**Example 1:**



Input: head = [4,2,1,3]

Output: [1,2,3,4]

**Example 2:**



Input: head = [-1,5,3,4,0]

Output: [-1,0,3,4,5]

**Example 3:**

Input: head = []

Output: []

### Constraints:

- The number of nodes in the list is in the range  $[0, 5 * 10]$ .
- $-10^5 < \text{Node.val} \leq 10^5$

**Follow up:** Can you sort the linked list in  $O(n \log n)$  time and  $O(1)$  memory (i.e. constant space)?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sort-list/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    ListNode* sortList(ListNode* head) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    ListNode* head;
    LeetCodeIO::scan(cin, head);

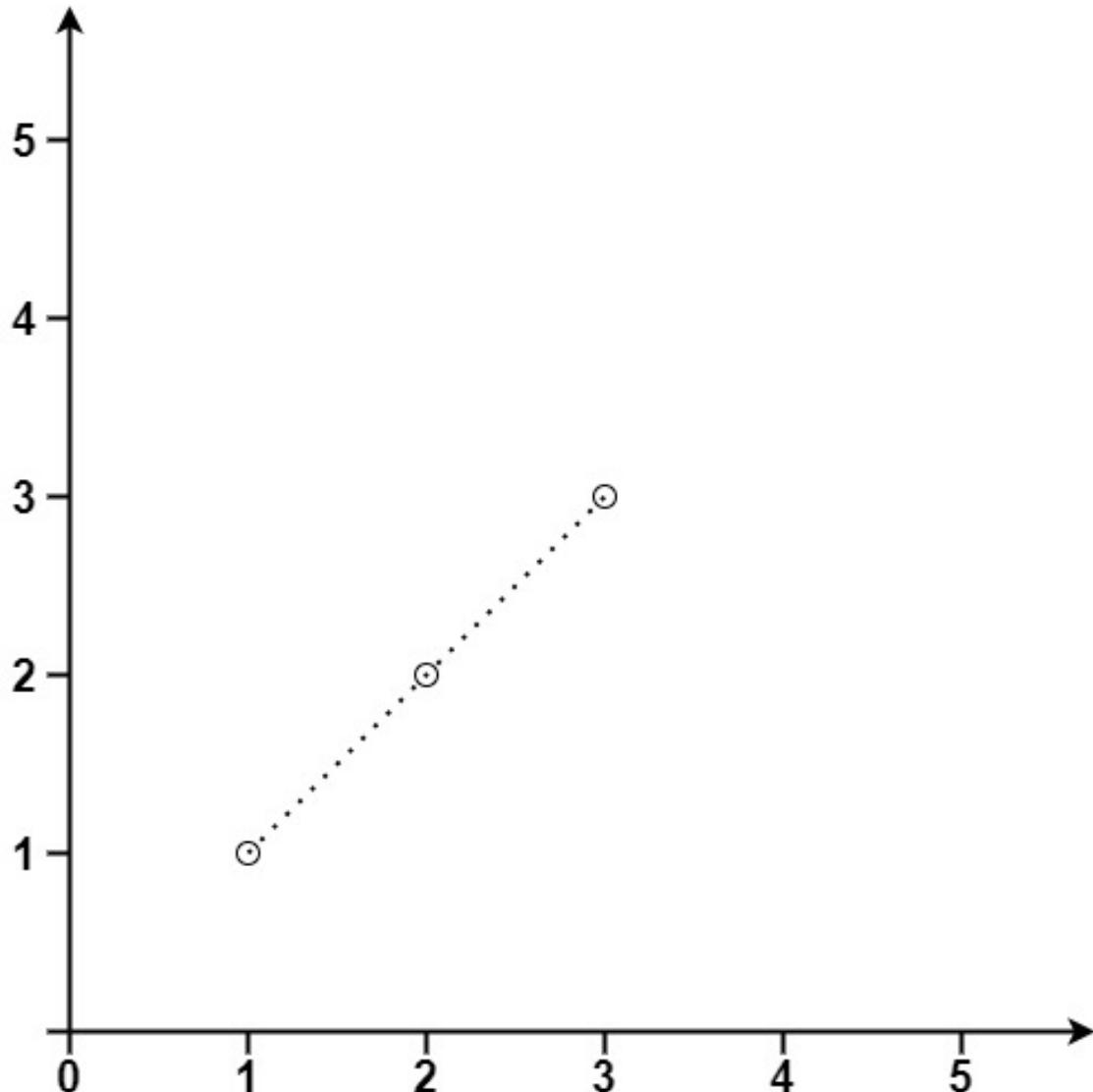
    Solution *obj = new Solution();
    auto res = obj->sortList(head);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.54 149. Max Points on a Line (Hard)

Given an array of points where `points[i] = [x, y]` represents a point on the X-Y plane, return the maximum number of points that lie on the same straight line.

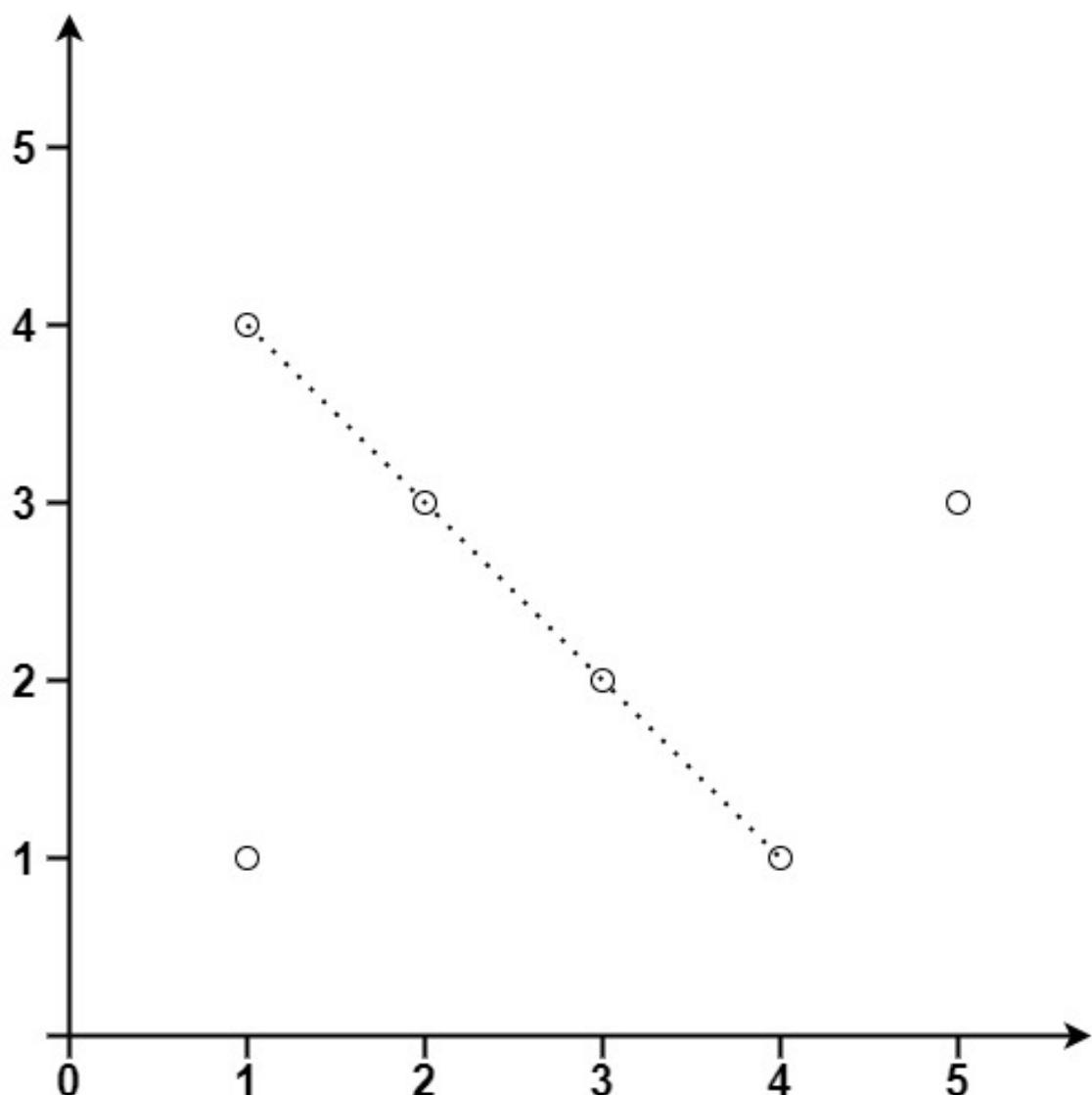
**Example 1:**



Input: `points = [[1,1],[2,2],[3,3]]`

Output: 3

**Example 2:**



Input: `points = [[1,1],[3,2],[5,3],[4,1],[2,3],[1,4]]`

Output: 4

**Constraints:**

- $1 < \text{points.length} \leq 300$
- $\text{points}[i].length = 2$
- $-10 \leq x, y \leq 10^4$
- All the points are **unique**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/max-points-on-a-line/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxPoints(vector<vector<int>>& points) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> points;
    LeetCodeIO::scan(cin, points);

    Solution *obj = new Solution();
    auto res = obj->maxPoints(points);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.55 154. Find Minimum in Rotated Sorted Array II (Hard)

Suppose an array of length  $n$  sorted in ascending order is **rotated** between 1 and  $n$  times. For example, the array `nums = [0, 1, 4, 4, 5, 6, 7]` might become:

- `[4, 5, 6, 7, 0, 1, 4]` if it was rotated 4 times.
- `[0, 1, 4, 4, 5, 6, 7]` if it was rotated 7 times.

Notice that **rotating** an array `[a[0], a[1], a[2], ..., a[n-1]]` 1 time results in the array `[a[n-1], a[0], a[1], a[2], ..., a[n-2]]`.

Given the sorted rotated array `nums` that may contain **duplicates**, return the minimum element of this array.

You must decrease the overall operation steps as much as possible.

### Example 1:

**Input:** `nums = [1, 3, 5]`

**Output:** 1

### Example 2:

**Input:** `nums = [2, 2, 2, 0, 1]`

**Output:** 0

### Constraints:

- $n = \text{nums.length} =$
- $1 < n \leq 5000 =$
- $-5000 < \text{nums}[i] \leq 5000 =$
- `nums` is sorted and rotated between 1 and  $n$  times.

**Follow up:** This problem is similar to [Find Minimum in Rotated Sorted Array](#), but `nums` may contain **duplicates**. Would this affect the runtime complexity? How and why?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/find-minimum-in-rotated-sorted-array-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findMin(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->findMin(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.56 155. Min Stack (Medium)

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

Implement the `MinStack` class:

- `MinStack()` initializes the stack object.
- `void push(int val)` pushes the element `val` onto the stack.
- `void pop()` removes the element on the top of the stack.
- `int top()` gets the top element of the stack.
- `int getMin()` retrieves the minimum element in the stack.

You must implement a solution with  $O(1)$  time complexity for each function.

**Example 1:**

**Input**

```
["MinStack","push","push","push","getMin","pop","top","getMin"]
[],[-2],[0],[-3],[],[],[],[]]
```

**Output**

```
[null,null,null,null,-3,null,0,-2]
```

**Explanation**

```
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();    // return 0
minStack.getMin(); // return -2
```

**Constraints:**

- $-2^{31} \leq \text{val} \leq 2^{31} - 1$
- Methods `pop`, `top` and `getMin` operations will always be called on **non-empty** stacks.
- At most  $3 * 10$  calls will be made to `push`, `pop`, `top`, and `getMin`.

## 題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/min-stack/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class MinStack {
public:
    MinStack() {

    }

    void push(int val) {

    }

    void pop() {

    }

    int top() {

    }

    int getMin() {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    MinStack *obj;
    const unordered_map<string, function<void()>> methods = {
        { "MinStack", [&]() {
            cin.ignore();
            obj = new MinStack();
            out_stream << "null,";
        } },
    };
}
```

```

{ "push", [&]() {
    int val; LeetCodeIO::scan(cin, val); cin.ignore();
    obj->push(val);
    out_stream << "null,";
} },
{ "pop", [&]() {
    cin.ignore();
    obj->pop();
    out_stream << "null,";
} },
{ "top", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->top()); out_stream << ',';
} },
{ "getMin", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->getMin()); out_stream << ',';
} },
};

cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

delete obj;
return 0;
}

```

## 8.57 162. Find Peak Element (Medium)

A peak element is an element that is strictly greater than its neighbors.

Given a **0-indexed** integer array `nums`, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that `nums[-1] = nums[n] = -∞`. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in  $O(\log n)$  time.

### Example 1:

Input: `nums = [1,2,3,1]`

Output: 2

Explanation: 3 is a peak element and your function should return the index number 2.

### Example 2:

Input: `nums = [1,2,1,3,5,6,4]`

Output: 5

Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

### Constraints:

- $1 < \text{nums.length} \leq 1000$
- $-2^{31} < \text{nums}[i] \leq 2^{31} - 1$
- $\text{nums}[i] \neq \text{nums}[i + 1]$  for all valid  $i$ .

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/find-peak-element/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findPeakElement(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->findPeakElement(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.58 167. Two Sum II - Input Array Is Sorted (Medium)

Given a **1-indexed** array of integers `numbers` that is already **sorted in non-decreasing order**, find two numbers such that they add up to a specific target number. Let these two numbers be `numbers[index1]` and `numbers[index2]` where  $1 < \text{index}_1 < \text{index}_2 \leq \text{numbers.length}$ .

Return the indices of the two numbers, `index1` and `index2`, **added by one** as an integer array `[index1, index2]` of length 2.

The tests are generated such that there is **exactly one solution**. You **may not** use the same element twice.

Your solution must use only constant extra space.

### Example 1:

`Input: numbers = [2,7,11,15], target = 9`

`Output: [1,2]`

`Explanation: The sum of 2 and 7 is 9. Therefore, index = 1, index = 2. We return [1, 2].`

### Example 2:

`Input: numbers = [2,3,4], target = 6`

`Output: [1,3]`

`Explanation: The sum of 2 and 4 is 6. Therefore index = 1, index = 3. We return [1, 3].`

### Example 3:

`Input: numbers = [-1,0], target = -1`

`Output: [1,2]`

`Explanation: The sum of -1 and 0 is -1. Therefore index = 1, index = 2. We return [1, 2].`

### Constraints:

- $2 < \text{numbers.length} \leq 3 * 10^4$
- $-1000 < \text{numbers}[i] \leq 1000$
- `numbers` is sorted in **non-decreasing order**.
- $-1000 < \text{target} \leq 1000$
- The tests are generated such that there is **exactly one solution**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/two-sum-ii-input-array-is-sorted/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> twoSum(vector<int>& numbers, int target) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> numbers;
    LeetCodeIO::scan(cin, numbers);
    int target;
    LeetCodeIO::scan(cin, target);

    Solution *obj = new Solution();
    auto res = obj->twoSum(numbers, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.59 168. Excel Sheet Column Title (Easy)

Given an integer `columnNumber`, return its corresponding column title as it appears in an Excel sheet.

For example:

A -> 1  
B -> 2  
C -> 3  
...  
Z -> 26  
AA -> 27  
AB -> 28  
...

### Example 1:

Input: `columnNumber = 1`

Output: "A"

### Example 2:

Input: `columnNumber = 28`

Output: "AB"

### Example 3:

Input: `columnNumber = 701`

Output: "ZY"

### Constraints:

- $1 < \text{columnNumber} \leq 2^{31} - 1$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/excel-sheet-column-title/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    string convertToTitle(int columnNumber) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int columnNumber;
    LeetCodeIO::scan(cin, columnNumber);

    Solution *obj = new Solution();
    auto res = obj->convertToTitle(columnNumber);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.60 169. Majority Element (Easy)

Given an array `nums` of size  $n$ , return the majority element.

The majority element is the element that appears more than  $n / 2$  times. You may assume that the majority element always exists in the array.

### Example 1:

Input: `nums = [3,2,3]`

Output: 3

### Example 2:

Input: `nums = [2,2,1,1,1,2,2]`

Output: 2

### Constraints:

- $n = \text{nums.length} =$
- $1 < n \leq 5 * 10^4 =$
- $-10 \leq \text{nums}[i] \leq 10^9 =$
- The input is generated such that a majority element will exist in the array.

**Follow-up:** Could you solve the problem in linear time and in  $O(1)$  space?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/majority-element/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int majorityElement(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->majorityElement(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.61 188. Best Time to Buy and Sell Stock IV (Hard)

You are given an integer array `prices` where `prices[i]` is the price of a given stock on the  $i^{\text{th}}$  day, and an integer  $k$ . Find the maximum profit you can achieve. You may complete at most  $k$  transactions: i.e. you may buy at most  $k$  times and sell at most  $k$  times.

**Note:** You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

### Example 1:

Input:  $k = 2$ , `prices = [2, 4, 1]`

Output: 2

Explanation: Buy on day 1 (price = 2) and sell on day 2 (price = 4), profit =  $4 - 2 = 2$ .

### Example 2:

Input:  $k = 2$ , `prices = [3, 2, 6, 5, 0, 3]`

Output: 7

Explanation: Buy on day 2 (price = 2) and sell on day 3 (price = 6), profit =  $6 - 2 = 4$ . Then buy on day 5 (price = 0) and sell on day 6 (price = 3), profit =  $3 - 0 = 3$ .

### Constraints:

- $1 < k \leq 100$ =
- $1 < \text{prices.length} \leq 1000$ =
- $0 < \text{prices}[i] \leq 1000$ =

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iv/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxProfit(int k, vector<int>& prices) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int k;
    LeetCodeIO::scan(cin, k);
    vector<int> prices;
    LeetCodeIO::scan(cin, prices);

    Solution *obj = new Solution();
    auto res = obj->maxProfit(k, prices);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.62 190. Reverse Bits (Easy)

Reverse bits of a given 32 bits signed integer.

**Example 1:**

**Input:** n = 43261596

**Output:** 964176192

**Explanation:**

Integer	Binary
43261596	00000010100101000001111010011100
964176192	00111001011110000010100101000000

**Example 2:**

**Input:** n = 2147483644

**Output:** 1073741822

**Explanation:**

Integer	Binary
2147483644	01111111111111111111111111111100
1073741822	00111111111111111111111111111110

**Constraints:**

- $0 < n \leq 2^{31} - 2$ =
- n is even.

**Follow up:** If this function is called many times, how would you optimize it?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/reverse-bits/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int reverseBits(int n) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->reverseBits(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.63 198. House Robber (Medium)

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connected and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array `nums` representing the amount of money of each house, return the maximum amount of money you can rob tonight without alerting the police.

### Example 1:

Input: `nums = [1,2,3,1]`

Output: 4

Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).

Total amount you can rob =  $1 + 3 = 4$ .

### Example 2:

Input: `nums = [2,7,9,3,1]`

Output: 12

Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).

Total amount you can rob =  $2 + 9 + 1 = 12$ .

### Constraints:

- $1 < \text{nums.length} \leq 100$ =
- $0 < \text{nums}[i] \leq 400$ =

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/house-robber/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int rob(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->rob(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.64 202. Happy Number (Easy)

Write an algorithm to determine if a number  $n$  is happy.

A **happy number** is a number defined by the following process:

- Starting with any positive integer, replace the number by the sum of the squares of its digits.
- Repeat the process until the number equals 1 (where it will stay), or it **loops endlessly in a cycle** which does not include 1.
- Those numbers for which this process **ends in 1** are happy.

Return =true=if  $=n=$ is a happy number, and =false=if not.

### Example 1:

Input:  $n = 19$

Output: true

Explanation:

$$1^2 + 9^2 = 82$$

$$8^2 + 2^2 = 68$$

$$6^2 + 8^2 = 100$$

$$1^2 + 0^2 + 0^2 = 1$$

### Example 2:

Input:  $n = 2$

Output: false

### Constraints:

- $1 < n \leq 2^{31} - 1 =$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/happy-number/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isHappy(int n) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->isHappy(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.65 205. Isomorphic Strings (Easy)

Given two strings  $s$  and  $t$ , determine if they are isomorphic.

Two strings  $s$  and  $t$  are isomorphic if the characters in  $s$  can be replaced to get  $t$ .

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

### Example 1:

**Input:**  $s = \text{"egg"}$ ,  $t = \text{"add"}$

**Output:** true

### Explanation:

The strings  $s$  and  $t$  can be made identical by:

- Mapping ' $e$ ' to ' $a$ '.
- Mapping ' $g$ ' to ' $d$ '.

### Example 2:

**Input:**  $s = \text{"foo"}$ ,  $t = \text{"bar"}$

**Output:** false

### Explanation:

The strings  $s$  and  $t$  can not be made identical as ' $o$ ' needs to be mapped to both ' $a$ ' and ' $r$ '.

### Example 3:

**Input:**  $s = \text{"paper"}$ ,  $t = \text{"title"}$

**Output:** true

### Constraints:

- $1 < s.length \leq 5 * 10^4$
- $t.length = s.length$
- $s$  and  $t$  consist of any valid ascii character.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/isomorphic-strings/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isIsomorphic(string s, string t) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);
    string t;
    LeetCodeIO::scan(cin, t);

    Solution *obj = new Solution();
    auto res = obj->isIsomorphic(s, t);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.66 208. Implement Trie (Prefix Tree) (Medium)

A **trie** (pronounced as "try") or **prefix tree** is a tree data structure used to efficiently store and retrieve keys in a dataset of strings. There are various applications of this data structure, such as autocomplete and spellchecker.

Implement the Trie class:

- `Trie()` Initializes the trie object.
- `void insert(String word)` Inserts the string `word` into the trie.
- `boolean search(String word)` Returns `true` if the string `word` is in the trie (i.e., was inserted before), and `false` otherwise.
- `boolean startsWith(String prefix)` Returns `true` if there is a previously inserted string `word` that has the prefix `prefix`, and `false` otherwise.

### Example 1:

Input

```
["Trie", "insert", "search", "search", "startsWith", "insert", "search"]
[[], ["apple"], ["apple"], ["app"], ["app"], ["app"], ["app"]]
```

Output

```
[null, null, true, false, true, null, true]
```

Explanation

```
Trie trie = new Trie();
trie.insert("apple");
trie.search("apple");    // return True
trie.search("app");      // return False
trie.startsWith("app"); // return True
trie.insert("app");
trie.search("app");      // return True
```

### Constraints:

- `1 < word.length, prefix.length <= 2000`
- `word` and `prefix` consist only of lowercase English letters.
- At most `3 * 10` calls **in total** will be made to `insert`, `search`, and `startsWith`.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/implement-trie-prefix-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Trie {
public:
    Trie() {

    }

    void insert(string word) {

    }

    bool search(string word) {

    }

    bool startsWith(string prefix) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    Trie *obj;
    const unordered_map<string, function<void()>> methods = {
        { "Trie", [&]() {
            cin.ignore();
            obj = new Trie();
            out_stream << "null,";
        } },
        { "insert", [&]() {
            string word; LeetCodeIO::scan(cin, word); cin.ignore();
            obj->insert(word);
            out_stream << "null,";
        } },
    };
}
```

```
    } },
    { "search", [&]() {
        string word; LeetCodeIO::scan(cin, word); cin.ignore();
        LeetCodeIO::print(out_stream, obj->search(word)); out_stream << ',';
    } },
    { "startsWith", [&]() {
        string prefix; LeetCodeIO::scan(cin, prefix); cin.ignore();
        LeetCodeIO::print(out_stream, obj->startsWith(prefix)); out_stream << ',';
    } },
};

cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

delete obj;
return 0;
}
```

## 8.67 210. Course Schedule II (Medium)

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] = [a, b]` indicates that you **must** take course `b` first if you want to take course `a`.

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

Return the ordering of courses you should take to finish all courses. If there are many valid answers, return **any** of them.

If it is impossible to finish all courses, return **an empty array**.

### Example 1:

Input: `numCourses = 2, prerequisites = [[1,0]]`

Output: `[0,1]`

Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is `[0,1]`.

### Example 2:

Input: `numCourses = 4, prerequisites = [[1,0],[2,0],[3,1],[3,2]]`

Output: `[0,2,1,3]`

Explanation: There are a total of 4 courses to take. To take course 3 you should have finished both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0.

So one correct course order is `[0,1,2,3]`. Another correct ordering is `[0,2,1,3]`.

### Example 3:

Input: `numCourses = 1, prerequisites = []`

Output: `[0]`

### Constraints:

- `1 < numCourses <= 2000`=
- `0 < prerequisites.length <= numCourses * (numCourses - 1)`=
- `prerequisites[i].length = 2`=
- `0 < a, b < numCourses`=
- `a != b`=
- All the pairs `[a, b]` are **distinct**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/course-schedule-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> findOrder(int numCourses, vector<vector<int>>& prerequisites) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int numCourses;
    LeetCodeIO::scan(cin, numCourses);
    vector<vector<int>> prerequisites;
    LeetCodeIO::scan(cin, prerequisites);

    Solution *obj = new Solution();
    auto res = obj->findOrder(numCourses, prerequisites);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.68 213. House Robber II (Medium)

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed. All houses at this place are **arranged in a circle**. That means the first house is the neighbor of the last one. Meanwhile, adjacent houses have a security system connected, and it will automatically contact the police if two adjacent houses were broken into on the same night.

Given an integer array `nums` representing the amount of money of each house, return the maximum amount of money you can rob tonight **without alerting the police**.

### Example 1:

Input: `nums = [2,3,2]`

Output: 3

Explanation: You cannot rob house 1 (`money = 2`) and then rob house 3 (`money = 2`), because they are adjacent houses.

### Example 2:

Input: `nums = [1,2,3,1]`

Output: 4

Explanation: Rob house 1 (`money = 1`) and then rob house 3 (`money = 3`).

Total amount you can rob =  $1 + 3 = 4$ .

### Example 3:

Input: `nums = [1,2,3]`

Output: 3

### Constraints:

- $1 < \text{nums.length} \leq 100$
- $0 < \text{nums}[i] \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/house-robber-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int rob(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->rob(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.69 215. Kth Largest Element in an Array (Medium)

Given an integer array `nums` and an integer `k`, return the `=kth=`largest element in the array.

Note that it is the `k` largest element in the sorted order, not the `k` distinct element.

Can you solve it without sorting?

### Example 1:

Input: `nums = [3,2,1,5,6,4]`, `k = 2`

Output: 5

### Example 2:

Input: `nums = [3,2,3,1,2,4,5,5,6]`, `k = 4`

Output: 4

### Constraints:

- `1 < k <= nums.length <= 105`
- `-10 < nums[i] <= 104`

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/kth-largest-element-in-an-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findKthLargest(vector<int>& nums, int k) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int k;
    LeetCodeIO::scan(cin, k);

    Solution *obj = new Solution();
    auto res = obj->findKthLargest(nums, k);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.70 217. Contains Duplicate (Easy)

Given an integer array `nums`, return `true` if any value appears **at least twice** in the array, and return `false` if every element is distinct.

### Example 1:

**Input:** `nums = [1,2,3,1]`

**Output:** `true`

**Explanation:**

The element 1 occurs at the indices 0 and 3.

### Example 2:

**Input:** `nums = [1,2,3,4]`

**Output:** `false`

**Explanation:**

All elements are distinct.

### Example 3:

**Input:** `nums = [1,1,1,3,3,4,3,2,4,2]`

**Output:** `true`

**Constraints:**

- $1 < \text{nums.length} \leq 10^5$ =
- $-10^9 < \text{nums}[i] \leq 10^9$ =

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/contains-duplicate/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool containsDuplicate(vector<int>& nums) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->containsDuplicate(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.71 218. The Skyline Problem (Hard)

A city's **skyline** is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return the **skyline** formed by these buildings collectively.

The geometric information of each building is given in the array `buildings` where `buildings[i] = [left, right, height]`:

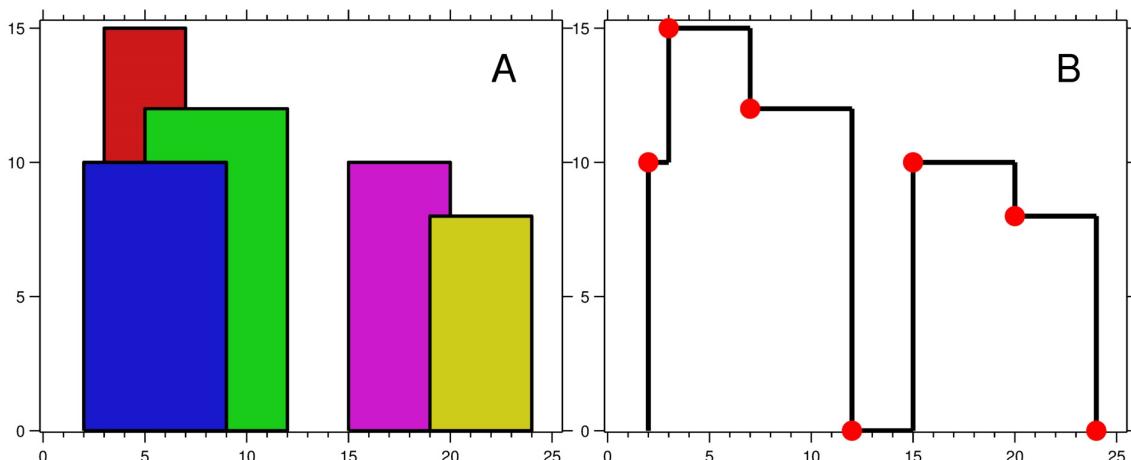
- `left` is the x coordinate of the left edge of the `i` building.
- `right` is the x coordinate of the right edge of the `i` building.
- `height` is the height of the `i` building.

You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.

The **skyline** should be represented as a list of "key points" **sorted by their x-coordinate** in the form `[[x, y], [x, y], ...]`. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in the list, which always has a y-coordinate 0 and is used to mark the skyline's termination where the rightmost building ends. Any ground between the leftmost and rightmost buildings should be part of the skyline's contour.

**Note:** There must be no consecutive horizontal lines of equal height in the output skyline. For instance, `[..., [2 3], [4 5], [7 5], [11 5], [12 7], ...]` is not acceptable; the three lines of height 5 should be merged into one in the final output as such: `[..., [2 3], [4 5], [12 7], ...]`

### Example 1:



Input: `buildings = [[2,9,10], [3,7,15], [5,12,12], [15,20,10], [19,24,8]]`

Output: `[[2,10], [3,15], [7,12], [12,0], [15,10], [20,8], [24,0]]`

Explanation:

Figure A shows the buildings of the input.

Figure B shows the skyline formed by those buildings. The red points in figure B represent the key points in the output list.

### Example 2:

Input: `buildings = [[0,2,3], [2,5,3]]`

Output: `[[0,3], [5,0]]`

### Constraints:

- `1 < buildings.length <= 10^4`
- `0 < left < right <= 2^{31} - 1`

- $1 < \text{height} \leq 2^{31} - 1$
- `buildings` is sorted by `left` in non-decreasing order.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/the-skyline-problem/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> buildings;
    LeetCodeIO::scan(cin, buildings);

    Solution *obj = new Solution();
    auto res = obj->getSkyline(buildings);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.72 221. Maximal Square (Medium)

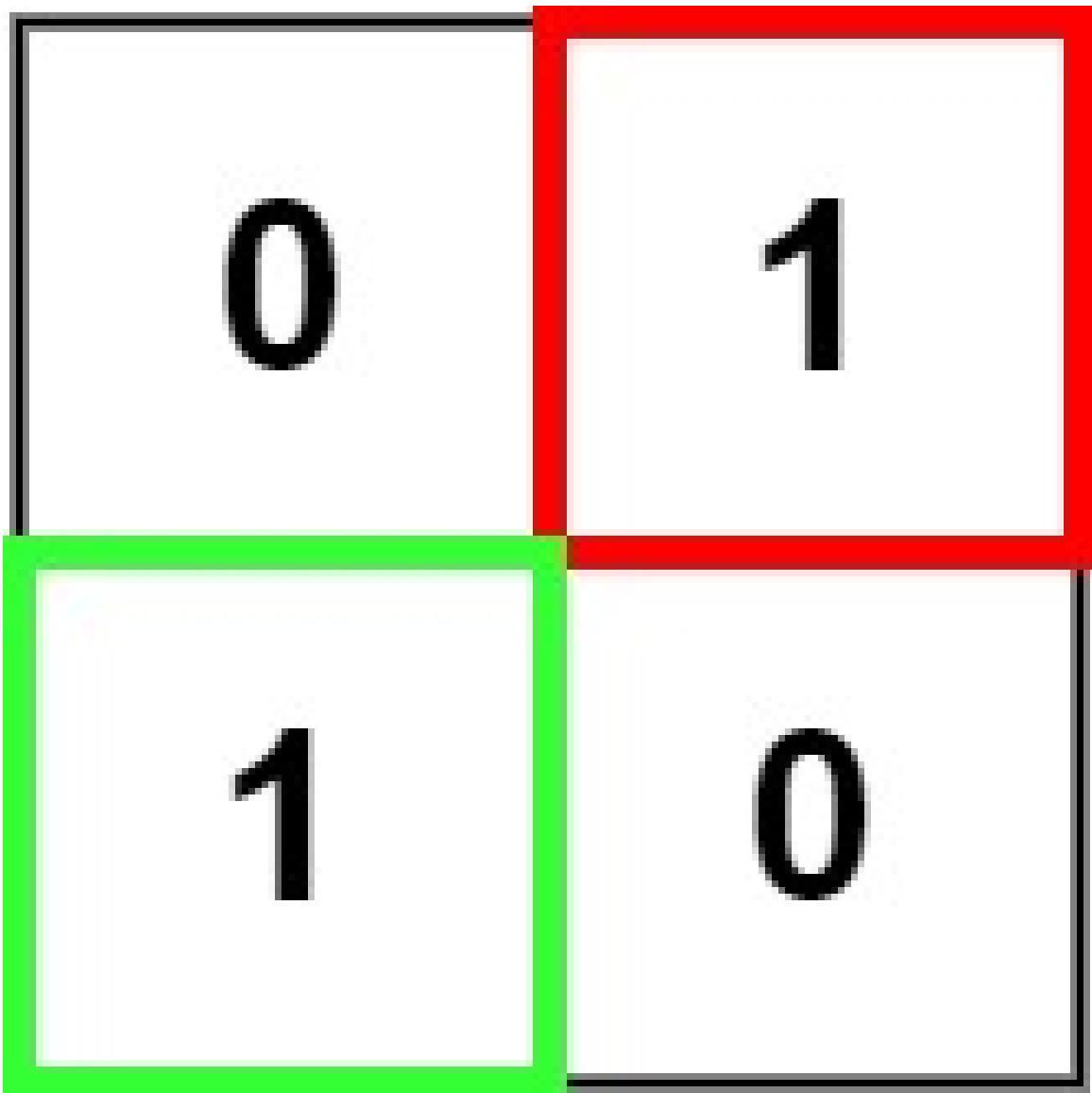
Given an  $m \times n$  binary matrix filled with 0's and 1's, find the largest square containing only 1's and return its area.

**Example 1:**

1	0	1	0	0
1	0	1	1	1
1	1	1	1	1
1	0	0	1	0

```
Input: matrix =
[[1,0,1,0,0],[1,0,1,1,1],[1,1,1,1,1],[1,0,0,1,0]]
Output: 4
```

**Example 2:**



Input: `matrix = [["0","1"],["1","0"]]`

Output: 1

**Example 3:**

Input: `matrix = [["0"]]`

Output: 0

**Constraints:**

- `m` = `matrix.length`=
- `n` = `matrix[i].length`=
- `1 < m, n <= 300`=
- `matrix[i][j]` is '0' or '1'.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/maximal-square/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maximalSquare(vector<vector<char>>& matrix) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<char>> matrix;
    LeetCodeIO::scan(cin, matrix);

    Solution *obj = new Solution();
    auto res = obj->maximalSquare(matrix);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.73 225. Implement Stack using Queues (Easy)

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the functions of a normal stack ( push, top, pop, and empty).

Implement the MyStack class:

- `void push(int x)` Pushes element x to the top of the stack.
- `int pop()` Removes the element on the top of the stack and returns it.
- `int top()` Returns the element on the top of the stack.
- `boolean empty()` Returns true if the stack is empty, false otherwise.

**Notes:**

- You must use **only** standard operations of a queue, which means that only `push to back`, `peek/pop from front`, `size` and `is empty` operations are valid.
- Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deque (double-ended queue) as long as you use only a queue's standard operations.

**Example 1:**

**Input**

```
["MyStack", "push", "push", "top", "pop", "empty"]
[[], [1], [2], [], [], []]
```

**Output**

```
[null, null, null, 2, 2, false]
```

**Explanation**

```
MyStack myStack = new MyStack();
myStack.push(1);
myStack.push(2);
myStack.top(); // return 2
myStack.pop(); // return 2
myStack.empty(); // return False
```

**Constraints:**

- $1 < x \leq 9$
- At most 100 calls will be made to push, pop, top, and empty.
- All the calls to pop and top are valid.

**Follow-up:** Can you implement the stack using only one queue?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/implement-stack-using-queues/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class MyStack {
public:
    MyStack() {

    }

    void push(int x) {

    }

    int pop() {

    }

    int top() {

    }

    bool empty() {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    MyStack *obj;
    const unordered_map<string, function<void()>> methods = {
        { "MyStack", [&]() {
            cin.ignore();
            obj = new MyStack();
            out_stream << "null,";
        } },
    };
}
```

```

{ "push", [&]() {
    int x; LeetCodeIO::scan(cin, x); cin.ignore();
    obj->push(x);
    out_stream << "null,";
} },
{ "pop", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->pop()); out_stream << ',';
} },
{ "top", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->top()); out_stream << ',';
} },
{ "empty", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->empty()); out_stream << ',';
} },
};

cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

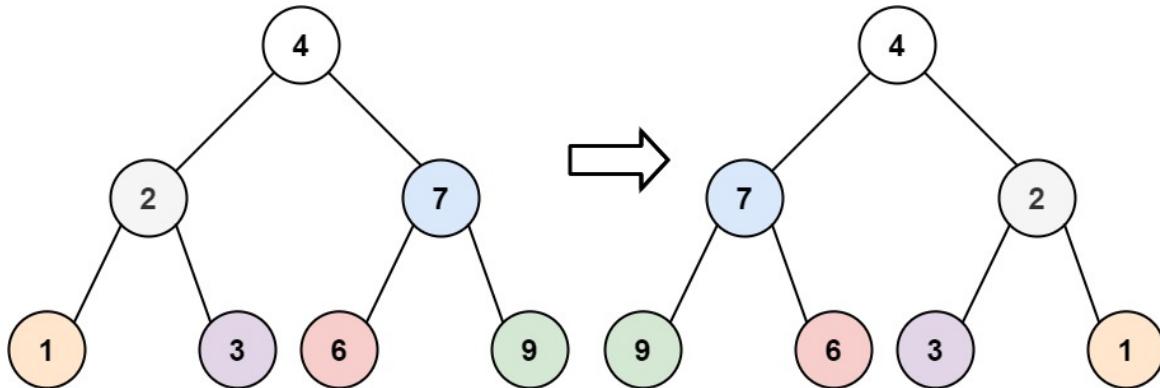
delete obj;
return 0;
}

```

## 8.74 226. Invert Binary Tree (Easy)

Given the root of a binary tree, invert the tree, and return its root.

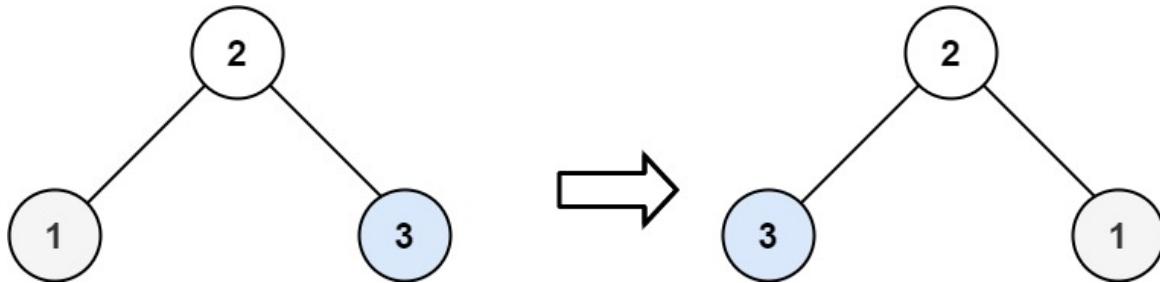
**Example 1:**



Input: `root = [4,2,7,1,3,6,9]`

Output: `[4,7,2,9,6,3,1]`

**Example 2:**



Input: `root = [2,1,3]`

Output: `[2,3,1]`

**Example 3:**

Input: `root = []`

Output: `[]`

**Constraints:**

- The number of nodes in the tree is in the range  $[0, 100]$ .
- $-100 < \text{Node.val} \leq 100$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/invert-binary-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* invertTree(TreeNode* root) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->invertTree(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.75 227. Basic Calculator II (Medium)

Given a string  $s$  which represents an expression, evaluate this expression and return its value.

The integer division should truncate toward zero.

You may assume that the given expression is always valid. All intermediate results will be in the range of  $[-2^{31}, 2^{31} - 1]$ .

**Note:** You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as `eval()`.

### Example 1:

Input:  $s = "3+2*2"$

Output: 7

### Example 2:

Input:  $s = "3/2"$

Output: 1

### Example 3:

Input:  $s = "3+5 / 2"$

Output: 5

### Constraints:

- $1 < s.length \leq 3 * 10^5$
- $s$  consists of integers and operators ('+', '-', '\*', '/') separated by some number of spaces.
- $s$  represents a **valid expression**.
- All the integers in the expression are non-negative integers in the range  $[0, 2^{31} - 1]$ .
- The answer is **guaranteed** to fit in a **32-bit integer**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/basic-calculator-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int calculate(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->calculate(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.76 232. Implement Queue using Stacks (Easy)

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the functions of a normal queue ( push, peek, pop, and empty).

Implement the MyQueue class:

- `void push(int x)` Pushes element x to the back of the queue.
- `int pop()` Removes the element from the front of the queue and returns it.
- `int peek()` Returns the element at the front of the queue.
- `boolean empty()` Returns true if the queue is empty, false otherwise.

**Notes:**

- You must use **only** standard operations of a stack, which means only `push to top`, `peek/pop from top`, `size`, and `is empty` operations are valid.
- Depending on your language, the stack may not be supported natively. You may simulate a stack using a list or deque (double-ended queue) as long as you use only a stack's standard operations.

**Example 1:**

**Input**

```
["MyQueue", "push", "push", "peek", "pop", "empty"]
[[], [1], [2], [], [], []]
```

**Output**

```
[null, null, null, 1, 1, false]
```

**Explanation**

```
MyQueue myQueue = new MyQueue();
myQueue.push(1); // queue is: [1]
myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)
myQueue.peek(); // return 1
myQueue.pop(); // return 1, queue is [2]
myQueue.empty(); // return false
```

**Constraints:**

- $1 < x \leq 9$
- At most 100 calls will be made to push, pop, peek, and empty.
- All the calls to pop and peek are valid.

**Follow-up:** Can you implement the queue such that each operation is **amortized**  $O(1)$  time complexity? In other words, performing  $n$  operations will take overall  $O(n)$  time even if one of those operations may take longer.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/implement-queue-using-stacks/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class MyQueue {
public:
    MyQueue() {

    }

    void push(int x) {

    }

    int pop() {

    }

    int peek() {

    }

    bool empty() {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    MyQueue *obj;
    const unordered_map<string, function<void()>> methods = {
        { "MyQueue", [&]() {
            cin.ignore();
            obj = new MyQueue();
            out_stream << "null,";
        } },
    };
}
```

```

{ "push", [&]() {
    int x; LeetCodeIO::scan(cin, x); cin.ignore();
    obj->push(x);
    out_stream << "null,";
} },
{ "pop", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->pop()); out_stream << ',';
} },
{ "peek", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->peek()); out_stream << ',';
} },
{ "empty", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->empty()); out_stream << ',';
} },
};

cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

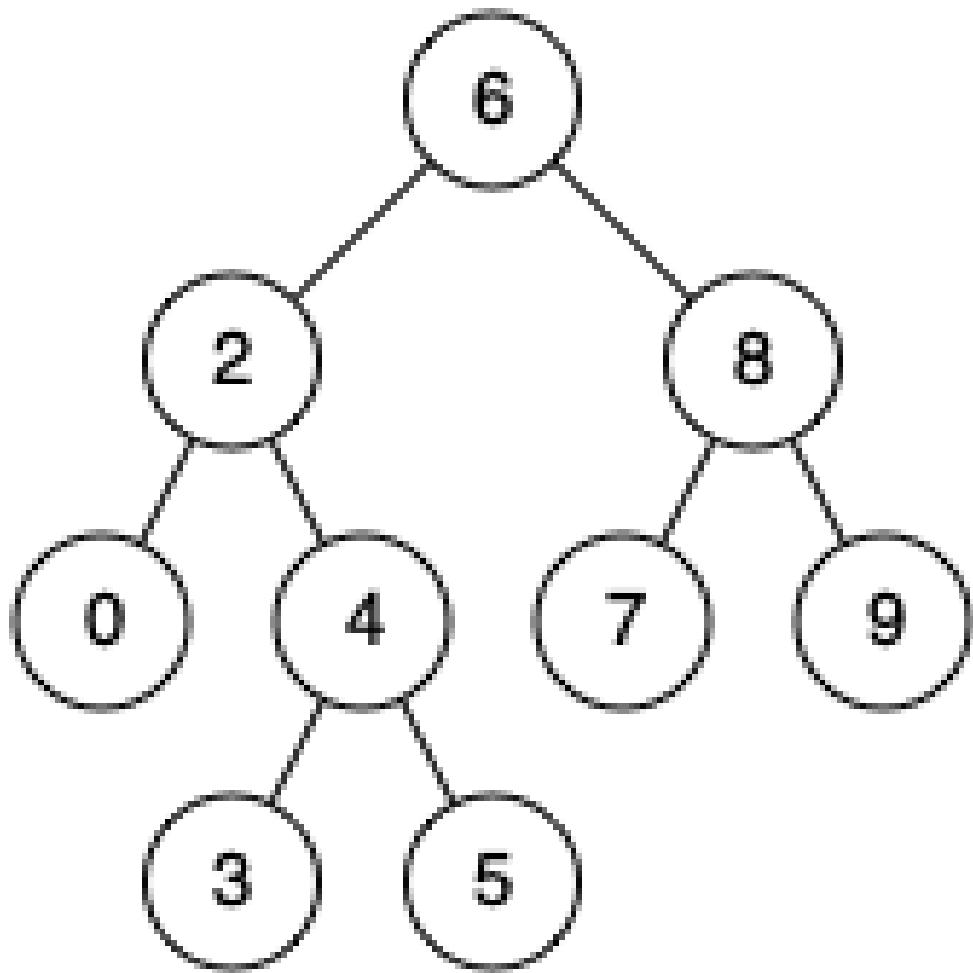
delete obj;
return 0;
}

```

## 8.77 235. Lowest Common Ancestor of a Binary Search Tree (Medium)

Given a binary search tree (BST), find the lowest common ancestor (LCA) node of two given nodes in the BST. According to the [definition of LCA on Wikipedia](#): "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)."

**Example 1:**

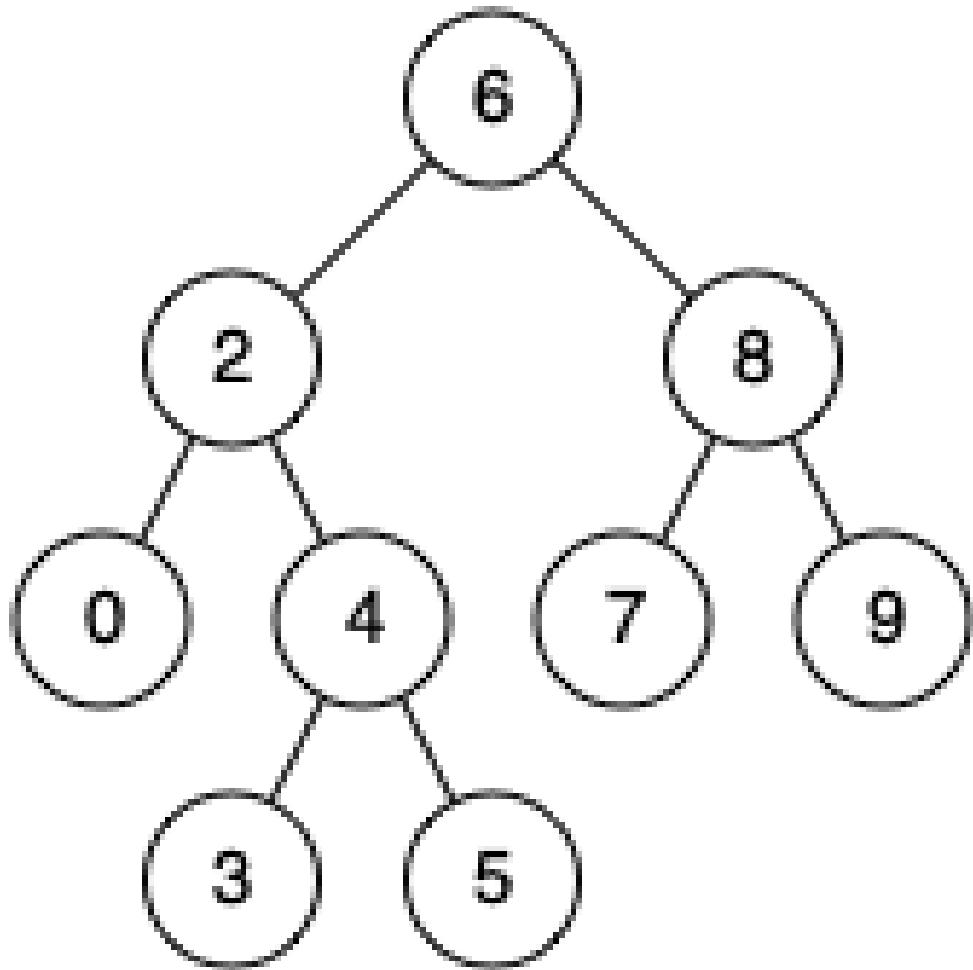


Input: `root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8`

Output: 6

Explanation: The LCA of nodes 2 and 8 is 6.

**Example 2:**



Input: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4

Output: 2

Explanation: The LCA of nodes 2 and 4 is 2, since a node can be a descendant of itself according to the LCA definition.

#### Example 3:

Input: root = [2,1], p = 2, q = 1

Output: 2

#### Constraints:

- The number of nodes in the tree is in the range [2, 10].
- $-10^9 < \text{Node.val} \leq 10^9$
- All `Node.val` are **unique**.
- $p \neq q$
- $p$  and  $q$  will exist in the BST.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        }

};

// @lc code=end

// Warning: this is a manual question, the generated test code may be incorrect.
int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    int p;
    LeetCodeIO::scan(cin, p);
    int q;
    LeetCodeIO::scan(cin, q);

    Solution *obj = new Solution();
    auto res = obj->lowestCommonAncestor(root, p, q);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

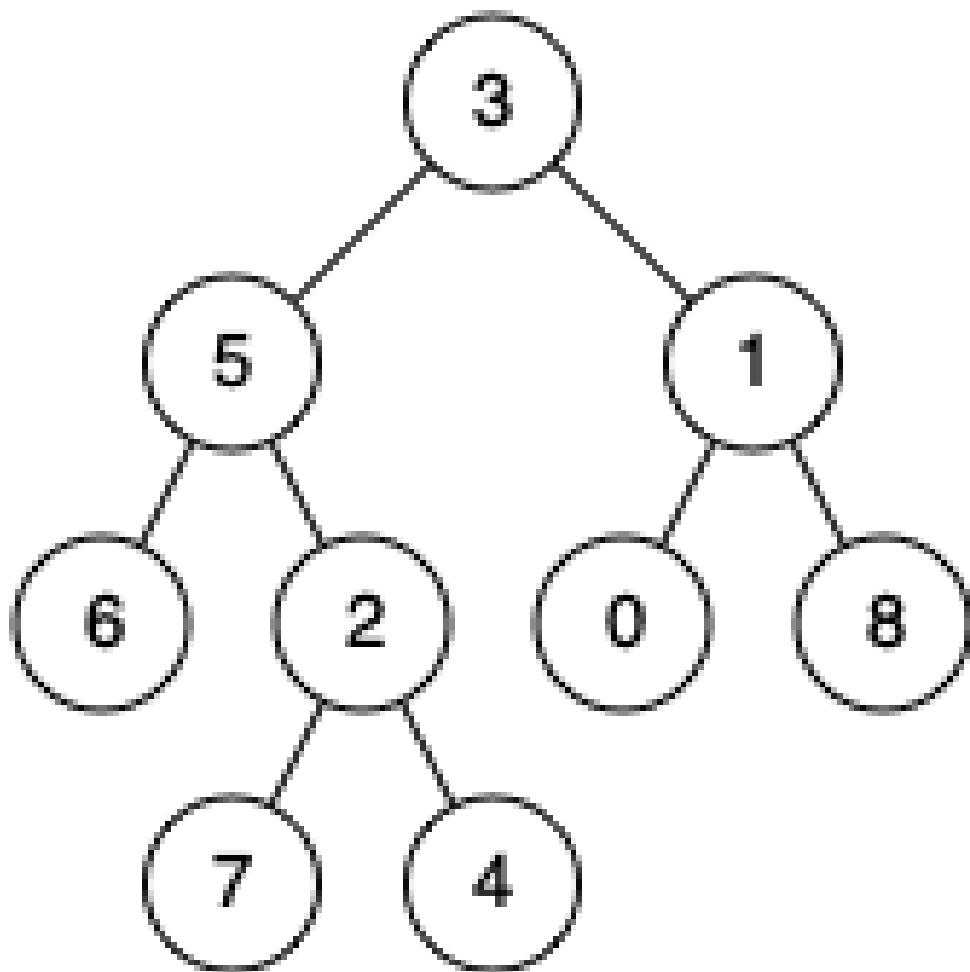
    delete obj;
    return 0;
}
```

## 8.78 236. Lowest Common Ancestor of a Binary Tree (Medium)

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](#): "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)."

**Example 1:**

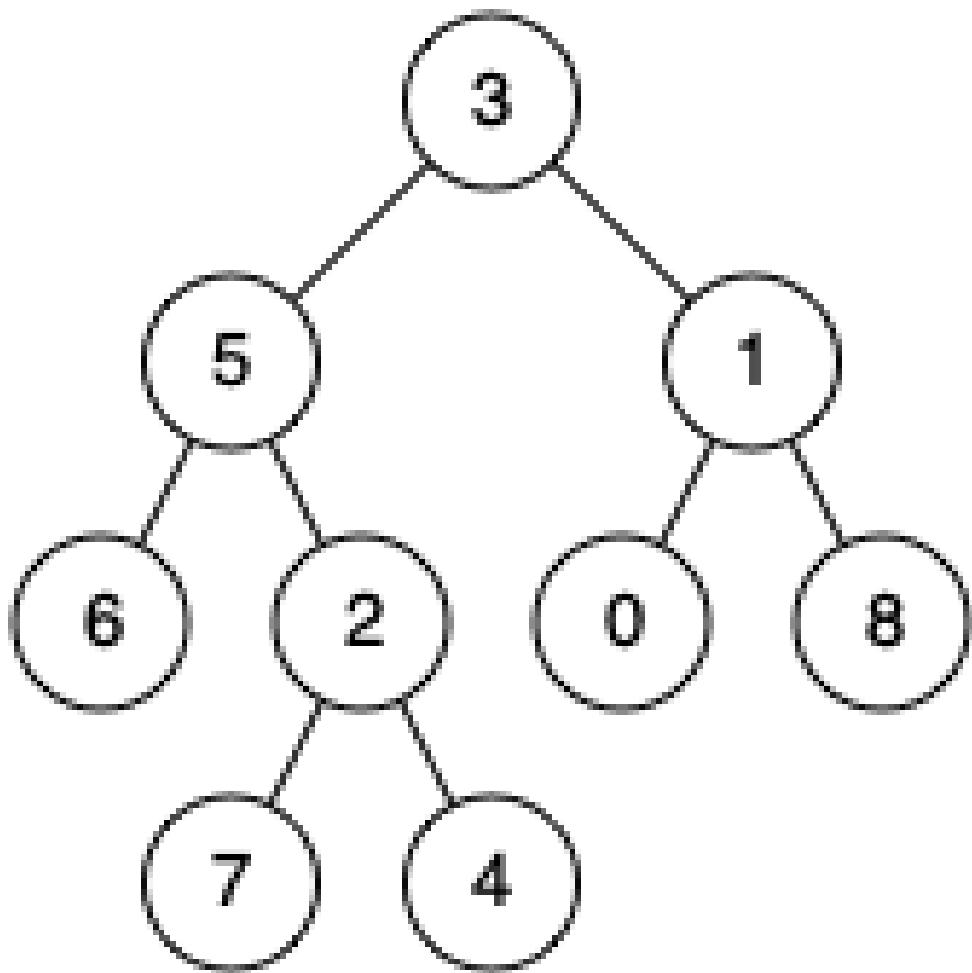


Input: `root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1`

Output: 3

Explanation: The LCA of nodes 5 and 1 is 3.

**Example 2:**



Input: root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 4

Output: 5

Explanation: The LCA of nodes 5 and 4 is 5, since a node can be a descendant of itself according to the LCA definition.

#### Example 3:

Input: root = [1,2], p = 1, q = 2

Output: 1

#### Constraints:

- The number of nodes in the tree is in the range [2, 10].
- $-10^9 < \text{Node.val} \leq 10^9$
- All `Node.val` are **unique**.
- $p \neq q$
- $p$  and  $q$  will exist in the tree.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
        }

};

// @lc code=end

// Warning: this is a manual question, the generated test code may be incorrect.
int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    int p;
    LeetCodeIO::scan(cin, p);
    int q;
    LeetCodeIO::scan(cin, q);

    Solution *obj = new Solution();
    auto res = obj->lowestCommonAncestor(root, p, q);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.79 238. Product of Array Except Self (Medium)

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.

### Example 1:

Input: `nums = [1,2,3,4]`

Output: `[24,12,8,6]`

### Example 2:

Input: `nums = [-1,1,0,-3,3]`

Output: `[0,0,9,0,0]`

### Constraints:

- $2 < \text{nums.length} \leq 10^5$
- $-30 < \text{nums}[i] \leq 30$
- The input is generated such that `answer[i]` is **guaranteed** to fit in a **32-bit** integer.

**Follow up:** Can you solve the problem in  $O(1)$  extra space complexity? (The output array **does not** count as extra space for space complexity analysis.)

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/product-of-array-except-self/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> productExceptSelf(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->productExceptSelf(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.80 239. Sliding Window Maximum (Hard)

You are given an array of integers `nums`, there is a sliding window of size `k` which is moving from the very left of the array to the very right. You can only see the `k` numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

### Example 1:

Input: `nums = [1,3,-1,-3,5,3,6,7]`, `k = 3`

Output: `[3,3,5,5,6,7]`

Explanation:

Window position	Max
-----	-----
[1 3 -1] -3 5 3 6 7	3
1 [3 -1 -3] 5 3 6 7	3
1 3 [-1 -3 5] 3 6 7	5
1 3 -1 [-3 5 3] 6 7	5
1 3 -1 -3 [5 3 6] 7	6
1 3 -1 -3 5 [3 6 7]	7

### Example 2:

Input: `nums = [1]`, `k = 1`

Output: `[1]`

### Constraints:

- `1 < nums.length <= 10^5`=
- `-10 < nums[i] <= 10^4`=
- `1 < k <= nums.length`=

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sliding-window-maximum/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> maxSlidingWindow(vector<int>& nums, int k) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int k;
    LeetCodeIO::scan(cin, k);

    Solution *obj = new Solution();
    auto res = obj->maxSlidingWindow(nums, k);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.81 240. Search a 2D Matrix II (Medium)

Write an efficient algorithm that searches for a value `target` in an  $m \times n$  integer matrix `matrix`. This matrix has the following properties:

- Integers in each row are sorted in ascending from left to right.
- Integers in each column are sorted in ascending from top to bottom.

**Example 1:**

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

```
Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]],  
target = 5  
Output: true
```

**Example 2:**

1	4	7	11	15
2	5	8	12	19
3	6	9	16	22
10	13	14	17	24
18	21	23	26	30

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 20

Output: false

#### Constraints:

- m = matrix.length=
- n = matrix[i].length=
- 1 < n, m <= 300=
- -10 < matrix[i][j] <= 10<sup>9</sup>=
- All the integers in each row are **sorted** in ascending order.
- All the integers in each column are **sorted** in ascending order.
- -10 < target <= 10<sup>9</sup>=

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/search-a-2d-matrix-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        }

};

// @lc code=end

// Warning: this is a manual question, the generated test code may be incorrect.
int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> matrix;
    LeetCodeIO::scan(cin, matrix);
    int target;
    LeetCodeIO::scan(cin, target);

    Solution *obj = new Solution();
    auto res = obj->searchMatrix(matrix, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.82 241. Different Ways to Add Parentheses (Medium)

Given a string **expression** of numbers and operators, return all possible results from computing all the different possible ways to group numbers and operators. You may return the answer in **any order**.

The test cases are generated such that the output values fit in a 32-bit integer and the number of different results does not exceed 10 .

### Example 1:

**Input:** expression = "2-1-1"

**Output:** [0,2]

**Explanation:**

$$((2-1)-1) = 0$$

$$(2-(1-1)) = 2$$

### Example 2:

**Input:** expression = "2\*3-4\*5"

**Output:** [-34,-14,-10,-10,10]

**Explanation:**

$$(2*(3-(4*5))) = -34$$

$$((2*3)-(4*5)) = -14$$

$$((2*(3-4))*5) = -10$$

$$(2*((3-4)*5)) = -10$$

$$(((2*3)-4)*5) = 10$$

### Constraints:

- $1 < \text{expression.length} \leq 20$ =
- **expression** consists of digits and the operator '+', '-', and '\*'.
- All the integer values in the input expression are in the range [0, 99].
- The integer values in the input expression do not have a leading '-' or '+' denoting the sign.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/different-ways-to-add-parentheses/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> diffWaysToCompute(string expression) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string expression;
    LeetCodeIO::scan(cin, expression);

    Solution *obj = new Solution();
    auto res = obj->diffWaysToCompute(expression);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.83 242. Valid Anagram (Easy)

Given two strings  $s$  and  $t$ , return `true` if  $t$  is an anagram of  $s$ , and `false` otherwise.

**Example 1:**

**Input:**  $s = \text{"anagram"}$ ,  $t = \text{"nagaram"}$

**Output:** `true`

**Example 2:**

**Input:**  $s = \text{"rat"}$ ,  $t = \text{"car"}$

**Output:** `false`

**Constraints:**

- $1 < s.length, t.length \leq 5 * 10^4$
- $s$  and  $t$  consist of lowercase English letters.

**Follow up:** What if the inputs contain Unicode characters? How would you adapt your solution to such a case?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/valid-anagram/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isAnagram(string s, string t) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);
    string t;
    LeetCodeIO::scan(cin, t);

    Solution *obj = new Solution();
    auto res = obj->isAnagram(s, t);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

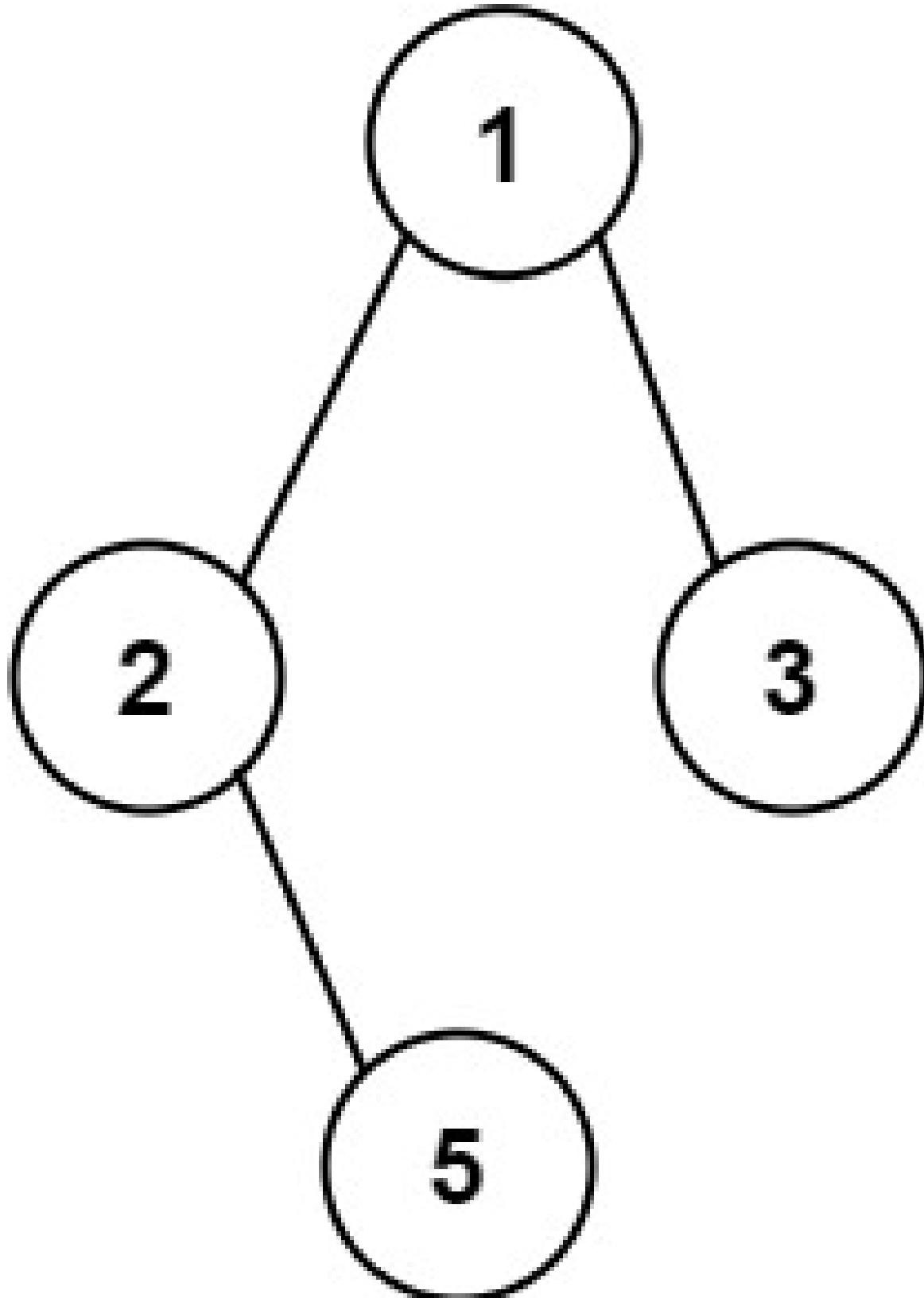
    delete obj;
    return 0;
}
```

## 8.84 257. Binary Tree Paths (Easy)

Given the root of a binary tree, return all root-to-leaf paths in **any order**.

A **leaf** is a node with no children.

**Example 1:**



Input: root = [1,2,3,null,5]

Output: ["1->2->5","1->3"]

**Example 2:**

Input: root = [1]

Output: ["1"]

**Constraints:**

- The number of nodes in the tree is in the range [1, 100].
- $-100 < \text{Node.val} \leq 100$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/binary-tree-paths/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<string> binaryTreePaths(TreeNode* root) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->binaryTreePaths(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.85 260. Single Number III (Medium)

Given an integer array `nums`, in which exactly two elements appear only once and all the other elements appear exactly twice. Find the two elements that appear only once. You can return the answer in **any order**.

You must write an algorithm that runs in linear runtime complexity and uses only constant extra space.

### Example 1:

Input: `nums = [1,2,1,3,2,5]`

Output: `[3,5]`

Explanation: `[5, 3]` is also a valid answer.

### Example 2:

Input: `nums = [-1,0]`

Output: `[-1,0]`

### Example 3:

Input: `nums = [0,1]`

Output: `[1,0]`

### Constraints:

- $2 < \text{nums.length} \leq 3 * 10^4$
- $-2^{31} < \text{nums}[i] \leq 2^{31} - 1$
- Each integer in `nums` will appear twice, only two integers will appear once.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/single-number-iii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> singleNumber(vector<int>& nums) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->singleNumber(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.86 268. Missing Number (Easy)

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.

**Example 1:**

**Input:** `nums = [3,0,1]`

**Output:** 2

**Explanation:**

$n = 3$  since there are 3 numbers, so all numbers are in the range  $[0, 3]$ . 2 is the missing number in the range since it does not appear in `nums`.

**Example 2:**

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

**Output:** 2

**Explanation:**

$n = 2$  since there are 2 numbers, so all numbers are in the range  $[0, 2]$ . 2 is the missing number in the range since it does not appear in `nums`.

**Example 3:**

**Input:** `nums = [9,6,4,2,3,5,7,0,1]`

**Output:** 8

**Explanation:**

$n = 9$  since there are 9 numbers, so all numbers are in the range  $[0, 9]$ . 8 is the missing number in the range since it does not appear in `nums`.

**Constraints:**

- $n = \text{nums.length} =$
- $1 < n \leq 10^4 =$
- $0 < \text{nums}[i] \leq n =$
- All the numbers of `nums` are **unique**.

**Follow up:** Could you implement a solution using only  $O(1)$  extra space complexity and  $O(n)$  runtime complexity?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/missing-number/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int missingNumber(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->missingNumber(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.87 279. Perfect Squares (Medium)

Given an integer  $n$ , return the least number of perfect square numbers that sum to  $n$ .

A **perfect square** is an integer that is the square of an integer; in other words, it is the product of some integer with itself.

For example, 1, 4, 9, and 16 are perfect squares while 3 and 11 are not.

### Example 1:

Input:  $n = 12$

Output: 3

Explanation:  $12 = 4 + 4 + 4$ .

### Example 2:

Input:  $n = 13$

Output: 2

Explanation:  $13 = 4 + 9$ .

### Constraints:

- $1 < n \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/perfect-squares/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int numSquares(int n) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->numSquares(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.88 287. Find the Duplicate Number (Medium)

Given an array of integers `nums` containing  $n + 1$  integers where each integer is in the range  $[1, n]$  inclusive.

There is only **one repeated number** in `nums`, return this repeated number.

You must solve the problem **without** modifying the array `nums` and using only constant extra space.

### Example 1:

Input: `nums` = [1,3,4,2,2]

Output: 2

### Example 2:

Input: `nums` = [3,1,3,4,2]

Output: 3

### Example 3:

Input: `nums` = [3,3,3,3,3]

Output: 3

### Constraints:

- $1 < n \leq 10^5$
- `nums.length = n + 1`
- $1 < \text{nums}[i] \leq n$
- All the integers in `nums` appear only **once** except for **precisely one integer** which appears **two or more** times.

### Follow up:

- How can we prove that at least one duplicate number must exist in `nums`?
- Can you solve the problem in linear runtime complexity?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/find-the-duplicate-number/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findDuplicate(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->findDuplicate(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.89 300. Longest Increasing Subsequence (Medium)

Given an integer array `nums`, return the length of the longest **strictly increasing subsequence**.

**Example 1:**

Input: `nums = [10,9,2,5,3,7,101,18]`

Output: 4

Explanation: The longest increasing subsequence is `[2,3,7,101]`, therefore the length is 4.

**Example 2:**

Input: `nums = [0,1,0,3,2,3]`

Output: 4

**Example 3:**

Input: `nums = [7,7,7,7,7,7,7]`

Output: 1

**Constraints:**

- $1 < \text{nums.length} \leq 2500$
- $-10 \leq \text{nums}[i] \leq 10^4$

**Follow up:** Can you come up with an algorithm that runs in  $O(n \log(n))$  time complexity?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/longest-increasing-subsequence/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->lengthOfLIS(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.90 303. Range Sum Query - Immutable (Easy)

Given an integer array `nums`, handle multiple queries of the following type:

1. Calculate the **sum** of the elements of `nums` between indices `left` and `right inclusive` where `left < right`=.

Implement the `NumArray` class:

- `NumArray(int[] nums)` Initializes the object with the integer array `nums`.
- `int sumRange(int left, int right)` Returns the **sum** of the elements of `nums` between indices `left` and `right inclusive` (i.e. `nums[left] + nums[left + 1] + ... + nums[right]`).

**Example 1:**

**Input**

```
["NumArray", "sumRange", "sumRange", "sumRange"]
[[[-2, 0, 3, -5, 2, -1]], [0, 2], [2, 5], [0, 5]]
```

**Output**

```
[null, 1, -1, -3]
```

**Explanation**

```
NumArray numArray = new NumArray([-2, 0, 3, -5, 2, -1]);
numArray.sumRange(0, 2); // return (-2) + 0 + 3 = 1
numArray.sumRange(2, 5); // return 3 + (-5) + 2 + (-1) = -1
numArray.sumRange(0, 5); // return (-2) + 0 + 3 + (-5) + 2 + (-1) = -3
```

**Constraints:**

- $1 < \text{nums.length} \leq 10^4$ =
- $-10^5 < \text{nums}[i] \leq 10^5$ =
- $0 < \text{left} \leq \text{right} < \text{nums.length}$ =
- At most 10 calls will be made to `sumRange`.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/range-sum-query-immutable/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class NumArray {
public:
    NumArray(vector<int>& nums) {

    }

    int sumRange(int left, int right) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    NumArray *obj;
    const unordered_map<string, function<void()>> methods = {
        { "NumArray", [&]() {
            vector<int> nums; LeetCodeIO::scan(cin, nums); cin.ignore();
            int numsSize; LeetCodeIO::scan(cin, numsSize); cin.ignore();
            obj = new NumArray(nums, numsSize);
            out_stream << "null,";
        } },
        { "sumRange", [&]() {
            int left; LeetCodeIO::scan(cin, left); cin.ignore();
            int right; LeetCodeIO::scan(cin, right); cin.ignore();
            LeetCodeIO::print(out_stream, obj->sumRange(left, right)); out_stream << ',';
        } },
    };
    cin >> ws;
    out_stream << '[';
    for (auto &method_name : method_names) {
        cin.ignore(2);
        methods.at(method_name)();
    }
}
```

```
}

cin.ignore();

out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

delete obj;
return 0;
}
```

## 8.91 304. Range Sum Query 2D - Immutable (Medium)

Given a 2D matrix `matrix`, handle multiple queries of the following type:

- Calculate the **sum** of the elements of `matrix` inside the rectangle defined by its **upper left corner** (`row1, col1`) and **lower right corner** (`row2, col2`).

Implement the `NumMatrix` class:

- `NumMatrix(int[][] matrix)` Initializes the object with the integer matrix `matrix`.
- `int sumRegion(int row1, int col1, int row2, int col2)` Returns the **sum** of the elements of `matrix` inside the rectangle defined by its **upper left corner** (`row1, col1`) and **lower right corner** (`row2, col2`).

You must design an algorithm where `sumRegion` works on  $O(1)$  time complexity.

**Example 1:**

3	0	1	4	2
5	6	3	2	1
1	2	0	1	5
4	1	0	1	7
1	0	3	0	5

**Input**

```
["NumMatrix", "sumRegion", "sumRegion", "sumRegion"]
[[[[3, 0, 1, 4, 2], [5, 6, 3, 2, 1], [1, 2, 0, 1, 5], [4, 1, 0, 1, 7], [1, 0, 3, 0, 5]], [2, 1, 4, 3], [1, 1, 2, 2], [1, 2, 2, 4]]]
```

**Output**

```
[null, 8, 11, 12]
```

#### Explanation

```
NumMatrix numMatrix = new NumMatrix([[3, 0, 1, 4, 2], [5, 6, 3, 2, 1], [1, 2, 0, 1, 5], [4, 1, 0, 1, 7], [1, 0, 3, 0, 5]]);  
numMatrix.sumRegion(2, 1, 4, 3); // return 8 (i.e sum of the red rectangle)  
numMatrix.sumRegion(1, 1, 2, 2); // return 11 (i.e sum of the green rectangle)  
numMatrix.sumRegion(1, 2, 2, 4); // return 12 (i.e sum of the blue rectangle)
```

#### Constraints:

- m = matrix.length=
- n = matrix[i].length=
- 1 < m, n <= 200=
- -10 < matrix[i][j] <= 10<sup>4</sup>=
- 0 < row1 <= row2 < m=
- 0 < col1 <= col2 < n=
- At most 10 calls will be made to sumRegion.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/range-sum-query-2d-immutable/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class NumMatrix {
public:
    NumMatrix(vector<vector<int>>& matrix) {

    }

    int sumRegion(int row1, int col1, int row2, int col2) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    NumMatrix *obj;
    const unordered_map<string, function<void()>> methods = {
        { "NumMatrix", [&]() {
            vector<vector<int>> matrix; LeetCodeIO::scan(cin, matrix); cin.ignore();
            int matrixRowSize; LeetCodeIO::scan(cin, matrixRowSize); cin.ignore();
            int matrixColSize; LeetCodeIO::scan(cin, matrixColSize); cin.ignore();
            obj = new NumMatrix(matrix, matrixRowSize, matrixColSize);
            out_stream << "null,";
        } },
        { "sumRegion", [&]() {
            int row1; LeetCodeIO::scan(cin, row1); cin.ignore();
            int col1; LeetCodeIO::scan(cin, col1); cin.ignore();
            int row2; LeetCodeIO::scan(cin, row2); cin.ignore();
            int col2; LeetCodeIO::scan(cin, col2); cin.ignore();
            LeetCodeIO::print(out_stream, obj->sumRegion(row1, col1, row2, col2)); out_stream << ',';
        } },
    };
    cin >> ws;
    out_stream << '[';
```

```
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}

cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

delete obj;
return 0;
}
```

## 8.92 307. Range Sum Query - Mutable (Medium)

Given an integer array `nums`, handle multiple queries of the following types:

1. **Update** the value of an element in `nums`.
2. Calculate the **sum** of the elements of `nums` between indices `left` and `right inclusive` where `left < right`=.

Implement the `NumArray` class:

- `NumArray(int[] nums)` Initializes the object with the integer array `nums`.
- `void update(int index, int val)` **Updates** the value of `nums[index]` to be `val`.
- `int sumRange(int left, int right)` Returns the **sum** of the elements of `nums` between indices `left` and `right inclusive` (i.e. `nums[left] + nums[left + 1] + ... + nums[right]`).

**Example 1:**

**Input**

```
["NumArray", "sumRange", "update", "sumRange"]
[[[1, 3, 5]], [0, 2], [1, 2], [0, 2]]
```

**Output**

```
[null, 9, null, 8]
```

**Explanation**

```
NumArray numArray = new NumArray([1, 3, 5]);
numArray.sumRange(0, 2); // return 1 + 3 + 5 = 9
numArray.update(1, 2); // nums = [1, 2, 5]
numArray.sumRange(0, 2); // return 1 + 2 + 5 = 8
```

**Constraints:**

- $1 < \text{nums.length} \leq 3 * 10^4$ =
- $-100 < \text{nums}[i] \leq 100$ =
- $0 < \text{index} < \text{nums.length}$ =
- $-100 < \text{val} \leq 100$ =
- $0 < \text{left} \leq \text{right} < \text{nums.length}$ =
- At most  $3 * 10$  calls will be made to `update` and `sumRange`.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/range-sum-query-mutable/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class NumArray {
public:
    NumArray(vector<int>& nums) {

    }

    void update(int index, int val) {

    }

    int sumRange(int left, int right) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    NumArray *obj;
    const unordered_map<string, function<void()>> methods = {
        { "NumArray", [&]() {
            vector<int> nums; LeetCodeIO::scan(cin, nums); cin.ignore();
            obj = new NumArray(nums);
            out_stream << "null,";
        } },
        { "update", [&]() {
            int index; LeetCodeIO::scan(cin, index); cin.ignore();
            int val; LeetCodeIO::scan(cin, val); cin.ignore();
            obj->update(index, val);
            out_stream << "null,";
        } },
        { "sumRange", [&]() {
            int left; LeetCodeIO::scan(cin, left); cin.ignore();
            out_stream << "null,";
        } }
    };
}
```

```
int right; LeetCodeIO::scan(cin, right); cin.ignore();
LeetCodeIO::print(out_stream, obj->sumRange(left, right)); out_stream << ',';
} },
};

cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

delete obj;
return 0;
}
```

## 8.93 309. Best Time to Buy and Sell Stock with Cooldown (Medium)

You are given an array `prices` where `prices[i]` is the price of a given stock on the `i` day.

Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times) with the following restrictions:

- After you sell your stock, you cannot buy stock on the next day (i.e., cooldown one day).

**Note:** You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).

### Example 1:

Input: `prices = [1,2,3,0,2]`

Output: 3

Explanation: `transactions = [buy, sell, cooldown, buy, sell]`

### Example 2:

Input: `prices = [1]`

Output: 0

### Constraints:

- `1 < prices.length <= 5000`
- `0 < prices[i] <= 1000`

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-cooldown/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxProfit(vector<int>& prices) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> prices;
    LeetCodeIO::scan(cin, prices);

    Solution *obj = new Solution();
    auto res = obj->maxProfit(prices);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.94 310. Minimum Height Trees (Medium)

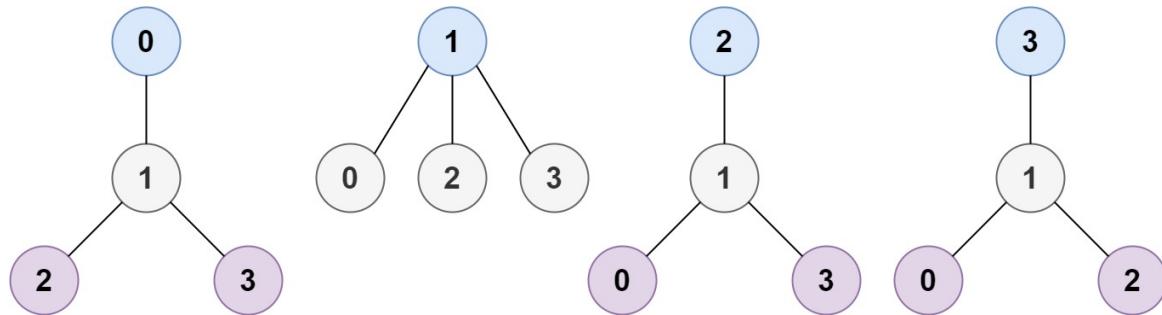
A tree is an undirected graph in which any two vertices are connected by *exactly* one path. In other words, any connected graph without simple cycles is a tree.

Given a tree of  $n$  nodes labelled from 0 to  $n - 1$ , and an array of  $n - 1$  edges where  $\text{edges}[i] = [a, b]$  indicates that there is an undirected edge between the two nodes  $a$  and  $b$  in the tree, you can choose any node of the tree as the root. When you select a node  $x$  as the root, the result tree has height  $h$ . Among all possible rooted trees, those with minimum height (i.e.  $\min(h)$ ) are called **minimum height trees** (MHTs).

Return a list of all MHTs' root labels. You can return the answer in **any order**.

The **height** of a rooted tree is the number of edges on the longest downward path between the root and a leaf.

**Example 1:**

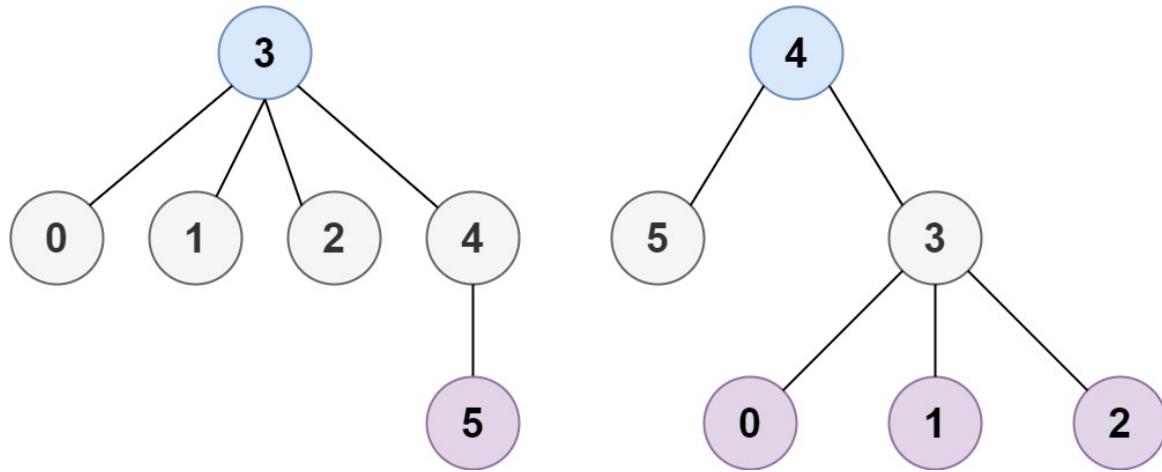


**Input:**  $n = 4$ ,  $\text{edges} = [[1,0], [1,2], [1,3]]$

**Output:** [1]

**Explanation:** As shown, the height of the tree is 1 when the root is the node with label 1 which is the only MHT.

**Example 2:**



**Input:**  $n = 6$ ,  $\text{edges} = [[3,0], [3,1], [3,2], [3,4], [5,4]]$

**Output:** [3,4]

**Constraints:**

- $1 < n \leq 2 * 10^4$
- $\text{edges.length} = n - 1$
- $0 < a, b < n$

- $a \neq b =$
- All the pairs  $(a, b)$  are distinct.
- The given input is **guaranteed** to be a tree and there will be **no repeated** edges.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/minimum-height-trees/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> findMinHeightTrees(int n, vector<vector<int>>& edges) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);
    vector<vector<int>> edges;
    LeetCodeIO::scan(cin, edges);

    Solution *obj = new Solution();
    auto res = obj->findMinHeightTrees(n, edges);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.95 312. Burst Balloons (Hard)

You are given  $n$  balloons, indexed from 0 to  $n - 1$ . Each balloon is painted with a number on it represented by an array `nums`. You are asked to burst all the balloons.

If you burst the  $i$  balloon, you will get  $\text{nums}[i - 1] * \text{nums}[i] * \text{nums}[i + 1]$  coins. If  $i - 1$  or  $i + 1$  goes out of bounds of the array, then treat it as if there is a balloon with a 1 painted on it.

Return the maximum coins you can collect by bursting the balloons wisely.

### Example 1:

Input: `nums = [3,1,5,8]`

Output: 167

Explanation:

`nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []`

`coins = 3*1*5 + 3*5*8 + 1*3*8 + 1*8*1 = 167`

### Example 2:

Input: `nums = [1,5]`

Output: 10

### Constraints:

- $n = \text{nums.length} =$
- $1 < n \leq 300 =$
- $0 < \text{nums}[i] \leq 100 =$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/burst-balloons/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxCoins(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->maxCoins(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.96 313. Super Ugly Number (Medium)

A **super ugly number** is a positive integer whose prime factors are in the array `primes`.

Given an integer `n` and an array of integers `primes`, return the `=nth=`\*super ugly number\*.

The `n` **super ugly number** is **guaranteed** to fit in a **32-bit** signed integer.

### Example 1:

**Input:** `n = 12, primes = [2,7,13,19]`

**Output:** 32

**Explanation:** `[1,2,4,7,8,13,14,16,19,26,28,32]` is the sequence of the first 12 super ugly numbers given `primes = [2,7,13,19]`.

### Example 2:

**Input:** `n = 1, primes = [2,3,5]`

**Output:** 1

**Explanation:** 1 has no prime factors, therefore all of its prime factors are in the array `primes = [2,3,5]`.

### Constraints:

- `1 < n <= 105`=
- `1 < primes.length <= 100`=
- `2 < primes[i] <= 1000`=
- `primes[i]` is **guaranteed** to be a prime number.
- All the values of `primes` are **unique** and sorted in **ascending order**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/super-ugly-number/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int nthSuperUglyNumber(int n, vector<int>& primes) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);
    vector<int> primes;
    LeetCodeIO::scan(cin, primes);

    Solution *obj = new Solution();
    auto res = obj->nthSuperUglyNumber(n, primes);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.97 318. Maximum Product of Word Lengths (Medium)

Given a string array `words`, return the maximum value of `length(word[i]) * length(word[j])` where the two words do not share common letters. If no such two words exist, return =0.

### Example 1:

Input: `words = ["abcw", "baz", "foo", "bar", "xtfn", "abcdef"]`

Output: 16

Explanation: The two words can be "abcw", "xtfn".

### Example 2:

Input: `words = ["a", "ab", "abc", "d", "cd", "bcd", "abcd"]`

Output: 4

Explanation: The two words can be "ab", "cd".

### Example 3:

Input: `words = ["a", "aa", "aaa", "aaaa"]`

Output: 0

Explanation: No such pair of words.

### Constraints:

- $2 < \text{words.length} \leq 1000$ =
- $1 < \text{words[i].length} \leq 1000$ =
- `words[i]` consists only of lowercase English letters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/maximum-product-of-word-lengths/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxProduct(vector<string>& words) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> words;
    LeetCodeIO::scan(cin, words);

    Solution *obj = new Solution();
    auto res = obj->maxProduct(words);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.98 322. Coin Change (Medium)

You are given an integer array `coins` representing coins of different denominations and an integer `amount` representing a total amount of money.

Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return `-1`.

You may assume that you have an infinite number of each kind of coin.

### Example 1:

Input: `coins = [1,2,5]`, `amount = 11`

Output: 3

Explanation:  $11 = 5 + 5 + 1$

### Example 2:

Input: `coins = [2]`, `amount = 3`

Output: -1

### Example 3:

Input: `coins = [1]`, `amount = 0`

Output: 0

### Constraints:

- $1 < \text{coins.length} \leq 12$
- $1 < \text{coins}[i] \leq 2^{31} - 1$
- $0 < \text{amount} \leq 10^4$

## 題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/coin-change/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> coins;
    LeetCodeIO::scan(cin, coins);
    int amount;
    LeetCodeIO::scan(cin, amount);

    Solution *obj = new Solution();
    auto res = obj->coinChange(coins, amount);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.99 328. Odd Even Linked List (Medium)

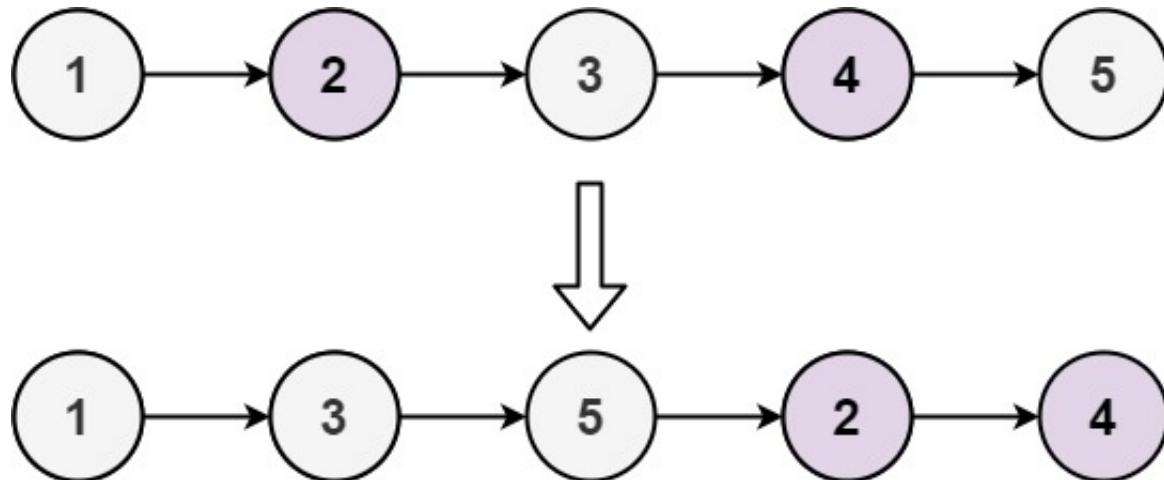
Given the head of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.

The **first** node is considered **odd**, and the **second** node is **even**, and so on.

Note that the relative order inside both the even and odd groups should remain as it was in the input.

You must solve the problem in  $O(1)$  extra space complexity and  $O(n)$  time complexity.

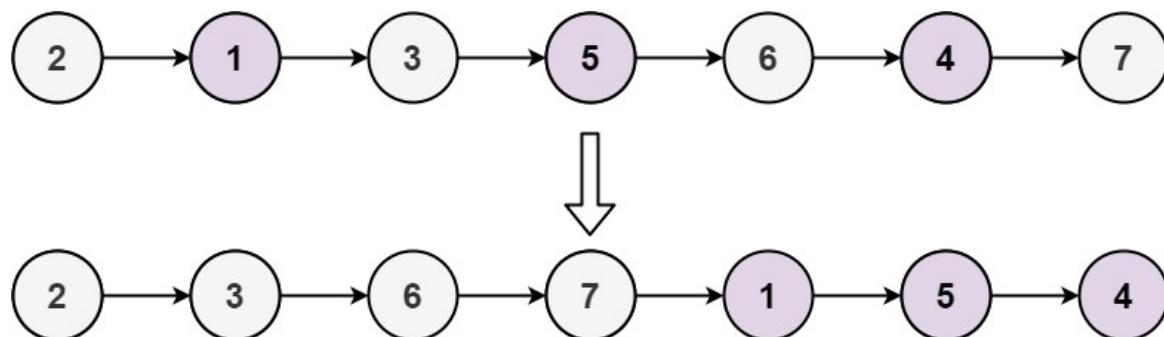
### Example 1:



Input: head = [1,2,3,4,5]

Output: [1,3,5,2,4]

### Example 2:



Input: head = [2,1,3,5,6,4,7]

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

### Constraints:

- The number of nodes in the linked list is in the range  $[0, 10]$ .
- $-10^6 < \text{Node.val} \leq 10^6$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/odd-even-linked-list/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    ListNode* oddEvenList(ListNode* head) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    ListNode* head;
    LeetCodeIO::scan(cin, head);

    Solution *obj = new Solution();
    auto res = obj->oddEvenList(head);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.100 332. Reconstruct Itinerary (Hard)

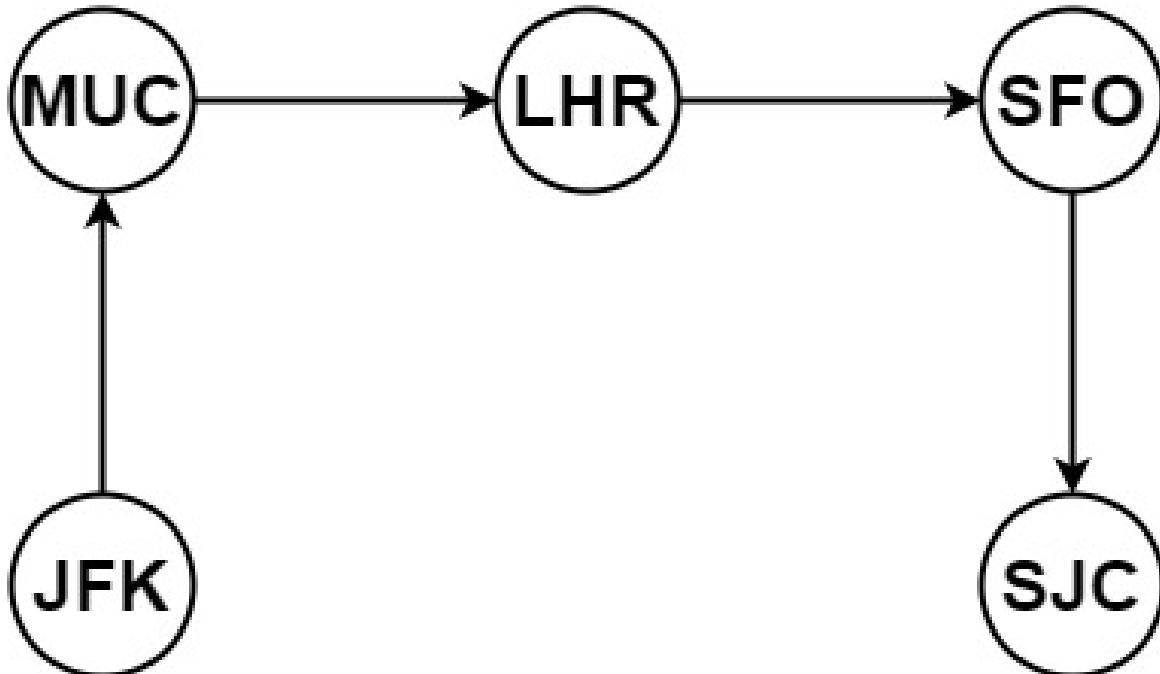
You are given a list of airline tickets where `tickets[i] = [from, to]` represent the departure and the arrival airports of one flight. Reconstruct the itinerary in order and return it.

All of the tickets belong to a man who departs from "JFK", thus, the itinerary must begin with "JFK". If there are multiple valid itineraries, you should return the itinerary that has the smallest lexical order when read as a single string.

- For example, the itinerary `["JFK", "LGA"]` has a smaller lexical order than `["JFK", "LGB"]`.

You may assume all tickets form at least one valid itinerary. You must use all the tickets once and only once.

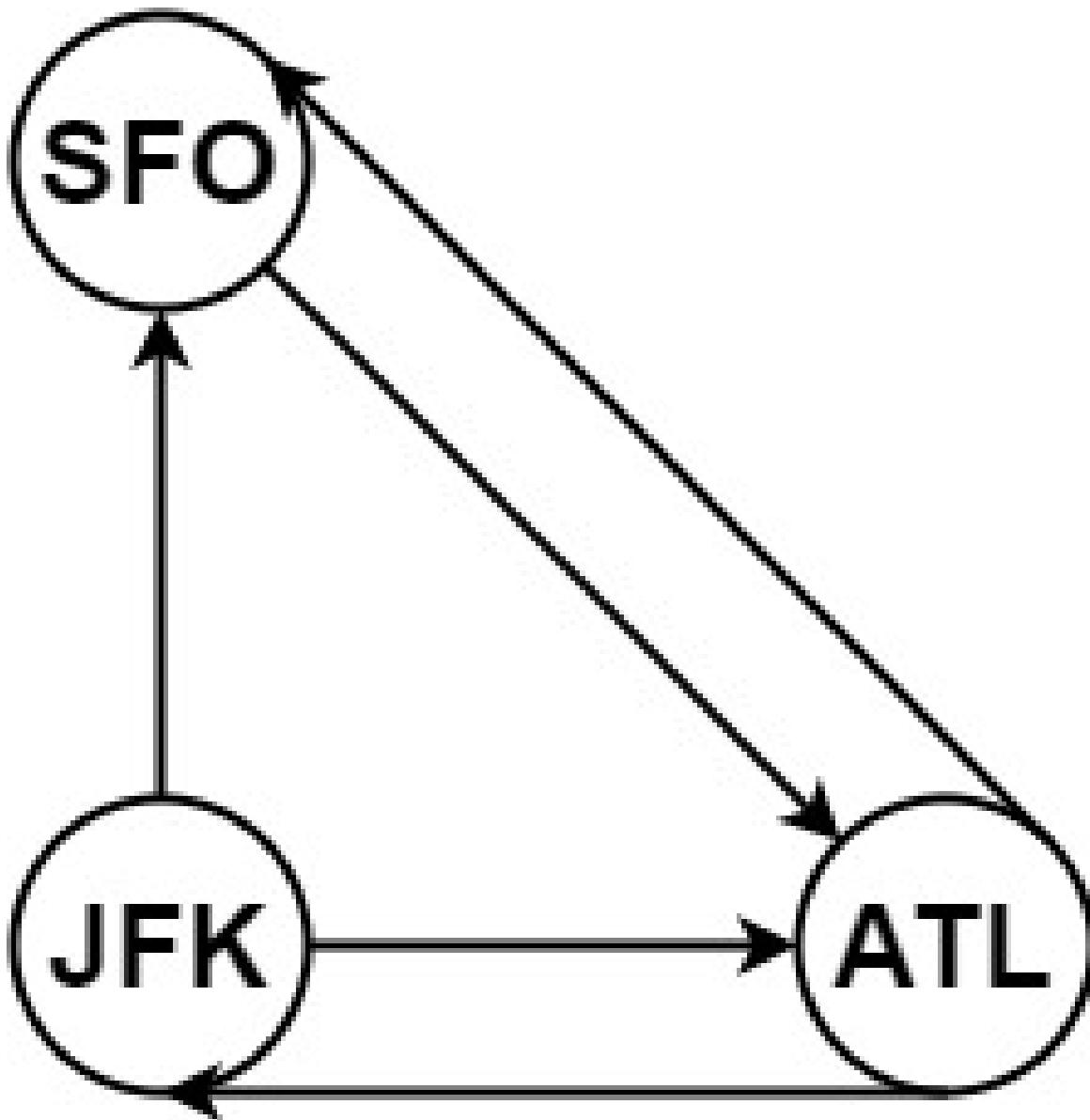
**Example 1:**



**Input:** `tickets = [["MUC", "LHR"], ["JFK", "MUC"], ["SFO", "SJC"], ["LHR", "SFO"]]`

**Output:** `["JFK", "MUC", "LHR", "SFO", "SJC"]`

**Example 2:**



Input: tickets = [["JFK", "SFO"], ["JFK", "ATL"], ["SFO", "ATL"], ["ATL", "JFK"], ["ATL", "SFO"]]

Output: ["JFK", "ATL", "JFK", "SFO", "ATL", "SFO"]

Explanation: Another possible reconstruction is ["JFK", "SFO", "ATL", "JFK", "ATL", "SFO"] but it is larger in lexical order.

#### Constraints:

- `1 < tickets.length <= 300`=
- `tickets[i].length = 2`=
- `from .length = 3`=
- `to .length = 3`=
- `from and to consist of uppercase English letters.`
- `from != to`=

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/reconstruct-itinerary/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<string> findItinerary(vector<vector<string>>& tickets) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<string>> tickets;
    LeetCodeIO::scan(cin, tickets);

    Solution *obj = new Solution();
    auto res = obj->findItinerary(tickets);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.101 338. Counting Bits (Easy)

Given an integer  $n$ , return an array  $\text{ans}$  of length  $n + 1$  such that for each  $i = ( =0 < i \leq n =)$ ,  $\text{ans}[i]$  is the **number of  $1=*\text{s}$**  in the binary representation of  $=i$ .

### Example 1:

Input:  $n = 2$

Output:  $[0,1,1]$

Explanation:

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow 10$

### Example 2:

Input:  $n = 5$

Output:  $[0,1,1,2,1,2]$

Explanation:

$0 \rightarrow 0$

$1 \rightarrow 1$

$2 \rightarrow 10$

$3 \rightarrow 11$

$4 \rightarrow 100$

$5 \rightarrow 101$

### Constraints:

- $0 < n \leq 10^5$

### Follow up:

- It is very easy to come up with a solution with a runtime of  $O(n \log n)$ . Can you do it in linear time  $O(n)$  and possibly in a single pass?
- Can you do it without using any built-in function (i.e., like `__builtin_popcount` in C++)?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/counting-bits/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> countBits(int n) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->countBits(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.102 343. Integer Break (Medium)

Given an integer  $n$ , break it into the sum of  $k$  **positive integers**, where  $k > 2$ , and maximize the product of those integers.

Return the maximum product you can get.

### Example 1:

Input:  $n = 2$

Output: 1

Explanation:  $2 = 1 + 1$ ,  $1 \times 1 = 1$ .

### Example 2:

Input:  $n = 10$

Output: 36

Explanation:  $10 = 3 + 3 + 4$ ,  $3 \times 3 \times 4 = 36$ .

### Constraints:

- $2 < n \leq 58$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/integer-break/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int integerBreak(int n) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->integerBreak(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.103 347. Top K Frequent Elements (Medium)

Given an integer array `nums` and an integer `k`, return the `=k=`most frequent elements. You may return the answer in **any order**.

**Example 1:**

**Input:** `nums = [1,1,1,2,2,3]`, `k = 2`

**Output:** `[1,2]`

**Example 2:**

**Input:** `nums = [1]`, `k = 1`

**Output:** `[1]`

**Example 3:**

**Input:** `nums = [1,2,1,2,1,2,3,1,3,2]`, `k = 2`

**Output:** `[1,2]`

**Constraints:**

- $1 < \text{nums.length} \leq 10^5$
- $-10 \leq \text{nums}[i] \leq 10^4$
- `k` is in the range `[1, the number of unique elements in the array]`.
- It is **guaranteed** that the answer is **unique**.

**Follow up:** Your algorithm's time complexity must be better than  $O(n \log n)$ , where  $n$  is the array's size.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/top-k-frequent-elements/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> topKFrequent(vector<int>& nums, int k) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int k;
    LeetCodeIO::scan(cin, k);

    Solution *obj = new Solution();
    auto res = obj->topKFrequent(nums, k);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.104 376. Wiggle Subsequence (Medium)

A **wiggle sequence** is a sequence where the differences between successive numbers strictly alternate between positive and negative. The first difference (if one exists) may be either positive or negative. A sequence with one element and a sequence with two non-equal elements are trivially wiggle sequences.

- For example, [1, 7, 4, 9, 2, 5] is a **wiggle sequence** because the differences (6, -3, 5, -7, 3) alternate between positive and negative.
- In contrast, [1, 4, 7, 2, 5] and [1, 7, 4, 5, 5] are not wiggle sequences. The first is not because its first two differences are positive, and the second is not because its last difference is zero.

A **subsequence** is obtained by deleting some elements (possibly zero) from the original sequence, leaving the remaining elements in their original order.

Given an integer array `nums`, return the length of the longest **wiggle subsequence** of `nums`.

### Example 1:

Input: `nums = [1,7,4,9,2,5]`

Output: 6

Explanation: The entire sequence is a wiggle sequence with differences (6, -3, 5, -7, 3).

### Example 2:

Input: `nums = [1,17,5,10,13,15,10,5,16,8]`

Output: 7

Explanation: There are several subsequences that achieve this length.

One is [1, 17, 10, 13, 10, 16, 8] with differences (16, -7, 3, -3, 6, -8).

### Example 3:

Input: `nums = [1,2,3,4,5,6,7,8,9]`

Output: 2

### Constraints:

- `1 < nums.length <= 1000`=
- `0 < nums[i] <= 1000`=

**Follow up:** Could you solve this in  $O(n)$  time?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/wiggle-subsequence/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int wiggleMaxLength(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->wiggleMaxLength(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.105 377. Combination Sum IV (Medium)

Given an array of **distinct** integers `nums` and a target integer `target`, return the number of possible combinations that add up to `target`.

The test cases are generated so that the answer can fit in a **32-bit** integer.

### Example 1:

**Input:** `nums = [1,2,3]`, `target = 4`

**Output:** 7

**Explanation:**

The possible combination ways are:

- (1, 1, 1, 1)
- (1, 1, 2)
- (1, 2, 1)
- (1, 3)
- (2, 1, 1)
- (2, 2)
- (3, 1)

Note that different sequences are counted as different combinations.

### Example 2:

**Input:** `nums = [9]`, `target = 3`

**Output:** 0

### Constraints:

- $1 < \text{nums.length} \leq 200$ =
- $1 < \text{nums}[i] \leq 1000$ =
- All the elements of `nums` are **unique**.
- $1 < \text{target} \leq 1000$ =

**Follow up:** What if negative numbers are allowed in the given array? How does it change the problem? What limitation we need to add to the question to allow negative numbers?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/combination-sum-iv/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int combinationSum4(vector<int>& nums, int target) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int target;
    LeetCodeIO::scan(cin, target);

    Solution *obj = new Solution();
    auto res = obj->combinationSum4(nums, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.106 380. Insert Delete GetRandom O(1) (Medium)

Implement the RandomizedSet class:

- RandomizedSet() Initializes the RandomizedSet object.
- bool insert(int val) Inserts an item val into the set if not present. Returns true if the item was not present, false otherwise.
- bool remove(int val) Removes an item val from the set if present. Returns true if the item was present, false otherwise.
- int getRandom() Returns a random element from the current set of elements (it's guaranteed that at least one element exists when this method is called). Each element must have the **same probability** of being returned.

You must implement the functions of the class such that each function works in **average**  $O(1)$  time complexity.

### Example 1:

Input

```
["RandomizedSet", "insert", "remove", "insert", "getRandom", "remove", "insert", "getRandom"]
[], [1], [2], [2], [], [1], [2], []]
```

Output

```
[null, true, false, true, 2, true, false, 2]
```

Explanation

```
RandomizedSet randomizedSet = new RandomizedSet();
randomizedSet.insert(1); // Inserts 1 to the set. Returns true as 1 was inserted successfully.
randomizedSet.remove(2); // Returns false as 2 does not exist in the set.
randomizedSet.insert(2); // Inserts 2 to the set, returns true. Set now contains [1,2].
randomizedSet.getRandom(); // getRandom() should return either 1 or 2 randomly.
randomizedSet.remove(1); // Removes 1 from the set, returns true. Set now contains [2].
randomizedSet.insert(2); // 2 was already in the set, so return false.
randomizedSet.getRandom(); // Since 2 is the only number in the set, getRandom() will always return 2.
```

### Constraints:

- $-2^{31} \leq \text{val} \leq 2^{31} - 1$
- At most  $2 * 10$  calls will be made to insert, remove, and getRandom.
- There will be **at least one** element in the data structure when getRandom is called.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/insert-delete-getrandom-o1/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class RandomizedSet {
public:
    RandomizedSet() {

    }

    bool insert(int val) {

    }

    bool remove(int val) {

    }

    int getRandom() {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    RandomizedSet *obj;
    const unordered_map<string, function<void()>> methods = {
        { "RandomizedSet", [&]() {
            cin.ignore();
            obj = new RandomizedSet();
            out_stream << "null,";
        } },
        { "insert", [&]() {
            int val; LeetCodeIO::scan(cin, val); cin.ignore();
            LeetCodeIO::print(out_stream, obj->insert(val)); out_stream << ',';
        } },
    };
}
```

```
{ "remove", [&]() {
    int val; LeetCodeIO::scan(cin, val); cin.ignore();
    LeetCodeIO::print(out_stream, obj->remove(val)); out_stream << ',';
} },
{ "getRandom", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->getRandom()); out_stream << ',';
} },
};

cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\n" << out_stream.rdbuf() << endl;

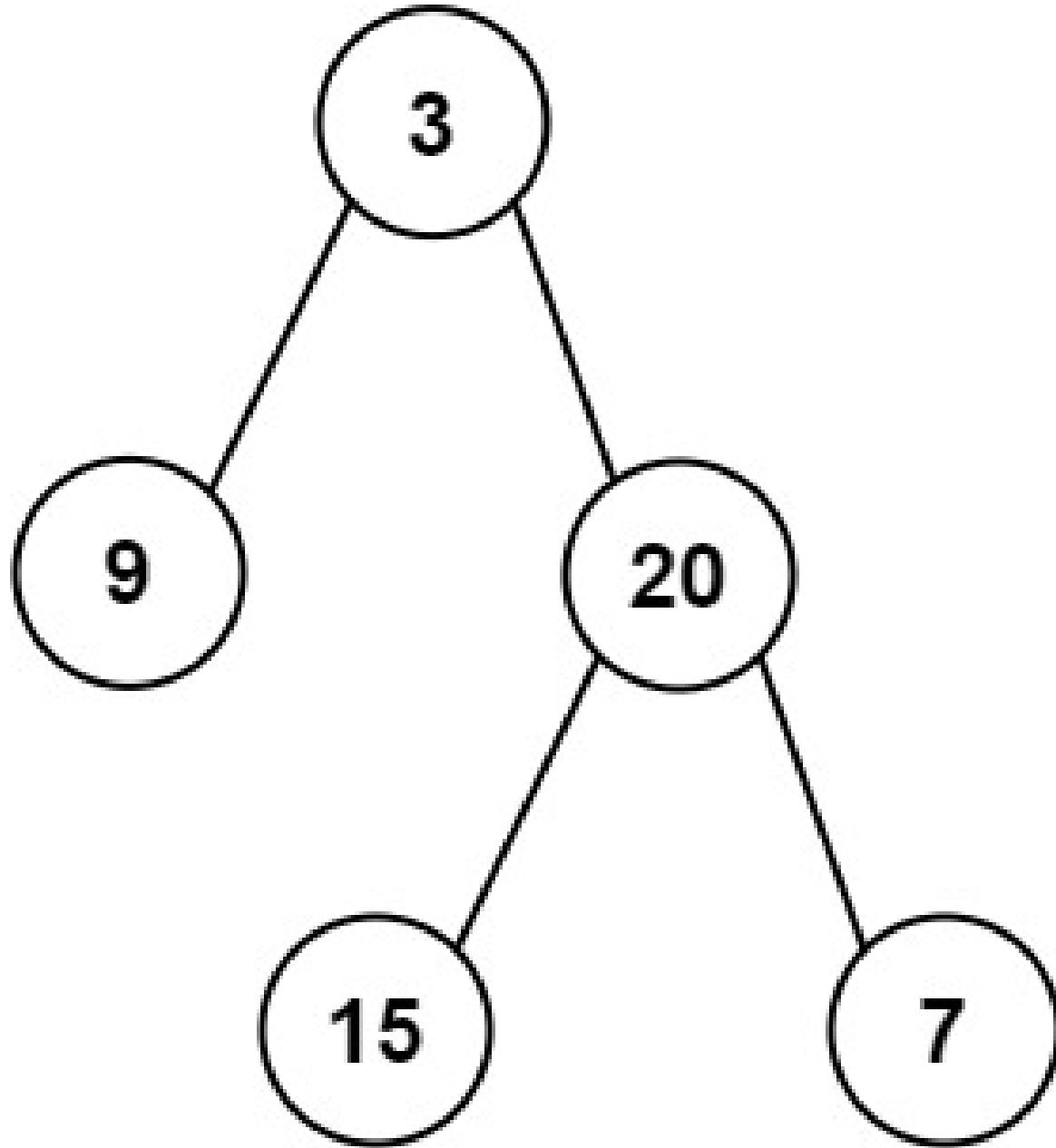
delete obj;
return 0;
}
```

## 8.107 404. Sum of Left Leaves (Easy)

Given the root of a binary tree, return the sum of all left leaves.

A **leaf** is a node with no children. A **left leaf** is a leaf that is the left child of another node.

**Example 1:**



Input: root = [3,9,20,null,null,15,7]

Output: 24

Explanation: There are two left leaves in the binary tree, with values 9 and 15 respectively.

**Example 2:**

Input: root = [1]

Output: 0

**Constraints:**

- The number of nodes in the tree is in the range [1, 1000].
- $-1000 < \text{Node.val} \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sum-of-left-leaves/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int sumOfLeftLeaves(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->sumOfLeftLeaves(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.108 406. Queue Reconstruction by Height (Medium)

You are given an array of people, `people`, which are the attributes of some people in a queue (not necessarily in order). Each `people[i] = [h, k]` represents the  $i$ -th person of height  $h$  with **exactly**  $k$  other people in front who have a height greater than or equal to  $h$ .

Reconstruct and return the queue that is represented by the input array `people`. The returned queue should be formatted as an array `queue`, where `queue[j] = [h, k]` is the attributes of the  $j$ -th person in the queue (`queue[0]` is the person at the front of the queue).

### Example 1:

**Input:** `people = [[7,0],[4,4],[7,1],[5,0],[6,1],[5,2]]`

**Output:** `[[5,0],[7,0],[5,2],[6,1],[4,4],[7,1]]`

**Explanation:**

Person 0 has height 5 with no other people taller or the same height in front.

Person 1 has height 7 with no other people taller or the same height in front.

Person 2 has height 5 with two persons taller or the same height in front, which are person 0 and 1.

Person 3 has height 6 with one person taller or the same height in front, which is person 1.

Person 4 has height 4 with four people taller or the same height in front, which are people 0, 1, 2, and 3.

Person 5 has height 7 with one person taller or the same height in front, which is person 1.

Hence `[[5,0],[7,0],[5,2],[6,1],[4,4],[7,1]]` is the reconstructed queue.

### Example 2:

**Input:** `people = [[6,0],[5,0],[4,0],[3,2],[2,2],[1,4]]`

**Output:** `[[4,0],[5,0],[2,2],[3,2],[1,4],[6,0]]`

### Constraints:

- $1 < \text{people.length} \leq 2000$ =
- $0 < h \leq 10^6$ =
- $0 < k < \text{people.length}$ =
- It is guaranteed that the queue can be reconstructed.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/queue-reconstruction-by-height/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> reconstructQueue(vector<vector<int>>& people) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> people;
    LeetCodeIO::scan(cin, people);

    Solution *obj = new Solution();
    auto res = obj->reconstructQueue(people);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.109 409. Longest Palindrome (Easy)

Given a string  $s$  which consists of lowercase or uppercase letters, return the length of the **longest palindrome** that can be built with those letters.

Letters are **case sensitive**, for example, "Aa" is not considered a palindrome.

### Example 1:

Input:  $s = \text{"abccccdd"}$

Output: 7

Explanation: One longest palindrome that can be built is "dccaccd", whose length is 7.

### Example 2:

Input:  $s = \text{"a"}$

Output: 1

Explanation: The longest palindrome that can be built is "a", whose length is 1.

### Constraints:

- $1 < s.length \leq 2000$
- $s$  consists of lowercase **and/or** uppercase English letters only.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/longest-palindrome/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int longestPalindrome(string s) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->longestPalindrome(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.110 413. Arithmetic Slices (Medium)

An integer array is called arithmetic if it consists of **at least three elements** and if the difference between any two consecutive elements is the same.

- For example, [1,3,5,7,9], [7,7,7,7], and [3,-1,-5,-9] are arithmetic sequences.

Given an integer array `nums`, return the number of arithmetic **subarrays** of `nums`.

A **subarray** is a contiguous subsequence of the array.

### Example 1:

Input: `nums` = [1,2,3,4]

Output: 3

Explanation: We have 3 arithmetic slices in `nums`: [1, 2, 3], [2, 3, 4] and [1,2,3,4] itself.

### Example 2:

Input: `nums` = [1]

Output: 0

### Constraints:

- $1 < \text{nums.length} \leq 5000$
- $-1000 \leq \text{nums}[i] \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/arithmetic-slices/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int numberOfArithmeticSlices(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->numberOfArithmeticSlices(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.111 416. Partition Equal Subset Sum (Medium)

Given an integer array `nums`, return `true`=if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or =`false` otherwise.

### Example 1:

Input: `nums` = [1,5,11,5]

Output: `true`

Explanation: The array can be partitioned as [1, 5, 5] and [11].

### Example 2:

Input: `nums` = [1,2,3,5]

Output: `false`

Explanation: The array cannot be partitioned into equal sum subsets.

### Constraints:

- `1 < nums.length <= 200`=
- `1 < nums[i] <= 100`=

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/partition-equal-subset-sum/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool canPartition(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->canPartition(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

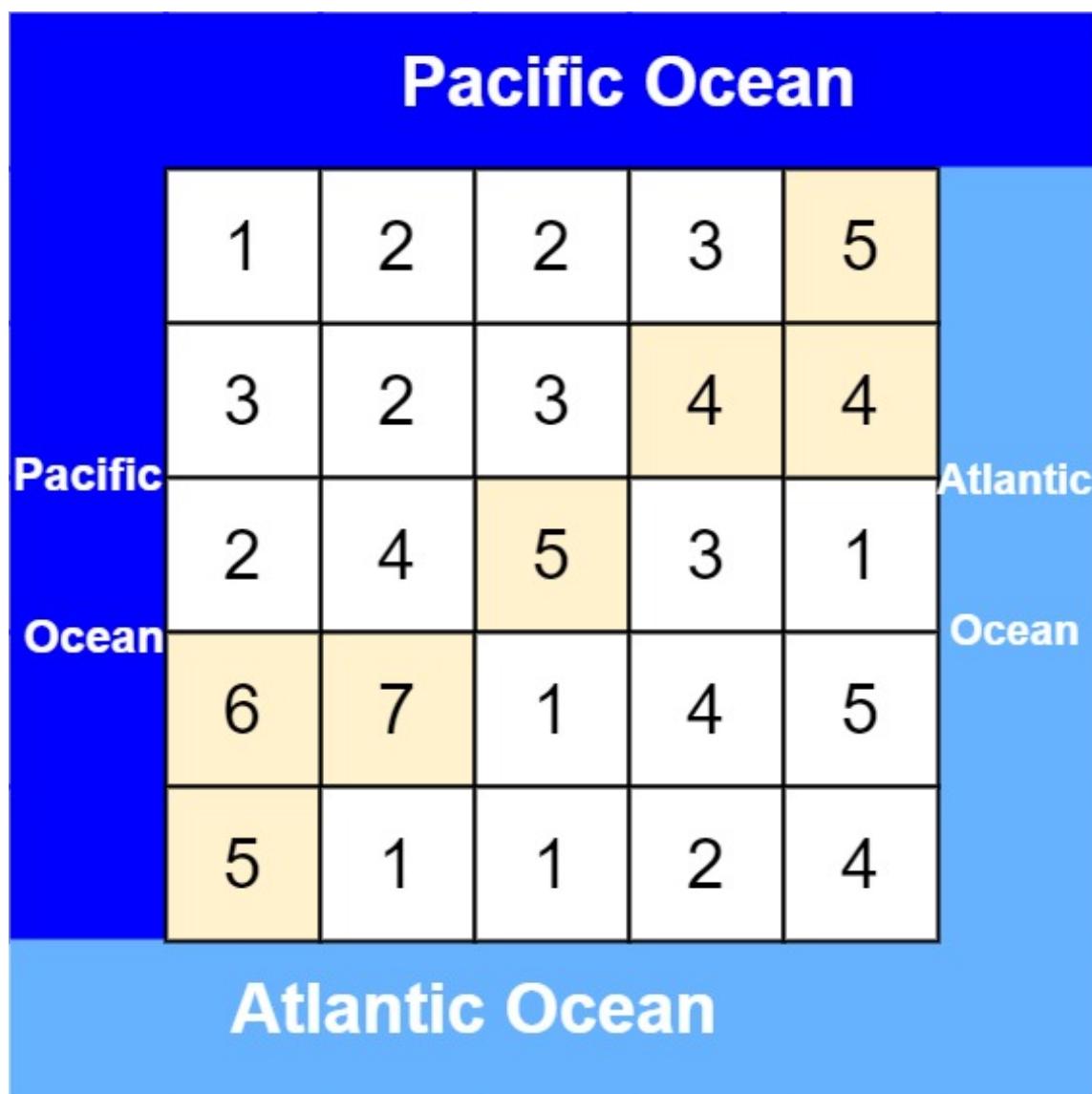
## 8.112 417. Pacific Atlantic Water Flow (Medium)

There is an  $m \times n$  rectangular island that borders both the **Pacific Ocean** and **Atlantic Ocean**. The **Pacific Ocean** touches the island's left and top edges, and the **Atlantic Ocean** touches the island's right and bottom edges. The island is partitioned into a grid of square cells. You are given an  $m \times n$  integer matrix **heights** where **heights[r][c]** represents the **height above sea level** of the cell at coordinate  $(r, c)$ .

The island receives a lot of rain, and the rain water can flow to neighboring cells directly north, south, east, and west if the neighboring cell's height is **less than or equal to** the current cell's height. Water can flow from any cell adjacent to an ocean into the ocean.

Return a **2D list** of grid coordinates **result** where **result[i] = [r, c]** denotes that rain water can flow from cell  $(r, c)$  to **both** the Pacific and Atlantic oceans.

**Example 1:**



Input: **heights** = `[[1,2,2,3,5],[3,2,3,4,4],[2,4,5,3,1],[6,7,1,4,5],[5,1,1,2,4]]`

Output: `[[0,4],[1,3],[1,4],[2,2],[3,0],[3,1],[4,0]]`

Explanation: The following cells can flow to the Pacific and Atlantic oceans, as shown below:  
`[0,4]: [0,4] -> Pacific Ocean`

```
[0,4] -> Atlantic Ocean
[1,3]: [1,3] -> [0,3] -> Pacific Ocean
        [1,3] -> [1,4] -> Atlantic Ocean
[1,4]: [1,4] -> [1,3] -> [0,3] -> Pacific Ocean
        [1,4] -> Atlantic Ocean
[2,2]: [2,2] -> [1,2] -> [0,2] -> Pacific Ocean
        [2,2] -> [2,3] -> [2,4] -> Atlantic Ocean
[3,0]: [3,0] -> Pacific Ocean
        [3,0] -> [4,0] -> Atlantic Ocean
[3,1]: [3,1] -> [3,0] -> Pacific Ocean
        [3,1] -> [4,1] -> Atlantic Ocean
[4,0]: [4,0] -> Pacific Ocean
        [4,0] -> Atlantic Ocean
```

Note that there are other possible paths for these cells to flow to the Pacific and Atlantic oceans.

#### **Example 2:**

Input: heights = [[1]]

Output: [[0,0]]

Explanation: The water can flow from the only cell to the Pacific and Atlantic oceans.

#### **Constraints:**

- m = heights.length=
- n = heights[r].length=
- 1 < m, n <= 200=
- 0 < heights[r][c] <=  $10^5$ =

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/pacific-atlantic-water-flow/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> pacificAtlantic(vector<vector<int>>& heights) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> heights;
    LeetCodeIO::scan(cin, heights);

    Solution *obj = new Solution();
    auto res = obj->pacificAtlantic(heights);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.113 432. All O'one Data Structure (Hard)

Design a data structure to store the strings' count with the ability to return the strings with minimum and maximum counts.

Implement the AllOne class:

- `AllOne()` Initializes the object of the data structure.
- `inc(String key)` Increments the count of the string `key` by 1. If `key` does not exist in the data structure, insert it with count 1.
- `dec(String key)` Decrements the count of the string `key` by 1. If the count of `key` is 0 after the decrement, remove it from the data structure. It is guaranteed that `key` exists in the data structure before the decrement.
- `getMaxKey()` Returns one of the keys with the maximal count. If no element exists, return an empty string "".
- `getMinKey()` Returns one of the keys with the minimum count. If no element exists, return an empty string "".

**Note** that each function must run in  $O(1)$  average time complexity.

### Example 1:

Input

```
["AllOne", "inc", "inc", "getMaxKey", "getMinKey", "inc", "getMaxKey", "getMinKey"]
[], ["hello"], ["hello"], [], [], ["leet"], [], []]
```

Output

```
[null, null, null, "hello", "hello", null, "hello", "leet"]
```

Explanation

```
AllOne allOne = new AllOne();
allOne.inc("hello");
allOne.inc("hello");
allOne.getMaxKey(); // return "hello"
allOne.getMinKey(); // return "hello"
allOne.inc("leet");
allOne.getMaxKey(); // return "hello"
allOne.getMinKey(); // return "leet"
```

### Constraints:

- $1 < \text{key.length} \leq 10$
- `key` consists of lowercase English letters.
- It is guaranteed that for each call to `dec`, `key` is existing in the data structure.
- At most  $5 * 10$  calls will be made to `inc`, `dec`, `getMaxKey`, and `getMinKey`.

## 題解如下:

```

// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/all-oone-data-structure/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class AllOne {
public:
    AllOne() {

    }

    void inc(string key) {

    }

    void dec(string key) {

    }

    string getMaxKey() {

    }

    string getMinKey() {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> method_names;
    LeetCodeIO::scan(cin, method_names);

    AllOne *obj;
    const unordered_map<string, function<void()>> methods = {
        { "AllOne", [&]() {
            cin.ignore();
            obj = new AllOne();
            out_stream << "null,";
        } },
    };
}

```

```

{ "inc", [&]() {
    string key; LeetCodeIO::scan(cin, key); cin.ignore();
    obj->inc(key);
    out_stream << "null,";
} },
{ "dec", [&]() {
    string key; LeetCodeIO::scan(cin, key); cin.ignore();
    obj->dec(key);
    out_stream << "null,";
} },
{ "getMaxKey", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->getMaxKey()); out_stream << ',';
} },
{ "getMinKey", [&]() {
    cin.ignore();
    LeetCodeIO::print(out_stream, obj->getMinKey()); out_stream << ',';
} },
};

cin >> ws;
out_stream << '[';
for (auto &&method_name : method_names) {
    cin.ignore(2);
    methods.at(method_name)();
}
cin.ignore();
out_stream.seekp(-1, ios_base::end); out_stream << ']';
cout << "\noutput:\u202a" << out_stream.rdbuf() << endl;

delete obj;
return 0;
}

```

## 8.114 435. Non-overlapping Intervals (Medium)

Given an array of intervals `intervals` where `intervals[i] = [start, end]`, return the minimum number of intervals you need to remove to make the rest of the intervals non-overlapping.

Note that intervals which only touch at a point are **non-overlapping**. For example, `[1, 2]` and `[2, 3]` are non-overlapping.

### Example 1:

Input: `intervals = [[1,2],[2,3],[3,4],[1,3]]`

Output: 1

Explanation: `[1,3]` can be removed and the rest of the intervals are non-overlapping.

### Example 2:

Input: `intervals = [[1,2],[1,2],[1,2]]`

Output: 2

Explanation: You need to remove two `[1,2]` to make the rest of the intervals non-overlapping.

### Example 3:

Input: `intervals = [[1,2],[2,3]]`

Output: 0

Explanation: You don't need to remove any of the intervals since they're already non-overlapping.

### Constraints:

- $1 < \text{intervals.length} \leq 10^5$
- $\text{intervals}[i].length = 2$
- $-5 * 10^4 < \text{start} < \text{end} \leq 5 * 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/non-overlapping-intervals/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int eraseOverlapIntervals(vector<vector<int>>& intervals) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> intervals;
    LeetCodeIO::scan(cin, intervals);

    Solution *obj = new Solution();
    auto res = obj->eraseOverlapIntervals(intervals);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

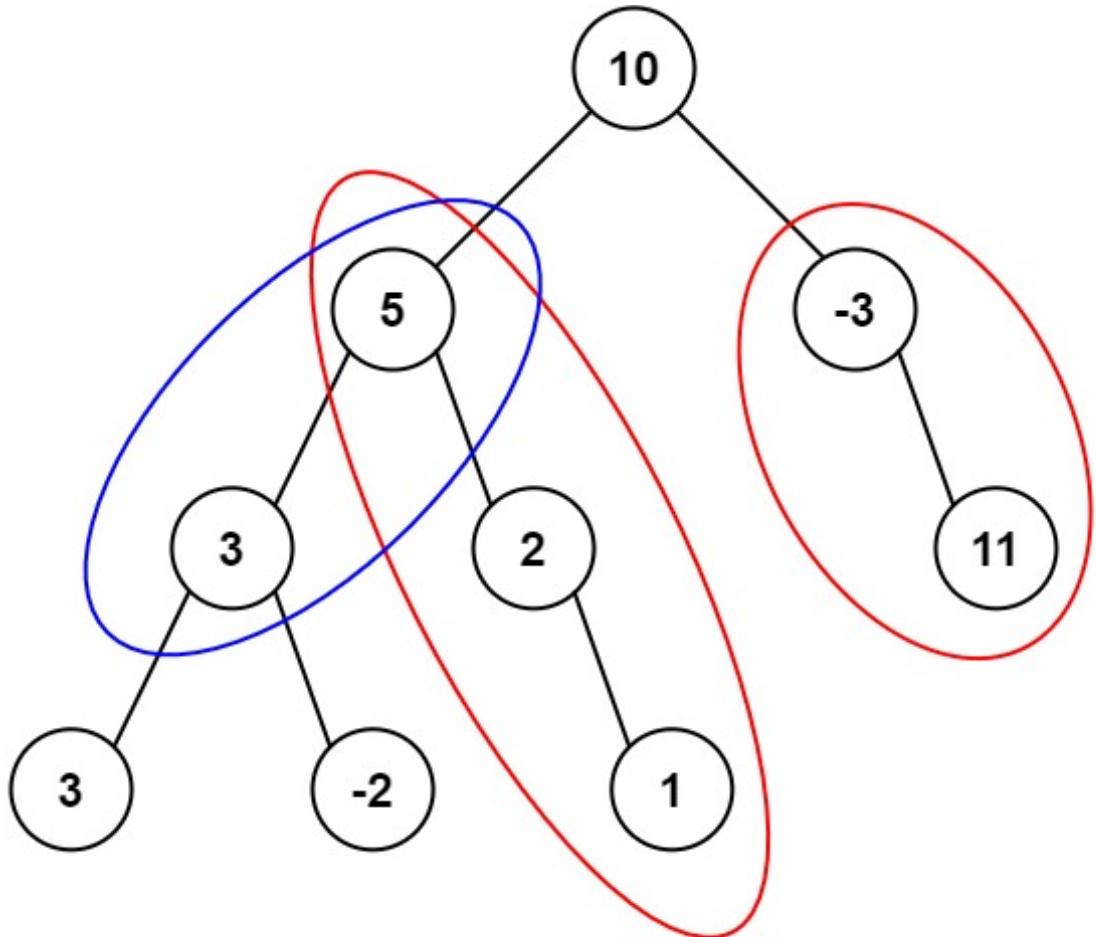
    delete obj;
    return 0;
}
```

## 8.115 437. Path Sum III (Medium)

Given the root of a binary tree and an integer targetSum, return the number of paths where the sum of the values along the path equals targetSum.

The path does not need to start or end at the root or a leaf, but it must go downwards (i.e., traveling only from parent nodes to child nodes).

**Example 1:**



Input: root = [10,5,-3,3,2,null,11,3,-2,null,1], targetSum = 8

Output: 3

Explanation: The paths that sum to 8 are shown.

**Example 2:**

Input: root = [5,4,8,11,null,13,4,7,2,null,null,5,1], targetSum = 22

Output: 3

### Constraints:

- The number of nodes in the tree is in the range [0, 1000].
- $-10^9 < \text{Node.val} \leq 10^9$
- $-1000 < \text{targetSum} \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/path-sum-iii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int pathSum(TreeNode* root, int targetSum) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    int targetSum;
    LeetCodeIO::scan(cin, targetSum);

    Solution *obj = new Solution();
    auto res = obj->pathSum(root, targetSum);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.116 448. Find All Numbers Disappeared in an Array (Easy)

Given an array `nums` of  $n$  integers where `nums[i]` is in the range  $[1, n]$ , return an array of all the integers in the range  $[1, n]$  that do not appear in `=nums`.

### Example 1:

Input: `nums` = [4,3,2,7,8,2,3,1]

Output: [5,6]

### Example 2:

Input: `nums` = [1,1]

Output: [2]

### Constraints:

- $n = \text{nums.length} =$
- $1 < n \leq 10^5 =$
- $1 < \text{nums}[i] \leq n =$

**Follow up:** Could you do it without extra space and in  $O(n)$  runtime? You may assume the returned list does not count as extra space.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/find-all-numbers-disappeared-in-an-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> findDisappearedNumbers(vector<int>& nums) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->findDisappearedNumbers(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

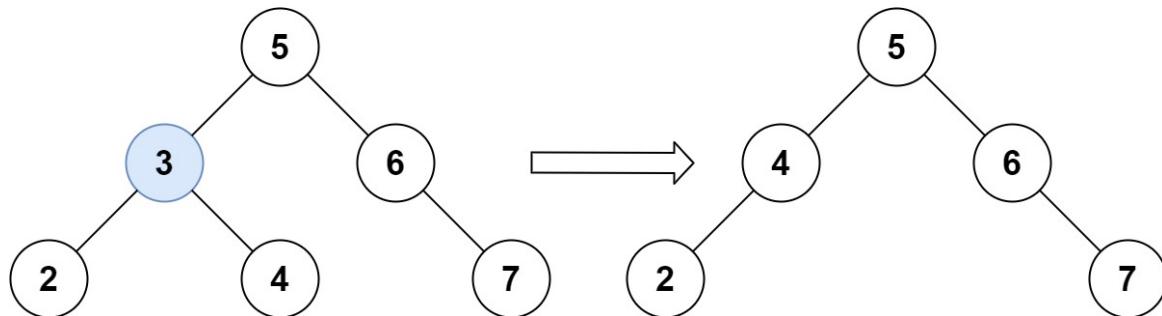
## 8.117 450. Delete Node in a BST (Medium)

Given a root node reference of a BST and a key, delete the node with the given key in the BST. Return the **root node reference** (possibly updated) of the BST.

Basically, the deletion can be divided into two stages:

1. Search for a node to remove.
2. If the node is found, delete the node.

### Example 1:



**Input:** root = [5,3,6,2,4,null,7], key = 3

**Output:** [5,4,6,2,null,null,7]

**Explanation:** Given key to delete is 3. So we find the node with value 3 and delete it.

One valid answer is [5,4,6,2,null,null,7], shown in the above BST.

Please notice that another valid answer is [5,2,6,null,4,null,7] and it's also accepted.

### Example 2:

**Input:** root = [5,3,6,2,4,null,7], key = 0

**Output:** [5,3,6,2,4,null,7]

**Explanation:** The tree does not contain a node with value = 0.

### Example 3:

**Input:** root = [], key = 0

**Output:** []

### Constraints:

- The number of nodes in the tree is in the range [0, 10 ].
- $-10^5 < \text{Node.val} \leq 10^5$
- Each node has a **unique** value.
- root is a valid binary search tree.
- $-10^5 < \text{key} \leq 10^5$

**Follow up:** Could you solve it with time complexity  $O(\text{height of tree})$ ?

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/delete-node-in-a-bst/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* deleteNode(TreeNode* root, int key) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    int key;
    LeetCodeIO::scan(cin, key);

    Solution *obj = new Solution();
    auto res = obj->deleteNode(root, key);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.118 451. Sort Characters By Frequency (Medium)

Given a string  $s$ , sort it in **decreasing order** based on the **frequency** of the characters. The **frequency** of a character is the number of times it appears in the string.

Return the sorted string. If there are multiple answers, return any of them.

### Example 1:

Input:  $s = \text{"tree"}$

Output:  $\text{"eert"}$

Explanation: 'e' appears twice while 'r' and 't' both appear once.

So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.

### Example 2:

Input:  $s = \text{"cccaaa"}$

Output:  $\text{"aaaccc"}$

Explanation: Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers.

Note that "cacaca" is incorrect, as the same characters must be together.

### Example 3:

Input:  $s = \text{"Aabb"}$

Output:  $\text{"bbAa"}$

Explanation: "bbaA" is also a valid answer, but "Aabb" is incorrect.

Note that 'A' and 'a' are treated as two different characters.

### Constraints:

- $1 < s.length \leq 5 * 10^5$
- $s$  consists of uppercase and lowercase English letters and digits.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sort-characters-by-frequency/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    string frequencySort(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->frequencySort(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.119 452. Minimum Number of Arrows to Burst Balloons (Medium)

There are some spherical balloons taped onto a flat wall that represents the XY-plane. The balloons are represented as a 2D integer array `points` where `points[i] = [x, x_d]` denotes a balloon whose **horizontal diameter** stretches between  $x$  and  $x + d$ . You do not know the exact y-coordinates of the balloons.

Arrows can be shot up **directly vertically** (in the positive y-direction) from different points along the x-axis. A balloon with  $x$  and  $x + d$  is **burst** by an arrow shot at  $x$  if  $x < x \leq x + d$ . There is **no limit** to the number of arrows that can be shot. A shot arrow keeps traveling up infinitely, bursting any balloons in its path.

Given the array `points`, return the **minimum** number of arrows that must be shot to burst all balloons.

### Example 1:

`Input: points = [[10,16],[2,8],[1,6],[7,12]]`

`Output: 2`

`Explanation: The balloons can be burst by 2 arrows:`

- Shoot an arrow at  $x = 6$ , bursting the balloons `[2,8]` and `[1,6]`.
- Shoot an arrow at  $x = 11$ , bursting the balloons `[10,16]` and `[7,12]`.

### Example 2:

`Input: points = [[1,2],[3,4],[5,6],[7,8]]`

`Output: 4`

`Explanation: One arrow needs to be shot for each balloon for a total of 4 arrows.`

### Example 3:

`Input: points = [[1,2],[2,3],[3,4],[4,5]]`

`Output: 2`

`Explanation: The balloons can be burst by 2 arrows:`

- Shoot an arrow at  $x = 2$ , bursting the balloons `[1,2]` and `[2,3]`.
- Shoot an arrow at  $x = 4$ , bursting the balloons `[3,4]` and `[4,5]`.

### Constraints:

- $1 < \text{points.length} \leq 10^5$
- $\text{points}[i].length = 2$
- $-2^{31} < x < x + d \leq 2^{31} - 1$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/minimum-number-of-arrows-to-burst-balloons/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findMinArrowShots(vector<vector<int>>& points) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> points;
    LeetCodeIO::scan(cin, points);

    Solution *obj = new Solution();
    auto res = obj->findMinArrowShots(points);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.120 455. Assign Cookies (Easy)

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.

Each child  $i$  has a greed factor  $g[i]$ , which is the minimum size of a cookie that the child will be content with; and each cookie  $j$  has a size  $s[j]$ . If  $s[j] > g[i]$ , we can assign the cookie  $j$  to the child  $i$ , and the child  $i$  will be content. Your goal is to maximize the number of your content children and output the maximum number.

### Example 1:

Input:  $g = [1, 2, 3]$ ,  $s = [1, 1]$

Output: 1

Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3. And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.

You need to output 1.

### Example 2:

Input:  $g = [1, 2]$ ,  $s = [1, 2, 3]$

Output: 2

Explanation: You have 2 children and 3 cookies. The greed factors of 2 children are 1, 2.

You have 3 cookies and their sizes are big enough to gratify all of the children,

You need to output 2.

### Constraints:

- $1 < g.length \leq 3 * 10^4$
- $0 < s.length \leq 3 * 10^4$
- $1 < g[i], s[j] \leq 2^{31} - 1$

**Note:** This question is the same as [2410: Maximum Matching of Players With Trainers](#).

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/assign-cookies/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findContentChildren(vector<int>& g, vector<int>& s) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> g;
    LeetCodeIO::scan(cin, g);
    vector<int> s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->findContentChildren(g, s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.121 461. Hamming Distance (Easy)

The **Hamming distance** between two integers is the number of positions at which the corresponding bits are different.

Given two integers  $x$  and  $y$ , return the **Hamming distance** between them.

**Example 1:**

Input:  $x = 1$ ,  $y = 4$

Output: 2

Explanation:

1 (0 0 0 1)

4 (0 1 0 0)

↑ ↑

The above arrows point to positions where the corresponding bits are different.

**Example 2:**

Input:  $x = 3$ ,  $y = 1$

Output: 1

**Constraints:**

- $0 < x, y \leq 2^{31} - 1$

**Note:** This question is the same as [2220: Minimum Bit Flips to Convert Number](#).

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/hamming-distance/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int hammingDistance(int x, int y) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int x;
    LeetCodeIO::scan(cin, x);
    int y;
    LeetCodeIO::scan(cin, y);

    Solution *obj = new Solution();
    auto res = obj->hammingDistance(x, y);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.122 462. Minimum Moves to Equal Array Elements II (Medium)

Given an integer array `nums` of size  $n$ , return the minimum number of moves required to make all array elements equal.

In one move, you can increment or decrement an element of the array by 1.

Test cases are designed so that the answer will fit in a **32-bit** integer.

### Example 1:

Input: `nums` = [1,2,3]

Output: 2

Explanation:

Only two moves are needed (remember each move increments or decrements one element):

[1,2,3] => [2,2,3] => [2,2,2]

### Example 2:

Input: `nums` = [1,10,2,9]

Output: 16

### Constraints:

- $n = \text{nums.length} =$
- $1 < \text{nums.length} \leq 10^5 =$
- $-10 < \text{nums}[i] \leq 10^9 =$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/minimum-moves-to-equal-array-elements-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int minMoves2(vector<int>& nums) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->minMoves2(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.123 474. Ones and Zeroes (Medium)

You are given an array of binary strings `strs` and two integers `m` and `n`.

Return the size of the largest subset of `strs` such that there are **at most** `m` 0's and `n` 1's in the subset.

A set `x` is a **subset** of a set `y` if all elements of `x` are also elements of `y`.

### Example 1:

**Input:** `strs = ["10", "0001", "111001", "1", "0"]`, `m = 5`, `n = 3`

**Output:** 4

**Explanation:** The largest subset with at most 5 0's and 3 1's is `{"10", "0001", "1", "0"}`, so the answer is 4.

Other valid but smaller subsets include `{"0001", "1"}` and `{"10", "1", "0"}`.

`{"111001"}` is an invalid subset because it contains 4 1's, greater than the maximum of 3.

### Example 2:

**Input:** `strs = ["10", "0", "1"]`, `m = 1`, `n = 1`

**Output:** 2

**Explanation:** The largest subset is `{"0", "1"}`, so the answer is 2.

### Constraints:

- `1 < strs.length <= 600`=
- `1 < strs[i].length <= 100`=
- `strs[i]` consists only of digits '0' and '1'.
- `1 < m, n <= 100`=

## 題解如下：

```

// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/ones-and-zeroes/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findMaxForm(vector<string>& strs, int m, int n) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<string> strs;
    LeetCodeIO::scan(cin, strs);
    int m;
    LeetCodeIO::scan(cin, m);
    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->findMaxForm(strs, m, n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}

```

## 8.124 476. Number Complement (Easy)

The **complement** of an integer is the integer you get when you flip all the 0's to 1's and all the 1's to 0's in its binary representation.

- For example, The integer 5 is "101" in binary and its **complement** is "010" which is the integer 2.

Given an integer num, return its complement.

### Example 1:

Input: num = 5

Output: 2

Explanation: The binary representation of 5 is 101 (no leading zero bits), and its complement is 010. So you need to output 2.

### Example 2:

Input: num = 1

Output: 0

Explanation: The binary representation of 1 is 1 (no leading zero bits), and its complement is 0. So you need to output 0.

### Constraints:

- $1 < \text{num} < 2^{31}$

**Note:** This question is the same as 1009: <https://leetcode.com/problems/complement-of-base-10-integer/>

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/number-complement/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findComplement(int num) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int num;
    LeetCodeIO::scan(cin, num);

    Solution *obj = new Solution();
    auto res = obj->findComplement(num);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.125 494. Target Sum (Medium)

You are given an integer array `nums` and an integer `target`.

You want to build an **expression** out of `nums` by adding one of the symbols `'+'` and `'-'` before each integer in `nums` and then concatenate all the integers.

- For example, if `nums = [2, 1]`, you can add a `'+'` before 2 and a `'-'` before 1 and concatenate them to build the expression `"+2-1"`.

Return the number of different **expressions** that you can build, which evaluates to `target`.

### Example 1:

`Input: nums = [1,1,1,1,1], target = 3`

`Output: 5`

`Explanation: There are 5 ways to assign symbols to make the sum of nums be target 3.`

```
-1 + 1 + 1 + 1 + 1 = 3
+1 - 1 + 1 + 1 + 1 = 3
+1 + 1 - 1 + 1 + 1 = 3
+1 + 1 + 1 - 1 + 1 = 3
+1 + 1 + 1 + 1 - 1 = 3
```

### Example 2:

`Input: nums = [1], target = 1`

`Output: 1`

### Constraints:

- `1 < nums.length <= 20`=
- `0 < nums[i] <= 1000`=
- `0 < sum(nums[i]) <= 1000`=
- `-1000 < target <= 1000`=

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/target-sum/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findTargetSumWays(vector<int>& nums, int target) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);
    int target;
    LeetCodeIO::scan(cin, target);

    Solution *obj = new Solution();
    auto res = obj->findTargetSumWays(nums, target);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.126 503. Next Greater Element II (Medium)

Given a circular integer array `nums` (i.e., the next element of `nums` [`nums.length - 1`] is `nums[0]`), return the **next greater number** for every element in `nums`.

The **next greater number** of a number `x` is the first greater number to its traversing-order next in the array, which means you could search circularly to find its next greater number. If it doesn't exist, return `-1` for this number.

### Example 1:

Input: `nums` = [1,2,1]

Output: [2,-1,2]

Explanation: The first 1's next greater number is 2;

The number 2 can't find next greater number.

The second 1's next greater number needs to search circularly, which is also 2.

### Example 2:

Input: `nums` = [1,2,3,4,3]

Output: [2,3,4,-1,4]

### Constraints:

- $1 < \text{nums.length} \leq 10^4$
- $-10^9 < \text{nums}[i] \leq 10^9$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/next-greater-element-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> nextGreaterElements(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

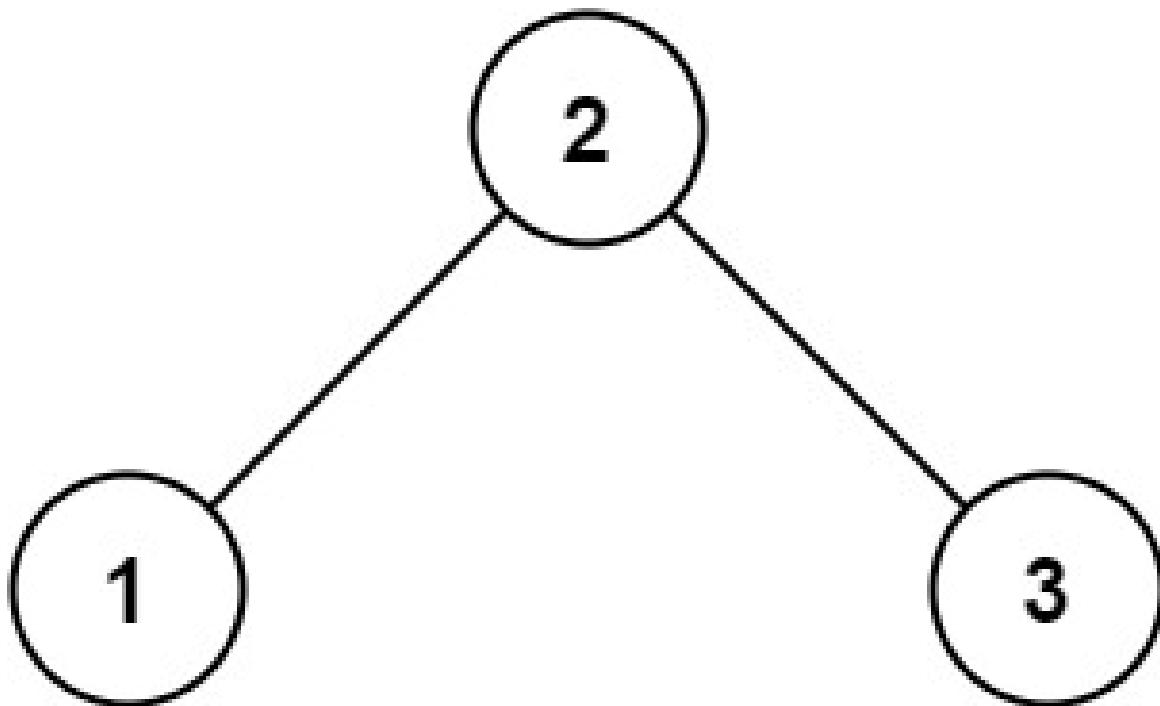
    Solution *obj = new Solution();
    auto res = obj->nextGreaterElements(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.127 513. Find Bottom Left Tree Value (Medium)

Given the root of a binary tree, return the leftmost value in the last row of the tree.

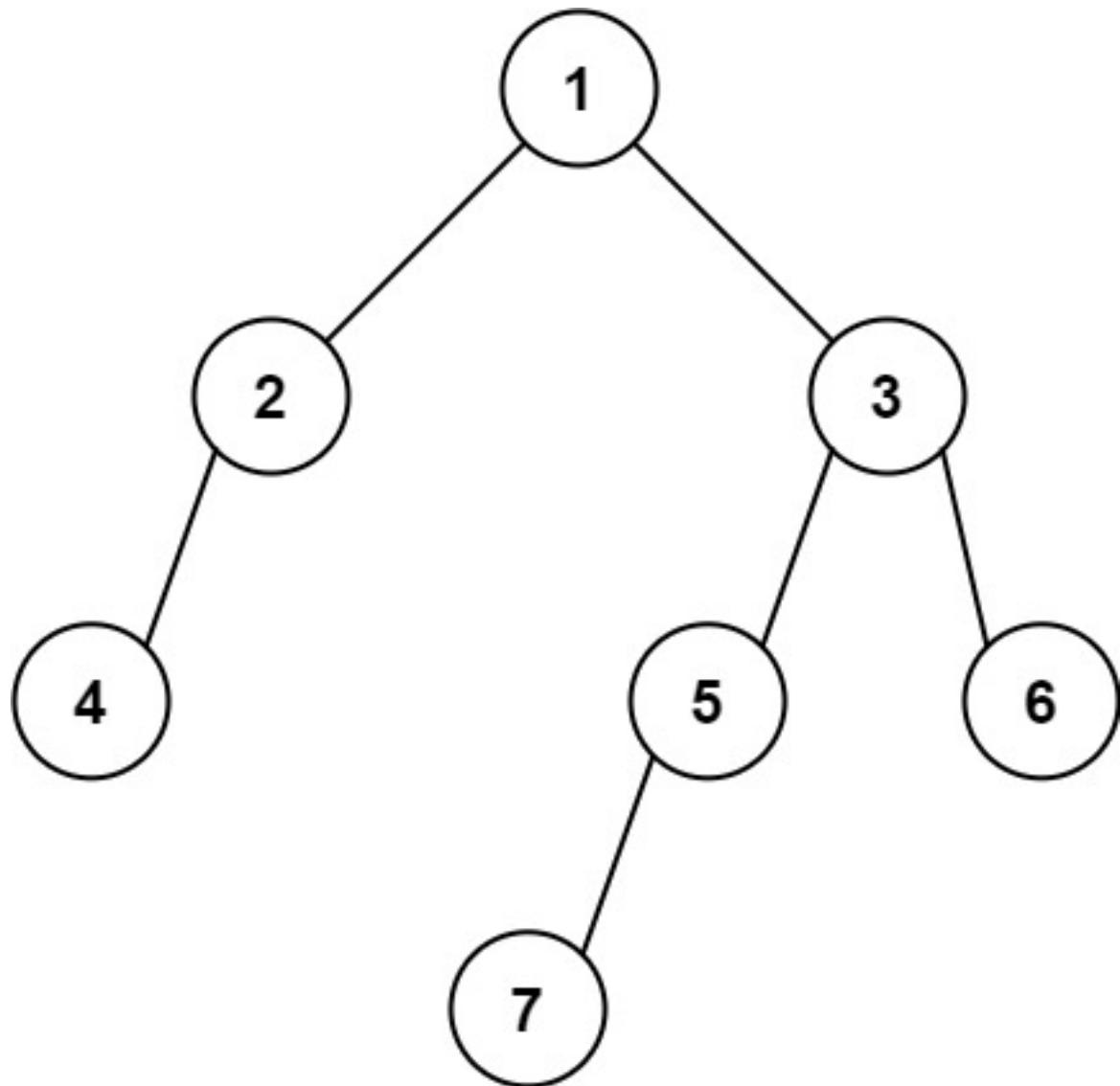
**Example 1:**



Input: `root = [2,1,3]`

Output: 1

**Example 2:**



Input: root = [1,2,3,4,null,5,6,null,null,7]

Output: 7

**Constraints:**

- The number of nodes in the tree is in the range [1, 10].
- $-2^{31} < \text{Node.val} \leq 2^{31} - 1$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/find-bottom-left-tree-value/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findBottomLeftValue(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

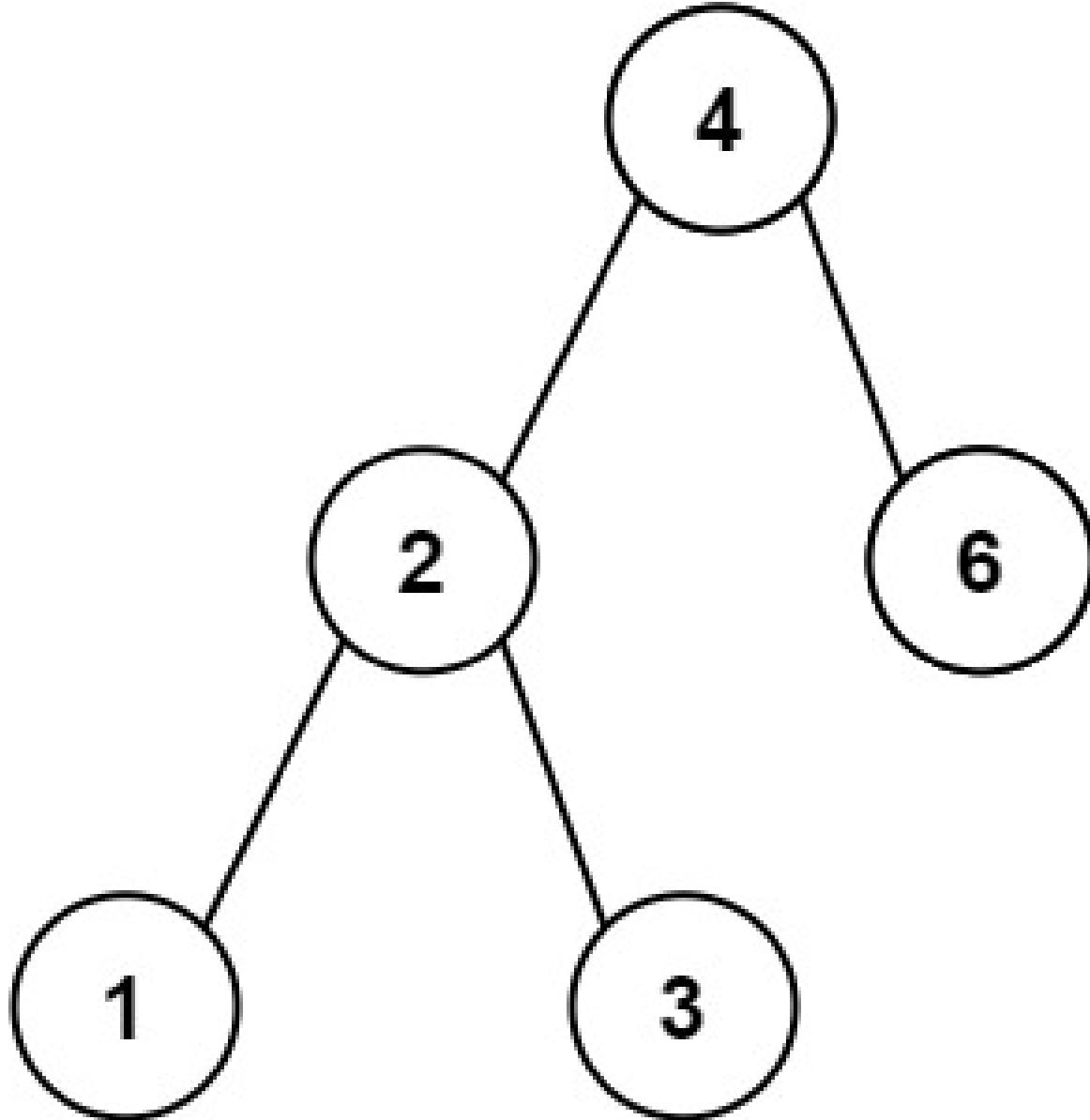
    Solution *obj = new Solution();
    auto res = obj->findBottomLeftValue(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

**8.128 530. Minimum Absolute Difference in BST (Easy)**

Given the root of a Binary Search Tree (BST), return the minimum absolute difference between the values of any two different nodes in the tree.

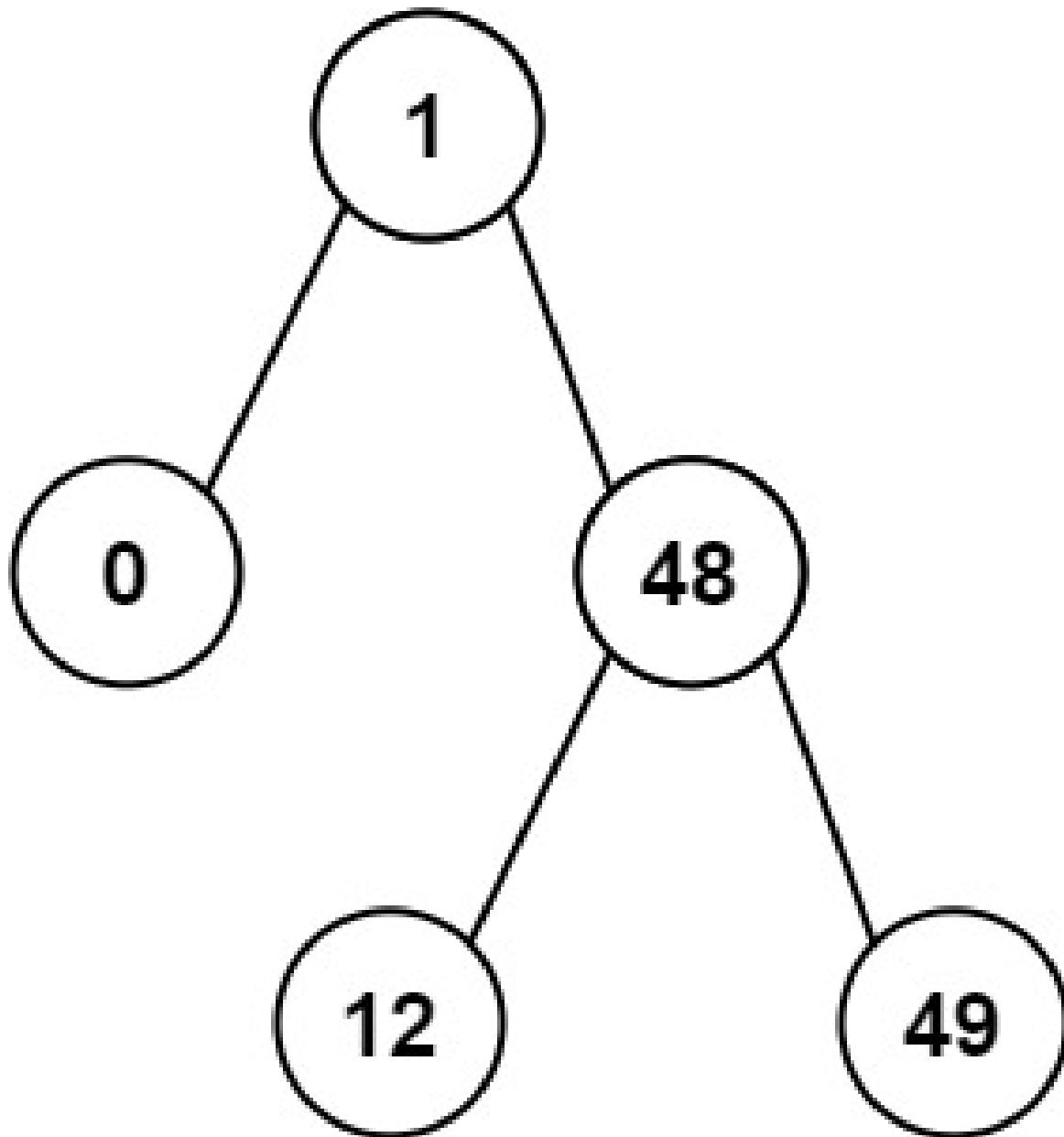
**Example 1:**



Input: root = [4,2,6,1,3]

Output: 1

**Example 2:**



Input: `root = [1,0,48,null,null,12,49]`

Output: 1

**Constraints:**

- The number of nodes in the tree is in the range  $[2, 10]$ .
- $0 < \text{Node.val} \leq 10^5$

**Note:** This question is the same as 783: <https://leetcode.com/problems/minimum-distance-between-bst-nodes/>

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/minimum-absolute-difference-in-bst/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int getMinimumDifference(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->getMinimumDifference(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

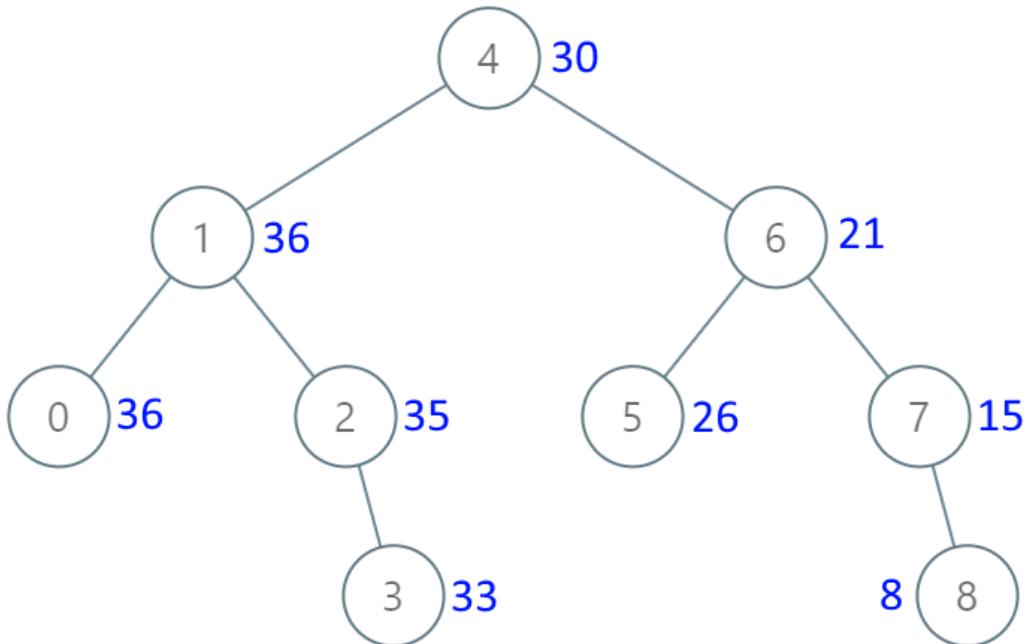
## 8.129 538. Convert BST to Greater Tree (Medium)

Given the root of a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus the sum of all keys greater than the original key in BST.

As a reminder, a binary search tree is a tree that satisfies these constraints:

- 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.

**Example 1:**



Input: `root = [4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]`

Output: `[30,36,21,36,35,26,15,null,null,null,33,null,null,8]`

**Example 2:**

Input: `root = [0,null,1]`

Output: `[1,null,1]`

**Constraints:**

- The number of nodes in the tree is in the range  $[0, 10]$ .
- $-10^4 < \text{Node.val} \leq 10^4$
- All the values in the tree are **unique**.
- `root` is guaranteed to be a valid binary search tree.

**Note:** This question is the same as 1038: <https://leetcode.com/problems/binary-search-tree-to-greater-sum-tree/>

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/convert-bst-to-greater-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* convertBST(TreeNode* root) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->convertBST(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.130 540. Single Element in a Sorted Array (Medium)

You are given a sorted array consisting of only integers where every element appears exactly twice, except for one element which appears exactly once.

Return the single element that appears only once.

Your solution must run in  $O(\log n)$  time and  $O(1)$  space.

**Example 1:**

Input: `nums = [1,1,2,3,3,4,4,8,8]`

Output: 2

**Example 2:**

Input: `nums = [3,3,7,7,10,11,11]`

Output: 10

**Constraints:**

- $1 < \text{nums.length} \leq 10^5$
- $0 < \text{nums}[i] \leq 10^5$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/single-element-in-a-sorted-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int singleNonDuplicate(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->singleNonDuplicate(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.131 542. 01 Matrix (Medium)

Given an  $m \times n$  binary matrix `mat`, return the distance of the nearest 0 for each cell.

The distance between two cells sharing a common edge is 1.

**Example 1:**

0	0	0
0	1	0
0	0	0

Input: `mat = [[0,0,0],[0,1,0],[0,0,0]]`

Output: `[[0,0,0],[0,1,0],[0,0,0]]`

**Example 2:**

0	0	0
0	1	0
1	1	1

Input: mat = [[0,0,0],[0,1,0],[1,1,1]]

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

**Constraints:**

- m = mat.length=
- n = mat[i].length=
- 1 < m, n <=  $10^4$ =
- 1 < m \* n <=  $10^4$ =
- mat [i] [j] is either 0 or 1.
- There is at least one 0 in mat.

**Note:** This question is the same as 1765: <https://leetcode.com/problems/map-of-highest- peak/>

## 題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/01-matrix/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> updateMatrix(vector<vector<int>>& mat) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> mat;
    LeetCodeIO::scan(cin, mat);

    Solution *obj = new Solution();
    auto res = obj->updateMatrix(mat);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

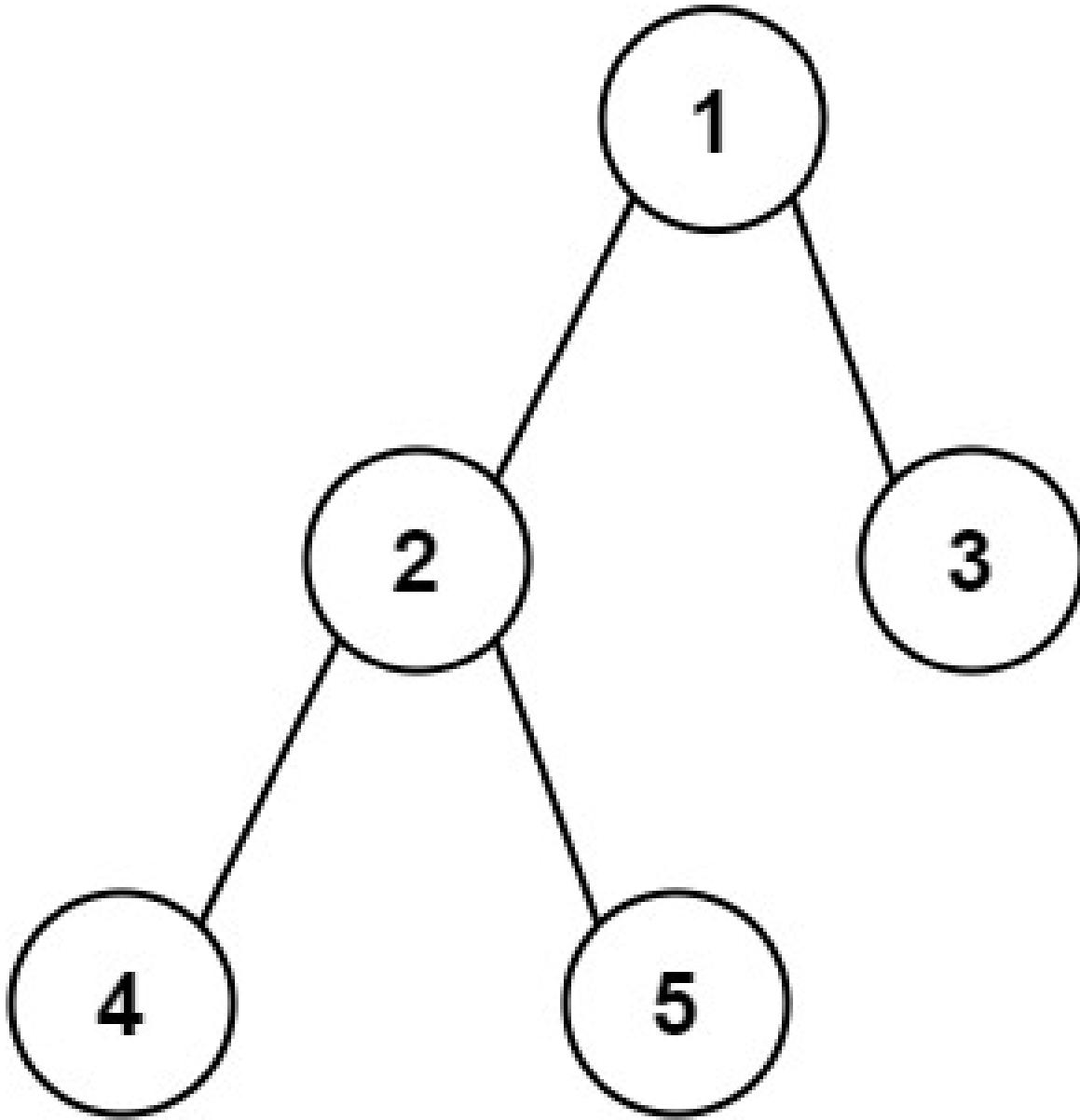
## 8.132 543. Diameter of Binary Tree (Easy)

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.

**Example 1:**



Input: root = [1,2,3,4,5]

Output: 3

Explanation: 3 is the length of the path [4,2,1,3] or [5,2,1,3].

**Example 2:**

Input: root = [1,2]

Output: 1

**Constraints:**

- The number of nodes in the tree is in the range [1, 10].
- $-100 < \text{Node.val} \leq 100$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/diameter-of-binary-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int diameterOfBinaryTree(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->diameterOfBinaryTree(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.133 547. Number of Provinces (Medium)

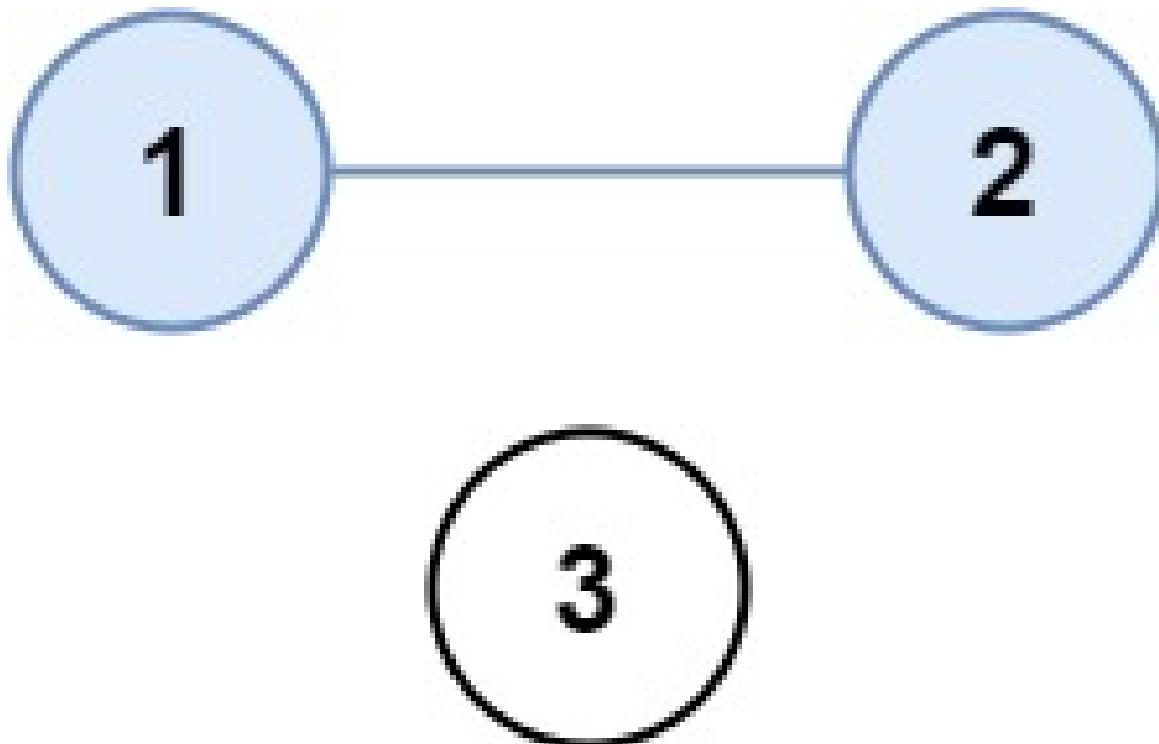
There are  $n$  cities. Some of them are connected, while some are not. If city  $a$  is connected directly with city  $b$ , and city  $b$  is connected directly with city  $c$ , then city  $a$  is connected indirectly with city  $c$ .

A **province** is a group of directly or indirectly connected cities and no other cities outside of the group.

You are given an  $n \times n$  matrix `isConnected` where `isConnected[i][j] = 1` if the  $i$ -th city and the  $j$ -th city are directly connected, and `isConnected[i][j] = 0` otherwise.

Return the total number of provinces.

**Example 1:**



Input: `isConnected = [[1,1,0],[1,1,0],[0,0,1]]`

Output: 2

**Example 2:**



Input: isConnected = [[1,0,0],[0,1,0],[0,0,1]]

Output: 3

**Constraints:**

- $1 < n \leq 200$ =
- $n = \text{isConnected.length}=$
- $n = \text{isConnected}[i].length=$
- $\text{isConnected}[i][j]$  is 1 or 0.
- $\text{isConnected}[i][i] = 1=$
- $\text{isConnected}[i][j] = \text{isConnected}[j][i]=$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/number-of-provinces/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findCircleNum(vector<vector<int>>& isConnected) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> isConnected;
    LeetCodeIO::scan(cin, isConnected);

    Solution *obj = new Solution();
    auto res = obj->findCircleNum(isConnected);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.134 566. Reshape the Matrix (Easy)

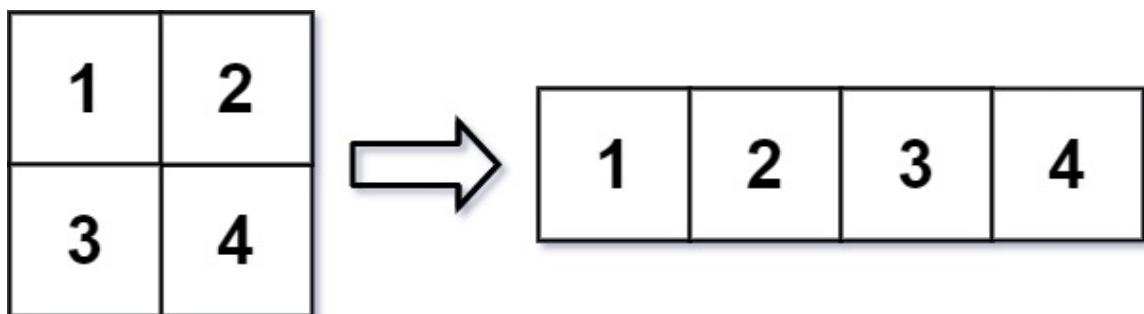
In MATLAB, there is a handy function called `reshape` which can reshape an  $m \times n$  matrix into a new one with a different size  $r \times c$  keeping its original data.

You are given an  $m \times n$  matrix `mat` and two integers `r` and `c` representing the number of rows and the number of columns of the wanted reshaped matrix.

The reshaped matrix should be filled with all the elements of the original matrix in the same row- traversing order as they were.

If the `reshape` operation with given parameters is possible and legal, output the new reshaped matrix; Otherwise, output the original matrix.

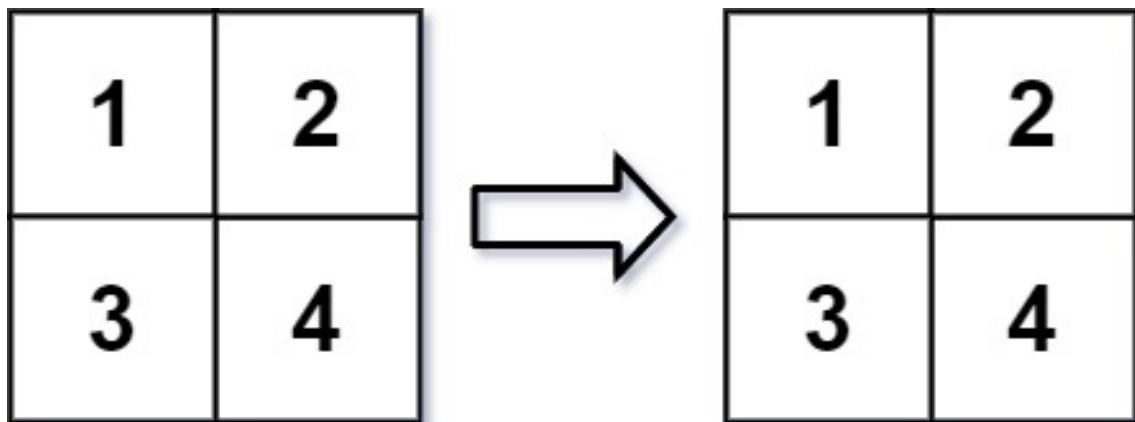
**Example 1:**



Input: `mat = [[1,2],[3,4]]`, `r = 1`, `c = 4`

Output: `[[1,2,3,4]]`

**Example 2:**



Input: `mat = [[1,2],[3,4]]`, `r = 2`, `c = 4`

Output: `[[1,2],[3,4]]`

**Constraints:**

- `m = mat.length=`
- `n = mat[i].length=`
- `1 < m, n <= 100=`
- `-1000 < mat[i][j] <= 1000=`
- `1 < r, c <= 300=`

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/reshape-the-matrix/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<vector<int>> matrixReshape(vector<vector<int>>& mat, int r, int c) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> mat;
    LeetCodeIO::scan(cin, mat);
    int r;
    LeetCodeIO::scan(cin, r);
    int c;
    LeetCodeIO::scan(cin, c);

    Solution *obj = new Solution();
    auto res = obj->matrixReshape(mat, r, c);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

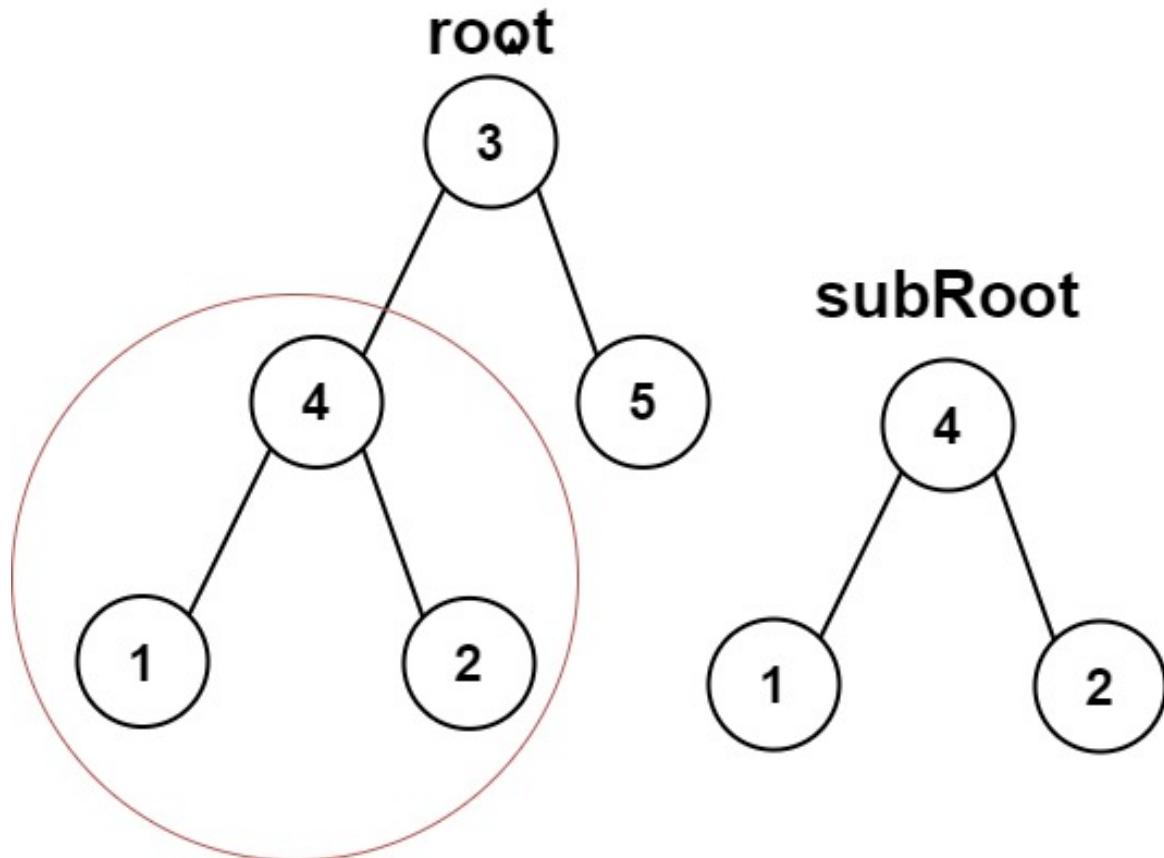
    delete obj;
    return 0;
}
```

## 8.135 572. Subtree of Another Tree (Easy)

Given the roots of two binary trees `root` and `subRoot`, return `true` if there is a subtree of `root` with the same structure and node values of `subRoot` and `false` otherwise.

A subtree of a binary tree `tree` is a tree that consists of a node in `tree` and all of this node's descendants. The tree `tree` could also be considered as a subtree of itself.

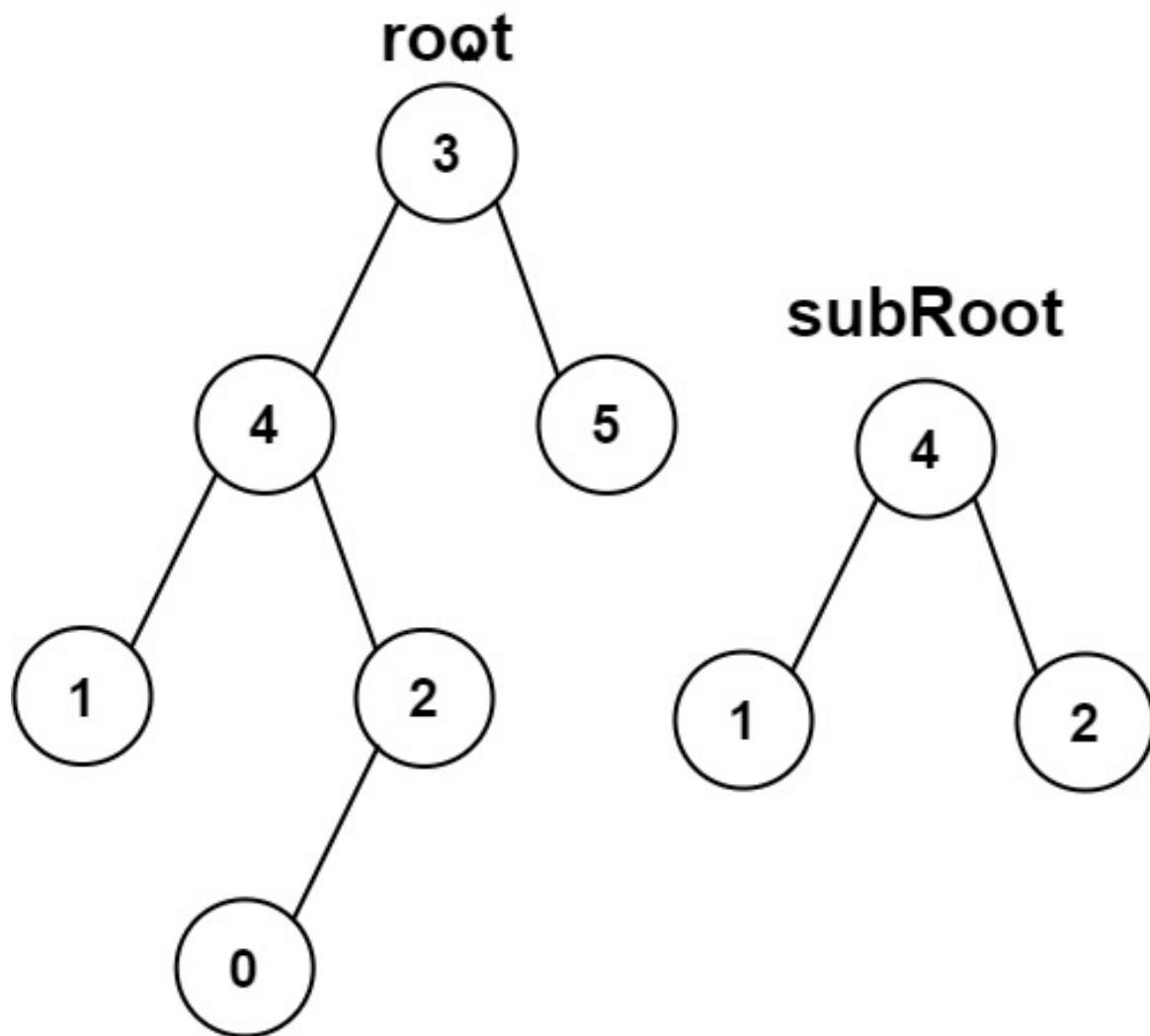
**Example 1:**



Input: `root = [3,4,5,1,2]`, `subRoot = [4,1,2]`

Output: `true`

**Example 2:**



Input: `root = [3,4,5,1,2,null,null,null,null,0]`, `subRoot = [4,1,2]`

Output: false

**Constraints:**

- The number of nodes in the root tree is in the range [1, 2000].
- The number of nodes in the subRoot tree is in the range [1, 1000].
- $-10^4 < \text{root.val} \leq 10^4$
- $-10^4 < \text{subRoot.val} \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/subtree-of-another-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isSubtree(TreeNode* root, TreeNode* subRoot) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    TreeNode* subRoot;
    LeetCodeIO::scan(cin, subRoot);

    Solution *obj = new Solution();
    auto res = obj->isSubtree(root, subRoot);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.136 583. Delete Operation for Two Strings (Medium)

Given two strings `word1` and `word2`, return the minimum number of **steps** required to make `=word1=` and `=word2=` the same.

In one **step**, you can delete exactly one character in either string.

### Example 1:

Input: `word1 = "sea"`, `word2 = "eat"`

Output: 2

Explanation: You need one step to make "sea" to "ea" and another step to make "eat" to "ea".

### Example 2:

Input: `word1 = "leetcode"`, `word2 = "etco"`

Output: 4

### Constraints:

- $1 < \text{word1.length}, \text{word2.length} \leq 500$
- `word1` and `word2` consist of only lowercase English letters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/delete-operation-for-two-strings/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int minDistance(string word1, string word2) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string word1;
    LeetCodeIO::scan(cin, word1);
    string word2;
    LeetCodeIO::scan(cin, word2);

    Solution *obj = new Solution();
    auto res = obj->minDistance(word1, word2);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.137 594. Longest Harmonious Subsequence (Easy)

We define a harmonious array as an array where the difference between its maximum value and its minimum value is **exactly 1**.

Given an integer array `nums`, return the length of its longest harmonious subsequence among all its possible subsequences.

### Example 1:

**Input:** `nums = [1,3,2,2,5,2,3,7]`

**Output:** 5

### Explanation:

The longest harmonious subsequence is `[3, 2, 2, 2, 3]`.

### Example 2:

**Input:** `nums = [1,2,3,4]`

**Output:** 2

### Explanation:

The longest harmonious subsequences are `[1, 2]`, `[2, 3]`, and `[3, 4]`, all of which have a length of 2.

### Example 3:

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

**Output:** 0

### Explanation:

No harmonic subsequence exists.

### Constraints:

- $1 < \text{nums.length} \leq 2 * 10^4$
- $-10 \leq \text{nums}[i] \leq 10^9$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/longest-harmonious-subsequence/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findLHS(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->findLHS(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.138 605. Can Place Flowers (Easy)

You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be planted in **adjacent** plots.

Given an integer array `flowerbed` containing 0's and 1's, where 0 means empty and 1 means not empty, and an integer `n`, return =true=if =n=new flowers can be planted in the =flowerbed=without violating the no-adjacent-flowers rule and =false=otherwise.

### Example 1:

Input: `flowerbed = [1,0,0,0,1]`, `n = 1`

Output: `true`

### Example 2:

Input: `flowerbed = [1,0,0,0,1]`, `n = 2`

Output: `false`

### Constraints:

- $1 < \text{flowerbed.length} \leq 2 * 10^4$
- `flowerbed[i]` is 0 or 1.
- There are no two adjacent flowers in `flowerbed`.
- $0 < n \leq \text{flowerbed.length}$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/can-place-flowers/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool canPlaceFlowers(vector<int>& flowerbed, int n) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> flowerbed;
    LeetCodeIO::scan(cin, flowerbed);
    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->canPlaceFlowers(flowerbed, n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.139 617. Merge Two Binary Trees (Easy)

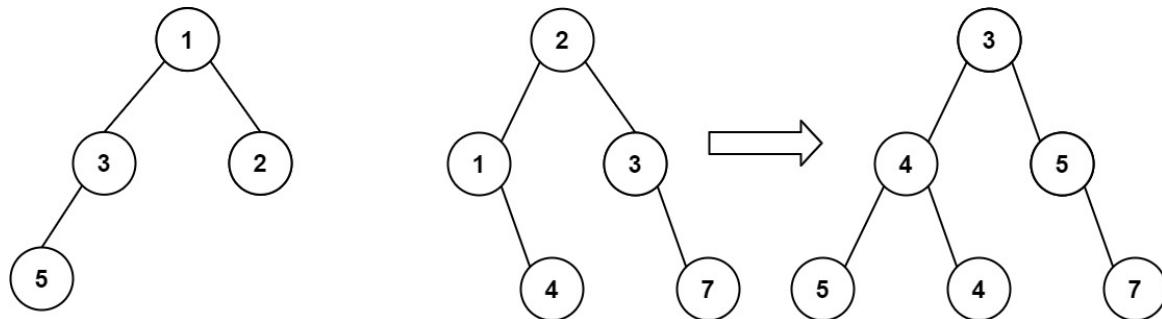
You are given two binary trees `root1` and `root2`.

Imagine that when you put one of them to cover the other, some nodes of the two trees are overlapped while the others are not. You need to merge the two trees into a new binary tree. The merge rule is that if two nodes overlap, then sum node values up as the new value of the merged node. Otherwise, the NOT null node will be used as the node of the new tree.

Return the merged tree.

**Note:** The merging process must start from the root nodes of both trees.

### Example 1:



Input: `root1 = [1,3,2,5]`, `root2 = [2,1,3,null,4,null,7]`

Output: `[3,4,5,5,4,null,7]`

### Example 2:

Input: `root1 = [1]`, `root2 = [1,2]`

Output: `[2,2]`

### Constraints:

- The number of nodes in both trees is in the range  $[0, 2000]$ .
- $-10 \leq \text{Node.val} \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/merge-two-binary-trees/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* mergeTrees(TreeNode* root1, TreeNode* root2) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root1;
    LeetCodeIO::scan(cin, root1);
    TreeNode* root2;
    LeetCodeIO::scan(cin, root2);

    Solution *obj = new Solution();
    auto res = obj->mergeTrees(root1, root2);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.140 633. Sum of Square Numbers (Medium)

Given a non-negative integer  $c$ , decide whether there're two integers  $a$  and  $b$  such that  $a^2 + b^2 = c$ .

**Example 1:**

Input:  $c = 5$

Output: true

Explanation:  $1 * 1 + 2 * 2 = 5$

**Example 2:**

Input:  $c = 3$

Output: false

**Constraints:**

- $0 < c \leq 2^{31} - 1$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/sum-of-square-numbers/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool judgeSquareSum(int c) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int c;
    LeetCodeIO::scan(cin, c);

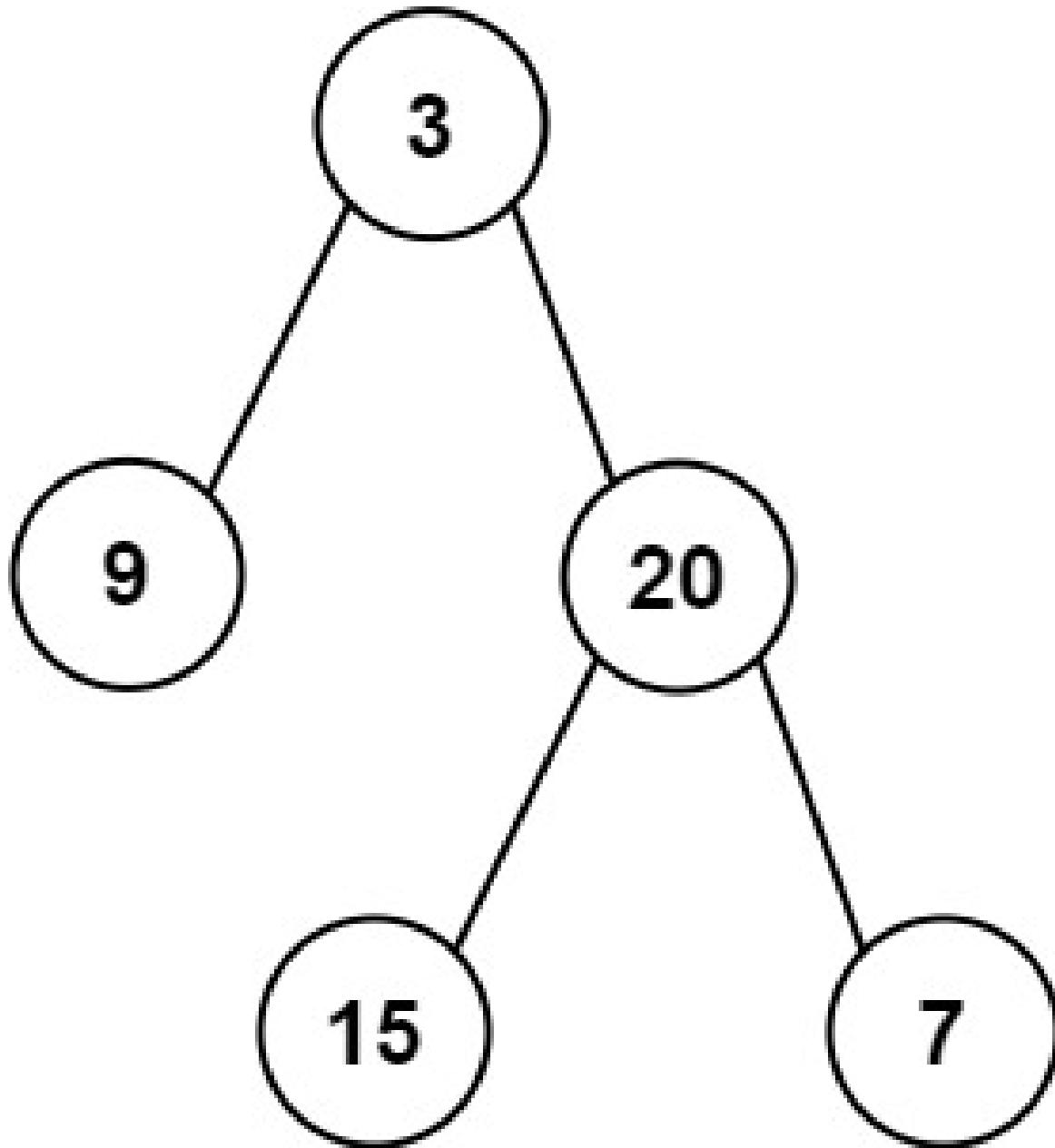
    Solution *obj = new Solution();
    auto res = obj->judgeSquareSum(c);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.141 637. Average of Levels in Binary Tree (Easy)

Given the root of a binary tree, return the average value of the nodes on each level in the form of an array. Answers within  $10^{-5}$  of the actual answer will be accepted.

**Example 1:**

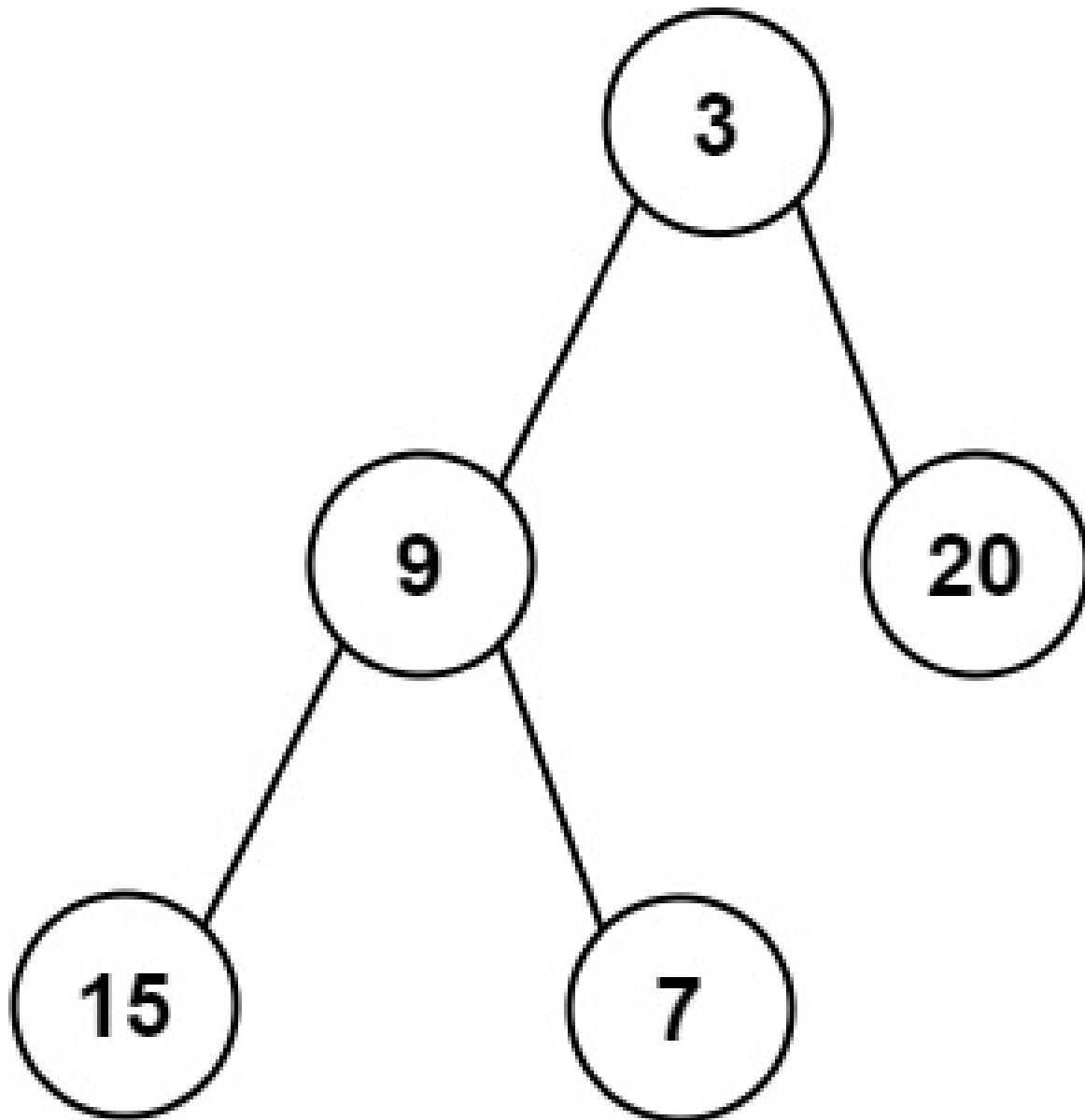


Input: root = [3,9,20,null,null,15,7]

Output: [3.00000,14.50000,11.00000]

Explanation: The average value of nodes on level 0 is 3, on level 1 is 14.5, and on level 2 is 11. Hence return [3, 14.5, 11].

**Example 2:**



Input: root = [3,9,20,15,7]

Output: [3.00000,14.50000,11.00000]

**Constraints:**

- The number of nodes in the tree is in the range [1, 10].
- $-2^{31} < \text{Node.val} \leq 2^{31} - 1$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/average-of-levels-in-binary-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<double> averageOfLevels(TreeNode* root) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->averageOfLevels(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.142 646. Maximum Length of Pair Chain (Medium)

You are given an array of  $n$  pairs  $\text{pairs}$  where  $\text{pairs}[i] = [\text{left}, \text{right}]$  and  $\text{left} < \text{right}$ . A pair  $p_2 = [c, d]$  follows a pair  $p_1 = [a, b]$  if  $b < c$ . A **chain** of pairs can be formed in this fashion.

Return the length longest chain which can be formed.

You do not need to use up all the given intervals. You can select pairs in any order.

### Example 1:

Input:  $\text{pairs} = [[1,2], [2,3], [3,4]]$

Output: 2

Explanation: The longest chain is  $[1,2] \rightarrow [3,4]$ .

### Example 2:

Input:  $\text{pairs} = [[1,2], [7,8], [4,5]]$

Output: 3

Explanation: The longest chain is  $[1,2] \rightarrow [4,5] \rightarrow [7,8]$ .

### Constraints:

- $n = \text{pairs.length} =$
- $1 < n \leq 1000 =$
- $-1000 < \text{left} < \text{right} \leq 1000 =$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/maximum-length-of-pair-chain/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findLongestChain(vector<vector<int>>& pairs) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> pairs;
    LeetCodeIO::scan(cin, pairs);

    Solution *obj = new Solution();
    auto res = obj->findLongestChain(pairs);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.143 647. Palindromic Substrings (Medium)

Given a string  $s$ , return the number of **palindromic substrings** in it.

A string is a **palindrome** when it reads the same backward as forward.

A **substring** is a contiguous sequence of characters within the string.

### Example 1:

Input:  $s = "abc"$

Output: 3

Explanation: Three palindromic strings: "a", "b", "c".

### Example 2:

Input:  $s = "aaa"$

Output: 6

Explanation: Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".

### Constraints:

- $1 < s.length \leq 1000$
- $s$  consists of lowercase English letters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/palindromic-substrings/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int countSubstrings(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->countSubstrings(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.144 650. 2 Keys Keyboard (Medium)

There is only one character 'A' on the screen of a notepad. You can perform one of two operations on this notepad for each step:

- Copy All: You can copy all the characters present on the screen (a partial copy is not allowed).
- Paste: You can paste the characters which are copied last time.

Given an integer  $n$ , return the minimum number of operations to get the character ='A'=exactly  $=n=$ times on the screen.

### Example 1:

Input:  $n = 3$

Output: 3

Explanation: Initially, we have one character 'A'.

In step 1, we use Copy All operation.

In step 2, we use Paste operation to get 'AA'.

In step 3, we use Paste operation to get 'AAA'.

### Example 2:

Input:  $n = 1$

Output: 0

### Constraints:

- $1 < n \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/2-keys-keyboard/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int minSteps(int n) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

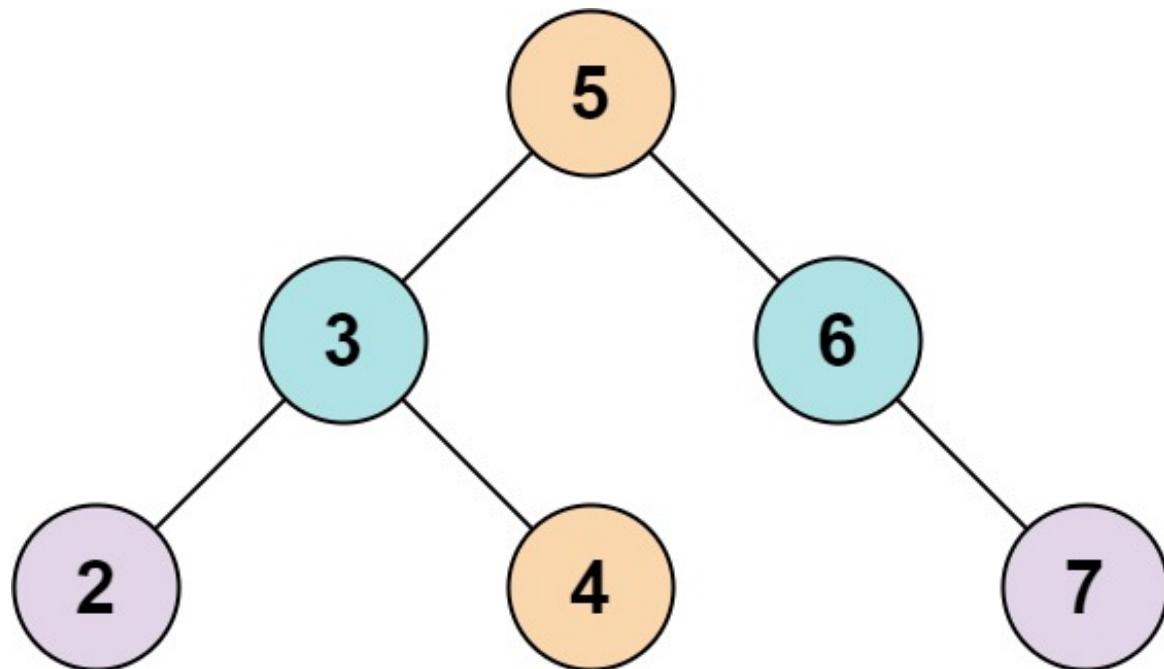
    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->minSteps(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

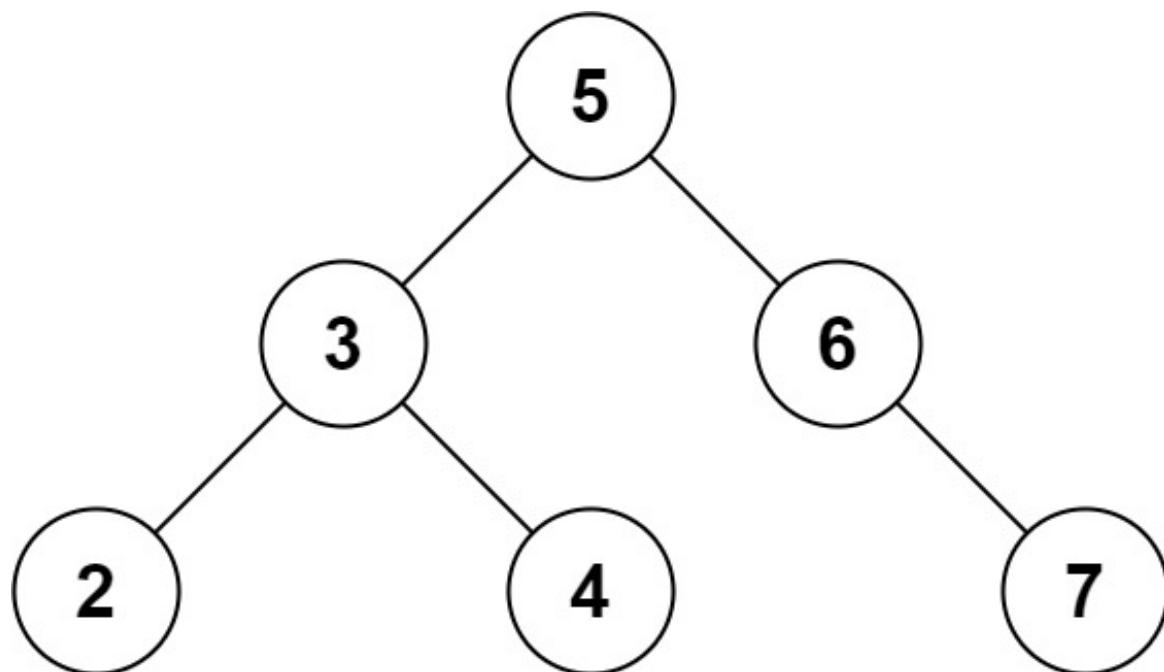
**8.145 653. Two Sum IV - Input is a BST (Easy)**

Given the root of a binary search tree and an integer k, return true if there exist two elements in the BST such that their sum is equal to =k, or =false=otherwise.

**Example 1:**

Input: root = [5,3,6,2,4,null,7], k = 9

Output: true

**Example 2:**

Input: root = [5,3,6,2,4,null,7], k = 28

Output: false

**Constraints:**

- The number of nodes in the tree is in the range [1, 10].
- $-10^4 < \text{Node.val} \leq 10^4$
- `root` is guaranteed to be a **valid** binary search tree.
- $-10^5 < k \leq 10^5$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/two-sum-iv-input-is-a-bst/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool findTarget(TreeNode* root, int k) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    int k;
    LeetCodeIO::scan(cin, k);

    Solution *obj = new Solution();
    auto res = obj->findTarget(root, k);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.146 665. Non-decreasing Array (Medium)

Given an array `nums` with  $n$  integers, your task is to check if it could become non-decreasing by modifying **at most one element**.

We define an array is non-decreasing if  $\text{nums}[i] \leq \text{nums}[i + 1]$  holds for every  $i$  (**0-based**) such that ( $0 < i \leq n - 2$ ).

### Example 1:

Input: `nums = [4, 2, 3]`

Output: `true`

Explanation: You could modify the first 4 to 1 to get a non-decreasing array.

### Example 2:

Input: `nums = [4, 2, 1]`

Output: `false`

Explanation: You cannot get a non-decreasing array by modifying at most one element.

### Constraints:

- $n = \text{nums.length} \geq 1$
- $1 < n \leq 10^4$
- $-10^4 \leq \text{nums}[i] \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/non-decreasing-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool checkPossibility(vector<int>& nums) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->checkPossibility(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

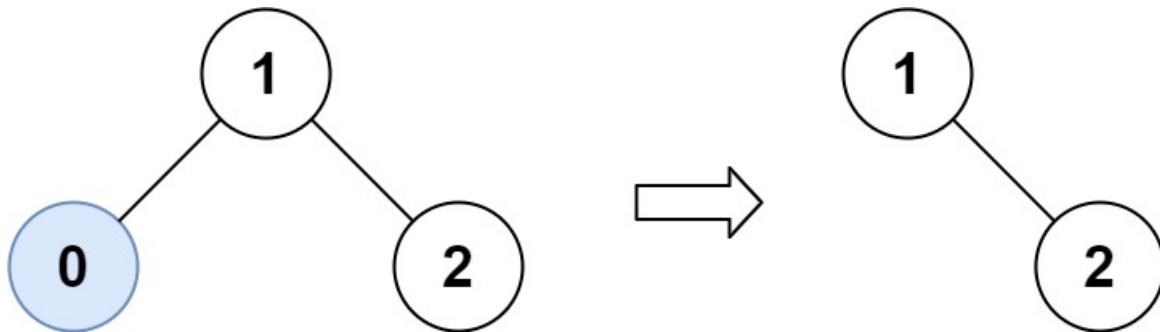
    delete obj;
    return 0;
}
```

## 8.147 669. Trim a Binary Search Tree (Medium)

Given the root of a binary search tree and the lowest and highest boundaries as `low` and `high`, trim the tree so that all its elements lies in `[low, high]`. Trimming the tree should **not** change the relative structure of the elements that will remain in the tree (i.e., any node's descendant should remain a descendant). It can be proven that there is a **unique answer**.

Return the root of the trimmed binary search tree. Note that the root may change depending on the given bounds.

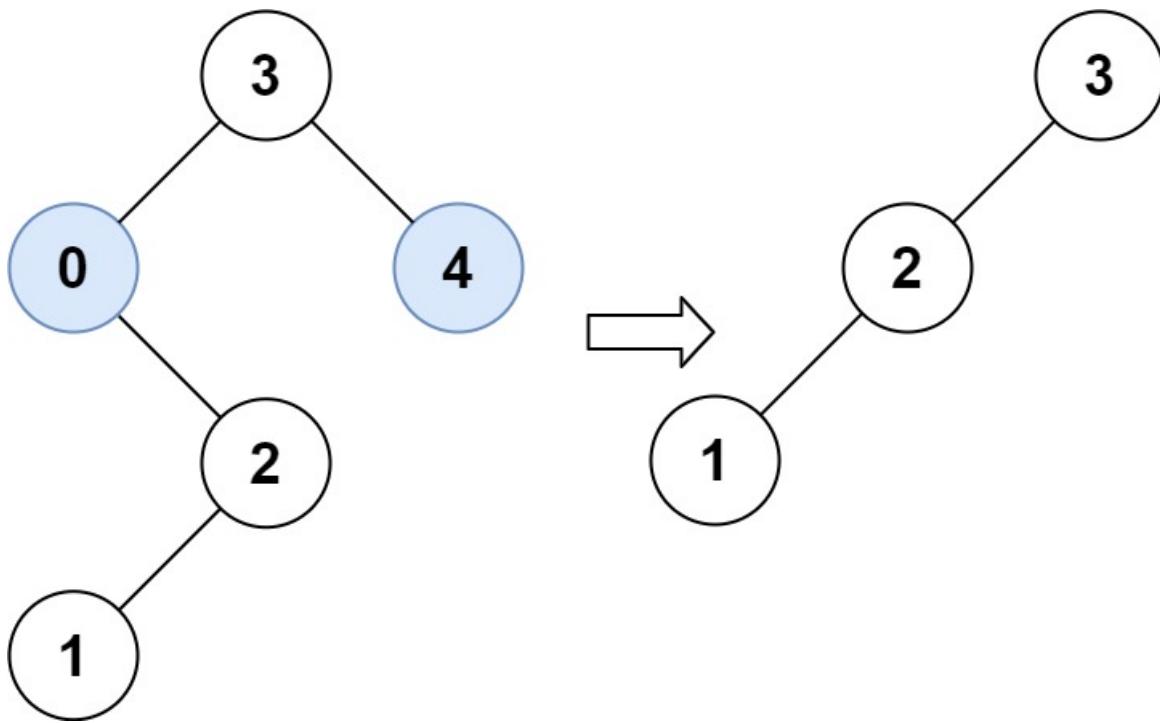
**Example 1:**



Input: `root = [1,0,2], low = 1, high = 2`

Output: `[1,null,2]`

**Example 2:**



Input: `root = [3,0,4,null,2,null,null,1], low = 1, high = 3`

Output: `[3,2,null,1]`

### Constraints:

- The number of nodes in the tree is in the range `[1, 10 ]`.
- $0 < \text{Node.val} \leq 10^4$

- The value of each node in the tree is **unique**.
- **root** is guaranteed to be a valid binary search tree.
- $0 < \text{low} \leq \text{high} \leq 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/trim-a-binary-search-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* trimBST(TreeNode* root, int low, int high) {
        if (!root) return NULL;
        if (root->val < low) return trimBST(root->right, low, high);
        if (root->val > high) return trimBST(root->left, low, high);
        root->left = trimBST(root->left, low, high);
        root->right = trimBST(root->right, low, high);
        return root;
    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    int low;
    LeetCodeIO::scan(cin, low);
    int high;
    LeetCodeIO::scan(cin, high);

    Solution *obj = new Solution();
    auto res = obj->trimBST(root, low, high);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
}
```

## 8.148 680. Valid Palindrome II (Easy)

Given a string  $s$ , return `true`=if the  $=s$  can be palindrome after deleting **at most one** character from it.

### Example 1:

Input:  $s = "aba"$

Output: `true`

### Example 2:

Input:  $s = "abca"$

Output: `true`

Explanation: You could delete the character 'c'.

### Example 3:

Input:  $s = "abc"$

Output: `false`

### Constraints:

- $1 < s.length \leq 10^5$
- $s$  consists of lowercase English letters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/valid-palindrome-ii/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool validPalindrome(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->validPalindrome(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

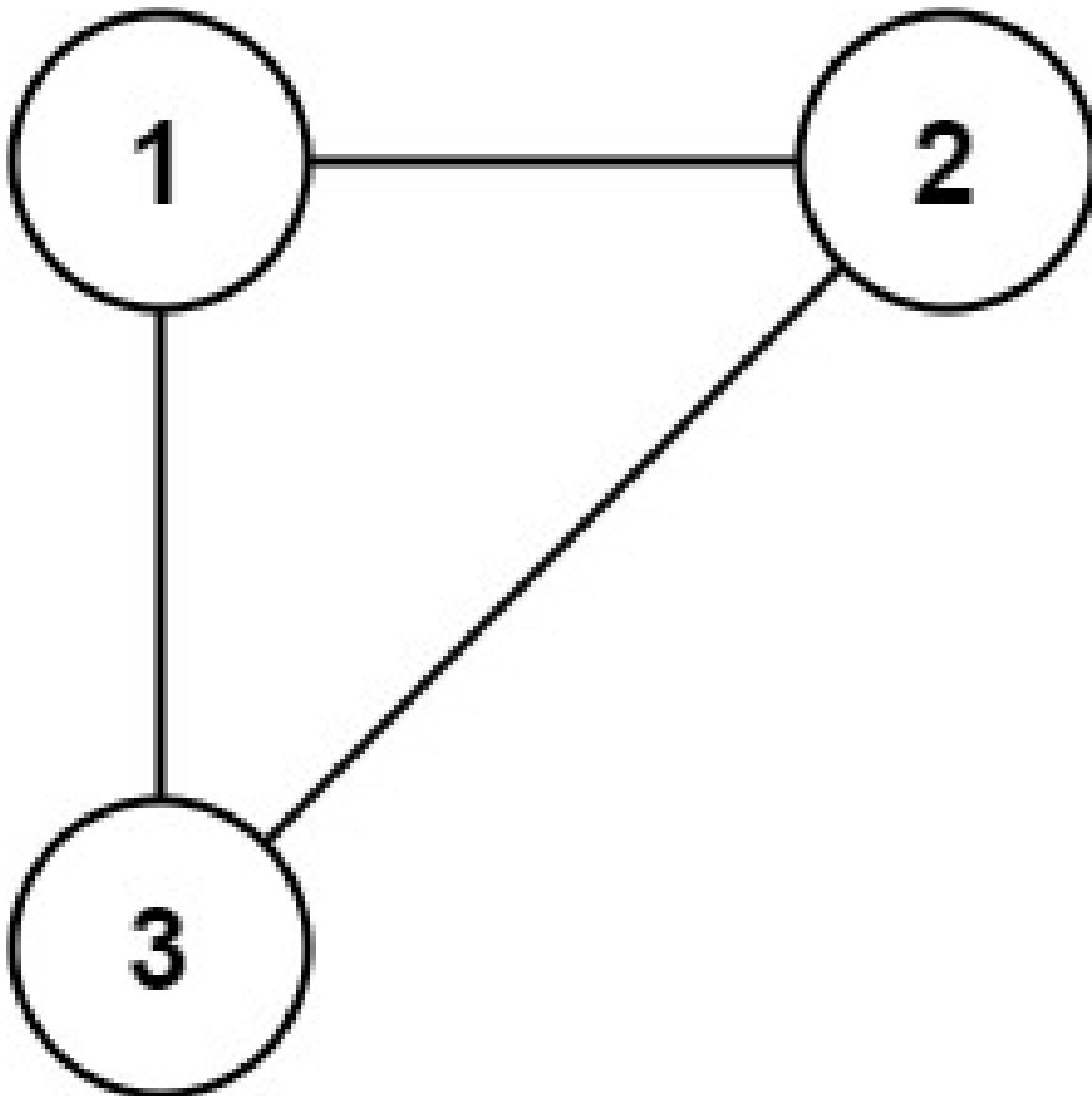
## 8.149 684. Redundant Connection (Medium)

In this problem, a tree is an **undirected graph** that is connected and has no cycles.

You are given a graph that started as a tree with  $n$  nodes labeled from 1 to  $n$ , with one additional edge added. The added edge has two **different** vertices chosen from 1 to  $n$ , and was not an edge that already existed. The graph is represented as an array `edges` of length  $n$  where `edges[i] = [a, b]` indicates that there is an edge between nodes  $a$  and  $b$  in the graph.

Return an edge that can be removed so that the resulting graph is a tree of  $n$  nodes. If there are multiple answers, return the answer that occurs last in the input.

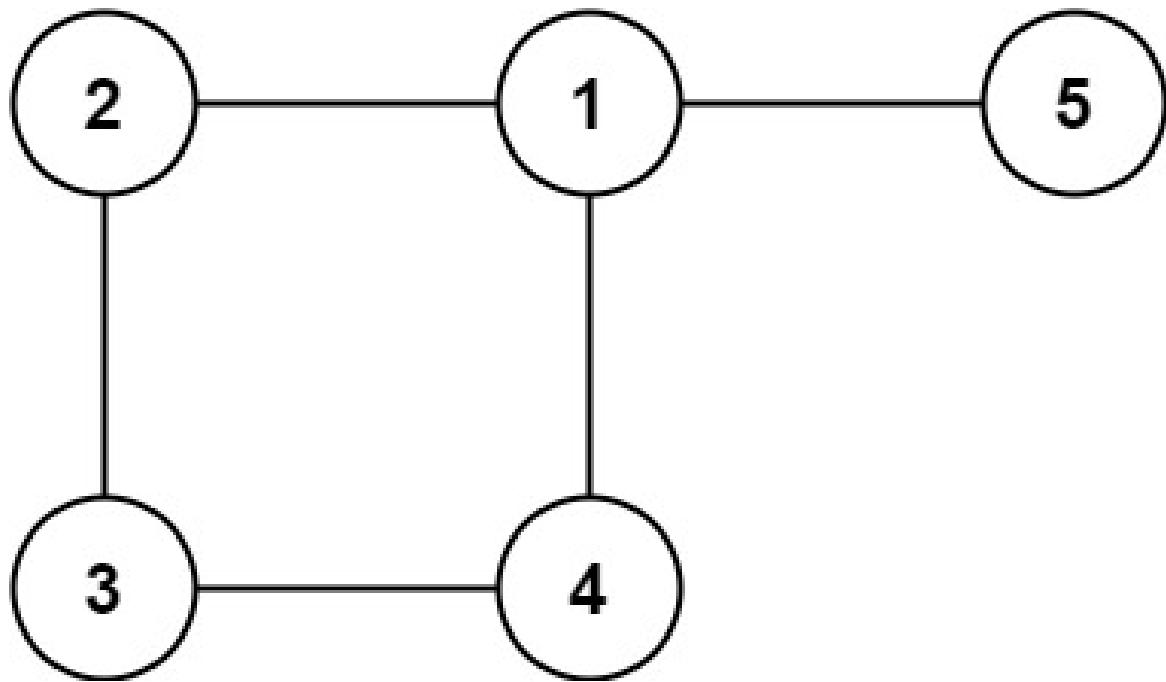
**Example 1:**



Input: `edges = [[1,2],[1,3],[2,3]]`

Output: `[2,3]`

**Example 2:**



Input: edges = [[1,2],[2,3],[3,4],[1,4],[1,5]]

Output: [1,4]

**Constraints:**

- n = edges.length=
- 3 < n <= 1000=
- edges[i].length = 2=
- 1 < a < b <= edges.length=
- a ! b =
- There are no repeated edges.
- The given graph is connected.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/redundant-connection/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> findRedundantConnection(vector<vector<int>>& edges) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> edges;
    LeetCodeIO::scan(cin, edges);

    Solution *obj = new Solution();
    auto res = obj->findRedundantConnection(edges);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.150 693. Binary Number with Alternating Bits (Easy)

Given a positive integer, check whether it has alternating bits: namely, if two adjacent bits will always have different values.

### Example 1:

Input: n = 5

Output: true

Explanation: The binary representation of 5 is: 101

### Example 2:

Input: n = 7

Output: false

Explanation: The binary representation of 7 is: 111.

### Example 3:

Input: n = 11

Output: false

Explanation: The binary representation of 11 is: 1011.

### Constraints:

- $1 < n \leq 2^{31} - 1$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/binary-number-with-alternating-bits/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool hasAlternatingBits(int n) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->hasAlternatingBits(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.151 695. Max Area of Island (Medium)

You are given an  $m \times n$  binary matrix  $\text{grid}$ . An island is a group of 1's (representing land) connected **4-directionally** (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.

The **area** of an island is the number of cells with a value 1 in the island.

Return the maximum **area** of an island in  $\text{grid}$ . If there is no island, return 0.

**Example 1:**

0	0	1	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0
0	1	1	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	0	1	0	1	0	0	0
0	1	0	0	1	1	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0

Input:  $\text{grid} =$

$[[0,0,1,0,0,0,0,1,0,0,0,0,0,0],[0,0,0,0,0,0,0,1,1,1,0,0,0],[0,1,1,0,1,0,0,0,0,0,0,0,0,0],[0,1,0,0,1,1,0,0,0,0,0,0,0,0]]$

Output: 6

Explanation: The answer is not 11, because the island must be connected 4-directionally.

**Example 2:**

Input:  $\text{grid} = [[0,0,0,0,0,0,0,0]]$

Output: 0

**Constraints:**

- $m = \text{grid.length} =$
- $n = \text{grid}[i].length =$
- $1 < m, n \leq 50 =$
- $\text{grid}[i][j]$  is either 0 or 1.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/max-area-of-island/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxAreaOfIsland(vector<vector<int>>& grid) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> grid;
    LeetCodeIO::scan(cin, grid);

    Solution *obj = new Solution();
    auto res = obj->maxAreaOfIsland(grid);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.152 696. Count Binary Substrings (Easy)

Given a binary string  $s$ , return the number of non-empty substrings that have the same number of 0's and 1's, and all the 0's and all the 1's in these substrings are grouped consecutively.

Substrings that occur multiple times are counted the number of times they occur.

### Example 1:

Input:  $s = "00110011"$

Output: 6

Explanation: There are 6 substrings that have equal number of consecutive 1's and 0's: "0011", "01", "1100", "10", "0011", and "01".

Notice that some of these substrings repeat and are counted the number of times they occur.

Also, "00110011" is not a valid substring because all the 0's (and 1's) are not grouped together.

### Example 2:

Input:  $s = "10101"$

Output: 4

Explanation: There are 4 substrings: "10", "01", "10", "01" that have equal number of consecutive 1's and 0's.

### Constraints:

- $1 < s.length \leq 10^5$
- $s[i]$  is either '0' or '1'.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/count-binary-substrings/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int countBinarySubstrings(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->countBinarySubstrings(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.153 697. Degree of an Array (Easy)

Given a non-empty array of non-negative integers `nums`, the **degree** of this array is defined as the maximum frequency of any one of its elements.

Your task is to find the smallest possible length of a (contiguous) subarray of `nums`, that has the same degree as `nums`.

### Example 1:

Input: `nums` = [1,2,2,3,1]

Output: 2

Explanation:

The input array has a degree of 2 because both elements 1 and 2 appear twice.

Of the subarrays that have the same degree:

[1, 2, 2, 3, 1], [1, 2, 2, 3], [2, 2, 3, 1], [1, 2, 2], [2, 2, 3], [2, 2]

The shortest length is 2. So return 2.

### Example 2:

Input: `nums` = [1,2,2,3,1,4,2]

Output: 6

Explanation:

The degree is 3 because the element 2 is repeated 3 times.

So [2,2,3,1,4,2] is the shortest subarray, therefore returning 6.

### Constraints:

- `nums.length` will be between 1 and 50,000.
- `nums[i]` will be an integer between 0 and 49,999.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/degree-of-an-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int findShortestSubArray(vector<int>& nums) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums;
    LeetCodeIO::scan(cin, nums);

    Solution *obj = new Solution();
    auto res = obj->findShortestSubArray(nums);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.154 714. Best Time to Buy and Sell Stock with Transaction Fee (Medium)

You are given an array `prices` where `prices[i]` is the price of a given stock on the `i` day, and an integer `fee` representing a transaction fee.

Find the maximum profit you can achieve. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction.

**Note:**

- You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy again).
- The transaction fee is only charged once for each stock purchase and sale.

**Example 1:**

Input: `prices = [1,3,2,8,4,9]`, `fee = 2`

Output: 8

Explanation: The maximum profit can be achieved by:

- Buying at `prices[0] = 1`
- Selling at `prices[3] = 8`
- Buying at `prices[4] = 4`
- Selling at `prices[5] = 9`

The total profit is  $((8 - 1) - 2) + ((9 - 4) - 2) = 8$ .

**Example 2:**

Input: `prices = [1,3,7,5,10,3]`, `fee = 3`

Output: 6

**Constraints:**

- $1 < \text{prices.length} \leq 5 * 10^4$
- $1 < \text{prices}[i] \leq 5 * 10^4$
- $0 < \text{fee} \leq 5 * 10^4$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/best-time-to-buy-and-sell-stock-with-transaction-fee/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxProfit(vector<int>& prices, int fee) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> prices;
    LeetCodeIO::scan(cin, prices);
    int fee;
    LeetCodeIO::scan(cin, fee);

    Solution *obj = new Solution();
    auto res = obj->maxProfit(prices, fee);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.155 739. Daily Temperatures (Medium)

Given an array of integers `temperatures` represents the daily temperatures, return an array `answer` such that `answer[i]` is the number of days you have to wait after the `i`-th day to get a warmer temperature. If there is no future day for which this is possible, keep `answer[i] = 0` instead.

### Example 1:

Input: `temperatures = [73,74,75,71,69,72,76,73]`

Output: `[1,1,4,2,1,1,0,0]`

### Example 2:

Input: `temperatures = [30,40,50,60]`

Output: `[1,1,1,0]`

### Example 3:

Input: `temperatures = [30,60,90]`

Output: `[1,1,0]`

### Constraints:

- $1 < \text{temperatures.length} \leq 10^5$
- $30 < \text{temperatures}[i] \leq 100$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/daily-temperatures/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> dailyTemperatures(vector<int>& temperatures) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> temperatures;
    LeetCodeIO::scan(cin, temperatures);

    Solution *obj = new Solution();
    auto res = obj->dailyTemperatures(temperatures);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.156 763. Partition Labels (Medium)

You are given a string  $s$ . We want to partition the string into as many parts as possible so that each letter appears in at most one part. For example, the string "ababcc" can be partitioned into ["abab", "cc"], but partitions such as ["aba", "bcc"] or ["ab", "ab", "cc"] are invalid.

Note that the partition is done so that after concatenating all the parts in order, the resultant string should be  $s$ .

Return a list of integers representing the size of these parts.

### Example 1:

Input:  $s = \text{"ababcbacadefegdehijhklij"}$

Output: [9,7,8]

Explanation:

The partition is "ababcbaca", "defegde", "hijhklij".

This is a partition so that each letter appears in at most one part.

A partition like "ababcbacadefegde", "hijhklij" is incorrect, because it splits  $s$  into less parts.

### Example 2:

Input:  $s = \text{"eccbbbbdec"}$

Output: [10]

### Constraints:

- $1 < s.length \leq 500$
- $s$  consists of lowercase English letters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/partition-labels/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> partitionLabels(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->partitionLabels(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.157 769. Max Chunks To Make Sorted (Medium)

You are given an integer array `arr` of length  $n$  that represents a permutation of the integers in the range  $[0, n - 1]$ . We split `arr` into some number of **chunks** (i.e., partitions), and individually sort each chunk. After concatenating them, the result should equal the sorted array.

Return the largest number of chunks we can make to sort the array.

### Example 1:

**Input:** arr = [4,3,2,1,0]

**Output:** 1

**Explanation:**

Splitting into two or more chunks will not return the required result.

For example, splitting into [4, 3], [2, 1, 0] will result in [3, 4, 0, 1, 2], which isn't sorted.

### Example 2:

**Input:** arr = [1,0,2,3,4]

**Output:** 4

**Explanation:**

We can split into two chunks, such as [1, 0], [2, 3, 4].

However, splitting into [1, 0], [2], [3], [4] is the highest number of chunks possible.

### Constraints:

- $n = \text{arr.length} =$
- $1 < n \leq 10 =$
- $0 < \text{arr}[i] < n =$
- All the elements of `arr` are **unique**.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/max-chunks-to-make-sorted/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int maxChunksToSorted(vector<int>& arr) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> arr;
    LeetCodeIO::scan(cin, arr);

    Solution *obj = new Solution();
    auto res = obj->maxChunksToSorted(arr);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.158 785. Is Graph Bipartite? (Medium)

There is an **undirected** graph with  $n$  nodes, where each node is numbered between 0 and  $n - 1$ . You are given a 2D array `graph`, where `graph[u]` is an array of nodes that node  $u$  is adjacent to. More formally, for each  $v$  in `graph[u]`,

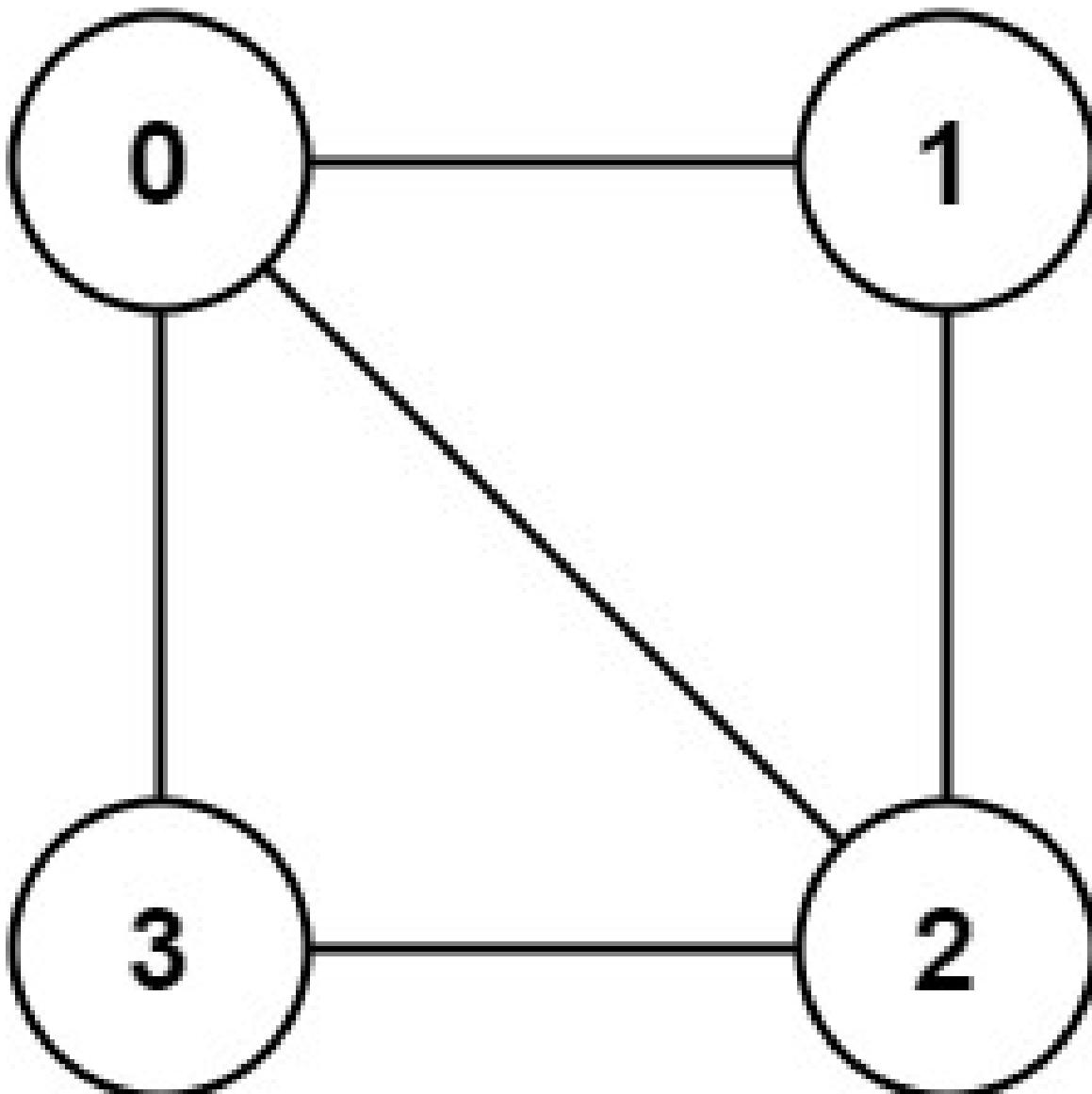
there is an undirected edge between node  $u$  and node  $v$ . The graph has the following properties:

- There are no self-edges (`graph[u]` does not contain  $u$ ).
- There are no parallel edges (`graph[u]` does not contain duplicate values).
- If  $v$  is in `graph[u]`, then  $u$  is in `graph[v]` (the graph is undirected).
- The graph may not be connected, meaning there may be two nodes  $u$  and  $v$  such that there is no path between them.

A graph is **bipartite** if the nodes can be partitioned into two independent sets  $A$  and  $B$  such that **every** edge in the graph connects a node in set  $A$  and a node in set  $B$ .

Return `true` if and only if it is **bipartite**.

**Example 1:**



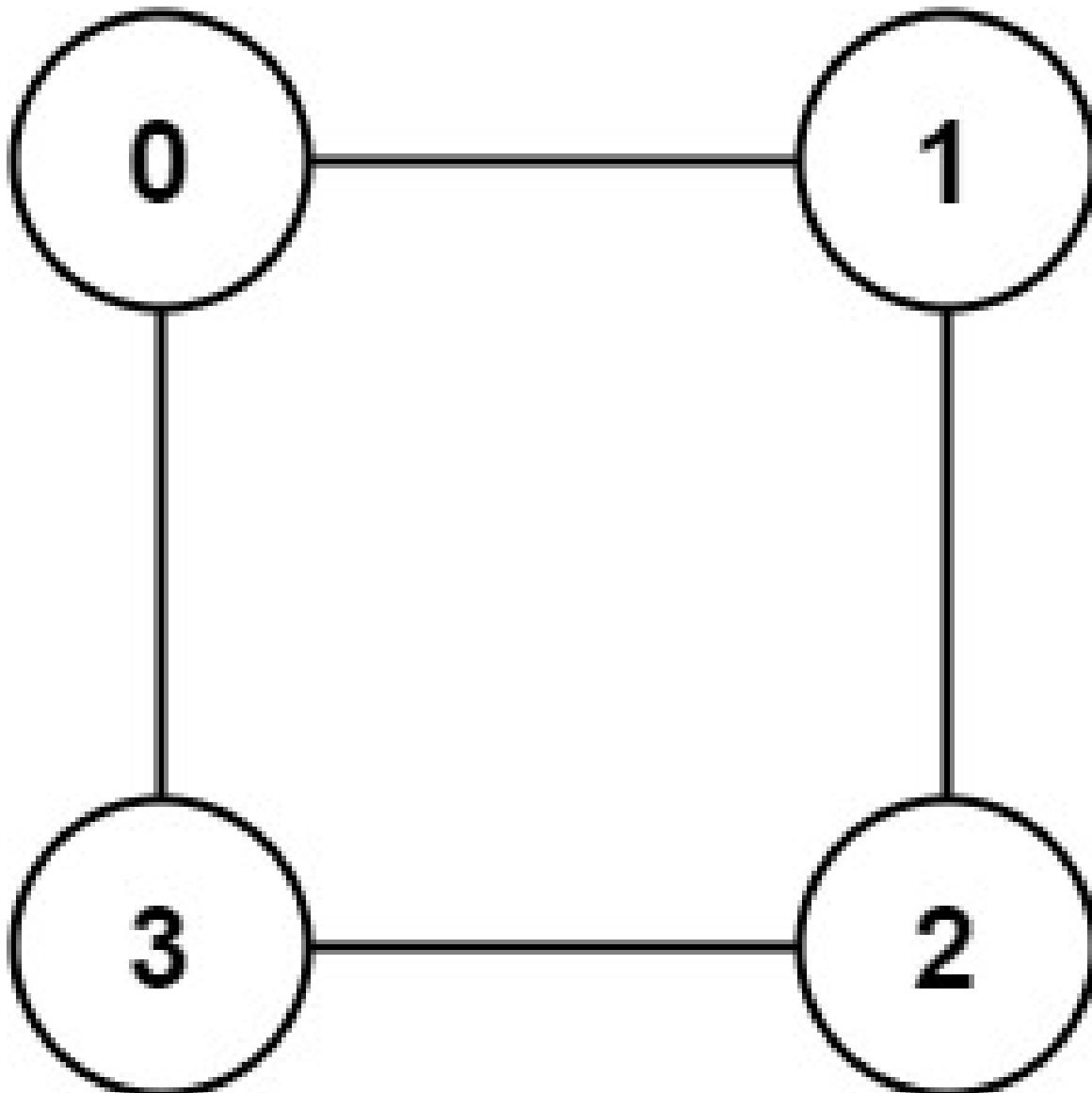
**Input:** `graph = [[1,2,3], [0,2], [0,1,3], [0,2]]`

**Output:** `false`

**Explanation:** There is no way to partition the nodes into two independent sets such that every edge

connects a node in one and a node in the other.

**Example 2:**



Input: graph = [[1,3],[0,2],[1,3],[0,2]]

Output: true

Explanation: We can partition the nodes into two sets: {0, 2} and {1, 3}.

**Constraints:**

- graph.length = n=
- 1 < n <= 100=
- 0 < graph[u].length < n=
- 0 < graph[u][i] <= n - 1=
- graph [u] does not contain u.
- All the values of graph [u] are **unique**.
- If graph [u] contains v, then graph [v] contains u.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/is-graph-bipartite/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    bool isBipartite(vector<vector<int>>& graph) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> graph;
    LeetCodeIO::scan(cin, graph);

    Solution *obj = new Solution();
    auto res = obj->isBipartite(graph);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.159 870. Advantage Shuffle (Medium)

You are given two integer arrays `nums1` and `nums2` both of the same length. The **advantage** of `nums1` with respect to `nums2` is the number of indices  $i$  for which  $\text{nums1}[i] > \text{nums2}[i]$ .

Return any permutation of `nums1` that maximizes its **advantage** with respect to `nums2`.

### Example 1:

Input: `nums1 = [2,7,11,15]`, `nums2 = [1,10,4,11]`

Output: `[2,11,7,15]`

### Example 2:

Input: `nums1 = [12,24,8,32]`, `nums2 = [13,25,32,11]`

Output: `[24,32,8,12]`

### Constraints:

- $1 < \text{nums1.length} \leq 10^5$
- $\text{nums2.length} = \text{nums1.length}$
- $0 < \text{nums1}[i], \text{nums2}[i] \leq 10^9$

## 題解如下：

```

// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/advantage-shuffle/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> advantageCount(vector<int>& nums1, vector<int>& nums2) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> nums1;
    LeetCodeIO::scan(cin, nums1);
    vector<int> nums2;
    LeetCodeIO::scan(cin, nums2);

    Solution *obj = new Solution();
    auto res = obj->advantageCount(nums1, nums2);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}

```

## 8.160 882. Reachable Nodes In Subdivided Graph (Hard)

You are given an undirected graph (the "original graph") with  $n$  nodes labeled from 0 to  $n - 1$ . You decide to **subdivide** each edge in the graph into a chain of nodes, with the number of new nodes varying between each edge. The graph is given as a 2D array of edges where  $\text{edges}[i] = [u, v, \text{cnt}]$  indicates that there is an edge between nodes  $u$  and  $v$  in the original graph, and  $\text{cnt}$  is the total number of new nodes that you will **subdivide** the edge into.

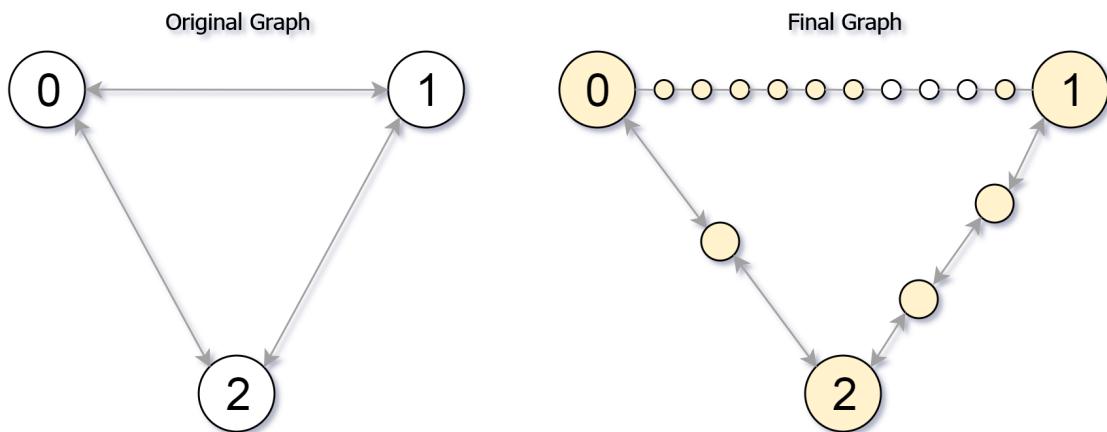
Note that  $\text{cnt} = 0$  means you will not subdivide the edge.

To **subdivide** the edge  $[u, v]$ , replace it with  $(\text{cnt} + 1)$  new edges and  $\text{cnt}$  new nodes. The new nodes are  $x_0, x_1, \dots, x_{\text{cnt}}$ , and the new edges are  $[u, x_0], [x_0, x_1], [x_1, x_2], \dots, [x_{\text{cnt}-1}, x_{\text{cnt}}], [x_{\text{cnt}}, v]$ .

In this **new graph**, you want to know how many nodes are **reachable** from the node 0, where a node is **reachable** if the distance is  $\text{maxMoves}$  or less.

Given the original graph and  $\text{maxMoves}$ , return the number of nodes that are **reachable** from node 0 in the new graph.

### Example 1:



**Input:**  $\text{edges} = [[0,1,10], [0,2,1], [1,2,2]]$ ,  $\text{maxMoves} = 6$ ,  $n = 3$

**Output:** 13

**Explanation:** The edge subdivisions are shown in the image above.

The nodes that are reachable are highlighted in yellow.

### Example 2:

**Input:**  $\text{edges} = [[0,1,4], [1,2,6], [0,2,8], [1,3,1]]$ ,  $\text{maxMoves} = 10$ ,  $n = 4$

**Output:** 23

### Example 3:

**Input:**  $\text{edges} = [[1,2,4], [1,4,5], [1,3,1], [2,3,4], [3,4,5]]$ ,  $\text{maxMoves} = 17$ ,  $n = 5$

**Output:** 1

**Explanation:** Node 0 is disconnected from the rest of the graph, so only node 0 is reachable.

### Constraints:

- $0 < \text{edges.length} \leq \min(n * (n - 1) / 2, 10^4)$
- $\text{edges}[i].length = 3$
- $0 < u < v < n$
- There are **no multiple edges** in the graph.

- $0 < \text{cnt} \leq 10^4$ =
- $0 < \text{maxMoves} \leq 10^9$ =
- $1 < n \leq 3000$ =

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/reachable-nodes-in-subdivided-graph/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int reachableNodes(vector<vector<int>>& edges, int maxMoves, int n) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> edges;
    LeetCodeIO::scan(cin, edges);
    int maxMoves;
    LeetCodeIO::scan(cin, maxMoves);
    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->reachableNodes(edges, maxMoves, n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

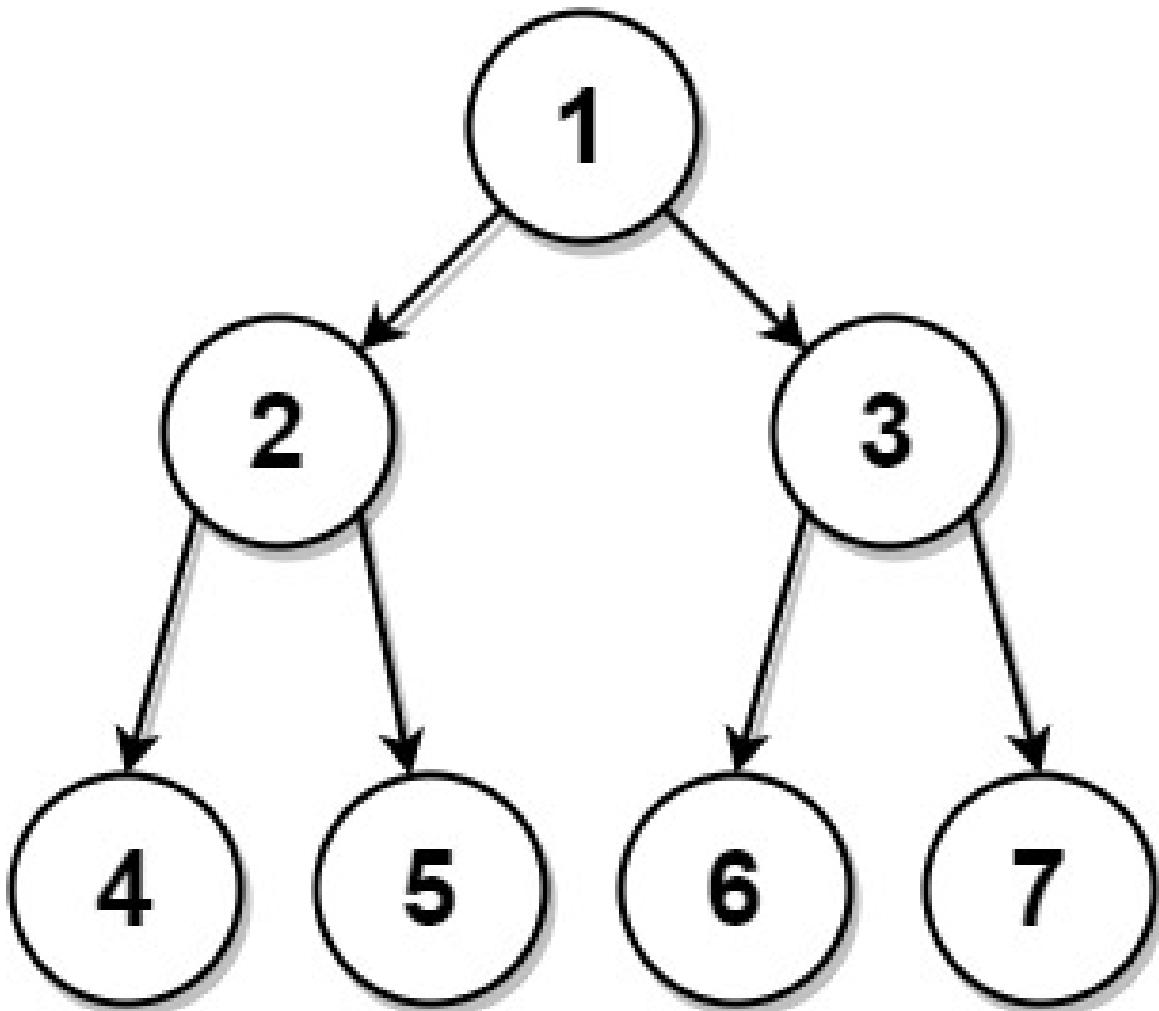
    delete obj;
    return 0;
}
```

## 8.161 889. Construct Binary Tree from Preorder and Postorder Traversal (Medium)

Given two integer arrays, preorder and postorder where preorder is the preorder traversal of a binary tree of **distinct** values and postorder is the postorder traversal of the same tree, reconstruct and return the binary tree.

If there exist multiple answers, you can **return any** of them.

### Example 1:



Input: preorder = [1,2,4,5,3,6,7], postorder = [4,5,2,6,7,3,1]

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

### Example 2:

Input: preorder = [1], postorder = [1]

Output: [1]

### Constraints:

- $1 < \text{preorder.length} \leq 30$
- $1 < \text{preorder}[i] \leq \text{preorder.length}$
- All the values of preorder are **unique**.
- $\text{postorder.length} = \text{preorder.length}$

- $1 < \text{postorder}[i] \leq \text{postorder.length}$
- All the values of `postorder` are **unique**.
- It is guaranteed that `preorder` and `postorder` are the preorder traversal and postorder traversal of the same binary tree.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/construct-binary-tree-from-preorder-and-postorder-traversal/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* constructFromPrePost(vector<int>& preorder, vector<int>& postorder) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<int> preorder;
    LeetCodeIO::scan(cin, preorder);
    vector<int> postorder;
    LeetCodeIO::scan(cin, postorder);

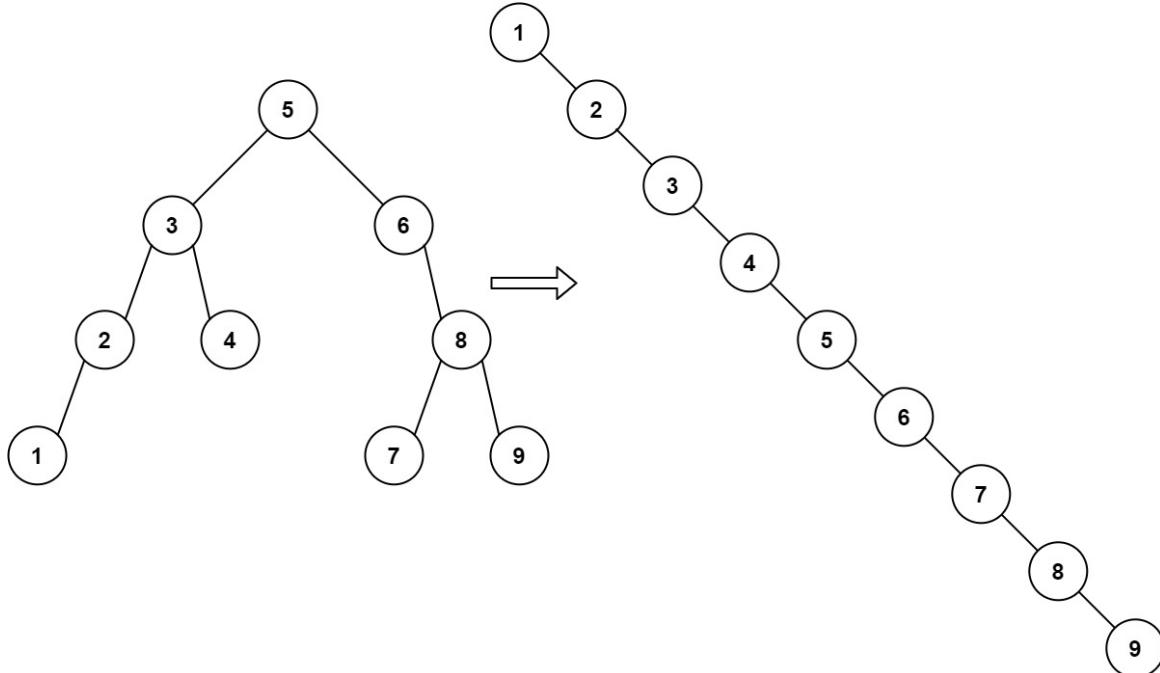
    Solution *obj = new Solution();
    auto res = obj->constructFromPrePost(preorder, postorder);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.162 897. Increasing Order Search Tree (Easy)

Given the root of a binary search tree, rearrange the tree in **in-order** so that the leftmost node in the tree is now the root of the tree, and every node has no left child and only one right child.

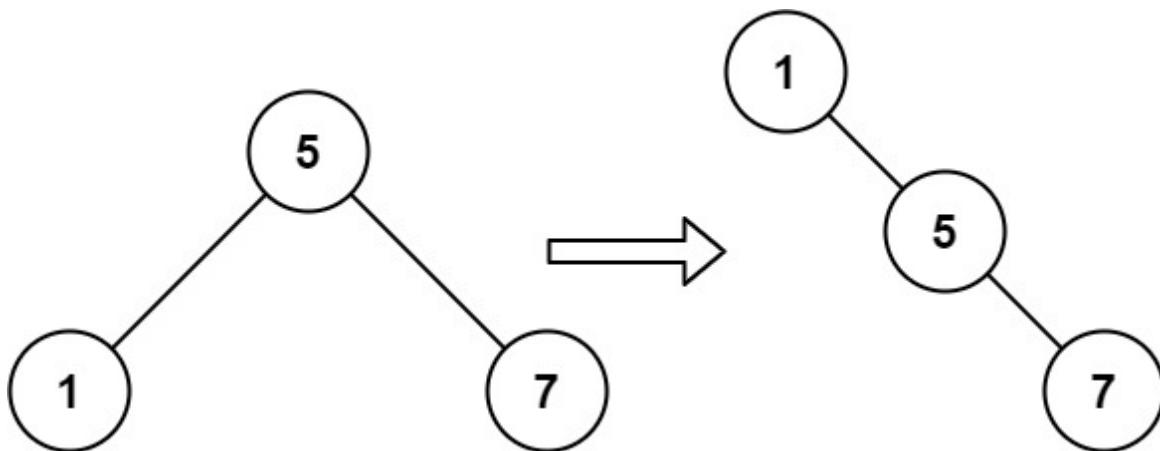
**Example 1:**



Input: `root = [5,3,6,2,4,null,8,1,null,null,null,7,9]`

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

**Example 2:**



Input: `root = [5,1,7]`

Output: `[1,null,5,null,7]`

### Constraints:

- The number of nodes in the given tree will be in the range `[1, 100]`.
- $0 < \text{Node.val} \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/increasing-order-search-tree/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    TreeNode* increasingBST(TreeNode* root) {
        }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);

    Solution *obj = new Solution();
    auto res = obj->increasingBST(root);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.163 932. Beautiful Array (Medium)

An array `nums` of length `n` is **beautiful** if:

- `nums` is a permutation of the integers in the range `[1, n]`.
- For every  $0 < i < j < n$ , there is no index `k` with  $i < k < j$  where  $2 * \text{nums}[k] = \text{nums}[i] + \text{nums}[j]$ .

Given the integer `n`, return any **beautiful** array `nums` of length `n`. There will be at least one valid answer for the given `n`.

**Example 1:**

Input: `n = 4`

Output: `[2,1,4,3]`

**Example 2:**

Input: `n = 5`

Output: `[3,1,2,5,4]`

**Constraints:**

- $1 < n \leq 1000$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/beautiful-array/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<int> beautifulArray(int n) {
        }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    int n;
    LeetCodeIO::scan(cin, n);

    Solution *obj = new Solution();
    auto res = obj->beautifulArray(n);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.164 934. Shortest Bridge (Medium)

You are given an  $n \times n$  binary matrix  $grid$  where 1 represents land and 0 represents water.

An **island** is a 4-directionally connected group of 1's not connected to any other 1's. There are **exactly two islands** in  $grid$ .

You may change 0's to 1's to connect the two islands to form **one island**.

Return the smallest number of 0's you must flip to connect the two islands.

### Example 1:

Input:  $grid = [[0,1],[1,0]]$

Output: 1

### Example 2:

Input:  $grid = [[0,1,0],[0,0,0],[0,0,1]]$

Output: 2

### Example 3:

Input:  $grid = [[1,1,1,1,1],[1,0,0,0,1],[1,0,1,0,1],[1,0,0,0,1],[1,1,1,1,1]]$

Output: 1

### Constraints:

- $n = grid.length = grid[i].length$
- $2 < n \leq 100$
- $grid[i][j]$  is either 0 or 1.
- There are exactly two islands in  $grid$ .

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/shortest-bridge/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int shortestBridge(vector<vector<int>>& grid) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> grid;
    LeetCodeIO::scan(cin, grid);

    Solution *obj = new Solution();
    auto res = obj->shortestBridge(grid);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.165 1091. Shortest Path in Binary Matrix (Medium)

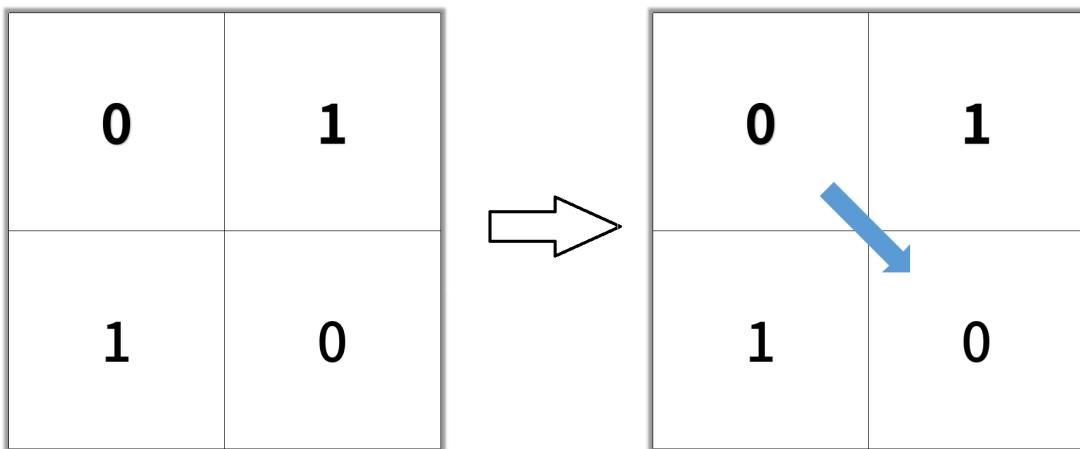
Given an  $n \times n$  binary matrix grid, return the length of the shortest **clear path** in the matrix. If there is no clear path, return -1.

A **clear path** in a binary matrix is a path from the **top-left** cell (i.e.,  $(0, 0)$ ) to the **bottom-right** cell (i.e.,  $(n - 1, n - 1)$ ) such that:

- All the visited cells of the path are 0.
- All the adjacent cells of the path are **8-directionally** connected (i.e., they are different and they share an edge or a corner).

The **length of a clear path** is the number of visited cells of this path.

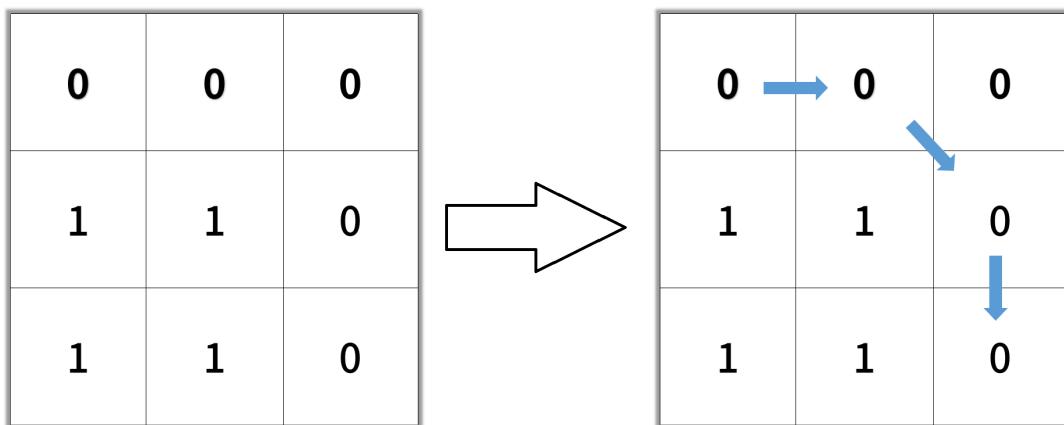
**Example 1:**



Input: `grid = [[0,1],[1,0]]`

Output: 2

**Example 2:**



Input: `grid = [[0,0,0],[1,1,0],[1,1,0]]`

Output: 4

**Example 3:**

Input: grid = [[1,0,0],[1,1,0],[1,1,0]]

Output: -1

**Constraints:**

- n = grid.length=
- n = grid[i].length=
- 1 < n <= 100=
- grid[i][j] is 0 or 1

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/shortest-path-in-binary-matrix/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int shortestPathBinaryMatrix(vector<vector<int>>& grid) {
        }

    }

};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> grid;
    LeetCodeIO::scan(cin, grid);

    Solution *obj = new Solution();
    auto res = obj->shortestPathBinaryMatrix(grid);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.166 1105. Filling Bookcase Shelves (Medium)

You are given an array `books` where `books[i] = [thickness, height]` indicates the thickness and height of the `i` book. You are also given an integer `shelfWidth`.

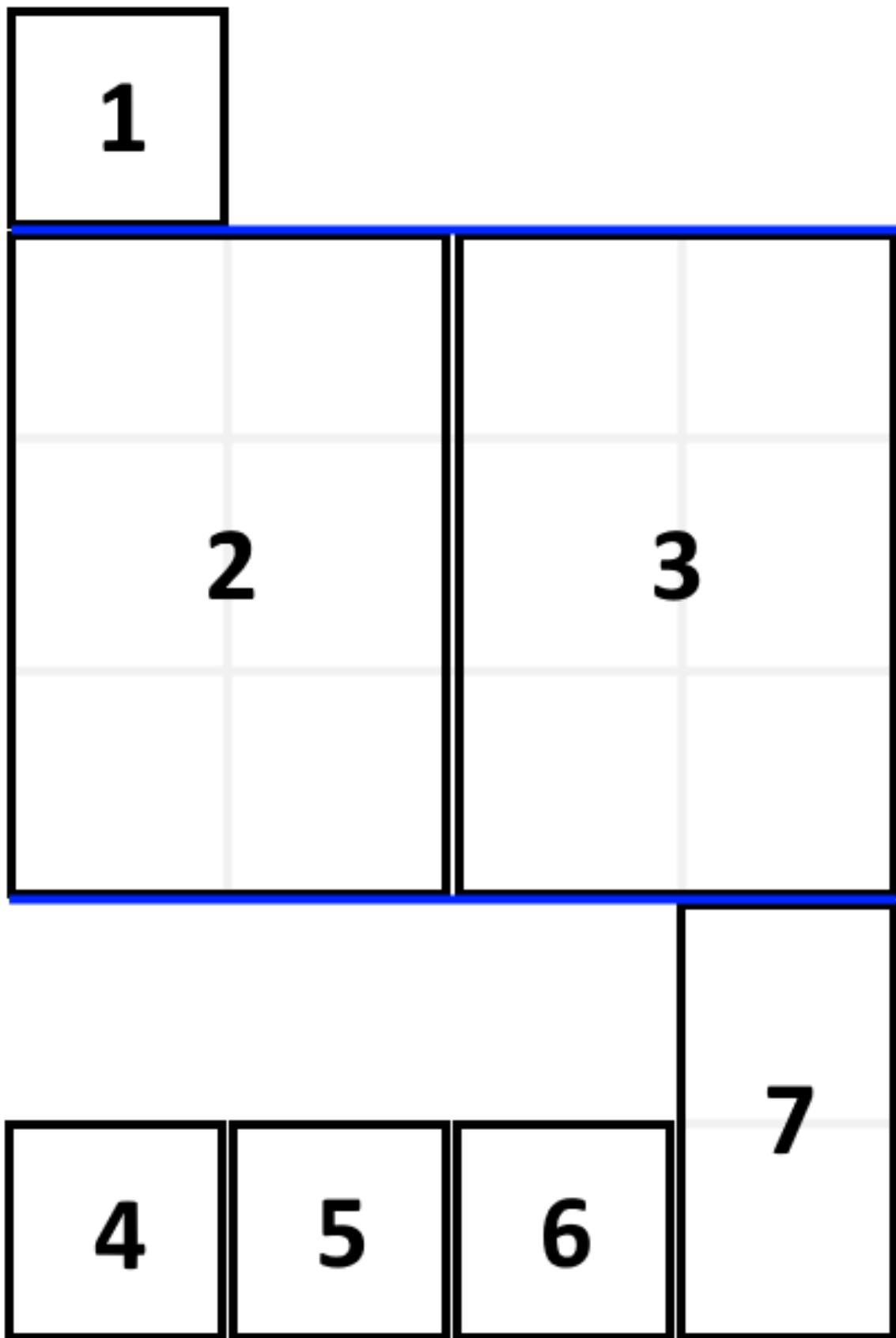
We want to place these books in order onto bookcase shelves that have a total width `shelfWidth`.

We choose some of the books to place on this shelf such that the sum of their thickness is less than or equal to `shelfWidth`, then build another level of the shelf of the bookcase so that the total height of the bookcase has increased by the maximum height of the books we just put down. We repeat this process until there are no more books to place. Note that at each step of the above process, the order of the books we place is the same order as the given sequence of books.

- For example, if we have an ordered list of 5 books, we might place the first and second book onto the first shelf, the third book on the second shelf, and the fourth and fifth book on the last shelf.

Return the minimum possible height that the total bookshelf can be after placing shelves in this manner.

**Example 1:**



Input: books = [[1,1],[2,3],[2,3],[1,1],[1,1],[1,1],[1,2]], shelfWidth = 4

Output: 6

Explanation:

The sum of the heights of the 3 shelves is  $1 + 3 + 2 = 6$ .

Notice that book number 2 does not have to be on the first shelf.

**Example 2:**

Input: books = [[1,3],[2,4],[3,2]], shelfWidth = 6

Output: 4

**Constraints:**

- $1 < \text{books.length} \leq 1000$ =
- $1 < \text{thickness} \leq \text{shelfWidth} \leq 1000$ =
- $1 < \text{height} \leq 1000$ =

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/filling-bookcase-shelves/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int minHeightShelves(vector<vector<int>>& books, int shelfWidth) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    vector<vector<int>> books;
    LeetCodeIO::scan(cin, books);
    int shelfWidth;
    LeetCodeIO::scan(cin, shelfWidth);

    Solution *obj = new Solution();
    auto res = obj->minHeightShelves(books, shelfWidth);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

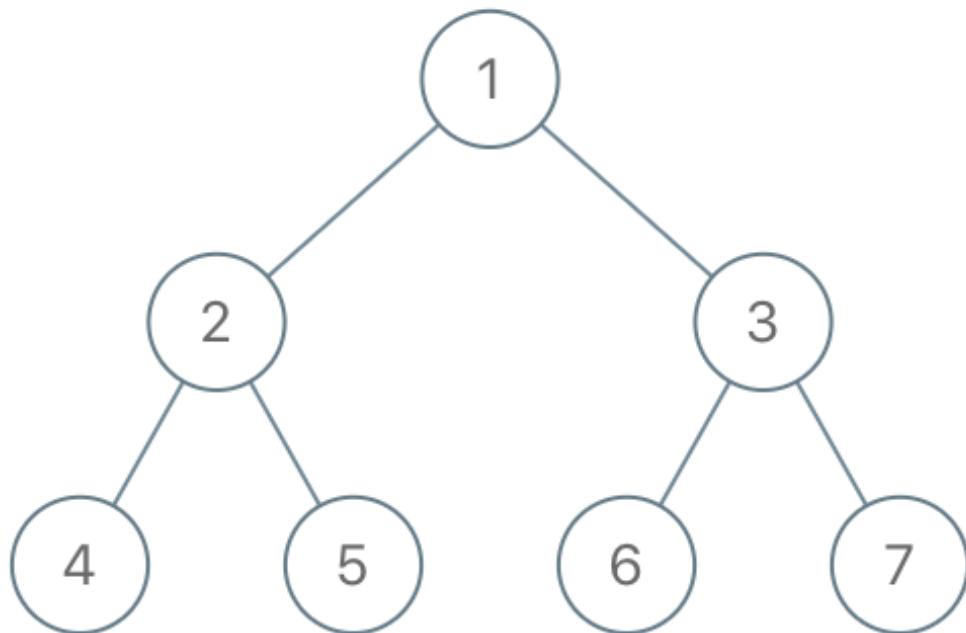
## 8.167 1110. Delete Nodes And Return Forest (Medium)

Given the `root` of a binary tree, each node in the tree has a distinct value.

After deleting all nodes with a value in `to_delete`, we are left with a forest (a disjoint union of trees).

Return the roots of the trees in the remaining forest. You may return the result in any order.

**Example 1:**



Input: `root = [1,2,3,4,5,6,7]`, `to_delete = [3,5]`

Output: `[[1,2,null,4],[6],[7]]`

**Example 2:**

Input: `root = [1,2,4,null,3]`, `to_delete = [3]`

Output: `[[1,2,4]]`

### Constraints:

- The number of nodes in the given tree is at most 1000.
- Each node has a distinct value between 1 and 1000.
- `to_delete.length < 1000`=
- `to_delete` contains distinct values between 1 and 1000.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/delete-nodes-and-return-forest/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    vector<TreeNode*> delNodes(TreeNode* root, vector<int>& to_delete) {

    }
};

// @lc code=end

// Warning: this is a manual question, the generated test code may be incorrect.
int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    vector<int> to_delete;
    LeetCodeIO::scan(cin, to_delete);

    Solution *obj = new Solution();
    auto res = obj->delNodes(root, to_delete);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.168 1143. Longest Common Subsequence (Medium)

Given two strings `text1` and `text2`, return the length of their longest **common subsequence**. If there is no **common subsequence**, return 0.

A **subsequence** of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

- For example, "ace" is a subsequence of "abcde".

A **common subsequence** of two strings is a subsequence that is common to both strings.

### Example 1:

Input: `text1 = "abcde"`, `text2 = "ace"`

Output: 3

Explanation: The longest common subsequence is "ace" and its length is 3.

### Example 2:

Input: `text1 = "abc"`, `text2 = "abc"`

Output: 3

Explanation: The longest common subsequence is "abc" and its length is 3.

### Example 3:

Input: `text1 = "abc"`, `text2 = "def"`

Output: 0

Explanation: There is no such common subsequence, so the result is 0.

### Constraints:

- $1 < \text{text1.length}, \text{text2.length} \leq 1000$
- `text1` and `text2` consist of only lowercase English characters.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/longest-common-subsequence/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int longestCommonSubsequence(string text1, string text2) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string text1;
    LeetCodeIO::scan(cin, text1);
    string text2;
    LeetCodeIO::scan(cin, text2);

    Solution *obj = new Solution();
    auto res = obj->longestCommonSubsequence(text1, text2);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.169 1249. Minimum Remove to Make Valid Parentheses (Medium)

Given a string s of '(', ')' and lowercase English characters.

Your task is to remove the minimum number of parentheses ('(' or ')', in any positions) so that the resulting parentheses string is valid and return **any** valid string.

Formally, a parentheses string is valid if and only if:

- It is the empty string, contains only lowercase characters, or
- It can be written as AB ( A concatenated with B), where A and B are valid strings, or
- It can be written as (A), where A is a valid string.

### Example 1:

Input: s = "lee(t(c)o)de"

Output: "lee(t(c)o)de"

Explanation: "lee(t(co)de)" , "lee(t(c)ode)" would also be accepted.

### Example 2:

Input: s = "a)b(c)d"

Output: "ab(c)d"

### Example 3:

Input: s = ")())()

Output: ""

Explanation: An empty string is also valid.

### Constraints:

- $1 < \text{s.length} \leq 10^5$
- $\text{s}[i]$  is either '(', ')', or lowercase English letter.

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/minimum-remove-to-make-valid-parentheses/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    string minRemoveToMakeValid(string s) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    string s;
    LeetCodeIO::scan(cin, s);

    Solution *obj = new Solution();
    auto res = obj->minRemoveToMakeValid(s);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\u2022" << out_stream.rdbuf() << endl;

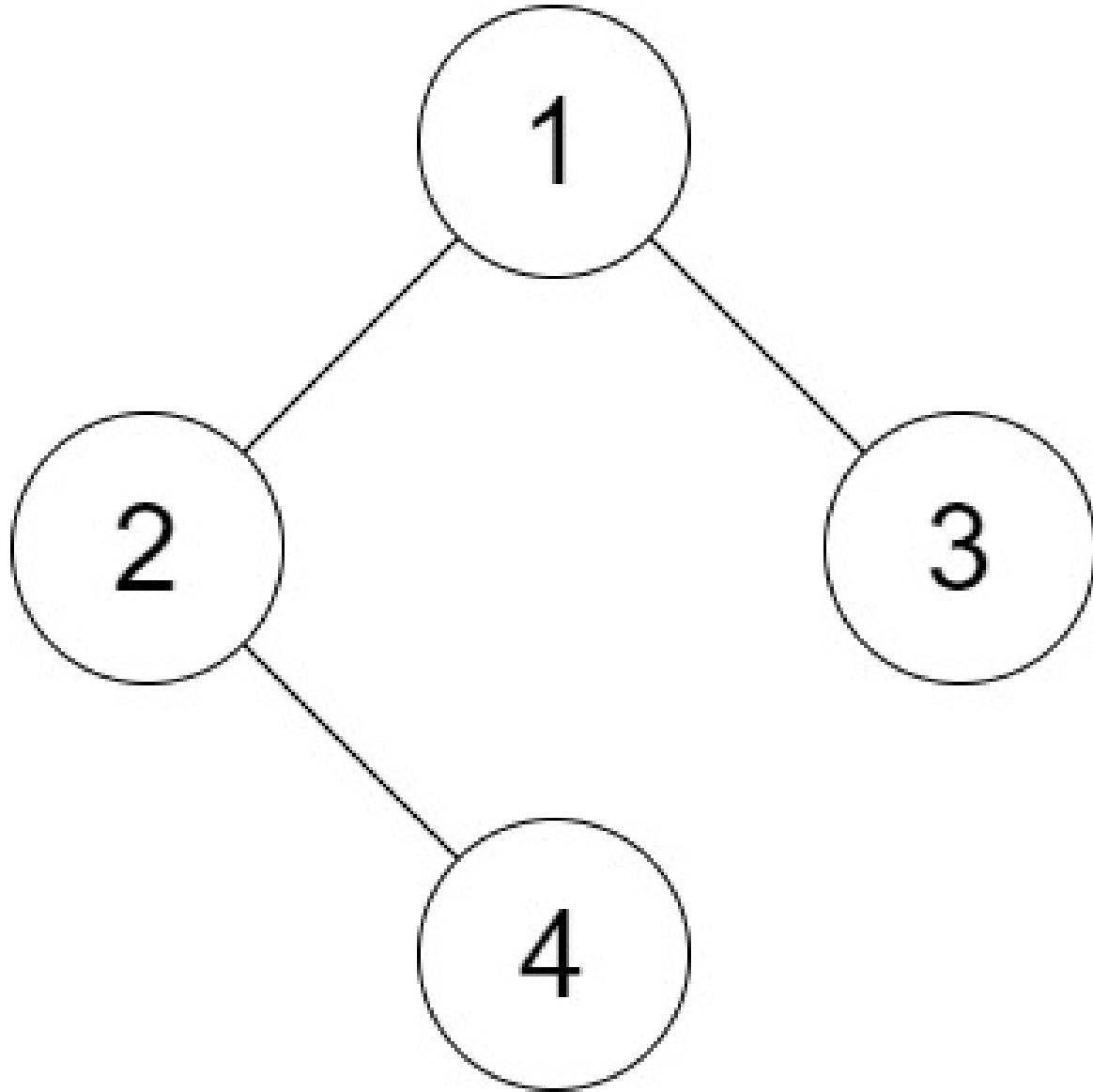
    delete obj;
    return 0;
}
```

## 8.170 1530. Number of Good Leaf Nodes Pairs (Medium)

You are given the root of a binary tree and an integer *distance*. A pair of two different **leaf** nodes of a binary tree is said to be good if the length of **the shortest path** between them is less than or equal to *distance*.

Return the number of good leaf node pairs in the tree.

**Example 1:**

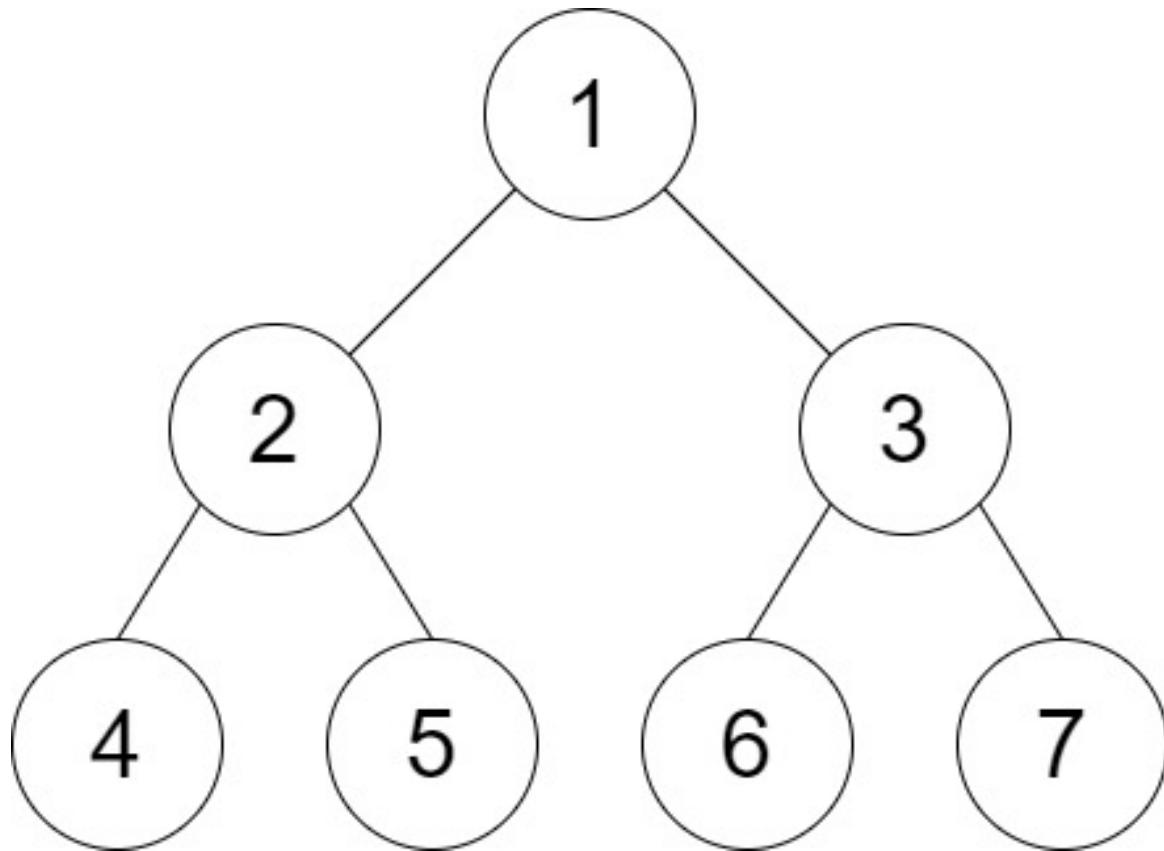


Input: root = [1,2,3,null,4], distance = 3

Output: 1

Explanation: The leaf nodes of the tree are 3 and 4 and the length of the shortest path between them is 3. This is the only good pair.

**Example 2:**



Input: root = [1,2,3,4,5,6,7], distance = 3

Output: 2

Explanation: The good pairs are [4,5] and [6,7] with shortest path = 2. The pair [4,6] is not good because the length of their shortest path between them is 4.

#### Example 3:

Input: root = [7,1,4,6,null,5,3,null,null,null,null,null,2], distance = 3

Output: 1

Explanation: The only good pair is [2,5].

#### Constraints:

- The number of nodes in the tree is in the range [1,  $2^1$  ].
- $1 < \text{Node.val} \leq 100$
- $1 < \text{distance} \leq 10$

題解如下：

```
// -*- compile-command: "make -f ../Makefile submit" -*-
// https://leetcode.com/problems/number-of-good-leaf-nodes-pairs/

#include <bits/stdc++.h>
#include "LC_IO.h"
using namespace std;

// @lc code=begin

class Solution {
public:
    int countPairs(TreeNode* root, int distance) {

    }
};

// @lc code=end

int main() {
    ios_base::sync_with_stdio(false);
    stringstream out_stream;

    TreeNode* root;
    LeetCodeIO::scan(cin, root);
    int distance;
    LeetCodeIO::scan(cin, distance);

    Solution *obj = new Solution();
    auto res = obj->countPairs(root, distance);
    LeetCodeIO::print(out_stream, res);
    cout << "\noutput:\n" << out_stream.rdbuf() << endl;

    delete obj;
    return 0;
}
```

## 8.171 设计刷题环境

### 8.171.1 leetgo

本来打算刷多门语言的，现在看了还是太高估自己了，把 cpp 刷好就可以了。

1. 准备 leetgo，使用 leetgo init，默认的模板就行，没多少需要修改的，如果需要多语言的可以改 outdir，cookie 不用自己找在浏览器里登录就行。
2. 准备 emacs

```
(defvar my/leetcode-root "~/leetcode/src/"
  "Root directory of leetgo-generated problems.")

(defun my/leetcode-format-number (n)
  "Return N formatted like 1 → \"0001\"."
  (format "%04d" n))

(defun my/leetcode--find-problem-dir (n)
  "Return directory path for problem number N."
  (let* ((prefix (my/leetcode-format-number n)))
    (dirs (directory-files my/leetcode-root t
                           (concat "^" prefix "\\.\\..")))
    (car dirs)))

(defun my/leetcode--pandoc-md-to-org (md-file org-file)
  "Convert MD-FILE → ORG-FILE using pandoc."
  (call-process "pandoc" nil nil nil md-file "-o" org-file "--wrap=none"))

(defun my/leetcode--download-image (url dest)
  "Download image URL to DEST file path."
  (url-copy-file url dest t))

(defun my/leetcode--process-org-images (org-file problem-dir)
  "Download remote images using wget and replace links with local filenames."
  (with-temp-buffer
    (insert-file-contents org-file)
    (goto-char (point-min)))

  ;; 匹配所有 http 图片链接
  (while (re-search-forward "\\\\[\\\[\\[https?://[^]]+\\.\\((png\\|jpg\\|jpeg\\|gif\\)\\)\\]\\]\\]\\]" nil t)
    (let* ((url (match-string 1))
           (filename (file-name-nondirectory url)) ;; 保留原始图片名
           (local-path (expand-file-name filename problem-dir)))

      ;; 如果图片不存在 -> 用 wget 下载
      (unless (file-exists-p local-path)
        (message "Downloading image via wget: %s" url)
        (call-process "wget" nil nil nil "-q" "-O" local-path url))

      ;; 替换 org 链接为相对路径
      (replace-regexp-in-string url (concat "#" (file-relative-name filename)))
```

```

(replace-match (format "[./%s]" filename) t t))

(write-region (point-min) (point-max) org-file nil 'quiet))

(defun my/leetcode-open (n)
  "Open Leetcode problem N. If not found, auto-fetch using `leetgo pick -l cpp N`."
  (interactive "nProblem_number:")
  (let* ((dir (my/leetcode--find-problem-dir n)))

    ;; If not found → auto call: leetgo pick -l cpp n
    (unless dir
      (message "Problem %d not found. Fetching via `leetgo pick -l cpp %d`..." n n)
      (let ((default-directory (expand-file-name "../" my/leetcode-root)))
        (call-process "leetgo" nil "*leetgo-pick*" t
                     "pick" "-l" "cpp" (number-to-string n)))
      ;; Re-scan after fetch
      (setq dir (my/leetcode--find-problem-dir n))
      (unless dir
        (error "After running `leetgo pick`, problem %d still not found. Check leetgo login." n)))

    ;; ---- If directory exists, continue normal code ----
    (let* ((md (expand-file-name "question.md" dir))
           (org (expand-file-name "question.org" dir))
           (cpp (expand-file-name "solution.cpp" dir)))

      ;; 1. Convert MD → ORG
      (my/leetcode--pandoc-md-to-org md org)

      ;; 2. Replace remote images → local images
      (my/leetcode--process-org-images org dir)

      ;; 3. Add include block to org
      (with-temp-buffer
        (insert-file-contents org)
        (goto-char (point-max))
        (insert "\n#+INCLUDE: ./solution.cpp\\src\\cpp\\n\\n\\newpage")
        (write-region (point-min) (point-max) org nil 'quiet))

      ;; 4. Add compile-command to solution.cpp
      (with-temp-buffer
        (insert-file-contents cpp)
        (goto-char (point-min))
        (unless (looking-at "//-*-compile-command:")
          (insert "//-*-compile-command: \"make -f ..\\Makefile\\submit\""))
        (write-region (point-min) (point-max) cpp nil 'quiet))

      ;; 5. Open org and cpp buffers
      (find-file org)
      (save-window-excursion

```

```

(find-file cpp))
(message "Loaded LeetCode %d from %s" n dir)))

(defun my/leetcode-generate-includes ()
"Generate #+INCLUDE lines for all LeetCode question.org files."
(let* ((root (expand-file-name "src" default-directory))
      (dirs (directory-files root t "^[0-9]+\\..+"))
      results)
  (dolist (d dirs (nreverse results))
    (let ((q (expand-file-name "question.org" d)))
      (when (file-exists-p q)
        (push (format "#+INCLUDE: \"%s\" :minlevel 1"
                     (file-relative-name q default-directory))
              results)))))

(defun my/leetcode-fetch-one-silent (n)
"Fetch problem N silently (no opening buffers), generating question.org and downloading images."
(let* ((dir (my/leetcode--find-problem-dir n)))

  ;; pick if missing
  (unless dir
    (message "Fetching %d via leetgo pick..." n)
    (let ((default-directory (expand-file-name "../" my/leetcode-root)))
      (call-process "leetgo" nil nil nil "pick" "-l" "cpp" (number-to-string n)))
    (setq dir (my/leetcode--find-problem-dir n)))

  (unless dir
    (message "Failed to fetch problem %d" n)
    (cl-return-from my/leetcode-fetch-one-silent nil))

  (let* ((md (expand-file-name "question.md" dir))
         (org (expand-file-name "question.org" dir))
         (cpp (expand-file-name "solution.cpp" dir)))

    ;; convert
    (my/leetcode--pandoc-md-to-org md org)
    (my/leetcode--process-org-images org dir)

    ;; add include to org
    (with-temp-buffer
      (insert-file-contents org)
      (goto-char (point-max))
      (insert "\n****_Solution\n#+INCLUDE: ./solution.cpp \"src cpp\\n\"")
      (write-region (point-min) (point-max) org nil 'quiet))

    ;; add compile-command
    (with-temp-buffer
      (insert-file-contents cpp)
      (goto-char (point-min)))

```

```

(unless (looking-at "//-*-compile-command:")
  (insert "//-*-compile-command:\\"make-f../Makefile_submit\"-*-\\n"))
  (write-region (point-min) (point-max) cpp nil 'quiet)))

(message "Fetched,%d OK" n))

(defun my/leetcode-fetch-batch (numbers)
  "Fetch multiple problems silently.
NUMBERS is a string like \"1,2,3,11,17,19\"."
  (interactive "sProblem numbers (e.g. \"1,2,3,11\"):")
  (let ((nums (mapcar #'string-to-number (split-string numbers))))
    (dolist (n nums)
      (ignore-errors
        (my/leetcode-fetch-one-silent n))))
  (message "Batch fetch done."))

(defun my/leetcode-update-includes ()
  "Find marker '#begin_of_leetcode_include', erase old includes, insert new ones.
#+INCLUDE: "src/0001.two-sum/question.org" :minlevel 2
#+INCLUDE: "src/0003.longest-substring-without-repeating-characters/question.org" :minlevel 2
#+INCLUDE: "src/0004.median-of-two-sorted-arrays/question.org" :minlevel 2
#+INCLUDE: "src/0005.longest-palindromic-substring/question.org" :minlevel 2
#+INCLUDE: "src/0010.regular-expression-matching/question.org" :minlevel 2
#+INCLUDE: "src/0015.3sum/question.org" :minlevel 2
#+INCLUDE: "src/0019.remove-nth-node-from-end-of-list/question.org" :minlevel 2
#+INCLUDE: "src/0020.valid-parentheses/question.org" :minlevel 2
#+INCLUDE: "src/0023.merge-k-sorted-lists/question.org" :minlevel 2
#+INCLUDE: "src/0028.find-the-index-of-the-first-occurrence-in-a-string/question.org" :minlevel 2
#+INCLUDE: "src/0034.find-first-and-last-position-of-element-in-sorted-array/question.org" :minlevel 2
#+INCLUDE: "src/0037.sudoku-solver/question.org" :minlevel 2
#+INCLUDE: "src/0040.combination-sum-ii/question.org" :minlevel 2
#+INCLUDE: "src/0046.permutations/question.org" :minlevel 2
#+INCLUDE: "src/0047.permutations-ii/question.org" :minlevel 2
#+INCLUDE: "src/0048.rotate-image/question.org" :minlevel 2
#+INCLUDE: "src/0051.n-queens/question.org" :minlevel 2
#+INCLUDE: "src/0053.maximum-subarray/question.org" :minlevel 2
#+INCLUDE: "src/0064.minimum-path-sum/question.org" :minlevel 2
#+INCLUDE: "src/0067.add-binary/question.org" :minlevel 2
#+INCLUDE: "src/0069.sqrtx/question.org" :minlevel 2
#+INCLUDE: "src/0070.climbing-stairs/question.org" :minlevel 2
#+INCLUDE: "src/0072.edit-distance/question.org" :minlevel 2
#+INCLUDE: "src/0075.sort-colors/question.org" :minlevel 2
#+INCLUDE: "src/0076.minimum-window-substring/question.org" :minlevel 2
#+INCLUDE: "src/0077.combinations/question.org" :minlevel 2
#+INCLUDE: "src/0079.word-search/question.org" :minlevel 2
#+INCLUDE: "src/0081.search-in-rotated-sorted-array-ii/question.org" :minlevel 2
#+INCLUDE: "src/0083.remove-duplicates-from-sorted-list/question.org" :minlevel 2
#+INCLUDE: "src/0088.merge-sorted-array/question.org" :minlevel 2
#+INCLUDE: "src/0091.decode-ways/question.org" :minlevel 2

```

```

#+INCLUDE: "src/0094.binary-tree-inorder-traversal/question.org" :minlevel 2
#+INCLUDE: "src/0099.recover-binary-search-tree/question.org" :minlevel 2
#+INCLUDE: "src/0101.symmetric-tree/question.org" :minlevel 2
#+INCLUDE: "src/0104.maximum-depth-of-binary-tree/question.org" :minlevel 2
#+INCLUDE: "src/0105.construct-binary-tree-from-preorder-and-inorder-traversal/question.org" :minlevel
  2
#+INCLUDE: "src/0106.construct-binary-tree-from-inorder-and-postorder-traversal/question.org" :
  minlevel 2
#+INCLUDE: "src/0109.convert-sorted-list-to-binary-search-tree/question.org" :minlevel 2
#+INCLUDE: "src/0110.balanced-binary-tree/question.org" :minlevel 2
#+INCLUDE: "src/0121.best-time-to-buy-and-sell-stock/question.org" :minlevel 2
#+INCLUDE: "src/0122.best-time-to-buy-and-sell-stock-ii/question.org" :minlevel 2
#+INCLUDE: "src/0126.word-ladder-ii/question.org" :minlevel 2
#+INCLUDE: "src/0128.longest-consecutive-sequence/question.org" :minlevel 2
#+INCLUDE: "src/0130.surrounded-regions/question.org" :minlevel 2
#+INCLUDE: "src/0135.candy/question.org" :minlevel 2
#+INCLUDE: "src/0136.single-number/question.org" :minlevel 2
#+INCLUDE: "src/0139.word-break/question.org" :minlevel 2
#+INCLUDE: "src/0142.linked-list-cycle-ii/question.org" :minlevel 2
#+INCLUDE: "src/0144.binary-tree-preorder-traversal/question.org" :minlevel 2
#+INCLUDE: "src/0145.binary-tree-postorder-traversal/question.org" :minlevel 2
#+INCLUDE: "src/0146.lru-cache/question.org" :minlevel 2
#+INCLUDE: "src/0148.sort-list/question.org" :minlevel 2
#+INCLUDE: "src/0149.max-points-on-a-line/question.org" :minlevel 2
#+INCLUDE: "src/0154.find-minimum-in-rotated-sorted-array-ii/question.org" :minlevel 2
#+INCLUDE: "src/0155.min-stack/question.org" :minlevel 2
#+INCLUDE: "src/0162.find-peak-element/question.org" :minlevel 2
#+INCLUDE: "src/0167.two-sum-ii-input-array-is-sorted/question.org" :minlevel 2
#+INCLUDE: "src/0168.excel-sheet-column-title/question.org" :minlevel 2
#+INCLUDE: "src/0169.majority-element/question.org" :minlevel 2
#+INCLUDE: "src/0188.best-time-to-buy-and-sell-stock-iv/question.org" :minlevel 2
#+INCLUDE: "src/0190.reverse-bits/question.org" :minlevel 2
#+INCLUDE: "src/0198.house-robber/question.org" :minlevel 2
#+INCLUDE: "src/0202.happy-number/question.org" :minlevel 2
#+INCLUDE: "src/0205.isomorphic-strings/question.org" :minlevel 2
#+INCLUDE: "src/0208.implement-trie-prefix-tree/question.org" :minlevel 2
#+INCLUDE: "src/0210.course-schedule-ii/question.org" :minlevel 2
#+INCLUDE: "src/0213.house-robber-ii/question.org" :minlevel 2
#+INCLUDE: "src/0215.kth-largest-element-in-an-array/question.org" :minlevel 2
#+INCLUDE: "src/0217.contains-duplicate/question.org" :minlevel 2
#+INCLUDE: "src/0218.the-skyline-problem/question.org" :minlevel 2
#+INCLUDE: "src/0221.maximal-square/question.org" :minlevel 2
#+INCLUDE: "src/0225.implement-stack-using-queues/question.org" :minlevel 2
#+INCLUDE: "src/0226.invert-binary-tree/question.org" :minlevel 2
#+INCLUDE: "src/0227.basic-calculator-ii/question.org" :minlevel 2
#+INCLUDE: "src/0232.implement-queue-using-stacks/question.org" :minlevel 2
#+INCLUDE: "src/0235.lowest-common-ancestor-of-a-binary-search-tree/question.org" :minlevel 2
#+INCLUDE: "src/0236.lowest-common-ancestor-of-a-binary-tree/question.org" :minlevel 2
#+INCLUDE: "src/0238.product-of-array-except-self/question.org" :minlevel 2

```

```

#+INCLUDE: "src/0239.sliding-window-maximum/question.org" :minlevel 2
#+INCLUDE: "src/0240.search-a-2d-matrix-ii/question.org" :minlevel 2
#+INCLUDE: "src/0241.different-ways-to-add-parentheses/question.org" :minlevel 2
#+INCLUDE: "src/0242.valid-anagram/question.org" :minlevel 2
#+INCLUDE: "src/0257.binary-tree-paths/question.org" :minlevel 2
#+INCLUDE: "src/0260.single-number-iii/question.org" :minlevel 2
#+INCLUDE: "src/0268.missing-number/question.org" :minlevel 2
#+INCLUDE: "src/0279.perfect-squares/question.org" :minlevel 2
#+INCLUDE: "src/0287.find-the-duplicate-number/question.org" :minlevel 2
#+INCLUDE: "src/0300.longest-increasing-subsequence/question.org" :minlevel 2
#+INCLUDE: "src/0303.range-sum-query-immutable/question.org" :minlevel 2
#+INCLUDE: "src/0304.range-sum-query-2d-immutable/question.org" :minlevel 2
#+INCLUDE: "src/0307.range-sum-query-mutable/question.org" :minlevel 2
#+INCLUDE: "src/0309.best-time-to-buy-and-sell-stock-with-cooldown/question.org" :minlevel 2
#+INCLUDE: "src/0310.minimum-height-trees/question.org" :minlevel 2
#+INCLUDE: "src/0312.burst-balloons/question.org" :minlevel 2
#+INCLUDE: "src/0313.super-ugly-number/question.org" :minlevel 2
#+INCLUDE: "src/0318.maximum-product-of-word-lengths/question.org" :minlevel 2
#+INCLUDE: "src/0322.coin-change/question.org" :minlevel 2
#+INCLUDE: "src/0328.odd-even-linked-list/question.org" :minlevel 2
#+INCLUDE: "src/0332.reconstruct-itinerary/question.org" :minlevel 2
#+INCLUDE: "src/0338.counting-bits/question.org" :minlevel 2
#+INCLUDE: "src/0343.integer-break/question.org" :minlevel 2
#+INCLUDE: "src/0347.top-k-frequent-elements/question.org" :minlevel 2
#+INCLUDE: "src/0376.wiggle-subsequence/question.org" :minlevel 2
#+INCLUDE: "src/0377.combination-sum-iv/question.org" :minlevel 2
#+INCLUDE: "src/0380.insert-delete-getrandom-o1/question.org" :minlevel 2
#+INCLUDE: "src/0404.sum-of-left-leaves/question.org" :minlevel 2
#+INCLUDE: "src/0406.queue-reconstruction-by-height/question.org" :minlevel 2
#+INCLUDE: "src/0409.longest-palindrome/question.org" :minlevel 2
#+INCLUDE: "src/0413.arithmetic-slices/question.org" :minlevel 2
#+INCLUDE: "src/0416.partition-equal-subset-sum/question.org" :minlevel 2
#+INCLUDE: "src/0417.pacific-atlantic-water-flow/question.org" :minlevel 2
#+INCLUDE: "src/0432.all-oone-data-structure/question.org" :minlevel 2
#+INCLUDE: "src/0435.non-overlapping-intervals/question.org" :minlevel 2
#+INCLUDE: "src/0437.path-sum-iii/question.org" :minlevel 2
#+INCLUDE: "src/0448.find-all-numbers-disappeared-in-an-array/question.org" :minlevel 2
#+INCLUDE: "src/0450.delete-node-in-a-bst/question.org" :minlevel 2
#+INCLUDE: "src/0451.sort-characters-by-frequency/question.org" :minlevel 2
#+INCLUDE: "src/0452.minimum-number-of-arrows-to-burst-balloons/question.org" :minlevel 2
#+INCLUDE: "src/0455.assign-cookies/question.org" :minlevel 2
#+INCLUDE: "src/0461.hamming-distance/question.org" :minlevel 2
#+INCLUDE: "src/0462.minimum-moves-to-equal-array-elements-ii/question.org" :minlevel 2
#+INCLUDE: "src/0474.ones-and-zeroes/question.org" :minlevel 2
#+INCLUDE: "src/0476.number-complement/question.org" :minlevel 2
#+INCLUDE: "src/0494.target-sum/question.org" :minlevel 2
#+INCLUDE: "src/0503.next-greater-element-ii/question.org" :minlevel 2
#+INCLUDE: "src/0513.find-bottom-left-tree-value/question.org" :minlevel 2
#+INCLUDE: "src/0530.minimum-absolute-difference-in-bst/question.org" :minlevel 2

```

```

#+INCLUDE: "src/0538.convert-bst-to-greater-tree/question.org" :minlevel 2
#+INCLUDE: "src/0540.single-element-in-a-sorted-array/question.org" :minlevel 2
#+INCLUDE: "src/0542.01-matrix/question.org" :minlevel 2
#+INCLUDE: "src/0543.diameter-of-binary-tree/question.org" :minlevel 2
#+INCLUDE: "src/0547.number-of-provinces/question.org" :minlevel 2
#+INCLUDE: "src/0566.reshape-the-matrix/question.org" :minlevel 2
#+INCLUDE: "src/0572.subtree-of-another-tree/question.org" :minlevel 2
#+INCLUDE: "src/0583.delete-operation-for-two-strings/question.org" :minlevel 2
#+INCLUDE: "src/0594.longest-harmonious-subsequence/question.org" :minlevel 2
#+INCLUDE: "src/0605.can-place-flowers/question.org" :minlevel 2
#+INCLUDE: "src/0617.merge-two-binary-trees/question.org" :minlevel 2
#+INCLUDE: "src/0633.sum-of-square-numbers/question.org" :minlevel 2
#+INCLUDE: "src/0637.average-of-levels-in-binary-tree/question.org" :minlevel 2
#+INCLUDE: "src/0646.maximum-length-of-pair-chain/question.org" :minlevel 2
#+INCLUDE: "src/0647.palindromic-substrings/question.org" :minlevel 2
#+INCLUDE: "src/0650.2-keys-keyboard/question.org" :minlevel 2
#+INCLUDE: "src/0653.two-sum-iv-input-is-a-bst/question.org" :minlevel 2
#+INCLUDE: "src/0665.non-decreasing-array/question.org" :minlevel 2
#+INCLUDE: "src/0669.trim-a-binary-search-tree/question.org" :minlevel 2
#+INCLUDE: "src/0680.valid-palindrome-ii/question.org" :minlevel 2
#+INCLUDE: "src/0684.redundant-connection/question.org" :minlevel 2
#+INCLUDE: "src/0693.binary-number-with-alternating-bits/question.org" :minlevel 2
#+INCLUDE: "src/0695.max-area-of-island/question.org" :minlevel 2
#+INCLUDE: "src/0696.count-binary-substrings/question.org" :minlevel 2
#+INCLUDE: "src/0697.degree-of-an-array/question.org" :minlevel 2
#+INCLUDE: "src/0714.best-time-to-buy-and-sell-stock-with-transaction-fee/question.org" :minlevel 2
#+INCLUDE: "src/0739.daily-temperatures/question.org" :minlevel 2
#+INCLUDE: "src/0763.partition-labels/question.org" :minlevel 2
#+INCLUDE: "src/0769.max-chunks-to-make-sorted/question.org" :minlevel 2
#+INCLUDE: "src/0785.is-graph-bipartite/question.org" :minlevel 2
#+INCLUDE: "src/0870.advantage-shuffle/question.org" :minlevel 2
#+INCLUDE: "src/0882.reachable-nodes-in-subdivided-graph/question.org" :minlevel 2
#+INCLUDE: "src/0889.construct-binary-tree-from-preorder-and-postorder-traversal/question.org" :
  minlevel 2
#+INCLUDE: "src/0897.increasing-order-search-tree/question.org" :minlevel 2
#+INCLUDE: "src/0932.beautiful-array/question.org" :minlevel 2
#+INCLUDE: "src/0934.shortest-bridge/question.org" :minlevel 2
#+INCLUDE: "src/1091.shortest-path-in-binary-matrix/question.org" :minlevel 2
#+INCLUDE: "src/1105.filling-bookcase-shelves/question.org" :minlevel 2
#+INCLUDE: "src/1110.delete-nodes-and-return-forest/question.org" :minlevel 2
#+INCLUDE: "src/1143.longest-common-subsequence/question.org" :minlevel 2
#+INCLUDE: "src/1249.minimum-remove-to-make-valid-parentheses/question.org" :minlevel 2
#+INCLUDE: "src/1530.number-of-good-leaf-nodes-pairs/question.org" :minlevel 2
(interactive)
(save-excursion
  (goto-char (point-min))
  ;; 找到标记行
  (if (search-forward "#_begin_of_leetcode_include" nil t)
    (progn
      
```

```

;; 到下一行 (开始删除旧的内容)
(forward-line 1)

;; 记录起点
(let ((start (point)))

;; 删除所有旧的 #+INCLUDE 行
(while (looking-at "^#\\"+INCLUDE:")
  (forward-line 1))
(delete-region start (point))

;; 生成新的 include 内容
(let* ((root (expand-file-name "src" default-directory))
       (dirs (directory-files root t "[0-9]+\\"..+"))
       (dolist (d dirs)
         (let ((q (expand-file-name "question.org" d)))
           (when (file-exists-p q)
             (insert
              (format "#+INCLUDE: \"%s\" :minlevel 2\n"
                     (file-relative-name q default-directory)))))))
  (message "Marker #'begin of leetcode include' not found!")))

```

## 8.171.2 本地刷题测试

由于本人的网络不是 24 小时都有，所以有一个能本地刷题的方式就好了，用测试例测好，明天再提交到网页也不失为一种方法。

```

BEGIN { testcase = 0; input_lines = ""; expected = ""; fail = 0 }

/^input:/ {
  testcase++;
  input_lines = "";
  expected = "";
  next
}

/^output:/ {
  getline expected;

  tmp = "/tmp/input_" testcase ".txt";
  print input_lines > tmp;
  close(tmp);

  cmd = prog "<" tmp;
  result = "";
  while ((cmd | getline line) > 0) {
    if (result != "") result = result "\n";
    result = result line;
  }
}

```

```

close(cmd);

sub(/^output:[ \t]*/, "", result);
sub(/[\t\r\n]+$/, "", expected);

if (result == expected) {
    print "Testcase" testcase "PASSED";
} else {
    print "Testcase" testcase "FAILED";
    print "Input:" input_lines;
    print "Expected:" expected;
    print "Got:" result;
    fail = 1 # 标记失败
}
next
}

{
if (input_lines == "") {
    input_lines = $0;
} else {
    input_lines = input_lines "\n" $0;
}

END {
    if (fail) exit 1
}

```

**Listing 8.1:** 利用本地测试 testcase.txt，先过本地才能上传网页

```

# Current directory (e.g. 0001.two-sum)
CURDIR_NAME := $(notdir $(CURDIR))
# Extract part after first dot, e.g. "0001.two-sum" -> "two-sum"
SPLIT_DIRNAME := $(word 2, $(subst ., ,$(CURDIR_NAME)))

a.out: solution.cpp
    $(CXX) solution.cpp -I.. -O2 -std=c++17 -Wall -o a.out

submit: testc
    leetgo submit $(SPLIT_DIRNAME) -l cpp

testc: a.out
    @awk -v prog="./a.out" -f ../../target/debug/$(SPLIT_DIRNAME) -f ../../tester.awk testcases.txt

# Default target
testr:
    cargo build --bin $(SPLIT_DIRNAME)
    awk -v prog="../../target/debug/$(SPLIT_DIRNAME)" -f ../../tester.awk testcases.txt

```

**Listing 8.2:** 用 makefile 测试，makefile 知道是从那一个 subdir 传了的 make 指令

```
[-] Leetcode
|[-] src
||-[+] 0001.two-sum
||-[+] 0002.add-two-numbers
||-[+] 0004.median-of-two-sorted-arrays
||-[+] 0005.longest-palindromic-substring
||-[+] 0006.zigzag-conversion
||-[+] bits
||— LCIO.h
| ‘— Makefile
|-[+] target
|—— Cargo.lock
|—— Cargo.toml
|—— README.org
|—— hh.elc
|—— leetgo.yaml
‘—— tester.awk
```

# 第九章 版本更新历史

---

2025-11-25 更新: 晴

① **走路:** 晚上和母亲一起在江边走路，她抱怨太远了，脚得走痛了，但还是我陪走完了一圈。我想我是很幸福的，有这样的母亲，虽然嘴碎但还是能陪伴我一行，在冷清的江边，只有两三个跑者与我俩，冬天好像萧条不少。我抡或背着二锤，吃了两个别人落下的柑子，敲敲死树、木桩，在草坪上练练投掷，作壶铃溜两下，中午还曾晒过太阳，晚上吃了红烧鱼，一天就这样，简单。简单也没什么不好，看着那些黄毛和抽烟的女友在摩托上追江风，也许他们的母亲就没有这样的耐心了，当然也许他们自己也无法耐受半小时的无聊，母亲不爱自己放荡的儿子，儿子不耐自己老气的母亲。吹着江风，默默无语的草和人在行走着，灯光亮着一路，一切干净又舒适，又有什么无聊呢？

② **Github:** 逛 Github 其实是最浪费时间的事，因为 github 就是伪装成编程平台的社交媒体。

为 feature 点赞 为爱豆的新推点赞

为语言传教 为爱豆打 call

awesome-xx 爱豆图集合订本

上传 toy project 自己为爱豆做的周边

争论语言/编辑器/操作系统/etc 争论韩流/日流/欧美圈/内娱…

新工具新框架 网红

学习分享 卖课网红

开源学习分享 盗版卖课网红

discuss/issue 粉丝群群管理答疑

怎么不把 github 当成社交媒体？

什么是社交媒体 交互性/排名性/圈子化/免费/易得…

- 不交互

不点赞 作个局外人，它好它坏不必说。

不乱传项目 少乱写代码上传 github，少立 flag。

把 author 看作工具人而不是教主 当然有问题还是得说 thanks。

- 不排名

不看 star/awesome/trend/weekly github 项目大多只是 blog。

绝不看中文 github 中文自带传销属性。

- 不易得

用 consult-gh/forge/eww 一分钟加载一个页面…

- 不圈子

工具论 一切只是工具，工具只是乐趣，乐趣不是越多越好。

---

2025-11-23 更新: 阴

① **leetcode:** 把题都抓完了，这还不开刷开学，想后半生吃草？

② **配 emacs:** 翻了半天别人的配置，就找两包 git-link, git-timemachine 感觉不错。

---

2025-11-22 更新: 晴

① **C 语言:** 对 C 语言的复习还没开始。妈的，互联网

② **leetcode:** 完成了 0 题。妈的，互联网

③ **又后悔了:** 本地有 chromium, hosts06 忘了加，想用 chromium 看一下翻墙网站的兼容性，莫名发现谷歌搜索能用，又看起片来！

- 
- ④ **从今天起**: 能只用 emacs 就只用 emacs, 能只用 eww 就只用 eww, 能用 gptel 就只用 gptel。eww 不能访问也别去试 firefox, firefox 不能访问也别去试 chromium。
- 

2025-11-21 更新: 晴

- ① **域名**: 为 columnddeeply/hosts 提交了 17868 行新域名, 这个 hosts 是我最近在网上看到的最大的 host 文件, 达到了近 410mb, 共 12.576.671 行, 专门用于 Porn, 今天比较闲就把之前使用 tempermonkey 写的自动根据关键词用 ublacklist 自动 block 和从 fackads/和 github u3m8 collection 提出来的 cdn 网站和以前又闲又愤青时期屏蔽的各种网站, 进行比较有二万三千行的不同 host, 用后缀比如 gov/org/cn 过滤了一些自己动手又删了些, 发现像很多云很多热门网站像 cloud.tencent.com./taobao/nvidia/xx.gov.cn 但用了这么久好像没有任何冲突, 可能是我实在是没有怎么上网, 只是用用 github 就够了。看起来很多, 17868 行, 几天检查不完, 但是也就只是它的 703 分之一, 我看了下很多是 blogspot/tumblr, 互联网的域名真得是故意弄成这样难管? 如果说 porn 的域名长成这样: 028b2c9ad2a24433ab97b8e5dbf69597.mediatailor.us-east-1.amazonaws.com(这还真是 porn), 这能管? 设立 whitelist 比 blacklist 更容易, 但网络的架构 (tcp/ip-dns-domain) 就不允许这种事, 我也好久没有更新我的 host 了, 用 ai 比用搜索引擎好用多了, ai 的用处就是过滤黄色网站, 当然 openai 要开放成人内容这事说明: 其实也没什么一定得搜的事。用 emacs 看文档, 把记忆留在自己的脑子里, 比起什么网页要重要的多。
- ② **折腾 emacs**: iedit/embark-next-symbol/forge。
- ③ **弄完了 leetgo**: 明天绝对开始刷题。
- ④ **研究了 github 热门 trending**: 发现中文代码质量之差, 全是卖课/卖舆论分析/卖盗版书 pdf/卖前端 ai 的… fork 外国大神后各种引流的, ruanyifeng 周刊是可以的, 但评论和想让他在 issue 里引流的项目也一言难尽的, 看看 nixpkg 几千人才 22k 星, 中文的一个 openwrt 教程就有 30k 星几百个人 contribute, 看看 openai 也才几十 k 星, deepseek 上百 k 的星, 看看各自的 issues, openai 很正常, deepseek 就好似精神病的故乡, 冷清又时不时出来几个弱智问题, 外国人看了怕是会笑掉大牙。
- 

2025-11-20 更新: 晴

- ① **leetcode**: 完成对 leetgo 工具的 elisp 改造。chatgpt 很给力。明天一定刷 leetcode。
- ② **real-mono-themes**: 今日 maintiner 还没有动, 看来工作很忙啊。
- ③ **博客/pdf**: 加入了新的 org snippet, 更深入学习了 org 转入 latex 更多方法, 更完美的使用了本 latex 模板。
- 

2025-11-19 更新: 晴

- ① **天气好, 玩了一下午**: 结果踩在沥水槽腿掉下去了, 还好只是擦伤。
- ② **leetcode**: 让 chatgpt 写 leetgo 的 elisp 工具, 写到最后凌晨没网了!
- ③ **后悔的一天**: reddit 里的 china\irl 我常常去看, 实在是没意思, 但总有一种引力让我去刷, 莫名的又想看片, 于是乎又成了看片加键政, 什么高市什么献忠, 吃瓜吃到 10 年的兽兽们, 再看现在的其他事件, 感觉 10 年代清新多了, 那时候的主角还是 87 年“小妹”, 现在则是 0 几年的“小妹”, 真是世风日下, 历史重是重复又重复, 技术重是加强又如强啊, 献忠机器人, 何时到来? 假韩炳哲的话改下: “互联网让道德世风内涵成了空气, 一切除了炫/爽/性没有他用”。
- ④ **折腾 nixos/emacs**: 精简所有文件到博客这一个仓库。

nixos 只用一个 flake.nix 和 secrets.yaml 再把一串 key 放到一个位置就能用了, 简直是极简完了。emacs 只用一个 post-init.init, 再 git clone 下 minimal-emacs 到位, org 再 src 运行一下, 就能用了。这两都是 2000 行的配置。总之我有了一个 blog/pdf/emacs/nixos/snippet/hosts 集于一体的仓库。完全可重现, 从配置到经历, 其它的代码也可以从其它 repo 里 clone 下来, 但总之目前这些就是我大学大概折腾的所有玩具了吧。越说越感自己的脑残了,

明明有女生喜欢我, 我只有折腾这些 sb 玩意去了! 现在想起了, 后悔得我都成了反科技主义了。

- **越说越后悔**: 总之, 刷题刷项目, 早点找工作, 早点重新联系她。别再做 reddit 上的支人了…别再看什么瓜片了…反技术的后果就是“深山老林里的怪人”!!! 第一, 你还有家人要养, 第二, 你也没生存能力, 第

三，我还是渴望家庭，第四，你都花了二十多年在技术上，现在去深山，那你学的拼音/加法/英文/编程都是屁吗？

2025-11-18 更新：阴

- ① 沉迷看书<sup>1</sup> 错过了我设定的上网时间二点到五点<sup>2</sup>…明天再测试吧<sup>3</sup>
  - ② C 语言: 刷一遍 flag 代码, 发现自己还是不太懂 C 语言。今天又没刷力扣, 感觉自己很废, 就这样还想搞什么?

2025/11/17 更新：雨

- ① **开始刷 Leetcode:** 完成 0 题，折腾 leetgo 去了，使用 async-shell-command 与 emacs 不全好用，两者的时间不一致问题，shell-command 又卡 emacs 本身，两者同样不能管理弹出的 shell 输出。只能用完整的 shell 命令和在 async 里 eval emacs function 套娃看看。
  - ② **将本“博客”的 html 和 pdf 上传了 github.io:** 只是不只为何几十分钟了 qingsongliao.github.io 还没有，明天再看吧。
  - ③ **完成了 real-mono-themes 的 emac 主题包:** 从此 emacs 又多了个别具一格的主题，不过也许 70 年代的 emacs 就长这样吧，oldfasion never die。

图 9.1: real-mono-old

2025/11/16 更新：阴

- **开始写作**
  - **使用 elegant 模板:** 搭建自己博客？写书还是写博客？干脆一起写吧。

<sup>1</sup>把什么现代诗选和中国皇帝传都丢了，再看下去这辈子就毁了！只留下两本书，互联网浅薄与雅思7天词汇，一是提醒我互联网对大脑的“伤害”另一个是提升下我的大脑，最近几年或十年特别是高考后，感觉自己的脑子雾雾的。

<sup>2</sup>我有点网瘾，所以通过家里的路由器限制一下。速度200kps，限制热门社交媒体的域名，时间上只有下午二点到五点能用。每次无限制上网都有一种沉迷的感觉，看黄片刷新闻作为瘾症生意在互联网真是完美载体，总之，戒断的痛苦真是难受啊!!!

<sup>3</sup>晚上又用母亲的手机上网了，怎么试都发现这个github.io是404，下载下来好像是html本身的问题。切换了一下账号发现看不了那个号了，问了下chatgpt发现github有新号防bot功能，没法只能重新用回我的NestorLiao账号[https://github.com/orgs/community/discussions/55609?utm\\_source=chatgpt.com](https://github.com/orgs/community/discussions/55609?utm_source=chatgpt.com)。

## 参考文献

- [1] LI Q, CHEN L, ZENG Y. The Mechanism and Effectiveness of Credit Scoring of P2P Lending Platform: Evidence from Renrendai.com. *China Finance Review International*, 2018, 8(3): 256-274.
- [2] CARLSTROM C T FUERST T S. Agency Costs, Net Worth, and Business Fluctuations: A Computable General Equilibrium Analysis. *The American Economic Review*, 1997: 893-910.
- [3] QUADRINI V. Financial Frictions in Macroeconomic Fluctuations. *FRB Richmond Economic Quarterly*, 2011, 97(3): 209-254.
- [4] 方军雄. 所有制、制度环境与信贷资金配置. *经济研究*, 2007(12): 82-92.
- [5] 刘凤良, 章潇萌, 于泽. 高投资、结构失衡与价格指数二元分化. *金融研究*, 2017(02): 54-69.
- [6] 吕捷 王高望. CPI 与 PPI “背离” 的结构性解释. *经济研究*, 2015, 50(04): 136-149.
- [7] Ted Kazynski. 工业革命与人类命运. *社会学研究*, 2017(02): 54-69.

# 附录 A 开发中遇见的各种软件问题

## A.0.1 General Contributions

内容提要
<ul style="list-style-type: none"><li>□ org-mode, 提交 pr 增加 INCLUDE 的行末倒数行数的行为代码 x, 加入 mailing-list 提出 FR</li><li>□ melpa, 提交 pr 增加自己的 leetgo package</li><li>☒ consult-gh, 提交 pr 修改 readme 中的 consult-gh-search-code 的使用</li><li>☒ melpa, 提交 pr 增加自己的 real-mono-themes</li><li>☒ nixpkg, 提交 issue 增加 biospy 等 nix 生理信号包</li><li>☒ columndeeplly/hosts, 提交 pr 增加 17868 行新域名, 修改脚本为 0.0.0.0</li><li>☒ pyim, 提交 issue 修改% 报错行为</li><li>☒ addons.mozilla, eink 插件</li></ul>

## A.0.2 TODO Advent of code in zig?

## A.0.3 TODO package leego elisp package as emacs package

- description: do leetcode in emacs with leetgo
- require: wget/pandoc/leetgo/
- wget for fetch image
- pandoc for markdown to org
- leetgo for pick/commit questions
- simple init, use leetgo
- batch fetch, (leetcode-fetch-batch "1 2 3") to fetch 1/2/3 questions
- also you can fetch contest,
- auto detected browser cookie, no need to fill cookies by yourself
- offline test, use pre fetched testcase/makefile/awk to local test tasks.
- org format, local image, and generally you can easily export all questions.
- wait I finished doing leetcode

## A.0.4 FIXME How to modify org-mode's #+INCLUDE:

1. how to contribute to org <https://orgmode.org/worg/org-contribute.html> <https://orgmode.org/worg/org-mailing-list.html>

2. explain FR Subject: [FR] Add support for reverse (negative) line ranges in '#+INCLUDE:' ':lines' Hi all,

Org's current '#+INCLUDE:' keyword supports selecting lines using the ':lines' parameter, e.g.:

- '5-10' —lines 5 to 10
- '-10' —lines 1 to 10
- '10-' —line 10 to EOF

However, Org currently does **not** support selecting lines counted from the end of the file.

I would like to propose extending the syntax to support “reverse line numbers”, similar to how many tools allow negative indexing. Examples:

```
#+INCLUDE: "x.txt" :lines "~5-" ; last 5 lines to EOF
#+INCLUDE: "x.txt" :lines "~5-~1" ; last 5 lines to last 1 line
```

---

```
#+INCLUDE: "x.txt" :lines "-~5"      ; from beginning to last 5 lines
#+INCLUDE: "x.txt" :lines "10-~1"     ; from line 10 to last 1 line
```

This would make it much easier to include “the last N lines” of logs, datasets, generated files, etc., without needing to preprocess the file externally.

If there is interest, I would be happy to help test or contribute patches.

Thanks! —Qingsong

## A.0.5 DONE how many hours of pc idle’s power consumption is equal to hibernate save ram to disk?

## A.0.6 DONE almost same vscode-emacs package.

## A.0.7 DONE how to run current region’s content (if not in region, run that line) into current project’s vterm

current I get a command called “multi-vterm-project”, when you are not in vterm and vterm is closed use multi-vterm-project to open(or turn on a vterm project shell window) it, and can use it again to toggle it close(not kill, just remove that window), if the project vterm winodw is open and cursor is not in that window, use multi-vterm-project will focus to vterm window. based on this, I want you to write run the current region/line’s content into vterm, how it works?

Below is the exact design that matches your workflow **with ‘multi-vterm-project‘ already handling window creation/toggle/focus**. You only need a function that:

1. Ensures the project vterm exists (by calling ‘multi-vterm-project‘ when needed).
2. Gets the project vterm buffer.
3. Sends region or current line.
4. Does not change your current window unless the vterm does not yet exist.
5. If the vterm window is already visible but not focused, **do not move focus**; only send text.

Assumption: ‘multi-vterm-project‘ creates or returns a vterm buffer named like “\*vterm-project: <project>\*“. If your naming differs, adjust the regexp.

Code:

```
““elisp (require ‘project) (require ‘vterm) (require ‘multi-vterm)
(defun my/get-project-vterm-buffer () ”Return the project vterm buffer, or nil if not exists.” (let* ((proj (project-current
t)) (root (project-root proj)) (name (file-name-nondirectory (directory-file-name root)))) (catch ‘found (dolist (buf
(buffer-list)) (when (string-match (format ”\vterm-project: %s” name) (buffer-name buf)) (throw ‘found buf)))) nil)))
  (defun my/ensure-project-vterm () ”Ensure project vterm exists. Do NOT focus it if already exists.” (let ((buf
(my/get-project-vterm-buffer))) (unless buf ;; Creates and shows it. Your toggle logic is inside multi-vterm-project.
(multi-vterm-project) (setq buf (my/get-project-vterm-buffer))) buf))
  (defun my/send-region-or-line-to-project-vterm () ”Send active region or current line to project vterm. Does not move
focus unless no vterm existed.” (interactive) (let* ((text (if (use-region-p) (buffer-substring-no-properties
(region-beginning) (region-end)) (buffer-substring-no-properties (line-beginning-position) (line-end-position)))))) ;; Make
sure vterm exists (let ((vbuf (my/ensure-project-vterm))) (when vbuf (with-current-buffer vbuf (vterm-send-string text)
(vterm-send-return))))))
  (global-set-key (kbd ”C-c e”) #'my/send-region-or-line-to-project-vterm) ““
```

Operation summary:

If project vterm **does not exist**, this creates it by calling ‘multi-vterm-project’. If project vterm **exists but hidden**, the function does not toggle it; it simply sends text silently. If project vterm **exists and is visible**, the function does not steal

---

focus; it sends text in place. Region has priority. No region → send current line.

This integrates cleanly with your workflow where ‘multi-vterm-project‘ handles all window toggle/focus behavior.

### A.0.8 DONE pyim % 字符触发 Not enough arguments for format string

使用虎码码表<https://github.com/welandx/huma-danzi.pyim> readme 发现输入1时报错: Debugger entered–Lisp error: (error "Not enough arguments for format string") message("[1]: 1. 而 2.% (1/1) \$ ") #f(compiled-function (string position tooltip)" 使用 minibuffer 来显示 STRING。" #<bytecode -0x26f1e7f81176ed3>("[1]: 1. 而 2.% (1/1) \$ " 620 minibuffer) apply(#f(compiled-function (string position tooltip)" 使用 minibuffer 来显示 STRING。" #<bytecode -0x26f1e7f81176ed3> "[1]: 1. 而 2.% (1/1) \$ " 620 minibuffer nil) pyim-page-show("[1]: 1. 而 2.% (1/1) \$ " 620 minibuffer) pyim-page-refresh(nil) pyim-process-ui-refresh() pyim-process-run() pyim-self-insert-command() funcall-interactively(pyim-self-insert-command) call-interactively(pyim-self-insert-command) pyim-process-input-method(108) pyim-input-method(108)

删除 hmdz.pyim 中的% 或使用中文的% 使用时不会报错, 发现 pyim 有% 就报错: Debugger entered–Lisp error: (error "Not enough arguments for format string") message("[1]: 1.% (1/1) \$ ") #f(compiled-function (string position tooltip)" 使用 minibuffer 来显示 STRING。" #<bytecode 0x1d8cdaf8ba1c112a>("[1]: 1.% (1/1) \$ " 624 minibuffer) apply(#f(compiled-function (string position tooltip)" 使用 minibuffer 来显示 STRING。" #<bytecode 0x1d8cdaf8ba1c112a> "[1]: 1.% (1/1) \$ " 624 minibuffer nil) pyim-page-show("[1]: 1.% (1/1) \$ " 624 minibuffer) pyim-page-refresh(nil) pyim-process-ui-refresh() pyim-process-run() pyim-self-insert-command() funcall-interactively(pyim-self-insert-command) call-interactively(pyim-self-insert-command) pyim-process-input-method(108) pyim-input-method(108)

### A.0.9 DONE explain this wget command

wget -r -p -np -k <url> 含义:

-r 启用递归下载。会下载 <url> 页面以及它引用的其他页面或资源 (根据递归规则)。

-p 下载页面显示所需的所有资源。包括图片、CSS、JS、字体等。这是“下载完整页面以便离线浏览”的选项。

-np 不进入父目录 (no parent)。递归下载时不会爬到 <url> 的上级目录。例如: 如果 <url> 是

<http://example.com/a/b/page.html> 它不会下载 <http://example.com/a/> 或 <http://example.com/> 下的内容。

-k 转换下载后的链接为本地相对路径。方便本地离线浏览时页面中的链接仍然能点击访问

### A.0.10 DONE how to do emacs overlay in nixos to config not to have somepackage natively exclude

how to disable gomoku, I am getting addiction to that game...

### A.0.11 DONE using consult-gh-notifications, and I just type wrong pass as my own pc's pass, not github token

it used to pop a dialog box at first time, then I input my linux user's password, then it just disappered. why? I want to reinput, but it just no dialog anymore and it just show me: ghub-handle-response-error: HTTP Error: 403, "Forbidden", "<https://api.github.com/graphql>", ((message . "Request forbidden by administrative rules. Please make sure your request has a User-Agent header (<https://docs.github.com/en/rest/overview/resources-in-the-rest-api#user-agent-required>).

Check <https://developer.github.com> for other possible causes.") (documentation\_url .

"<https://github.com/magit/ghub/wiki/Github-Errors>")

it's the issue of url-user-agent.

---

## A.0.12 DONE how the `~/.mozilla/firefox/profiles.ini` looks like?

the profiles name is set in nix as " profiles.firefox = {} ", I want you to write the default looks of that file.

```
[General]
StartWithLastProfile=1

[Profile0]
Name=firefox
IsRelative=1
Path=firefox
Default=1
```

## A.0.13 DONE add sway emacs pkg?

## A.0.14 DONE how to make rm safer?

let rm delete file to trash just don't use term+shell directly, use dired/magit...

## A.0.15 DONE recentf moving file persisnt

every time I move my file and the path changed but the recentf didn't follow, causing production and memory lost. just maintin my own recentf list.

## A.0.16 DONE pull request to the porn site list project

fix the columndeeplly/hosts 's 127.0.0.0 to 0.0.0.0

by clone and compare the problem is that "I include a lost non-porn site in to it". not just "non-porn" sites, I also get a lot cdn server to block, which fetch from a lot of m3u8 streaming server. it may also block some movies which using the same server as porn sites... so, there is only KLUDGE for thing like blocking porn site, first question: what is porn?

really? to be, the social media like twitter which spread sexual clip? the pornhub teaching mathmatic? the reddit/4chan/zhihu/weibo/... even the taobao can sell sex toys with sexual pictures, so blocking internet is blocking porn really? I don't know, it's a question for everyone.

finished, by

```
#!/usr/bin/env bash
set -euo pipefail

HOST_URLS=(
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts00"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts01"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts02"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts03"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts04"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts05"
)

TMPDIR=$(mktemp -d)
REMOTE_DOMAINS="$TMPDIR/remote_domains"
YOUR_DOMAINS="$TMPDIR/your_domains"
```

```

OUTPUT="hosts6"

echo "[*] Downloading hostlists..."
> "$REMOTE_DOMAINS"
for url in "${HOST_URLS[@]}"; do
    echo "uuuu->$url"
    curl -sL "$url" \
        | grep -E "^[0-9:\.]+"
        | awk '{print $2}' \
        >> "$REMOTE_DOMAINS"
done

echo "[*] Normalizing remote domains..."
sort -u "$REMOTE_DOMAINS" > "$TMPDIR/remote_sorted"

echo "[*] Extracting your domains..."
grep -E "^[0-9:\.]+" /etc/hosts \
    | awk '{print $2}' \
    | sort -u \
    > "$YOUR_DOMAINS"

echo "[*] Generating unique domains (hosts6)..."
comm -23 "$YOUR_DOMAINS" "$TMPDIR/remote_sorted" \
    | awk '{print "0.0.0.0", $1}' \
    > "$OUTPUT"

echo "[+] Done. Unique domains saved in $OUTPUT"
echo "uuuu Total: $(wc -l < "$OUTPUT")"

```

get uniq hosts, and update to github. it's I only get 23240 unique lines.

add 17868 new hosts

Used to maintain my hostslist, write a tempermonkey script with ublacklist addon to block chinese keyword search result automatically.

mainly it blocked sexual model gallary sites/Pirate JAV/and chinese porn m3u8 CDN which I filter from a github m3u8 collection repos/or anything non-programming like socialmedia/news/shopping..., occationally the script blocks org/edu/gov sites too.

I remove duplicate hosts in yours and delete non-porn sites, but there still too many which I can't checkout they all.  
BTW, I didn't update my hostslist for a long time, because I just find blocking search engine and "hot" social-media instead porn sites is way more easier, I now use tavily/chatgpt/github in emacs only.

BTW, seems like use 0.0.0.0 instaed of 127.0.0.1 is faster.

### A.0.17 DONE make Leetcode fresh tasks list

- have already?
- yes/no, yes, edit that file, no, get the files(testcase/question.md/solution.xxx)
- edit the files question to include src in end, solution.xxx to have mode line in header.
- download image to local, turn md in org format, and delete md one.
- add a allorg, to include all under src dir's question.org

- 
- 支持无网本地测试，在源代码文件头部中加入 mode line，在题目文件尾部中加入 include 源代码
  - 根据题号在相同的目录下抓取不同题目和几个指定语言，并下载至本地图片，再转化 md 为 org 格式
  - 支持在根目录中得到 src 的所有 org file 的 include
  - md to org 的 == 问题
  - based on leetgo to creat a leetcode elisp pkg
  - test leetcode emacs pkg instead

## A.0.18 DONE learn how to reference in org

<empty citation> org-cite-insert unable to insert problem **M-RET** to enter cite list.

## A.0.19 DONE upload large files into the github repo

use sed to filter, use what to split into small files.

## A.0.20 DONE try to package real-mono-theme to melpa

minic almost-mono-theme, creat recipe, make pr wait for maintainer check. I am familiarizing with github, feeling awesome!

1. reply from melpa maintainer

Thanks for this. I'm encouraged that you looked at other monochrome themes and went with a comprehensive approach here.

Typically themes are not byte compiled but you still want to maintain conventions across these "stub" files -

☒real-mono-sea-theme.el with byte-compile using Emacs 30.1:

real-mono-sea-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line  
☒real-mono-old-theme.el with byte-compile using Emacs 30.1:

real-mono-old-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line  
☒real-mono-girl-theme.el with byte-compile using Emacs 30.1:

real-mono-girl-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line  
☒real-mono-eink-theme.el with byte-compile using Emacs 30.1:

real-mono-eink-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line  
☒real-mono-dark-theme.el with byte-compile using Emacs 30.1:

real-mono-dark-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line  
☒real-mono-sea-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":

1 issue found: 1:0: error: There is no 'provide' form.

☒real-mono-old-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no 'provide' form.

☒real-mono-girl-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no 'provide' form.

☒real-mono-eink-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no 'provide' form.

☒real-mono-dark-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no 'provide' form.

☒real-mono-themes.el with melpazoid:

- real-mono-themes.el#L469: It's safer to sharp-quote function names; use '#'

☒real-mono-sea-theme.el with checkdoc 0.6.2 (fix within reason):

---

real-mono-sea-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-sea-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-sea-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-sea-theme.el:2: The footer should be: (provide ‘real-mono-sea-theme);; real-mono-sea-  
theme.el ends here

[F]real-mono-old-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-old-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-old-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-old-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-old-theme.el:2: The footer should be: (provide ‘real-mono-old-theme);; real-mono-old-  
theme.el ends here

[F]real-mono-girl-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-girl-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-girl-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-girl-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-girl-theme.el:2: The footer should be: (provide ‘real-mono-girl-theme);; real-mono-girl-  
theme.el ends here

[F]real-mono-eink-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-eink-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-eink-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-eink-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-eink-theme.el:2: The footer should be: (provide ‘real-mono-eink-theme);; real-mono-eink-  
theme.el ends here

[F]real-mono-dark-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-dark-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-dark-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-dark-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-dark-theme.el:2: The footer should be: (provide ‘real-mono-dark-theme);; real-mono-dark-  
theme.el ends here

[F]Package and license:

Please specify :fetcher before :repo in your recipe real-mono-dark-theme.el needs formal license boilerplate and/or  
an SPDX-License-Identifier real-mono-eink-theme.el needs formal license boilerplate and/or an SPDX-License-  
Identifier real-mono-girl-theme.el needs formal license boilerplate and/or an SPDX-License-Identifier real-mono-  
old-theme.el needs formal license boilerplate and/or an SPDX-License-Identifier real-mono-sea-theme.el needs for-  
mal lic  
ense boilerplate and/or an SPDX-License-Identifier

## A.0.21 DONE creat new account's github repos for github.io

have a `username.github.io` public repo, have `index.html` in it, in `url`

## A.0.22 DONE upload Purezen to github

- description: theme that really monochrome A collection of real monochrome emacs themes in a couple of variants.
- clone a monochrome and study it <https://github.com/cryon/almost-mono-themes>
- minic a theme to creat a repo <https://github.com/qingsongliao/real-mono-themes>

## 附录 B 嵌入式软件招聘常见要求

- 本科及以上学历，嵌入式软件工作经验，电子、通信或计算机相关专业；
- 有 ARM Linux 软件的开发经验，熟练使用 C++ 和 C 语言；
- 熟悉 Linux 常用设备操作，有 spi、can、WiFi、audio、video 相关驱动开发经验优先；
- 熟悉多线程、多进程和 socket 网络编程，有并发程序开发经验和良好的设计思路；
- 有机器人相关嵌入式设计或者实时系统经验优先。
- 负责 yocto & android 的构建（Build）与升级（OS Upgrade），确保系统的稳定性和兼容性。
- 参与新硬件平台的 Bringup，包括 CPU、GPU、Memory 等核心组件的初始化和调试。
- 分析和解决系统稳定性问题，优化系统架构设计，提升整体性能和可靠性。
- 研究客户系统功能需求，定制和优化系统功能，满足客户特定场景的需求。
- 解决系统性能问题，包括 CPU、GPU、Memory 等资源的优化与调度。
- 主导基线升级（Baseline Upgrade），确保系统与最新技术和安全标准同步。
- 具备操作系统（如 Android、Linux）的构建与升级经验，熟悉系统启动流程和内核开发。
- 有硬件平台 Bringup 经验，熟悉 CPU、GPU、Memory 等核心组件的初始化和调试。
- 具备系统架构设计能力，能够优化系统性能并解决稳定性问题。
- 有基线升级经验，熟悉版本管理和代码合并流程。
- 具备客户需求分析和功能定制能力，能够根据客户需求优化系统功能。
- 熟悉性能优化工具（如 perf, ftrace, gprof 等），能够解决 CPU、GPU、Memory 相关的性能问题。
- 有嵌入式系统开发经验，熟悉低功耗设计和优化。
- 熟悉虚拟化技术（如 KVM, QEMU）和容器化技术（如 Docker, Kubernetes）。
- 有 MTK 芯片平台开发经验

## 附录 C 嵌入式招聘常见笔试问题

## 附录 D 嵌入式招聘常见面试问题