**257. Binary Tree Paths**

迭代写法，相当于bfs。如果用dfs将会需要额外的空间记录信息

**258. Add Digits**

数学题

基论：The original number is divisible by 9 if and only if the sum of its digits is divisible by 9

结论：n为0，答案为零；n为9的倍数，答案为9；其他答案为n除9取余。推导见<https://leetcode.com/problems/add-digits/solution/>

**136. Single Number**

利用异或运算的性质，相同的数异或之后为0，0和任何数异或之后还是这个数，所以将所有的数异或，即得解

**137. Single Number II**

方法1：统计每个数位上1出现的次数，如果出现3次，则说明重复，如果只有一次，则最终结果的这个数位上也是1，这样最终结果就是只出现一次的数。复杂度为32n，但实际中意外的快。缺点是可扩展性不行，如果有两个出现一次的数就不行了。方法来自<https://www.cnblogs.com/grandyang/p/4263927.html>

**264. Ugly Number II**

枚举法，把n之前的丑数都枚举出来，然后查找只要O(1)的时间

方法1：用堆，每次找出当前最小的丑数加入返回数组，然后用它乘以2,3,5再放回堆里（需要判断重复）

方法2：优化了方法1，方法1的本质是把丑数序列在乘以235来找下一个丑数，可以设置三个数组，每个数组分别是丑数序列乘以235。我们用三个指针分别记录每个数组的乘数分别在丑数序列的位置，然后每次对比三个数组的下一个值来选取最小值。需要注意的是如果有重复，对应的指针位置都要后移，这样就不用判断重复了。其思路类似于合并三个有序的数组，一边计算丑数数组一边利用丑数数组。答案见<https://leetcode.com/problems/ugly-number-ii/solution/>，中文讲解<https://zhuanlan.zhihu.com/p/111954850>

**268. Missing Number**

最简单的办法，求和。或者可以用异或运算的性质来做，把数组的index和value求亦或，最后在亦或上n就可以求得缺失的值。可以推导只有缺失的值异或了一次。

**274. H-Index**

**275. H-Index II**

首先将题目转为直方图形解释，更直观。其次O(n)的解法为使用计数排序，注意到任何大于n的数字都可以转换为n而不影响最终结果。从而限制范围在0-n就可以使用计数排序了。排序之后反向遍历就可以找出n

**279. Perfect Squares**

方法1：动态规划，复杂度是多项式级别，提供了下限

方法2：枚举法。注意因为要找最短路径，所以应该用BFS。代码实现上，BFS可以用两个队列来实现，每次把上层队列中的东西拿出来形成下一个队列，这样每个数被push进去的时候不需要记录额外信息。另外实际中每层队列中集合来实现去重可以大大加速

方法3：数学法。任意一个数可以表示为4个平方数之和（包含零），如果数的形式是，那就需要四个不为零的平方数之和，如果不是，我们检查它是否可以由两个或一个平方数组成，如果行就返回，不行刘返回4。具体见<https://leetcode.com/problems/perfect-squares/solution/>

**290. Word Pattern**

遍历一遍的方法是把每个字母和单词都映射到一个数，每次遇到一样的，就检查对应的数是不是一样，如果一个有，一个没有，或者两个数不一样，就返回false

**299. Bulls and Cows**

只遍历一遍的方法是，用一个数组记录每个digit在其中的次数，在secret中就加一，在guess中就减一。当遇到一样的，bull加一，如果不一样，如果s中的小于零则cow加一，而g中的是大于零cow加一。相当于记录了前面所有遍历过的信息。

**303. Range Sum Query – Immutable**

利用缓存原理，提前将0-i之间的和算好，然后i-j之间的和就是两者相减，常数时间可得

**304. Range Sum Query 2D – Immutable**

同样利用缓存，我们缓存一个起点在(0,0)终点在(i,j)的矩形的和，这样对于任意一个矩形ABCD，面积就是OD-OB-OC+OA（其实和一维中缓存原理和方法一样）

**351. Android Unlock Patterns**

回溯枚举。利用visit记录每个数的访问状态，还需要记录上一个访问的数来决定接下来可以走哪几步。优化在于利用对称性，从1379出发的图案对称，同理2468也一样，设置其中一个为起点然后乘以4即可，最后5单独算。优化和完整代码在<https://leetcode.com/problems/android-unlock-patterns/discuss/82463/Java-DFS-solution-with-clear-explanations-and-optimization-beats-97.61-(12ms)>

**419. Battleships in a Board**

找出所有战舰的起点即可，起点的特点是指为X而且上边和左边不是X。

方法来自<https://www.cnblogs.com/grandyang/p/5979207.html>

**448.Find All Numbers Disappeared in an Array**

一个数组，每个元素都是1到n，有重复，有遗漏，找出所有遗漏

解：把每个数放置到它本来的位置，通过交换两个数实现。直到当前位置或即将交换的位置对应正确的数，重新遍历，所有不对应的就是遗漏的数

答案：1.不考虑空间复杂度，哈希表

2.考虑空间，利用数组本身进行标记，把遇到过的数的对应位置标记为负数

**679. 24 Game**

枚举法，具体的枚举方法是先挑两个数进行四则运算，然后合并为1个数，将这三个数算24,。重复直到最后结果