O(1): 1

O(log n): 2

O(n): 3

O(n log n): 4

O(n^2): 5

O(n^3): 6

O(2^n): 7

O(n \* 2^n): 8

ArabicNumToChinese完全相同 O(log(N)) O(log(N))

AddSubset完全相同O(N \* 2^N) O(N \* 2^N)

SetZeroMatrix完全相同甚至多了注释O(M \* N) O(1)

Shuffle完全相同 O(N^2) O(N)

SpiralMatrix完全相同O(m \* n) O(m \* n)

HierarchicalTraversal 5行/43不同 生成的代码没有冗余变量O(N) O(max(N, W))

PreorderTraversal 完全相同 O(N) O(max(N, H))

DeleteBTSNode 完全相同 O(n) O(n)

ReplaceWord 2行/40不同 生成的代码采用正则表达式//s匹配空格更优 字典树的时间复杂度为O(n) 单词数和字符数，记为m和k 遍历完整个字典树O(n + m \* k) O(n + k)

Dijkstra 代码解释第一次为中文 第二次失败 手动描述后生成失败

MaxArea 逻辑几乎相同但写法不同 原始代码有是否有两根柱子的判断（容错）O(n) O(1)

MaxProfit 逻辑完全相同 生成代码可读性更强O(n) O(1)

MinSubArray 完全相同O(n) O(1)

Rain 完全相同O(n^2) O(1)

RemoveDuplicate 完全相同O(n) O(1)

ThreeNumSum 完全相同甚至多了注释O(n^2) O(n^2)

All1SquareMatrix 逻辑完全相同但生成代码有注释 可读性更强 虽然我认为纯看代码第一段可读性更强O(m \* n) O(m \* n)

MaxSubString 同上

MinStairs 完全相同O(n) O(1)

YHTriangle 完全相同O(numRows^2) O(numRows^2)

CircleList 逻辑几乎相同但写法不同 原始代码有是否存在第二个节点的判断（容错）O(n) O(1)

HappyNum 逻辑完全相同但写法不同O(log n) O(1)

MiddleNode完全相同 生成代码多一句无效注释O(n) O(1)

GlassBall 逻辑完全相同 但生成代码将原始代码有效部分抽象成了函数 更具有可读性

FirstUniqueChar 逻辑完全相同 生成的代码更简洁但比较难以理解O(n) O(n)

SumOf2Num 完全相同O(n) O(n)

FindKthLargest 完全相同 保持队列的大小为k O(n log k) O(k)

Add2Nums 完全相同 O(max(m, n)) O(max(m, n))

CopyRandomList 几乎完全相同但生成的代码报错 缺少一行定义O(n) O(n)

DeleteNNodeFromEnd 完全相同 O(n) O(1)

GetIntersectionNode 完全相同 O(m+n) O(m)

ReverseList 完全相同O(n) O(1)

RotateList 完全相同 O(n) O(1)

Merge 完全相同 O(nlogn) O(n)

IntReverse 完全相同 O(log|x|) O(1)

NumOfReplies 逻辑不同 但都正确 原始代码更具效率O(log|x|) O(1)

StringMultiplication 逻辑不同 但都正确 生成的代码效率更高但更难以理解O(m \* n) O(m + n)

StringToInt 逻辑基本相同 生成的代码容错性更好O(n) O(1)

TrailingZeros 逻辑基本相同 生成的代码更好理解O(logn) O(1)

MergeLists 完全相同 O(m+n)

SearchSpanSortedArray 生成的代码不完全 报错 O(log n) O(log n)

InvertTree 逻辑不同 原代码用递归 生成代码用栈 O(n) 取决于树的高度，最坏情况下为O(h)，平均情况下为O(log(n))。

LowestCommonAncestor 完全相同 生成代码用栈 O(n) 取决于树的高度，最坏情况下为O(h)，平均情况下为O(log(n))。

MaxDepth 完全相同生成代码用栈 O(n) 取决于树的高度，最坏情况下为O(h)，平均情况下为O(log(n))。

Merge2Lists 完全相同 生成的代码有注释 O(m+n) O(m+n)

RemoveLeafNodes 完全相同 O(n) O(n)

SymmetricBinaryTree 完全相同 O(n) O(n)

Brackets 完全相同 生成有效括号对的解的数量为卡特兰数，即Catalan(n) = (2n)! / ((n+1)! \* n!)。因此，时间复杂度为O(Catalan(n))。 O(n)

CombinationSum 完全相同 O(2^n) O(n)

RegularExpressionMatching 完全相同 O(m \* n) O(m + n)

Calculator 功能相同但实现细节不同 原始代码容错性更好更严谨 O(n) O(n)

Calculator2 功能相同但实现细节不同 原始代码可读性和可复用性强 O(n) O(n)

LongestValidBrackets 功能相同但实现细节不同 原始代码容错性更好 生成代码更具效率 O(n) O(n)

MaxQueue 同上 O(k) O(n)

MinStack 完全相同 O(1) O(n)

RemoveDuplicateLetters 功能相同但实现细节不同 原始代码可读性更好 生成代码更具效率 O(n) O(n)

ValidBrackets 同上 O(n) O(n)

DelineateLetterRange 完全相同 O(n) O(n)

ZTransfer 功能相同但实现细节不同 生成代码更具效率 在可读性上各有千秋O(n + numRows \* m) O(n)

ProducerConsumer 完全相同

TreeTraversal 基本相同 生成代码在前序遍历中效率略逊于原始代码 O(n) O(n)