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 totally same O(log(N)) O(log(N))

AddSubset totally same O(N \* 2^N) O(N \* 2^N)

SetZeroMatrix totally same Even more notes O(M \* N) O(1)

Shuffle totally same O(N^2) O(N)

SpiralMatrix totally same O(m \* n) O(m \* n)

HierarchicalTraversal 5 lines/43 different Generated code without redundant variables O(N) O(max(N, W))

PreorderTraversal totally same O(N) O(max(N, H))

DeleteBTSNode totally same O(n) O(n)

ReplaceWord 2 lines/40 different Generated code uses regular expression //s matching spaces is better Time complexity of dictionary tree is O(n) Number of words and number of characters, denoted as m and k Traversing the complete dictionary tree O(n + m \* k) O(n + k)

Dijkstra Code Interpretation First Chinese Second Failure Generation Failure After Manual Description O(n^2)O(n^2)

MaxArea The logic is almost the same but written differently The original code has a judgment of whether there are two columns (fault tolerance) O(n) O(1)

MaxProfit logically totally same Generate code that is more readable O(n) O(1)

MinSubArray totally sameO(n) O(1)

Rain totally same O(n^2) O(1)

RemoveDuplicate totally same O(n) O(1)

ThreeNumSum totally sameEven more notes O(n^2) O(n^2)

All1SquareMatrix logically totally same But the generated code is more readable with comments, although I think it's more readable purely by looking at the first paragraph of the code. O(m \* n) O(m \* n)

MaxSubString ibid

MinStairs totally sameO(n) O(1)

YHTriangle totally same O(numRows^2) O(numRows^2)

CircleList The logic is almost the same but written differently The original code has a judgment of whether a second node exists (fault tolerance) O(n) O(1)

HappyNum logically totally same But it's written differently. O(log n) O(1)

MiddleNode totally same Generate code with one more invalid comment O(n) O(1)

GlassBall logically totally same But the generated code abstracts the valid parts of the original code into functions that are more readable. O(n^2)O(n^2)

FirstUniqueChar logically totally same The generated code is more concise but harder to understand O(n) O(n)

SumOf2Num totally sameO(n) O(n)

FindKthLargest totally same Keep the size of the queue as k O(n log k) O(k)

Add2Nums totally same O(max(m, n)) O(max(m, n))

CopyRandomList totally sameBut the generated code gives an error Missing one line of definition O(n) O(n)

DeleteNNodeFromEnd totally same O(n) O(1)

GetIntersectionNode totally same O(m+n) O(m)

ReverseList totally sameO(n) O(1)

RotateList totally same O(n) O(1)

Merge totally same O(nlogn) O(n)

IntReverse totally same O(log|x|) O(1)

NumOfReplies in different logic But it's all right. The original code was more efficient. O(log|x|) O(1)

StringMultiplication in different logic The resulting code is more efficient but harder to understand. O(m \* n) O(m + n)

StringToInt basically logically same Generated code is more fault tolerant O(n) O(1)

TrailingZeros basically logically same Generated code is better understood O(logn) O(1)

MergeLists totally same O(m+n) O(1)

SearchSpanSortedArray Incomplete code generated. O(log n) O(log n)

InvertTree in diferent logic Recursion for original code Stack for generated code O(n) depends on the height of the tree, O(h) in the worst case and O(log(n)) in the average case.

LowestCommonAncestor totally same Stack for generated code O(n) depends on the height of the tree, O(h) in the worst case and O(log(n)) in the average case.

MaxDepth totally sameGenerating code with a stack O(n) depends on the height of the tree, O(h) in the worst case and O(log(n)) in the average case.

Merge2Lists totally same The generated code is commented O(m+n) O(m+n)

RemoveLeafNodes totally same O(n) O(n)

SymmetricBinaryTree totally same O(n) O(n)

Brackets totally same The number of solutions that generate valid bracket pairs is the Catalan number, i.e. Catalan(n) = (2n)! / ((n+1)! \* n!). Therefore, the time complexity is O(Catalan(n)). O(n)

CombinationSum totally same O(2^n) O(n)

RegularExpressionMatching totally same O(m \* n) O(m + n)

Calculator Same functionality but different implementation details Original code is more fault tolerant and rigorous O(n) O(n)

Calculator2 Same functionality but different implementation details Original code is readable and reusable O(n) O(n)

LongestValidBrackets Same functionality but different implementation details Original code is more fault tolerant Generated code is more efficient O(n) O(n)

MaxQueue ibid O(k) O(n)

MinStack totally same O(1) O(n)

RemoveDuplicateLetters Same functionality but different implementation details Original code is more readable Generated code is more efficient O(n) O(n)

ValidBrackets ibid O(n) O(n)

DelineateLetterRange totally same O(n) O(n)

ZTransfer Functionality is the same but implementation details are different Generating code is more efficient Each has its own advantages in terms of readability O(n + numRows \* m) O(n)

ProducerConsumer totally same

TreeTraversal logically same The generated code is slightly less efficient than the original code in the preorder traversal O(n) O(n)