# Binary search

## Template

* while start + 1 < end 🡪 不会造成dead loop，但要double check
* mid = start + (end – start) /2
* A[mid] ==, <, >
* A[start] A[end] ? target

public int findPosition(int[] nums, int target) {

if (nums == null || nums.length == 0) {

return -1;

}

int start = 0, end = nums.length - 1;

// while start < end: 相邻就退出，start=1, end=2 就退出

while (start + 1 < end) {

// start，end=2^ 31，(start + end)/2对32bits int越界

int mid = start + (end - start) / 2;

if (nums[mid] == target) {

start = mid;

} else if (nums[mid] < target) {

start = mid;

} else {

end = mid;

}

}

// double check

if (nums[end] == target) {

return end;

}

if (nums[start] == target) {

return start;

}

return -1;

}

## Types

* OOXX (there’s a point allows to divide the list, sorted)
  + First match
  + Search Insert position: find first position >= target, the position is the insertion
  + Exponential search: search in a big sorted array
  + Find minimum in rotated sorted array
    - i.e., 0,1,2,4,5,6,7 become 4,5,6,7,0,1,2
* Half half
  + Find peak element (follow-up: find peak element II)
  + Search in rotated sorted array
* Others
  + Sqrt(x): y – 1~x
  + Wood cut
  + Copy books
  + 2D Matrix II - O(n + m)

A screenshot of a computer

Description automatically generated

If m[i][j] < target, travel horizontally; if m[i][j] > target, travel vertically.

## Questions

|  |  |  |
| --- | --- | --- |
| Basic binary search | | |
| [704](https://leetcode.com/problems/binary-search/) | Binary Search |  |
| [69](https://leetcode.com/problems/sqrtx/) | Sqrt(x) |  |
| [278](https://leetcode.com/problems/first-bad-version/) | First bad version |  |
| Search insert position | | |
| [35](https://leetcode.com/problems/search-insert-position/) | Search Insert Position |  |
| [34](https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/) | Find First and Last Position of Element |  |
| [702](https://leetcode.com/problems/search-in-a-sorted-array-of-unknown-size/) | Search in Unknown Size array |  |
| [162/852](https://leetcode.com/problems/find-peak-element/) | Find peak element | peak left side is going up while right side going down |
| Rotated Sorted Array | | |
| [33](https://leetcode.com/problems/search-in-rotated-sorted-array) | Search in RSA |  |
| [81](https://leetcode.com/problems/search-in-rotated-sorted-array-ii/) | Search in RSA II (duplicates) | remove dups: a[m] == a[s], s++ |
| [153](https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/) | Find Min in RSA |  |
| [154](https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/) | Find Min in RSA II (duplicates) | remove dups |
| 2D Matrix | | |
| [74](https://leetcode.com/problems/search-a-2d-matrix/) | Search in a 2D matrix |  |
| [240](https://leetcode.com/problems/search-a-2d-matrix-ii/) | Search in a 2D matrix II | row starts at 0, col starts at end of row |
| Wood Cut | | |
| 183 | woodcut |  |
| 875 | Koko eating bananas | eat SLOWLY, find min of all possibility |
| [1011](https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/) | Capacity to ship packages within days | day starts at 1 not 0 |
| (lint)437 | Copy book |  |
| Kth Smallest in Matrix | | |
| [378](https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/) | Kth Smallest in a Sorted Matrix | row starts at 0, col starts at end of row |
| 668 | Kth Smallest in a Sorted Matrix |  |
| [719](https://leetcode.com/problems/find-k-th-smallest-pair-distance/) | Find Kth Smallest Pair Distance | nums[j] - nums[i] increase when i++ and decrease j-- thus j should starts at 0 not end |
| [786](https://leetcode.com/problems/k-th-smallest-prime-fraction/) | Kth Smallest Prime Fraction |  |

# Binary Tree – Recursion: Divide & Conquer

## Traverse a Binary Tree

A screenshot of a cell phone

Description automatically generated

Recursion: Traverse || Divide & Conquer

* Preorder: add root value – traverse left – traverse right
* Inorder:
* Postorder:

Non-recursion

* Preorder: stack, root – right – left
* Inorder:
* Postorder:

## Recursion Template

// Template 1: Traverse

// global return value / result to update

// recursion does not return anything

public class Solution {

public void traverse(TreeNode root) {

if (root == null) {

return;

}

// do something with root

// do something with left

traverse(root.left);

// do something with right

traverse(root.right);

}

}

// Template 2: Divide & Conquer

// recursion returns the left / right result - result type could be int, boolean or selfdefined

public class Solution {

public ResultType traversal(TreeNode root) {

// null or leaf

if (root == null) {

// do something and return

}

// Divide

ResultType left = traversal(root.left);

ResultType right = traversal(root.right);

// Conquer

ResultType result = Merge from left and right.

return result;

}

}

## Questions

|  |  |  |
| --- | --- | --- |
| Traverse a Binary Tree | | |
| [144](https://leetcode.com/problems/binary-tree-preorder-traversal/)  **必背** | Preorder traversal | DFS: stack to push right and push left, res stores the pop value. |
| [145](https://leetcode.com/problems/binary-tree-postorder-traversal/)  **必背** | Postorder traversal | DFS: stack to push left and right, res stores the pop value + return res[::-1] |
| [94](https://leetcode.com/problems/binary-tree-inorder-traversal/)  **必背** | Inorder traversal | DFS: go to the very left children, and back to the parent level to right, for right node continue to the very left children |
| Divide and Conquer Algorithm – Path and Sum | | |
| [110](https://leetcode.com/problems/balanced-binary-tree/) | Balanced Binary Tree |  |
| [104](https://leetcode.com/problems/maximum-depth-of-binary-tree/) | Maximum depth of Binary Tree | BFS Iteration: level order traversal, stack |
| [257](https://leetcode.com/problems/binary-tree-paths/) | Binary Tree Paths | BFS Iteration: stack = [(node, PATH to this node)] |
| [114](https://leetcode.com/problems/flatten-binary-tree-to-linked-list/) | Flatten Binary Tree to Linked List | Iteration: keep left forward to append into stack, and pop to be right node. Recursion: return left most and right most |
|  |  | Related:   |  |  | | --- | --- | | [430](https://leetcode.com/problems/flatten-a-multilevel-doubly-linked-list/) | Flatten a Multilevel Doubly LL | |
| [236](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/) | Lowest Common Ancestor BT | Recursion |
|  |  | Related:   |  |  | | --- | --- | | [235](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/) | Lowest Common Ancestor BST | | L474 | LCA II with parent pointer | | L578 | LCA III – may not exist  add recursion method DoesExist() | |
| [549](https://leetcode.com/problems/binary-tree-longest-consecutive-sequence-ii/) | Longest Consecutive Sequence II | circulation |
|  |  | Related & locked:   |  |  | | --- | --- | | [298](https://leetcode.com/problems/binary-tree-longest-consecutive-sequence/) | Longest Consecutive Sequence | |
| [L596](http://buttercola.blogspot.com/2019/04/lintcode-596-minimum-subtree.html) | Minimum Subtree |  |
| [L597](https://mrleonhuang.gitbooks.io/lintcode/binary-tree-and-divide-conquer/subtree-with-maximum-average.html) | Subtree with Max Average |  |
| [112](https://leetcode.com/problems/path-sum/) | Binary Tree Path Sum | Iteration: stack = [(node, SUM to the node.val)] |
| [113](https://leetcode.com/problems/path-sum-ii/) | Binary Tree Path Sum II | Recursion: traverse: need to be careful about in-place “append” method of list, if left do sth with the path and if right do sth with the path again. |
| [124](https://leetcode.com/problems/binary-tree-maximum-path-sum/) | Binary Tree Max Path Sum | Recursion: d & q, cross-left-right!!! circulation |
| Binary Search Tree | | |
| [98](https://leetcode.com/problems/validate-binary-search-tree/) | Validate BST | 3 parameters in helper(), min = float(‘-inf’), max = float(‘inf’) |
| [173](https://leetcode.com/problems/binary-search-tree-iterator/)  **必背** | BST iterator |  |
| [99](https://leetcode.com/problems/recover-binary-search-tree/) | Recover BST | Iteration: in-order traversal |
| [230](https://leetcode.com/problems/kth-smallest-element-in-a-bst/) | Kth Smallest Element in BST | Iteration: in-order traversal |
| [108](https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/) | Convert Sorted Array to BST | Divide & conquer; avoiding slicing |
| [109](https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree/) | Convert Sorted List to BST | Two pointers – slow and fast, st: root k, left child 2k, right child 2k+1 |
| [538](https://leetcode.com/problems/convert-bst-to-greater-tree/)  [1038](https://leetcode.com/problems/binary-search-tree-to-greater-sum-tree/) | Convert BST to Greater Tree | Recursion: traverse – global add-on value, right -> left  Iteration: in-order traversal-right to left |
| [426](https://leetcode.com/problems/convert-binary-search-tree-to-sorted-doubly-linked-list/) | Convert BST to Sorted Doubly Linked List | **LOCKED!!!!!** |
| [285](https://leetcode.com/problems/inorder-successor-in-bst/) | Inorder Successor in BST | **LOCKED!!!!!** |
| L11 | Search Range in BST |  |
| L85 | Insert Node in a BST |  |
| L87 | Remove Node in a BST | 1. find the node;  2. find the max node in the left subtree  3. replace the node with the max node  Special cases:  1. the node doesn’t have a left child;  2. the max node in the left subtree has a left child  3. the node is the root of the tree |

# Binary Tree

## Questions

|  |  |  |
| --- | --- | --- |
| Traverse a Binary Tree | | |
| [144](https://leetcode.com/problems/binary-tree-preorder-traversal/) | Preorder traversal | DFS: stack to push right and push left, res stores the pop value. |
| [145](https://leetcode.com/problems/binary-tree-postorder-traversal/) | Postorder traversal | DFS: stack to push left and right, res stores the pop value + return res[::-1] |
| [94](https://leetcode.com/problems/binary-tree-inorder-traversal/) | Inorder traversal | DFS: go to the very left children, and back to the parent level to right, for right node continue to the very left children |
| [590](https://leetcode.com/problems/n-ary-tree-postorder-traversal/) | N-ary Postorder Traversal | BFS: stack.extend instead stack.append, because node.children is a list to extend |
| [102](https://leetcode.com/problems/binary-tree-level-order-traversal/)/107 | Level order traversal I & II | II: reverse of I’s result; BFS. – NONE RECURSION |
| [103](https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/) | ZigZag Level Order Traversal | BFS |
| [314](https://leetcode.com/problems/binary-tree-vertical-order-traversal/) | Vertical Order Traversal | **LOCKED!!!!** |
| Binary Tree determination / Convert | | |
| [110](https://leetcode.com/problems/balanced-binary-tree/) | Balanced Binary Tree |  |
| [100](https://leetcode.com/problems/same-tree/) | Same Tree | Iteration: stack = [(p, q)] |
| [101](https://leetcode.com/problems/symmetric-tree/) | Symmetric Tree | Iteration: stack = [(p, q)] |
| [156](https://leetcode.com/problems/binary-tree-upside-down/) | Binary Tree Upside Down | **LOCKED!!!!** |
| [199](https://leetcode.com/problems/binary-tree-right-side-view/) | Binary Tree Right Side View | Recursion: if len(right) < len(left): res.extend(left[len(right):]);  BFS |
| [226](https://leetcode.com/problems/invert-binary-tree/) | Invert Binary Tree | Recursion – divide and conquer |
| [297](https://leetcode.com/problems/serialize-and-deserialize-binary-tree/) | Serialize and Deserialize of Binary Tree | DFS or BFS |
| Binary Tree Paths | | |
| [104](https://leetcode.com/problems/maximum-depth-of-binary-tree/) | Maximum depth of Binary Tree | BFS Iteration: level order traversal, stack |
| [111](https://leetcode.com/problems/minimum-depth-of-binary-tree/) | Minimum depth of Binary Tree | BFS Iteration: level order traversal, stack |
| [257](https://leetcode.com/problems/binary-tree-paths/) | Binary Tree Paths | BFS Iteration: stack = [(node, PATH to this node)] |
| [114](https://leetcode.com/problems/flatten-binary-tree-to-linked-list/) | Flatten Binary Tree to Linked List | Iteration: keep left forward to append into stack, and pop to be right node. Recursion: return left most and right most |
|  |  | Related:   |  |  | | --- | --- | | [430](https://leetcode.com/problems/flatten-a-multilevel-doubly-linked-list/) | Flatten a Multilevel Doubly LL | |
| [236](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/) | Lowest Common Ancestor BT | Recursion |
|  |  | Related:   |  |  | | --- | --- | | [235](https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-search-tree/) | Lowest Common Ancestor BST | | L474 | LCA II with parent pointer | | L578 | LCA III – may not exist  add recursion method DoesExist() | |
| [549](https://leetcode.com/problems/binary-tree-longest-consecutive-sequence-ii/) | Longest Consecutive Sequence II | circulation |
|  |  | Related & locked:   |  |  | | --- | --- | | [298](https://leetcode.com/problems/binary-tree-longest-consecutive-sequence/) | Longest Consecutive Sequence | |
| [366](https://leetcode.com/problems/find-leaves-of-binary-tree/) | Find Leaves of Binary Tree | **LOCKED!!!!!** |
| [337](https://leetcode.com/problems/house-robber-iii/) | House Robber III | Recursion: d & q, return (rob, not rob) |
| [257](https://leetcode.com/problems/binary-tree-paths/) | Binary Tree Paths |  |
| Binary Tree Sum | | |
| [L596](http://buttercola.blogspot.com/2019/04/lintcode-596-minimum-subtree.html) | Minimum Subtree |  |
| [L597](https://mrleonhuang.gitbooks.io/lintcode/binary-tree-and-divide-conquer/subtree-with-maximum-average.html) | Subtree with Max Average |  |
| [112](https://leetcode.com/problems/path-sum/) | Binary Tree Path Sum | Iteration: stack = [(node, SUM to the node.val)] |
| [113](https://leetcode.com/problems/path-sum-ii/) | Binary Tree Path Sum II | Recursion: traverse: need to be careful about in-place “append” method of list, if left do sth with the path and if right do sth with the path again. |
| [124](https://leetcode.com/problems/binary-tree-maximum-path-sum/) | Binary Tree Max Path Sum | Recursion: d & q, cross-left-right!!! circulation |
| [129](https://leetcode.com/problems/sum-root-to-leaf-numbers/) | Sum Root to Leaf Numbers | Recursion: need to practice Iteration: stack = [(node, NUMBER to the node.val)] |
| [250](https://leetcode.com/problems/count-univalue-subtrees/) | Count Univalue Subtrees | **LOCKED!!!!!** |
| Binary Search Tree | | |
| [98](https://leetcode.com/problems/validate-binary-search-tree/) | Validate BST | 3 parameters in helper(), min = float(‘-inf’), max = float(‘inf’) |
| [173](https://leetcode.com/problems/binary-search-tree-iterator/) | BST iterator |  |
| [669](https://leetcode.com/problems/trim-a-binary-search-tree/) | Trim a Binary Search Tree |  |
| [99](https://leetcode.com/problems/recover-binary-search-tree/) | Recover BST | Iteration: in-order traversal |
| [230](https://leetcode.com/problems/kth-smallest-element-in-a-bst/) | Kth Smallest Element in BST | Iteration: in-order traversal |
| [108](https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/) | Convert Sorted Array to BST | Divide & conquer; avoiding slicing |
| [109](https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree/) | Convert Sorted List to BST | Two pointers – slow and fast, st: root k, left child 2k, right child 2k+1 |
| [538](https://leetcode.com/problems/convert-bst-to-greater-tree/)  [1038](https://leetcode.com/problems/binary-search-tree-to-greater-sum-tree/) | Convert BST to Greater Tree | Recursion: traverse – global add-on value, right -> left  Iteration: in-order traversal-right to left |
| [426](https://leetcode.com/problems/convert-binary-search-tree-to-sorted-doubly-linked-list/) | Convert BST to Sorted Doubly Linked List | **LOCKED!!!!!** |
| 270 | Closest BST Value | **LOCKED!!!!!** |
| 272 | Closest BST Value II | **LOCKED!!!!!** |
| [285](https://leetcode.com/problems/inorder-successor-in-bst/) | Inorder Successor in BST | **LOCKED!!!!!** |
| L11 | Search Range in BST |  |
| L85 | Insert Node in a BST |  |
| L87 | Remove Node in a BST | 1. find the node;  2. find the max node in the left subtree  3. replace the node with the max node  Special cases:  1. the node doesn’t have a left child;  2. the max node in the left subtree has a left child  3. the node is the root of the tree |
| Graph, DP, related … | | |
| [95](https://leetcode.com/problems/unique-binary-search-trees-ii/) | Unique BST II | Wait for DP |
| [96](https://leetcode.com/problems/unique-binary-search-trees/) | Unique BST | Wait for DP |
| [116](https://leetcode.com/problems/populating-next-right-pointers-in-each-node/) | Populating Next Right Pointers in Each Node |  |
| [117](https://leetcode.com/problems/populating-next-right-pointers-in-each-node-ii/) | Populating Next Right Pointers in Each Node II | Level order traversal add the next pointer |
| [310](https://leetcode.com/problems/minimum-height-trees/) | Minimum Height Trees |  |

# Graph – BFS

## Types

* Traversal in Graph
  + Level order traversal层级遍历: size = queue.size()
  + Connected components
  + **Topological sorting: DFS to find cycle path**
* Shortest path in simple graph (undirected & unweighted graph):
  + Cannot use DFS;
  + DFS will find all the paths to the required vertex
* Serialization: object to string – string to object将内存中结构化的数据变成字符串
* 坐标变换数组：
  + deltaX, deltaY
  + inbound 有没有越界，或者墙啊房啊

## Template

### Binary Tree – level order - Queue

public class Solution {

private ArrayList<Integer> levelOrder(TreeNode root) {

ArrayList<ArrayList<Integer>> results = new ArrayList<>();

if (root == null) {

return results;

}

Queue<TreeNode> queue = new LinkedList<TreeNode>();

queue.offer(root);

while (!queue.isEmpty()) {

ArrayList<Integer> level = new ArrayList<>();

int size = queue.size();

for (int i = 0; i < size; i++) {

TreeNode node = queue.poll();

level.add(node.val);

if (node.left != null) {

queue.offer(node.left);

}

if (node.right != null) {

queue.offer(node.right);

}

}

results.add(level);

}

return results;

}

}

### Graph

*Complete BFS in three bullets*

1. Count (either count level or visit total amount)
2. Another list to record each level (no need to know the level)
3. For loop to travel all the neighbors (if it is a tree, this is not needed)

public class Solution {

private ArrayList<UndirectedGraphNode> getNodes(UndirectedGraphNode node) {

Queue<UndirectedGraphNode> queue = new LinkedList<UndirectedGraphNode>();

HashSet<UndirectedGraphNode> set = new HashSet<>();

queue.offer(node);

set.add(node);

int visited = 0;

while (!queue.isEmpty()) {

UndirectedGraphNode head = queue.poll();

visited++;

for (UndirectedGraphNode neighbor : head.neighbors) {

if (set.contains(neighbor)) {

continue;

}

set.add(neighbor);

queue.offer(neighbor);

}

}

return new ArrayList<UndirectedGraphNode>(set);

// or

return (visited == n);

}

}

## Questions

|  |  |  |
| --- | --- | --- |
| BFS in Binary Tree - QUEUE | | |
| [102](https://leetcode.com/problems/binary-tree-level-order-traversal/)  107 | Level order traversal I & II | II: reverse of I’s result; BFS. – NONE RECURSION  Related:   |  |  | | --- | --- | | L242 | Convert Binary Tree to LinkedList by Depth | |
| [103](https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/) | ZigZag Level Order Traversal | BFS |
| [297](https://leetcode.com/problems/serialize-and-deserialize-binary-tree/) | Serialize and Deserialize of Binary Tree | BFS & DFS |
| BFS in Graph – HashMap – **由点及面** 一个点去找到图中所有的点 N vertices, M edges: max(M) = N^2 Complexity = **O(M) = O(N^2)** at worst case | | |
| [261](https://leetcode.com/problems/graph-valid-tree/)  [L178](https://www.lintcode.com/en/old/problem/graph-valid-tree/) | Graph Valid Tree | 1. edge = n – 1 is tree  2. connected but no cycle 通过一个点连了所有的点  Hashmap of set, vertex:{connected vertices} |
| [133](https://leetcode.com/problems/clone-graph/) | Clone Graph | 0. get nodes from graph and store in a list  1. copy nodes  2. copy edges  Related:   |  |  | | --- | --- | | [78](https://leetcode.com/problems/subsets/) | Subsets – BFS level by level; DFS | | [90](https://leetcode.com/problems/subsets-ii/) | Subsets II - DFS | |
| L618 | Search Graph Nodes | Related?   |  |  | | --- | --- | | [787](https://leetcode.com/problems/cheapest-flights-within-k-stops/) | Cheapest Flights within K stops | |
| [863](https://leetcode.com/problems/all-nodes-distance-k-in-binary-tree/) | All Nodes Distance K in Binary Tree |  |
| [323](https://leetcode.com/problems/number-of-connected-components-in-an-undirected-graph/) | Number of Connected Component in an Undirected Graph | LOCKED! - Union find – important  Same?   |  |  | | --- | --- | | L431 | Connected Component in undirected graph | |
| [302](https://leetcode.com/problems/smallest-rectangle-enclosing-black-pixels/) | Smallest Rectangle Enclosing Black Pixels | LOCKED |
| [127](https://leetcode.com/problems/word-ladder/) | Word Ladder | Using 26 letters to replace words: wordLength \* 26 |
| Topological Sorting | | |
| [L127](https://www.lintcode.com/problem/topological-sorting/my-submissions) | Topological Sorting | BFS:  1. Collect in-degree  2. pull all nodes that indegree = 0 into queue  3. bfs  IF CYCLE MAY EXIST: topoOrder.size() != graph.size()  (BFS 1’47’’)  DFS |
| [210](https://leetcode.com/problems/course-schedule-ii/) | Course Schedule II | Related:   |  |  | | --- | --- | | [207](https://leetcode.com/problems/course-schedule/) | Course Schedule: - DFS | | 630 | Course Schedule III - greedy | |
| 269  [L892](https://www.lintcode.com/problem/alien-dictionary/note/163019) | Alien Dictionary | IF CYCLE MAY EXIST: topoOrder.size() != graph.size() |
| [444](https://leetcode.com/problems/sequence-reconstruction/)/  [L605](https://www.lintcode.com/problem/sequence-reconstruction/description) | Sequence reconstruction | Cannot pass corner cases  Related:   |  |  | | --- | --- | | [406](https://leetcode.com/problems/queue-reconstruction-by-height/) | Queue Reconstruction by height | |
| BFS in Matrix – easily to think start with DFS, but BFS is easier N row\* M col nodes with N\*M\*2 edges (每个点上下左右4条边，每边2个点共享): **O(N \* M)** | | |
| [200](https://leetcode.com/problems/number-of-islands/) | Number of Islands | Coordination:  deltaX = {1, 0, 0, -1}  deltaY = {0, 1, -1, 0}  either BFS or DFS  Related:   |  |  | | --- | --- | | [305](https://leetcode.com/problems/number-of-islands-ii/) | Number of Islands II (Locked)  Union find | |
| [130](https://leetcode.com/problems/surrounded-regions/) | Surrounded Regions | Search from the border ‘O’ |
| [L598](https://www.lintcode.com/problem/zombie-in-matrix/description) | Zombie in Matrix |  |
| 286 | Walls and Gates | LOCKED |
| L611 | Knight Shortest Path | LOCKED  8 direction coordination:  deltaX = {1, 1, 2, 2, -1, -1, -2, -2}  deltaY = {2, -2, 1, -1, 2, -2, 1, -1} |
| L573 | Build Post Office II | LOCKED  Same as 127 word ladder to build next possibility |

# Graph – DFS / Recursion

## Types

* IsCycle()
* UnionFind()
* All possible ……:找所有方案的题，一定是DFS，要么是排列，要么是组合;

#### Complexity

* 搜索的时间复杂度：O(答案总数 构造每个答案的时间)
  + Example: Subsets问题，求所有的子集。 子集的个数一共有2^n，每个集合的平均长度是O(n)的，所以时间复杂度为O(n 2^n)，同理permutations问题的复杂度就是O(n n!)
* 动态规划的时间复杂度：O(答案总数 构造每个答案的时间)
  + Example: triangle问题，数字三角形的最短路径。 状态总数约有2^n，每个状态的时间复杂度是O(1)-就是求一下min，所以总的复杂度O(2^n)
* 分治方法的二叉树问题: O(二叉树节点个数 每个节点时间)
  + Example: 二叉树最大深度。二叉树节点个数为N，每个节点上的时间为O(1)。所以总的复杂度O(N)

## Template

* Sequence

1. Add a number
2. keep looking for the all possibility with the added number prefix;
3. Remove a number, getting back 回溯

A close up of a logo

Description automatically generated

* Difference between combination and permutation
  + Combination:
    - O(2^n)
    - startIndex: without the order, so should start at the next one
  + Permutation:
    - O(n!)
    - No startIndex

public class Solution {

public List<List<Integer>> combination

Sum(int[] candidates, int target) {

List<List<Integer>> results = new ArrayList<>();

if (candidates == null || candidiates.length == 0) {

return results;

}

int[] nums = removeDuplicates(candidates);

helper(nums, 0, new ArrayList<Integer>(), target, results);

}

// 1. 递归的定义

// 从nums中的startIndex开始挑选一些书，放到combination中，且他们的和为target

private void helper(int[] nums, int startIndex, List<Integer> combination, int target,

List<List<Integer>> results) {

// 3. 递归的出口

if (target == 0) {

// deep

results.add(new ArrayList<Integer>(combination));

return;

}

// 2. 递归的拆解

// [1,2], [1,3], [1,4], ......[1,5]

// [2,3], [2,4], ......[2,5]

for (int i = startIndex; i < nums.length; i++) {

if (nums[i] > target) {

break;

}

// [1] --> [1, 2]

combination.add(nums[i]);

// 把所有[1, 2]开头的（剩余的）和为remainTarget的集合，都找到，放到results

helper(nums, i, combination, target - nums[i], results);

// 回溯！[1,2] --> [1]

combination.remove(combination.size() - 1);

}

}

private int[] removeDuplicates(int[] candidates) {

Array.sort(candidates);

int index = 0;

for (int i = 0; i < candidates.length; i++) {

if (candidates[i] != candidates[index]) {

candidates[++index] = candidates[i];

}

}

int[] nums = new int[index + 1];

for (int i = 0; i < index + 1; i++) {

nums[i] = candidates[i];

}

return nums;

}

}

## Questions

|  |  |  |
| --- | --- | --- |
| DFS combination组合 / Partition 切分组合中的元素与顺序无关; O(2^n) | | |
| [78](https://leetcode.com/problems/subsets/)  **必背** | Subsets | BFS level by level; DFS recursion – saving space  NON-recursion: bit manipulation (bitwise operator)  Related:   |  |  | | --- | --- | | L142 | O(1) check power of 2  if (n <= 0) {  return false;  }  return (n & (n - 1)) == 0; | |
| [90](https://leetcode.com/problems/subsets-ii/) | Subsets II  Unique subset | 去重：Sorting +  If (i != startIndex && candidates[i] == candidates[i-1]) { continue;} |
| [39](https://leetcode.com/problems/combination-sum/) | Combination Sum | Related:   |  |  | | --- | --- | | [377](https://leetcode.com/problems/combination-sum-iv/) | Combination Sum IV - DP | |
| [40](https://leetcode.com/problems/combination-sum-ii/) | Combination Sum II | 如何去重? == subset II |
| [216](https://leetcode.com/problems/combination-sum-iii/) | Combination Sum III |  |
| [131](https://leetcode.com/problems/palindrome-partitioning/) | Palindrome Partitioning | Related:   |  |  | | --- | --- | | 132 | Palindrome Partitioning II -DP | |
| DFS – Permutation 排列 求出所有满足条件的排列; 组合中的元素与顺序相关; O(n!) | | |
| [46](https://leetcode.com/problems/permutations/)  **必背** | Permutations | Related:   |  |  | | --- | --- | | [31](https://leetcode.com/problems/next-permutation/)  Math | Next Permutation  **(not DFS, but need to remember)** | |
| [47](https://leetcode.com/problems/permutations-ii/) | Permutations II  Unique Permutation | 去除重复元素前一个没有使用那就坚决不能用后一个  if (i != 0 && nums[i] == nums[i-1]&&visited[i-1]=0) |
| [51](https://leetcode.com/problems/n-queens/) | [N-Queens](#nqueen) | Iterative rows, and dfs columns |
| [126](https://leetcode.com/problems/word-ladder-ii/) | Word Ladder II | BFS + DFS  BFS: end -> start -> hashmap:distance[“abc”] = 10  DFS: start -> end find all possibilities of shortest paths |

# Linked List

## Template:

### Find Middle:

if not head or not head.next:

return False

slow, fast = head, head.next

### Find Intersection/entrance:

if not head or not head.next:

return False

slow, fast = head, head

### Reverse Template:

def reverseList(self, head: ListNode) -> ListNode:

curr, prev = head, None

while curr:

nxt = curr.next

curr.next = prev

prev = curr

curr = nxt

return prev

|  |  |  |
| --- | --- | --- |
| Basic | | |
| [21](https://leetcode.com/problems/merge-two-sorted-lists/) | Merge two sorted lists |  |
| [206](https://leetcode.com/problems/reverse-linked-list/) | Reverse Linked List \* |  |
| [92](https://leetcode.com/problems/reverse-linked-list-ii/) | Reverse Linked List II |  |
| [24](https://leetcode.com/problems/swap-nodes-in-pairs/) | Swap Nodes in Pairs | Iteration + recursion |
| [L511](https://www.lintcode.com/problem/swap-two-nodes-in-linked-list/note/178277) | Swap two nodes in Linked List | Corner cases: v1 == v2; v1 and v2 neighbors |
| [143](https://leetcode.com/problems/reorder-list/) | Reorder List | Two pointers + reverse + connect  ! slow.next = None |
| [61](https://leetcode.com/problems/rotate-list/) | Rotate List | Count length and k % length |
| [86](https://leetcode.com/problems/partition-list/) | Partition List | Small head and small, large head and large |
|  | | |
| [328](https://leetcode.com/problems/odd-even-linked-list/) | Odd Even Linked List |  |
| [237](https://leetcode.com/problems/delete-node-in-a-linked-list/) | Delete Node in a Linked List \* |  |
| 369  [L904](https://www.lintcode.com/problem/plus-one-linked-list/note) | Plus One Linked List |  |
| [2](https://leetcode.com/problems/add-two-numbers/submissions/) | Add Two Numbers | carry + l1.val + l2.val one by one |
| [19](https://leetcode.com/problems/remove-nth-node-from-end-of-list/) | Remove Nth Node from end of List |  |
| [83](https://leetcode.com/problems/remove-duplicates-from-sorted-list/) | Remove Duplicates from Sorted List |  |
| [82](https://leetcode.com/problems/remove-duplicates-from-sorted-list-ii/) | Remove Duplicates from Sorted List II |  |
| [203](https://leetcode.com/problems/remove-linked-list-elements/) | Remove Linked List Elements |  |
| Advanced | | |
| [25](https://leetcode.com/problems/reverse-nodes-in-k-group/) | Reverse Nodes in K-Groups \*\* |  |
| [138](https://leetcode.com/problems/copy-list-with-random-pointer/) | Copy List with Random Pointer\*\* | Dictionary || in-place  7->13->11->10->1->None  7->7'->13->13'->11->11'->10->10'->1->1'->None |
| [141](https://leetcode.com/problems/linked-list-cycle/) | Linked List Cycle \*\* | Slow and fast pointer |
| [142](https://leetcode.com/problems/linked-list-cycle-ii/) | Linked List Cycle II \*\* | fast = a + 2b + c; slow = a + b; a == c  slow and head meet at the entrance node |
| [160](https://leetcode.com/problems/intersection-of-two-linked-lists/) | Intersection of Two Linked List \*\* | Corner case: no intersection – check p not p.next |
| [148](https://leetcode.com/problems/sort-list/) | Sort List \*\*\* |  |
| [109](https://leetcode.com/problems/convert-sorted-list-to-binary-search-tree/) | Convert Sorted List to Balanced BST \* |  |
| 426  [L1534](https://www.lintcode.com/problem/convert-binary-search-tree-to-sorted-doubly-linked-list/description) | Convert BST to doubly linked list |  |
|  | | |
| [234](https://leetcode.com/problems/palindrome-linked-list/) | Palindrome Linked List |  |
| [23](https://leetcode.com/problems/merge-k-sorted-lists/) | Merge K Sorted Lists | Similar 148 |
| [147](https://leetcode.com/problems/insertion-sort-list/) | Insertion Sort List |  |

# Array

## Types

### Subarray

* 令 PrefixSum[i] = A[0] + A[1] + ... A[i - 1], PrefixSum[0] = 0 易知构造 PrefixSum 耗费 O(n) 时间和 O(n) 空间。如需计算子数组从下标i到下标j之间的所有数之和 **Sum(i~j) = PrefixSum[j + 1] – PrefixSum[i]**
* **Sum(i~j) = PrefixSum[j] – PrefixSum[i - 1] if prefixSum[i] = A[0] +…A[i]**

### Sorted Array

* mergeSort, quicksort, heapsort

## Template:

## A screenshot of a social media post Description automatically generated

## Questions

|  |  |  |
| --- | --- | --- |
| Subarray | | |
| [53](https://leetcode.com/problems/maximum-subarray/) | Maximum Subarray | Current – minSum; maxCurrentIndex, max |
| [643](https://leetcode.com/problems/maximum-average-subarray-i/) | Maximum Average Subarray I |  |
| [L42](https://www.lintcode.com/problem/maximum-subarray-ii/description) | Maximum Subarray II | Maximum Subarray left + right  Related:   |  |  | | --- | --- | | [L43](https://www.lintcode.com/problem/maximum-subarray-iii/) | Maximum Subarray III – DP | |
| [L604](https://www.lintcode.com/problem/window-sum/) | Window Sum |  |
| [L138](https://www.lintcode.com/problem/subarray-sum/) | Subarray Sum |  |
| [L139](https://www.lintcode.com/problem/subarray-sum-closest/) | Subarray Sum Closest | Sort prefixSum array of tuple, sumSort = sorted(summ, key = lambda el:el[0]) |
| [560](https://leetcode.com/problems/subarray-sum-equals-k/) | Subarray Sum Equals K |  |
| [523](https://leetcode.com/problems/continuous-subarray-sum/) | Continuous Subarray Sum | Similar:   |  |  | | --- | --- | | [974](https://leetcode.com/problems/subarray-sums-divisible-by-k/) | Subarray Sums Divisible by K | |
| 206 | Minimum Size Subarray Sum | Subarray + two pointers |
| 152 | Maximum Product Subarray | Be careful of negative | basic max subarray methods, but cal num, num \* max and num\*min together. |
| Sorted Array, Partition | | |
|  | Merge Two Sorted Arrays | 小数归并到大数组里 |
|  | Merge Sorted Array | A = 1,2,3,empty, empty,B=4,5 |
|  | Intersection of Two Arrays | hash; merge |
|  | Median of Two Sorted Arrays\*\* | Related:  Kth largest element: quick select  Median: quick select, findK(nums, len(nums)/2) |
|  |  |  |
| Two pointers, Partition: quick sort 不是稳定排序，因为交换后不能保证相对位置 | | |
|  | Valid palindrome | 相向双指针 |
|  | Remove Duplicate Numbers in Array |  |
|  | Move Zeros | 同向双指针 |
|  | Rotate String | 三步反转法 |
|  | Partition Array |  |
|  | Kth Largest Element \*\* | Quick select |
|  | Partition Array by Odd and Even |  |
|  | Interleaving Positive and Negative Nums |  |
|  | Sort Letters by Case |  |
|  | Sort Colors |  |
|  | Sort Colors II | Rainbow sort  Related:  Pancake sorting: flip: each time largest of the rest pancake, and flip again to the bottom. |
| Two Sum | | |
|  | Two Sum \* | Hashmap; two pointers, sort |
|  | Two Sum – return 返回值 | Two pointers |
|  | Two Sum – unique pairs | Two pointers |
|  | 3Sum |  |
|  | Two Sum – less than or equal to target |  |
|  | Two Sum – greater to target |  |
|  | Two Sum – closest |  |
|  | 3Sum - closest |  |
|  | 4Sum |  |
|  | Two Sum – Difference equals to target |  |

A screenshot of a social media post

Description automatically generated

# Data Structure

## Types

* DFS: stack, BFS: level

## Questions

|  |  |  |
| --- | --- | --- |
| Stack | | |
|  | Expression Expand | recursion |
|  |  |  |
| 划分型 | | |
|  |  |  |
| 双序型 | | |
|  |  |  |
| 背包型 | | |
|  |  |  |
| 区间型 | | |
|  |  |  |

# DP

## Types

* 求最大值最小值
* 判断是否可行
* 统计方案个数（求所有方案，不能用dp）

## Template

### DP只能记录一种最优方案或者count所有方案

### 坐标型：

* State:
  + f[x]从起点走到坐标x
  + f[x][y]从起点走到坐标x,y
* function: 研究走到x,y这个点之前的一步
* initialize: 起点
  + 2D:初始化0th col& 0th row
* answer: 终点

## Questions

|  |  |  |
| --- | --- | --- |
| 坐标型dp -接龙型dp | | |
|  | Triangle \*\* |  |
|  | Minimum Path Sum |  |
|  | Unique Paths |  |
|  | Unique Paths II |  |
|  | Climbing Stairs |  |
|  | Jump Games | Greedy O(n), DP O(n2) |
|  | Jump Games II |  |
|  | Longest Increasing Subsequence \*\* |  |
|  | Perfect Square |  |
|  | Largest Divisible Subset |  |
|  | | |
|  |  |  |
|  | | |
|  |  |  |
|  | | |
|  |  |  |
|  | | |
|  |  |  |

|  |  |
| --- | --- |
| [377](https://leetcode.com/problems/combination-sum-iv/) | Combination Sum IV - DP |
|  |  |
| 132 | Palindrome Partitioning II -DP |