面试流程总结.

1. clarification
   - Edge cases
   - input type + input size and range
   - output type
   - time + Space complicity
   - $\boxed{\text{give examples}}$ 多举例子

Edge Cases:
① 负数, 0, 正数
② 数的 type
③ 极值
④ empty

2. Think out loud: speak whatever you are thinking
   实在 3~5分钟想不出来, 要hint

3. Think before coding:
   - write idea and steps: ifs else
   - Edge cases.
   - Run an example
   - ☆ 自己想 Test case. 问面试官对不对
   - ☆ 写注释 → 把所有步骤 用comment 写出来
     ☆ 先用 function名, 后看是否需要实现

4. Test in real time:
   - Edge cases
   - example
   - in-out memory
   - large input

5. Refactor the code:
   - Make it more efficient.
   - Fiequenty operation → make it a value

datastructure 底层实现
写伪 → one step by step
Communicate
code Imp
System design

Leet Code 总结 → Array

注: Peak → Binary Search

1. 数组有序？
   - yes: binary search：是否有相同值
   - no:
     - $O(n)$：双指针，Hash table，DP
     - $>O(n)$：先 sort，后用 binary search

2. 如果主体算法 $> O(n\log n)$，那么可以先 sort 来看是否会简单些

3. 乱序 + $O(n)$：Hash table，双指针
   乱序 + $O(n)$ + Space $O(1)$：双指针

4. Inplace 操作 $O(1)$：
   - 将 index 作为值；eg arr[index] = arr[arr[index]];
   - 利用 swap 来做
   - 用取负式将异值标记

5. 问 Min / Max 的种类数
   - DP
   - Greedy

6. 不要求 space 的时候：temp array 也许会大大简化过程

7. 双指针：可以相互追逐 或 两端同时扫
   - 追击
   - 相遇

8. 对于找规律题，一定要注意问题特征，一步一步发现线索

9. 组合问题：
   - Back tracking
   - Bit Minpulation

10. Range 问题
    - step = current + arr[current]
    - Union Find

11. 相同模式问题：Recursion

12. Carry 问题：stepResult = digit[i] + digit[j] + carry；
    - digit = stepResult % 10;
    - carry = stepResult / 10;

13. 数字种类有限：Counting Sort

14. 有时候差操作无需移动数组元素

15. Majority：Moor-Voting-Algo

DP

1. 字符串 dp：

Meaning: dp[i][j] text(0, i) pattern(0, j)

init : ..."" a b c

b

c

d.

function: 填 dp[][] 找规律.

Leet Code 总结 → Linked List

1. Fake Head
2. { fast / slow pointer
3. Reverse Linked List：当需要逆向走时，先reverse，多个reverse可将其变为双向链表.
4. 链表可以修改 { 值 / 结构 } 以达到目的
5. Linked List len { fast, slow 找中点，倒数点，one pass. / 先求长度，再操作 ⇒ two passes.
6. two or three pointers.
7. 做题时一定要画图.
8. 用 Hash Map 存节点，也许会简化操作
9. Insert Node

Leet Code 总结 → Tree

1. In order, pre-order, post-order Traversal

2. Level order → Queue + count

3. Tree Depth

4. 节点上有值的，传入 sum，使其为 sum - value 进行更新

5. Tree Node Insert

6. 不连 root，且除图的挑战

```
left = Math.max(0, helper(root.left));
right = Math.max(0, helper(root.right));
result = Math.max(result, left + right + root.val);
    return Math.max(left, right) + root.val;
```

7. BST + inorde = 有序数组

8. balanced BST = 取中点为 root
   - 中点.left = recursion( )
   - 中点.right = recursion( )

9. Lonest Common Ancestor of BST：是否在一个 subtree 中.

10. Leaf： root.left == null 且 root.right == null

11. Same Tree?

12. 从 n=1 → n=4 举例 ⇒ 找 Base case ⇒ 找递归模式 ⇒ 确定参数.

13. 序列化，重重 serialize

14. 将树 Hash 成一个值

```
int l = dfs(root.left);
int r = dfs(root.right);
int root = ((l^1253 ×31) + r^1235) ×31 + root.val;
return root ✓ ;
```

Array:

Edge cases?
正数, 0, 负数
① 重数, 小数
有相同值
有序
Time, Space
例子

Time →
- $O(n^2)$ → dp
- $O(n\log n)$ → { sort / n × binary search / n × Tree
- $O(n)$ → { 2 ptr. / hashtable / dP / stack
- $O(\log n)$ → { Binary Search / Tree Method

Space →
- $O(n)$ → { hash table / temp array / stack
- $O(1)$ → { 2 ptr / arr[arr[index]] / swap / 特殊值标记 / 取反 / 自身做自身的 temp array

有序?
- Y → { 局部有序 / 全有序 } → Binary Search
- N → sort $O(n\log n)$ → Binary Search
- N →
  - Hash Table
  - 2ptr
  - sliding window
  - DP
  - Greedy } min, max
  - Counting sort (有限数)
  - 取 max, min temp array
  - Stack → 记录之前状态
  - Heap → min, max

组合问题 { Backtracking / Bit Manipulation

0.1 knapsack ← 取 vs 不取 ← Range { interval / union find / step += current + arr[current]

区间问题

Recursive: 相对子问题有相同操作
找到规律 多举例子

Linked List

① Edge Cases?
head == null?
root.left == null?
root.right == null?
单向?
Time, Space
得好
One or two pass?

简化问题 → FakeHead
随时查 null

len? →
- 相对 → fast, low ptr
  先走 n, 再一起走
- 绝对 → One pass count len
- N → 结构 / 值
  结构 → reverse: 单向变双向
  HashMap 存节点

基本操作 →

② Edge Cases? Binary Tree →
- Y →
  - BST → left < root < right
    inorder → sorted
    leaf → root.left == null 或 root.right == null
  - !BST → In, pre, post
    level traversal
    Tree height
- N → 总想对 left 和 right 做相同操作
  
  + Memo
  序列化, Hash
  - Tree 的序列化

基本操作 →

Graph.

抽象：

① 将问题转化为边，点问题 →

- DFS：
- BFS：shortest path
- Topo sort：prerequisits
- Union find：sets merge

对上下左右做相同操作

param 和 return 的选择

Graph

1. DFS 模板，Basic Idea, param pass

```
public void dfs(int[][] map, int row, int col, int[][] visited)
{   if(row<0 || col<0 || row≥map.length || col≥rooms.length || visited[row][col] == true)
        return;

    visited[row][col] = true;
    dfs(row-1, col);
    dfs(row+1, col);
    dfs(row, col-1);
    dfs(row, col+1);
}.
```

2. Topological sort.
   ① 图 HashMap<Key, Set<child>> Graph.
   ② 入度 HashMap(Key, Integer>
   ③ BFS

3. Union find.
   ① Make Set : 自己指向自己.
   ② Union : path compression + always union with parent node
   ③ find : walk along parent pointer

Data structure:
```
int rank
int data
Node parent
```

4. BFS.