

面试流程总结

1. Clarification

Edge cases

input type + input size and range

output type

time + space complexity

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: if

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, 再看是否需要实现

4. Test in real time:

Edge cases

example

in-out memory

large input

5. Refactor the code:

Make it more efficient.

Frequently operation → make it a value

data structure 的实现

写代码 → one step by step

Communicate

code jump

System design

Leet Code 总结 \rightarrow Array

注: Peak \rightarrow Binary Search

- 数组有序? $\left\{ \begin{array}{l} \text{Yes: binary search: 是否有相同值} \\ \text{no: } \left\{ \begin{array}{l} O(n): \text{双指针, Hash table, DP} \\ >O(n): \text{先 sort, 后用 binary search} \end{array} \right. \end{array} \right.$
- 如果主体算法 $> O(n \log n)$, 那么可以先 sort 来看是否会简单些
- 乱序 $+ O(n)$: Hash table, 双指针
乱序 $+ O(n) + \text{Space } O(1)$: 双指针
- Inplace 操作: $\left\{ \begin{array}{l} \text{将 index 作为值: eg } arr[index] = arr[arr[index]]; \\ O(1) \quad \left\{ \begin{array}{l} \text{利用 swap 来做} \\ \text{用取负或特殊值标记} \end{array} \right. \end{array} \right.$
- 问 $\left\{ \begin{array}{l} \text{Min} \\ \text{Max} \end{array} \right.$ 的种数 $\left\{ \begin{array}{l} \text{DP} \\ \text{Greedy} \end{array} \right.$
- 不要求 space 的时候: temp array 也许会大大简化过程
- 双指针: 可以相互追逐 或 两端同时扫 $\left\{ \begin{array}{l} \text{追击} \\ \text{相遇} \end{array} \right.$
- 对于找规律题, 一定要注意问题特征, 一步步发现线索
- 组合问题: $\left\{ \begin{array}{l} \text{Back tracking} \\ \text{Bit Manipulation} \end{array} \right.$
- Range 问题 $\left\{ \begin{array}{l} \text{step} = \text{current} + arr[\text{current}] \\ \text{Union Find} \end{array} \right.$
- 相同模式问题: Recursion $\left\{ \begin{array}{l} \text{digit} = \text{stepResult} \% 10; \\ \text{Carry} = \text{stepResult} / 10; \end{array} \right.$
- Carry 问题: $\text{stepResult} = \text{digit}[i] + \text{digit}[j] + \text{Carry}$
- 数字种类有限: Counting Sort
- 有时倒差操作无需移动数组元素
- Majority: Moore-Voting-Algo

DP

1. 字符串 dp:

Meaning: $dp[i][j]$ $\xrightarrow{\text{text}(0,i)}$ $\xrightarrow{\text{pattern}(0,j)}$
init: ... " " a b c
 b
 2.
function: 填 $dp[i][j]$ 找规律.

leet Code 总结 \rightarrow 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, 例数点, one pass.
先求长度, 再操作 \Rightarrow two passes.

6. two or three pointers.

7. 做题时 - 定要画图.

8. 用 Hash Map 存节点 也许会简化操作

9. Insert Node

Leet Code 树的 → 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. 不传 root, 且递归的操作

```
{ 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. Lowest Common Ancestor of BST: 是否存在一个 subtree 中.

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

11. Same Tree?

12. 从 $n=1 \rightarrow n=4$ 举例 \Rightarrow 找 Base case \Rightarrow 找递归模式 \Rightarrow 确定参数.

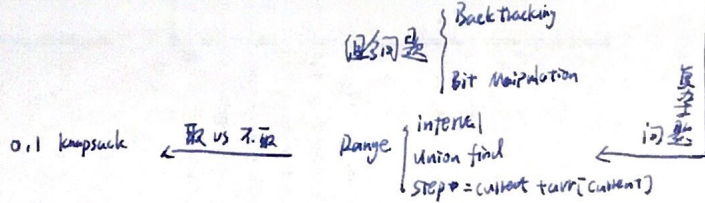
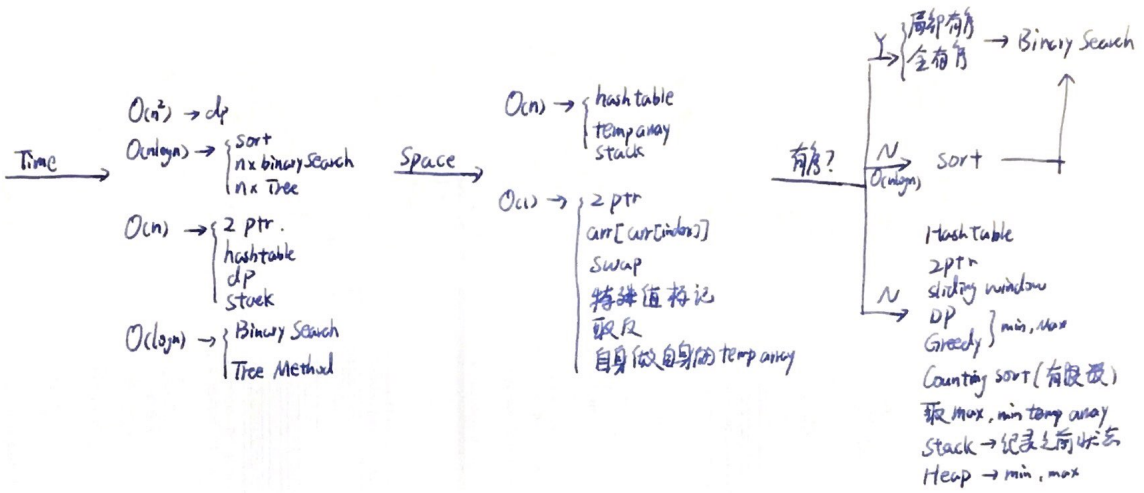
13. 序列化, 重建 serialize

14. 将树 Hash 成一个值

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

Array:

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

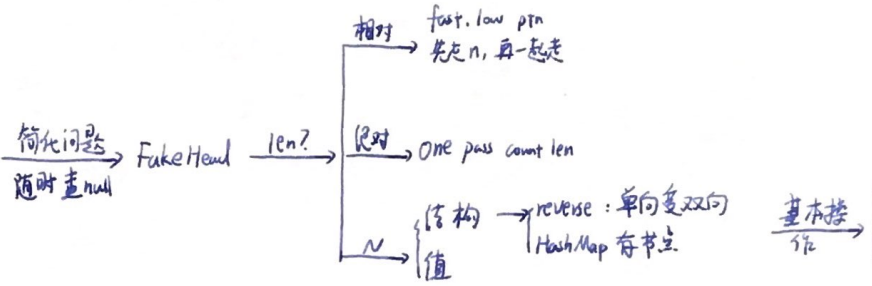


Recursive: 相对子问题有相同操作
找到规律 多举例子

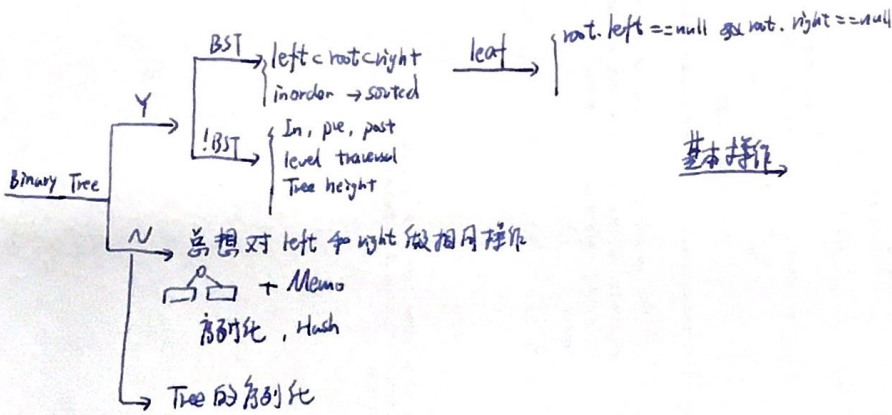
Linked List

Edge Cases?

- ①
 head == null?
 root.left == null?
 root.right == null?
 单向?
 Time, Space
 递归
 One or two pass?



② Edge Cases?



Graph.

① 抽象:
将问题转化为
边, 顶点
问题



DFS:

BFS: shortest path

Topo sort: prerequisites

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 >= map[0].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 $\text{HashMap} < \text{Key}, \text{Set} < \text{child} > > \text{Graph}$.
- ② 入度 $\text{HashMap} < \text{Key}, \text{Integer} >$
- 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.