

# 1 总体

- 态度：不卑不亢，有自信，不要说自己的发音不好，这题很困难等。
  - 面试官在接受面试时通常会有一种心态，即如果他/她可以与应聘者一起每天工作，那么请不要在面试时表现出傲慢或自我的迹象。。
- 发音：慢而清晰。只要放慢速度，不要急于求成。这也可以让你听起来更有信心。
- 书写：行距适中，头要对其，字要始终，工整，缩进4个空格，行距适中

## 2 BQ问题

- 背熟练，眼睛要对上，自然
- 简明回答，诱导对方进一步提问

## 3 确认题目要求

- 理解问题。
  - 请求画图等帮助理解
  - 按照我的理解说一遍，再次确认
- 确认数据范围
  - **Input:**
    - API是什么样的，需要自己写结构？
    - 确定数据的结构类型，是否“重，序”，取值范围
    - 链表有几个环等，可否改变node数值，二叉树值是什么类型的
  - **Output:** int? head?
  - **corner case:** (corner case包含edge case)
  - 横行 **corner case:** input output
  - 横行 '空1' xxx
- 明确：
  - 注重time complexity or space complexity?
  - 如果问题过于复杂，耗时，给出大概的时间，及是否能完成。问可否简化

## 4 写出数据结构和算法，解释

- 整体步骤：
  - 暴力+优化的层次分析
    - linklist: list法 + two pointer法等
- 具体步骤：
  - 提出可以尝试某一种idea (brute force or others), 说要思考，并解释简要原因
  - 写出idea及method(如有必要，请画图)
    - idea: A. brute force - list save / B. XX 算法
    - method:
      - 描述大概作用，流程简化书写，用scan而不是traversal, e.g. scan linklist, save node in list
      - 用case1 case2 case3表示各个条件情况
      - 解释的时候用例子更易懂
  - 给出space/time complexity
    - 主动给出
    - 对于dic等space comp, 要具体分析。由于题目条件，可能存在只有有限个(2个等)，故此时sO(2) = sO(1),而不是通常的O(N)
- 交流：
  - 注意聆听，如果你的面试官试图帮助你，不要错过任何提示！
  - 与面试环交流有歧义，要及时纠正，不要就坡下驴
- tips:
  - 先想清楚，再写
  - 注意编号的清晰
  - 如果发现corner case不完全，要写到2部分，不要在本部分插入

- 变量名用骆驼命名法 `mySolution`, 避免使用下划线 `new_head`

## 5 code之前:

- 'should I implement it in code? or you want me to continue to optimize it?'
- 'we have ... minutes. It seems that I can finish it in time.'

## 6 right coding

- 具体步骤:
  - `class` 命名
  - `function` 命名
    - 加功能注释及特殊输入格式的提醒, 不用说理由, 基础语法不用说
    - 注意要包含全部输入变量, 不要漏了
  - 写出框架, 再擦掉即可:
 

```
def func(self, input): #inp: XXX, outp: numb of string...
    res = 0
    方框
    return res
```
  - 书写代码, 并同时书写注释:
    - 如果是电面, 尽可能多的注释, 并说出来
    - 要严苛的写在行尾, 对齐, 不要写在行中间
    - 对复杂操作和无法一目了然的代码, 在行尾添加描述代码目的的注释
    - 不要描述代码, 比如 `traversal i from 1 to 19`
    - 针对'method'中列出的case, 针对性的注释 `case1/case2/case3` 即可
- 交流:
  - 简要交流目前在写什么, 不要不说话
- tips:
  - 只要写代码就行了, 不要问有无听懂。此处考察代码能力。
  - 变量名定义用骆驼命名法, 要有意义, 不要后补: `temp/ans/res`。不要用 `new`
  - 各部分功能清晰划分时, 最好用不同的函数分块去写。先写主函数, 再写分块函数
  - 发现method不完全, 需要添加内容时: 同时添加method和code
  - 求list中的长度, `j-i` 即可
  - 保证书写出代码的有效性, e.g. `a.next.next`, 要保证 `a.next` 的有效, 故在判断条件中要书写出来

## 7 跑test case

- 具体步骤:
  1. 跑corner case
  2. 跑面试官的case
    - 先验主函数, 再验证helper function
    - test要清晰的隔绝开, 不乱
  3. 再跑自己的case
- 格式: 前后风格格式要统一, 要易懂
- 题目类型test:
  - 链表: 画图指示箭头 + table
  - array: 分为两个部分, 总结果表示各个关键变量的值, 用table表示; 小结果写在code的右边, 即写即擦, 展现具体的计算过程
- Tips:
  - case 1-2-3-4-None, 前面空四格, 不要顶格写, 比如 `linklist` 加 `dummy` 和 变量名时就不好了

## 8 考官交流

- 问了必答, 比如在idea区域问space comp时, 也必须回答, 虽然不清楚具体的结果, 也要说我预计xxx, 下面可能会有改变
- 当你发现自己没有取得进展时, 说清楚你的问题, 考官会给与提示, 此时要做到:

- 及时跟进自己的想法，明白他说的细节
- 他说完后，要整理明白这段所有的思路，并叙述出来，与他交流。
- 永远不要说你做不到。即使存在您之前从未解决过的问题，或者似乎无法解决的问题，请从不同的角度来不断解决问题，面试官会给您提示。但是，如果您说无法解决问题，那将是一个很大的危险信号，并且您最终可能会被拒绝。
- **following up** 没时间写时，可以只讲思路，怎么做就行。等面试结束后，可以查面试官邮箱，把具体方法发到他邮箱里。
- 整个过程中，都要无拘束的问面试官问题，来保证完全理解了他们的问题！！
- 我们问问题时，也可以自由发问。内容可以包括：团队，文化等。这样利于判断这个工作是否适合我。

## 9 电面特殊之处

- 使用 Google Doc, 并建议使用扩音器或者耳机
- 大声说出自己的想法，吐字清晰（困难点），告诉 idea 和 method
- 能写多少写多少，先写函数头，然后写 **return**，再填充里面的代码 (**write what comes and but then refine it later**)
- **coding** 中，尽可能多写 **comments**，并口头解释
- 考虑生产就绪 (production ready) -> 很稳定，可维护的 (代码分功能写成不同的函数)，可扩展的 (用类)，有记录 (注释)
- 在电话面试中不太可能有系统设计问题，所以我的建议只是把它当作现场编程面试
- 找一个安静的环境