

# Google/Facebook 大公司高频题风格解析

课程不允许录像，否则将追究法律责任，赔偿损失



扫描二维码关注微信/微博  
获取最新面试题及权威解答

微信: [ninechapter](#)

知乎专栏: <http://zhuanlan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: [www.jiuzhang.com](http://www.jiuzhang.com)

- 高中信息学竞赛，湖北省队第一名，冬令营金牌，5年信息学竞赛教练
  - 本科保送浙大计院
  - 哈佛大学交流一年
  - 计算机转行金融（Quant）
- 
- 业余爱好：音乐剧，浙大灵韵音乐剧社创始人，常春藤盟校春晚导演

- ZOOM 的用法
- 老师/助教回答提问方式
- 课程Ladder: <http://www.lintcode.com/zh-cn/ladder/14/>
- 九章QA: <http://www.jiuzhang.com/qa/>
- 九章刷题小助手chrome插件: <http://www.jiuzhang.com/qa/6207/>
- 位运算入门教程: <http://www.jiuzhang.com/tutorial/bit-manipulation>
  - 访问密码: jiuzhang

- 高频题是啥？学了高频题可以上天吗？
- **FLAG** 高频题有什么特点——四道题目举例
- 高频题课会怎么上？

# 高频题是啥？学了高频题可以上天吗？

- 高频题是啥？
  - 不是难题，不是偏题，是面试出现频率很高的题
- 面试出现频率很高？
  1. 总结出公司出题风格
  2. 现在刷，面试时可能遇见原题



# FLAG 高频题有什么特点

## ——四道题目举例

# Facebook: Decode Ways

<http://www.lintcode.com/zh-cn/problem/decode-ways/>

<http://www.jiuzhang.com/solutions/decode-ways/>

## 面试小技巧 套原题



- 这一题类似哪道题呢？
  - 上楼梯问题 （动态规划基础题）
  - 上楼梯问题：  $n$  节楼梯从第0层开始，每次只能1层或2层，问上到第  $n$  层有多少种方法？

思路一：

- 枚举所有的走法？
  - 时间复杂度太高  $\approx O(2^{n/2}) \approx O(2^n)$

思路二：

- 考虑最后一步怎么走？（假设 $n=10$ ）
  - 最后一步一定是走1层 or 2层
  - 跳到第10层楼有且仅有这两类方法，并且两类方法不重合
  - 所以跳到第10层楼的方法数一定等于跳到第8层方法数+跳到第9层方法数



# FLAG 高频题有什么特点——四道题目举例

思路:

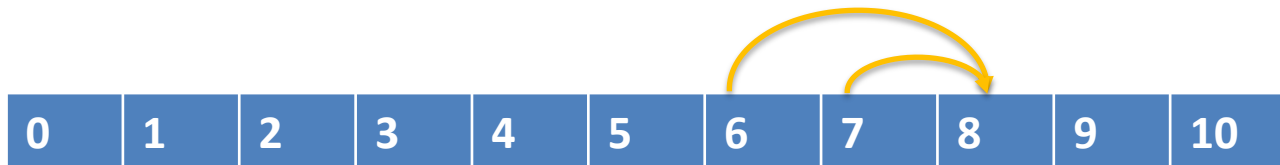
- 数学表示:  $f[n]$ 表示从第0层走到第n层一共有多少种方法
  - $f[10]=f[8]+f[9]$

要计算 $f[10]$ ,我们得先知道  
 $f[8]$ ,  $f[9]$ 等于多少



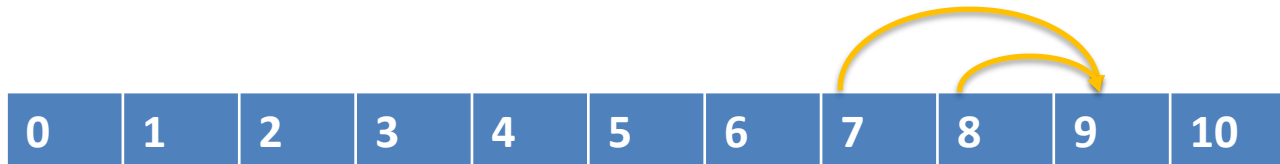
思路:

- 数学表示:  $f[n]$ 表示从第0层走到第n层一共有多少种方法
  - $f[10]=f[8]+f[9]$
  - $f[8] = f[6]+f[7]$



思路:

- 数学表示:  $f[n]$  表示从第0层走到第n层一共有多少种方法
  - $f[10] = f[8] + f[9]$
  - $f[8] = f[6] + f[7]$
  - $f[9] = f[7] + f[8]$



# FLAG 高频题有什么特点——四道题目举例

思路:

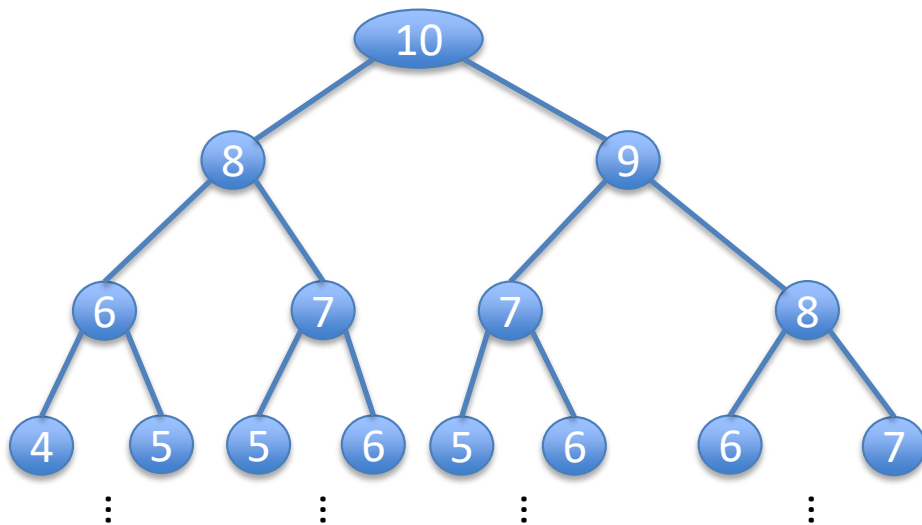
- 数学表示:  $f[n]$  表示从第0层走到第n层一共有多少种方法
  - $f[n] = f[n-1] + f[n-2]$



# FLAG 高频题有什么特点——四道题目举例

思路:

- 数学表示:  $f[n]$  表示从第0层走到第n层一共有多少种方法
  - $f[n] = f[n-1] + f[n-2]$



# FLAG 高频题有什么特点——四道题目举例

思路:

- 如何快速的计算 $f[10]$ ?
  - $f[n] = f[n-1] + f[n-2]$   $f[n]$ 只和比 $n$ 小的状态有关
  - 按照 $f[1]$   $f[2]$   $f[3]$  ....  $f[10]$ 的顺序依次计算

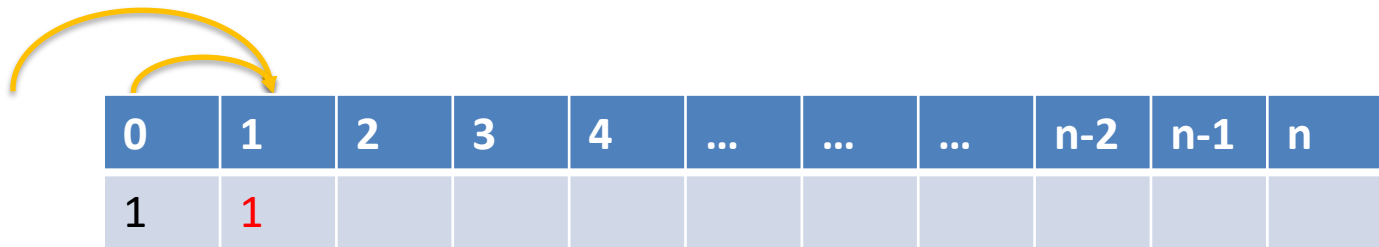




# FLAG 高频题有什么特点——四道题目举例

思路:

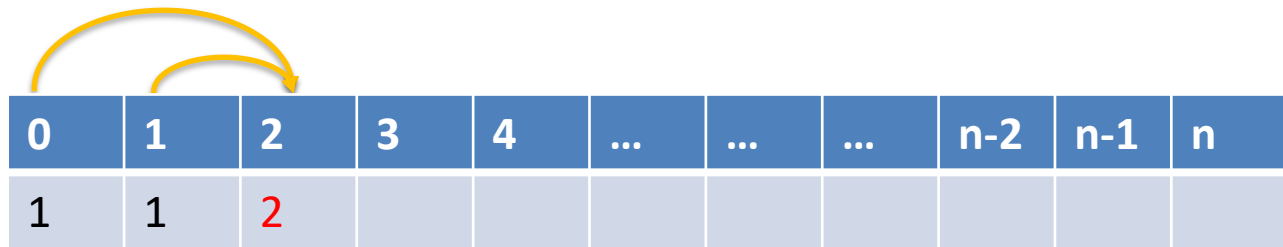
- 如何快速的计算 $f[10]$ ?
  - $f[n] = f[n-1] + f[n-2]$   $f[n]$ 只和比 $n$ 小的状态有关
  - 按照 $f[1]$   $f[2]$   $f[3]$  ....  $f[10]$ 的顺序依次计算



# FLAG 高频题有什么特点——四道题目举例

思路:

- 如何快速的计算 $f[10]$ ?
  - $f[n] = f[n-1] + f[n-2]$   $f[n]$ 只和比 $n$ 小的状态有关
  - 按照 $f[1]$   $f[2]$   $f[3]$  ....  $f[10]$ 的顺序依次计算




0	1	2	3	4	...	...	...	n-2	n-1	n
1	1	2								

# FLAG 高频题有什么特点——四道题目举例

思路:

- 如何快速的计算 $f[10]$ ?
  - $f[n] = f[n-1] + f[n-2]$   $f[n]$ 只和比 $n$ 小的状态有关
  - 按照 $f[1]$   $f[2]$   $f[3]$  ....  $f[10]$ 的顺序依次计算

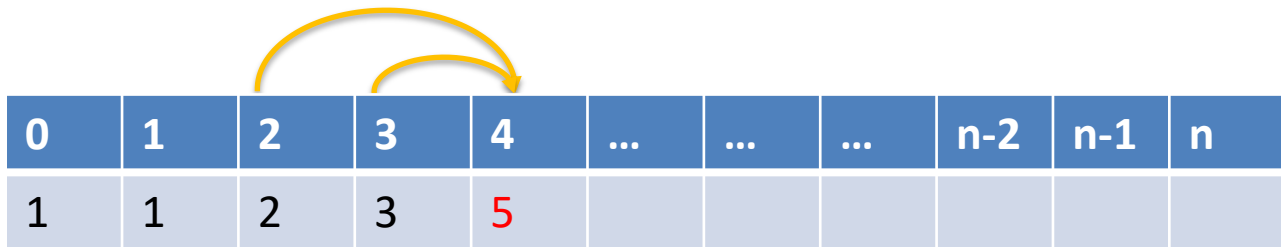


0	1	2	3	4	...	...	...	n-2	n-1	n
1	1	2	3							

# FLAG 高频题有什么特点——四道题目举例

思路:

- 如何快速的计算 $f[10]$ ?
  - $f[n] = f[n-1] + f[n-2]$   $f[n]$ 只和比 $n$ 小的状态有关
  - 按照 $f[1]$   $f[2]$   $f[3]$  ....  $f[10]$ 的顺序依次计算



0	1	2	3	4	...	...	...	n-2	n-1	n
1	1	2	3	5						

总体时间复杂度 $O(n)$

## Decode way 解法:

- 考虑最后一位怎么分解?
  - 最后1个数字当作一个字符or 最后2个数字当作一个字符
  - 类似上楼梯跳1步 or 跳2步

s="7452.....519"

idx	0	1	2	3	4	...	...	...	n-2	n-1	n
s		7	4	5	2	...	...	...	5	1	9

9 → 'i'

OR

19 → 's'

Decode way 解法:

- $f[n]$  代表  $s[1..n]$  有多少种分解办法 (s从1开始计数)

$s = "7452.....519"$

idx	0	1	2	3	4	...	...	...	n-2	n-1	n
s		7	4	5	2	...	...	...	5	1	9


9 → 'i'

OR

19 → 's'

Decode way 解法:

- $f[n] = f[n-1]$  (条件  $s[n] \neq 0$ ) +  $f[n-2]$  (条件是  $s[n-1]$  与  $s[n]$  组成的数字在 10-26 之间)



idx	0	1	2	3	4	...	...	...	n-2	n-1	n
s		7	4	5	2	...	...	...	5	1	9

9 → 'i'

OR

19 → 's'

## FLAG 高频题有什么特点——四道题目举例

- $f[n] = f[n-1](\text{条件 } s[n] \neq 0) + f[n-2](\text{条件是 } s[n-1] \text{ 与 } s[n] \text{ 组成的数字在 } 10-26 \text{ 之间})$
- 数据模拟:
  - $s = \text{"7452310519"}$

idx	0	1	2	3	4	5	6	7	8	9	10
s		7	4	5	2	3	1	0	5	1	9
f	1	1	1	1	1	2	2	2	2	2	4



# FLAG 高频题有什么特点——四道题目举例

## ◆ Facebook特点:

- OJ (Online Judge)上的原题    DP (Dynamic Programming 动态规划)
- E.g.: [Wildcard Matching](#)    [Regular Expression Matching](#)
- <http://www.lintcode.com/en/tag/facebook/>

额外扩展:

- 怎样识别动态规划类题?
  - 最优化, 问方法数, 不需要打印所有路径的大概率是
  - 看下数据范围
- 动态规划的解题步骤:
  - 0.考虑最后一步 (怎么分解, 怎么走)**      直觉、解题灵感来源
  - 1.定状态**
  - 2.写转移方程**
  - 3.初始条件、边界情况**

# LinkedIn: Strings Homomorphism

<http://www.lintcode.com/zh-cn/problem/strings-homomorphism/>

<http://www.jiuzhang.com/solutions/strings-homomorphism/>

# FLAG 高频题有什么特点——四道题目举例

题意：

- 验证相同位置上的两个字符是否是一一对应的

算法：

- 从左往右映射一遍，依次记录下映射关系，发现矛盾就false（比如a映射到c后 a又映射到b）
- 从右往左再做一遍

数据模拟：

- abad fekg 不是
- abad fefg 是

- 看到这里时，大家会发现这是一道**简单题**，算法都清楚了我们下一题吧 —— 不！我还要继续讲！
- 讲什么？——面试遇到简单题时该怎么玩？
  - 简单题时**TODO**: 怎么样简单的实现代码
  - 花式秀代码时间

注意：我们这里的“花式”代码是为了让程序更简洁、更易读、更容易 **bug free**，不是单纯的为了花式而花式

- 怎么表示“a 映射到f”？
  - Hash     `map['a'] = 'f'`
- map用什么数据类型？（3种实现Hash的方式）
  - Java: `HashMap`   C++: `unordered_map`   Python: `{}`   `dictionary`
  - 数组 `int[] map = new int[256];`   （当标号是数字或者类数字时）
  - 数组 `int[] map = new int[26];`   （扩展方法）

## ◆ 小技巧总结:

- Hash的3种方法（原始1种，特殊情况时数组2种）
  - 原始 Java: Hashmap C++: unordered\_map Python: {} dictionary
  - 特殊情况：如果被hash的对象是数字/类数字，如'a' 'g' 'Z'，Java / C++ 的实现可以使用数组（Python还是用{} 因为dictionary已经很简单了）
    1. 数组大小可以定义为256 （ASCII码最大值）
    2. 也可以定为被Hash对象的范围 （如 'a' - 'z' 就定义26）

数组实现优势：代码短，key可以方便的顺序访问

劣势：只能在被Hash的对象是数字或者字符时使用，且受内存大小限制

## ◆ LinkedIn 特点:

- 基础算法、数据结构的快速实现
- 难度适中
- <http://www.lintcode.com/en/tag/linkedin/>



- 课程定位：
  - 短期冲刺班
  - 一定的算法基础后，了解但不太熟练 → 熟练（+ 一点运气）
- 刚刚的两道题能在10分钟内写完一次AC（Accept 数据测试全过）吗？
- 明明做过的题第二遍就是写不对？
- 课后亲手把题写一遍
- 基础算法要写不只一遍（为什么？）
- 所以学了高频题真的可以上天吗？

# Amazon: Rectangle Overlap

<http://www.lintcode.com/zh-cn/problem/rectangle-overlap/>

<http://www.jiuzhang.com/solutions/rectangle-overlap/>

思路:

- 最基础的想法分类讨论
- 假设一个固定，另一个从左往右移，那么在一个维度上分别是：
- 不重叠 重叠 重叠 重叠 不重叠
- 两个维度上都重叠矩形才重叠

## FLAG 高频题有什么特点——四道题目举例

---

- 正着想麻烦就反着来
- 考虑下不重叠的时候是什么情况？
- 要么上下、要么左右

面试套路  
就是原题

# FLAG 高频题有什么特点——四道题目举例



## ◆ Amazon特点:

- 题库比较固定 OA（Online Assessment）老9题等
- <http://www.lintcode.com/en/tag/amazon/>

# FLAG 高频题有什么特点——四道题目举例



## OA9题概要

<a href="#">Rectangle Overlap</a>	<a href="#">Solution</a>	基础题
<a href="#">K Closest Points</a>	<a href="#">Solution</a>	堆
<a href="#">Window Sum</a>	<a href="#">Solution</a>	Sliding window类
<a href="#">Longest Palindrome</a>	<a href="#">Solution</a>	模拟
<a href="#">Copy List with Random Pointer</a>	<a href="#">Solution</a>	链表操作
<a href="#">Course Schedule II</a>	<a href="#">Solution</a>	拓扑排序
<a href="#">Minimum Spanning Tree</a>	<a href="#">Solution</a>	基础最小生成树
<a href="#">High Five</a>	<a href="#">Solution</a>	Hash+堆
<a href="#">Maximum Subtree</a>	<a href="#">Solution</a>	树上DFS

# FLAG 高频题有什么特点——四道题目举例

★ 0 - Amazon OA High Frequent 9 Problems		☰ 必做 (1/1)	☕ 选做 (8/8)	🔗 Related (0/0)
容易	627. 最长回文串 ☆		☑️	25 %
容易	628. 最大子树 ☆		☑️	43 %
容易	604. 滑动窗口内数的和 ☆		☑️	33 %
中等	616. 安排课程 ☆		☑️	22 %
中等	613. 优秀成绩 ☆		☑️	30 %
中等	612. K个最近的点 ☆		☑️	20 %
中等	105. 复制带随机指针的链表 ☆		☑️	33 %
困难	629. 最小生成树 ☆		☑️	26 %



# Google: Check Word Abbreviation

<http://www.lintcode.com/zh-cn/problem/check-word-abbreviation/>

<http://www.jiuzhang.com/solutions/check-word-abbreviation/>

字符串收缩操作：

apple



a 3 e

apple



a 2 le

apple



4 e

思路:

- 直接模拟
- `s = "internationalization"`
- `t = "i12iz4n"`
- 一个串上一个指针 见代码
- 麻烦点: 逻辑小细节处理
- 怎么办: 写一遍才有体会 `coding time`

## 面试小技巧

遇到麻烦题不要怕，现场debug也加分

## Google: Words Abbreviation

<http://www.lintcode.com/zh-cn/problem/words-abbreviation/>

<http://www.jiuzhang.com/solutions/words-abbreviation/>

## FLAG 高频题有什么特点——四道题目举例

like	god	internal	me	internet	interval	intension	face	intrusion
l2 e	god	i6l	me	i6t	i6l	i7n	f2e	i7n
		in5l			in5l	in6n		in6n
		int4l			int4l	int5n		int5n
		inte3l			inte3l	inte4n		intr4n
		inter2l			inter2l			
		internal			interval			

l2e god internal me i6t interval inte4n f2e intr4n

# FLAG 高频题有什么特点——四道题目举例

思路:

- 直接模拟
- 求出abbr 有重复就增加prefix 继续求 abbr
- 怎么判断重复 Hash桶计数

## ◆ Google特点:

- 网上原题少 原题变种 实现麻烦 代码功底
- <http://www.lintcode.com/en/tag/google/>



- Facebook: OJ (Online Judge)上的原题    DP (Dynamic Programming)
- LinkedIn: 基础算法、数据结构的快速实现、难度适中
- Amazon: 题库比较固定、OA老9题等
- Google: 网上原题少、原题变种、实现麻烦、代码功底

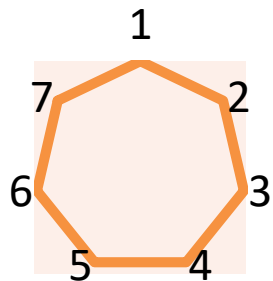
- 面试考察能力7维度：
  1. 理解问题
  2. 代码基础功力
  3. 基础数据结构/算法
  4. 逻辑思维/算法优化能力
  5. 细节处理（**corner case**）
  6. 算法分析（时间/空间复杂度）
  7. **debug**能力

- Strings Homomorphism
  1. 理解问题
  3. 基础数据结构/算法
- Decode Ways
  4. 逻辑思维/算法优化能力
  5. 细节处理 (corner case)
  7. debug能力

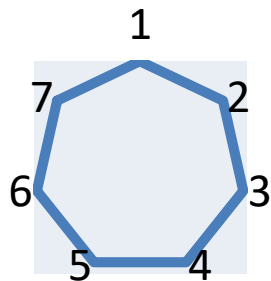
- Rectangle Overlap
  1. 理解问题
  4. 逻辑思维/算法优化能力
- Check Word Abbreviation
  2. 代码基础功力
  5. 细节处理 (corner case)
  7. debug能力
- Words Abbreviation
  1. 理解问题
  2. 代码基础功力
  3. 基础数据结构/算法
  7. debug能力

# FLAG 高频题有什么特点——四道题目举例

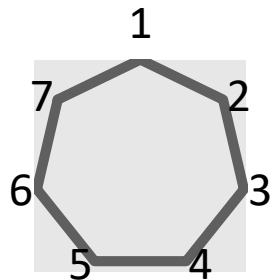
LinkedIn



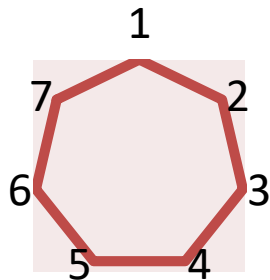
Facebook



Amazon



Google



# 高频题课会怎么上？

## 高频题课大纲

<http://www.jiuzhang.com/course/9/>

## 高频题课大纲：

第一节 Google/Facebook 大公司高频题风格解析

第二节 模拟算法和字符串处理技巧

第三节 基础算法和数据结构高频题 I

第四节 基础算法和数据结构高频题 II

第五节 搜索类题目如何高效实现

第六节 数学、计算几何、位运算常见问题详解



课程时间：

- 美国时间每周五六
  - 美西 18:30-20:30
  - 美东 21:30-23:30
- 北京时间每周六日
  - 早上09:30 – 11:30
- 请在九章官网“我的课程”中查看具体每节课的时间安排

## 课程特色：

- 九章老师亲自总结各公司的高频题list
- **FLAG**个公司面试风格解剖
- 快速突破经典面试题
- 短期快速补充各种算法小知识点

## 常见问题 Q & A:

### 1. 什么人适合上这门课？

有一定的语言基础（C++/Java/Python），  
但没有算法基础，或算法能力薄弱

想听**FLAG**各个公司的面试技巧，面试偏好，已有高频题总结  
希望短期高效有重点的准备大公司面试题

### 2. 课程目标是什么？

更加详细的分析**FLAG**各个公司面试风格，快速突破经典题目

## 常见问题 Q & A:

### 3. 有什么福利？

FLAG 常考题目，常考题目类型分布：

<http://www.lintcode.com/en/tag/linkedin/>

<http://www.lintcode.com/en/tag/facebook/>

<http://www.lintcode.com/en/tag/amazon/>

<http://www.lintcode.com/en/tag/google/>

## 常见问题 Q & A:

### 4. 上这门课有先修课程吗？

需要语言基础，对算法有基础了解即可（建议有30+以上的题量）

### 5. 高频题班 和九章算法、强化班之间是什么关系？

定位不同、互为补充，《九章算法班》《九章算法强化班》是从知识点的梳理和讲解出发，《高频题班》是从热点题目出发，是两个不同的课程体系，两门课程题目重叠度非常低，仅有2题左右相同

<http://www.jiuzhang.com/course/5/>

## 常见问题 Q & A:

### 6. 自己刷题不上课可以吗？

上课更高效、节省时间，和大家一起有学习的感觉和动力，更有独门的代码经验、小技巧的分享

### 7. 高频题班的高频题是如何统计的？

高频题由最新面经+OJ中的题目经过分类统计、结合老师经验归纳整理筛选而来，最近面试中最常见、最热门的题目都会在高频题班中出现

### 8. 这门课大概会有多少高频题？

总共50多道题目

## 常见问题 Q & A:

### 9. 课程题目难度如何？

题目难度适中，不会特别难，但是面试常考题，与今天的四道题类似



扫描二维码关注微信/微博  
获取最新面试题及权威解答

微信: [ninechapter](#)

知乎专栏: <http://zhuankan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: [www.jiuzhang.com](http://www.jiuzhang.com)