

< / > 在这里 / 有面试笔试常见技巧的提炼与总结
< / > 在这里 / 有面试笔试高频知识点的整理与剖析
< / > 在这里 / 有面试笔试历年真题的解答与拓展

PROGRAMMER

> Interview and written examination

程序员

面试笔试宝典

第③版

何昊 叶向阳 窦浩 / 等编著



本书覆盖了近**3年**程序员面试笔试中超过**98%**的高频知识点

当你细细品读完本书后，各类企业的offer将任由你挑选

一 书 在 手 / 工 作 不 愁 > .

 **机械工业出版社**
CHINA MACHINE PRESS



程序员面试笔试宝典

第3版

何 昊 叶向阳 窦 浩 等编著

机械工业出版社

本书针对当前各大 IT 企业面试笔试中常见的问题以及注意事项，进行了深层次地分析。本书除了对传统的计算机相关知识（C/C++、数据结构与算法、操作系统等）进行介绍外，还根据当前计算机技术的发展潮流，对面试笔试中常见的海量数据处理进行了详细地分析。同时，为了更具说服力，本书对面试过程中求职者存在的问题进行了深度剖析，总结提炼了大量程序员面试笔试技巧，这些技巧生动形象，对求职者能够起到一定的指引作用。本书也从历年的程序员面试笔试真题中精挑细选多套完整试题，以供读者学完本书后检测自我能力，通过这些试卷与讲解，能够帮助求职者有针对性地进行求职准备。

本书是一本计算机相关专业毕业生面试笔试的求职用书，同时也适合期望在计算机软硬件行业大显身手的计算机爱好者阅读。

图书在版编目（CIP）数据

程序员面试笔试宝典 / 何昊等编著. —3 版. —北京：机械工业出版社，2018.4
ISBN 978-7-111-59889-3

I. ①程… II. ①何… ②叶… ③窦… III. ①程序设计—资格考试—自学参考资料
IV. ①TP311.1

中国版本图书馆 CIP 数据核字（2018）第 092924 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：时 静 责任编辑：时 静

责任校对：张艳霞 责任印制：张 博

三河市宏达印刷有限公司印刷

2018 年 6 月第 3 版·第 1 次印刷

184mm×260mm·22.5 印张·538 千字

0001—3000 册

标准书号：ISBN 978-7-111-59889-3

定价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

网络服务

服务咨询热线：（010）88361066

机工官网：www.cmpbook.com

读者购书热线：（010）68326294

机工官博：weibo.com/cmp1952

（010）88379203

教育服务网：www.cmpedu.com

封面无防伪标均为盗版

金 书 网：www.golden-book.com

前言

自本书第1版、第2版发行以来，在读者群中产生了强烈反响，被广大读者奉为求职必备之宝典，获取工作之利器。图书的畅销并没有让我们作者团队产生丝毫懈怠，我们也并没有因此而沾沾自喜，反而是如履薄冰。使用本书的读者多了，自然而然地我们肩上的责任与担当就更重了，如果我们的图书中有任何错误或者是无法完全满足当前求职者的需求，我们必然会被读者唾弃、抛弃，所以，在图书出版后，我们并没有停止前进的步伐，我们一直在思考，如何才能让图书与时俱进，让读者看完该书后能够尽可能好地找到自己满意的工作。

《程序员面试笔试宝典 第3版》在保留第1版、第2版原有精华内容的基础上，考虑到近两年IT行业的背景以及程序员的求职情况，进行了以下几方面的工作：

(1) 结合当下的企业招聘侧重点，去掉了第1版、第2版中部分不常出现在程序员面试笔试中的相关内容：智力题、英语面试攻略、软件工程等内容，新增面试笔试经验技巧，更加突出重点，从而节省了读者的大量时间，保证每一部分内容都是重点、难点，提高阅读效率。

(2) 以附录的形式新增近两年各大IT名企的面试笔试真题两套，同时给出答案，使得读者能够及时了解当前企业招聘的重点、难点，把握复习方向，提高求职的成功率。

(3) 结合近两年以来程序员求职面试笔试真题的一些变化，本书在讲解知识点的同时，引入了更多的真题，并给出了解答，力求做到真题全覆盖。

希望本书能够继续为求职者提供必要的帮助，希望每个阅读过本书内容的人都能获得一份理想的工作。

有求职困惑的程序员或是对本书内容存在疑惑的读者都可以通过yuancoder@foxmail.com联系作者。

编者

目 录

前言

上篇：面试笔试经验技巧篇

经验技巧 1	如何巧妙地回答面试官的问题	2
经验技巧 2	如何回答技术性的问题	3
经验技巧 3	如何回答非技术性问题	5
经验技巧 4	如何回答快速估算类问题	5
经验技巧 5	如何回答算法设计问题	6
经验技巧 6	如何回答系统设计题	9
经验技巧 7	如何解决求职中的时间冲突问题	11
经验技巧 8	如果面试问题曾经遇见过，是否要告知面试官	12
经验技巧 9	在被企业拒绝后是否可以再申请	13
经验技巧 10	如何应对自己不会回答的问题	13
经验技巧 11	如何应对面试官的“激将法”语言	14
经验技巧 12	如何处理与面试官持不同观点这个问题	15
经验技巧 13	什么是职场暗语	15

下篇：面试笔试技术攻克篇

第 1 章	程序设计基础	21
1.1	C/C++关键字	21
1.1.1	static（静态）变量有什么作用	21
1.1.2	const 有哪些作用	23
1.1.3	switch 语句中的 case 结尾是否必须添加 break 语句	28
1.1.4	volatile 在程序设计中有什么作用	29
1.1.5	断言 ASSERT()是什么	31
1.1.6	枚举变量的值如何计算	32
1.1.7	char str1[] = "abc"; char str2[] = "abc"; str1 与 str2 不相等，为什么	32
1.1.8	为什么有时候 main()函数会带参数？参数 argc 与 argv 的含义是什么	34
1.1.9	C++里面是不是所有的动作都是 main()函数引起的	35
1.1.10	*p++与(*p)++等价吗？为什么	36
1.1.11	前置运算与后置运算有什么区别	36

1.1.12	a 是变量, 执行(a++) += a 语句是否合法	38
1.1.13	如何进行 float、bool、int、指针变量与“零值”的比较	38
1.1.14	new/delete 与 malloc/free 的区别是什么	40
1.1.15	什么时候需要将引用作为返回值	42
1.1.16	变量名为 618Software 是否合法	43
1.1.17	C 语言中, 整型变量 x 小于 0, 是否可知 x×2 也小于 0	43
1.1.18	exit(status)是否与从 main()函数返回的 status 等价	43
1.1.19	已知 String 类定义, 如何实现其函数体	44
1.1.20	在 C++语言中如何实现模板函数的外部调用	46
1.1.21	在 C++语言中, 关键字 explicit 有什么作用	47
1.1.22	C++中异常的处理方法以及使用了哪些关键字	48
1.1.23	如何定义和实现一个类的成员函数为回调函数	49
1.2	内存分配	50
1.2.1	内存分配的形式有哪些	50
1.2.2	什么是内存泄漏	52
1.2.3	栈空间的最大值是多少	61
1.2.4	什么是缓冲区溢出	62
1.3	sizeof	64
1.3.1	strlen("\0")=? sizeof("\0")=?	64
1.3.2	对于结构体而言, 为什么 sizeof 返回的值一般大于期望值	65
1.3.3	指针进行强制类型转换后与地址进行加法运算, 结果是什么	67
1.4	指针	67
1.4.1	使用指针有哪些好处	68
1.4.2	引用还是指针	68
1.4.3	指针和数组是否表示同一概念	69
1.4.4	指针是否可进行>、<、>=、<=、==运算	70
1.4.5	指针与数字相加的结果是什么	70
1.4.6	野指针? 空指针	71
1.5	预处理	72
1.5.1	C/C++头文件中的 ifndef/define/endif 的作用有哪些	72
1.5.2	#include <filename.h>和#include "filename.h" 有什么区别	73
1.5.3	#define 有哪些缺陷	74
1.5.4	如何使用 define 声明一个常数, 用以表明 1 年中有多少秒 (忽略闰年问题)	74
1.5.5	含参数的宏与函数有什么区别	75
1.5.6	宏定义平方运算#define SQR(X) X*X 是否正确	75
1.5.7	不能使用大于、小于、if 语句, 如何定义一个宏来比较两个整数 a、b 的大小	76
1.5.8	如何判断一个变量是有符号数还是无符号数	77
1.5.9	#define TRACE(S) (printf("%s\n", #S), S)是什么意思	79
1.5.10	不使用 sizeof, 如何求 int 占用的字节数	80

1.5.11	如何使用宏求结构体的内存偏移地址	81
1.5.12	如何用 sizeof 判断数组中有多少个元素	82
1.5.13	枚举和 define 有什么不同	82
1.5.14	typedef 和 define 有什么区别	83
1.5.15	C++中宏定义与内联函数有什么区别	84
1.5.16	定义常量谁更好? #define 还是 const	85
1.6	结构体与类	85
1.6.1	C 语言中 struct 与 union 的区别是什么	86
1.6.2	C 和 C++中 struct 的区别是什么	87
1.6.3	C++中 struct 与 class 的区别是什么	87
1.7	位操作	88
1.7.1	一些结构声明中的冒号和数字是什么意思	88
1.7.2	最有效的计算 2 乘以 8 的方法是什么	89
1.7.3	如何使用位操作求两个数的平均值	89
1.7.4	如何求解整型数的二进制表示中 1 的个数	91
1.7.5	不能用 sizeof()函数, 如何判断操作系统是 16 位, 还是 32 位	92
1.7.6	嵌入式编程中, 什么是大端? 什么是小端	93
1.7.7	考虑 n 位二进制数, 有多少个数中不存在两个相邻的 1	96
1.7.8	不用除法操作符如何实现两个正整数的除法	97
1.8	函数	101
1.8.1	怎么样写一个接受可变参数的函数	102
1.8.2	函数指针与指针函数有什么区别	102
1.8.3	C++函数传递参数的方式有哪些	108
1.8.4	重载与覆盖有什么区别	110
1.8.5	无参数构造函数是否可以调用单参数构造函数	114
1.8.6	C++中函数调用有哪几种方式	115
1.8.7	什么是可重入函数? C 语言中如何写可重入函数	116
1.9	数组	117
1.9.1	int a[2][2]={ {1},{2,3} }, 则 a[0][1]的值是多少	117
1.9.2	如何合法表示二维数组	118
1.9.3	a 是数组, (int*)(&a+1)表示什么意思	118
1.9.4	不使用流程控制语句, 如何打印出 1~1000 的整数	119
1.9.5	char str[1024];?scanf("%s",str)是否安全	122
1.9.6	行存储与列存储中哪种存储效率高	122
1.10	变量	123
1.10.1	全局变量和静态变量有什么异同	123
1.10.2	局部变量需要“避讳”全局变量吗	124
1.10.3	如何建立和理解非常复杂的声明	125
1.10.4	变量定义与变量声明有什么区别	126

1.10.5	不使用第三方变量，如何交换两个变量的值	127
1.10.6	C 与 C++变量初始化有什么不同	128
1.10.7	类型转换	128
1.11	字符串	130
1.11.1	如何实现 memmove	130
1.11.2	不使用 C/C++字符串库函数，如何自行编写 strcpy()函数	133
1.11.3	如何把数字转换成字符串	135
1.12	编译	137
1.12.1	编译和链接的区别是什么	137
1.12.2	编译型语言与解释型语言的区别是什么	138
1.12.3	如何判断一段程序是由 C 编译程序，还是由 C++编译程序编译的	139
1.12.4	C++程序中调用被 C 编译器编译后的函数，为什么要加 extern “C”	139
1.12.5	两段代码共存于一个文件，编译时有选择地编译其中的一部分，如何实现	140
1.13	面向对象相关	140
1.13.1	面向对象与面向过程有什么区别	140
1.13.2	面向对象的基本特征有哪些	142
1.13.3	什么是深拷贝？什么是浅拷贝	142
1.13.4	什么是友元	144
1.13.5	拷贝构造函数与赋值运算符的区别	146
1.13.6	基类的构造函数/析构函数是否能被派生类继承	148
1.13.7	初始化列表和构造函数初始化的区别	148
1.13.8	C++中有哪些情况只能用初始化列表，而不能用赋值	150
1.13.9	类的成员变量的初始化顺序	151
1.13.10	当一个类为另一个类的成员变量时，如何对其进行初始化	152
1.13.11	C++能设计实现一个不能被继承的类吗	152
1.13.12	构造函数没有返回值，那么如何得知对象是否构造成功	154
1.13.13	public 继承、protected 继承、private 继承的区别	155
1.13.14	C++提供默认参数的函数吗	156
1.13.15	如何解决多重继承中存在的钻石问题	158
1.14	虚函数	159
1.14.1	什么是虚函数	159
1.14.2	C++如何实现多态	163
1.14.3	纯虚函数指的是什么	164
1.14.4	什么函数不能声明为虚函数	164
1.14.5	C++中如何阻止一个类被实例化	167
1.15	编程技巧	168
1.15.1	当 while()的循环条件是赋值语句时会出现什么情况	168
1.15.2	不使用 if/?:/switch 及其他判断语句如何找出两个 int 型变量中的最大值和最小值	168
1.15.3	C 语言获取文件名的宏定义是什么	169

1.15.4	表达式 $a > b > c$ 是什么意思	170
1.15.5	如何打印自身代码	171
1.15.6	如何实现一个最简单的病毒	171
1.15.7	如何只使用一条语句实现 x 是否为 2 的若干次幂的判断	172
1.15.8	如何定义一对相互引用的结构	172
1.15.9	什么是逗号表达式	173
1.15.10	<code>\n</code> 是否与 <code>\nr</code> 等价	174
1.15.11	什么是短路求值	174
1.15.12	已知随机数函数 <code>rand7()</code> ，如何构造 <code>rand10()</code> 函数	175
1.15.13	<code>printf("%p\n", (void *)x)</code> 与 <code>printf("%p\n", &x)</code> 有什么区别	177
1.15.14	<code>printf()</code> 函数是否有返回值	177
1.15.15	不能使用任何变量，如何实现计算字符串长度函数	177
1.15.16	负数除法与正数除法的运算原理是否一样	178
1.15.17	<code>main()</code> 主函数执行完毕后，是否会再执行一段代码	179
第 2 章	操作系统	180
2.1	进程管理	180
2.1.1	进程与线程有什么区别	180
2.1.2	线程同步有哪些机制	181
2.1.3	内核线程和用户线程的区别	181
2.2	内存管理	182
2.2.1	内存管理有哪几种方式	182
2.2.2	什么是虚拟内存	183
2.2.3	什么是内存碎片？什么是内碎片？什么是外碎片	183
2.2.4	虚拟地址、逻辑地址、线性地址、物理地址有什么区别	184
2.2.5	Cache 替换算法有哪些	184
2.3	用户编程接口	186
2.3.1	库函数调用与系统调用有什么不同	186
2.3.2	静态链接与动态链接有什么区别	186
2.3.3	静态链接库与动态链接库有什么区别	187
2.3.4	用户态和核心态有什么区别	187
2.3.5	用户栈与内核栈有什么区别	188
第 3 章	数据结构与算法	189
3.1	数组	189
3.1.1	如何用递归实现数组求和	189
3.1.2	如何用一个 <code>for</code> 循环打印出一个二维数组	190
3.1.3	在顺序表中插入和删除一个结点平均移动多少个结点	191
3.1.4	如何用递归算法判断一个数组中的元素是否递增	191
3.1.5	如何分别使用递归与非递归实现二分查找算法	192

3.1.6	如何在排序数组中找出给定数字出现的次数	193
3.1.7	如何计算两个有序整型数组的交集	195
3.1.8	如何找出数组中重复次数最多的数	196
3.1.9	如何在 $O(n)$ 的时间复杂度内找出数组中出现次数超过了一半的数	198
3.1.10	如何找出数组中唯一的重复元素	200
3.1.11	如何判断一个数组中的数值是否连续相邻	203
3.1.12	如何找出数组中出现奇数次的元素	204
3.1.13	如何找出数列中符合条件的数对的个数	206
3.1.14	如何寻找出数列中缺失的数	209
3.1.15	如何判定数组是否存在重复元素	209
3.1.16	如何重新排列数组, 使得数组左边为奇数, 右边为偶数	211
3.1.17	如何把一个整型数组中重复的数字去掉	212
3.1.18	如何找出一个数组中第二大的数	214
3.1.19	如何寻找数组中的最小值和最大值	215
3.1.20	如何将数组的后面 m 个数移动为前面 m 个数	217
3.1.21	如何计算出序列的前 n 项数据	218
3.1.22	如何判断一个整数 x 是否可以表示成 n ($n \geq 2$) 个连续正整数的和	219
3.2	链表	220
3.2.1	数组和链表的区别是什么	220
3.2.2	何时选择顺序表、何时选择链表作为线性表的存储结构为宜	221
3.2.3	如何使用链表头	221
3.2.4	如何实现单链表的插入、删除操作	222
3.2.5	如何找出单链表中的倒数第 k 个元素	225
3.2.6	如何实现单链表反转	227
3.2.7	如何从尾到头输出单链表	230
3.2.8	如何寻找单链表的中间结点	231
3.2.9	如何对链表进行重新排序	232
3.2.10	如何把链表相邻元素翻转	234
3.2.11	如何检测一个较大的单链表是否有环	235
3.2.12	如何判断两个单链表(无环)是否交叉	237
3.2.13	如何删除单链表中的重复结点	240
3.2.14	如何合并两个有序链表(非交叉)	241
3.2.15	什么是循环链表	242
3.2.16	如何实现双向链表的插入、删除操作	244
3.2.17	一个链表不知道头结点, 有一个指针指向其中一个结点, 请问如何删除这个指针指向的结点	246
3.2.18	如何实现双向循环链表的删除与插入操作	246
3.2.19	如何在不知道头指针的情况下将结点删除	247
3.3	字符串	248

3.3.1	如何统计一行字符中有多少个单词	248
3.3.2	如何将字符串逆序	249
3.3.3	如何找出一个字符串中第一个只出现一次的字符	252
3.3.4	如何输出字符串的所有组合	253
3.3.5	如何检查字符是否是整数？如果是，返回其整数值	259
3.3.6	如何查找字符串中每个字符出现的个数	259
3.4	STL 容器	260
3.4.1	什么是泛型编程	260
3.4.2	栈与队列的区别有哪些	261
3.4.3	vector 与 list 的区别有哪些	261
3.4.4	如何实现循环队列	262
3.4.5	如何使用两个栈模拟队列操作	264
3.5	排序	266
3.5.1	如何进行选择排序	266
3.5.2	如何进行插入排序	267
3.5.3	如何进行冒泡排序	269
3.5.4	如何进行归并排序	272
3.5.5	如何进行快速排序	274
3.5.6	如何进行希尔排序	277
3.5.7	如何进行堆排序	278
3.5.8	各种排序算法有什么优劣	280
3.6	二叉树	281
3.6.1	基础知识	281
3.6.2	如何递归实现二叉树的遍历	283
3.6.3	已知先序遍历和中序遍历，如何求后序遍历	284
3.6.4	如何非递归实现二叉树的后序遍历	286
3.6.5	如何使用非递归算法求二叉树的深度	289
3.6.6	如何判断两棵二叉树是否相等	291
3.6.7	如何判断二叉树是否是平衡二叉树	292
3.6.8	什么是霍夫曼编解码	293
3.7	图	295
3.7.1	什么是拓扑排序	295
3.7.2	什么是 DFS？什么是 BFS	296
3.7.3	如何求关键路径	299
3.7.4	如何求最短路径	300
第 4 章	数据库原理	303
4.1	SQL 的功能	303
4.2	内连接与外连接	304

4.3	事务	306
4.4	存储过程与函数	307
4.5	数据库范式	308
4.6	触发器	310
4.7	游标	311
4.8	数据库日志	312
4.9	union 和 union all	312
4.10	视图	313
4.11	数据库分类	313
4.12	死锁	317
第 5 章	海量数据处理	318
5.1	问题分析	318
5.2	基本方法	318
5.3	经典实例分析	332
5.3.1	top K 问题	332
5.3.2	重复问题	335
5.3.3	排序问题	337
附录		338
附录 A		338
某互联网公司笔试真题 1		338
某互联网公司笔试真题 2		342
附录 B		343
某互联网公司笔试真题 1 答案		343
某互联网公司笔试真题 2 答案		345

上篇：面试笔试经验技巧篇

想找到一份程序员的工作，一点技术都没有显然是不行的，但是，只有技术也是不够的。面试笔试经验技巧篇主要针对程序员面试、笔试中遇到的 13 个常见问题进行深度解析，并且结合实际情景，给出一套较为合理的参考答案，以供读者学习与应用。掌握这 13 个问题的解答精髓，对于求职者大有裨益。

经验技巧 1 如何巧妙地回答面试官的问题

所谓“来者不善，善者不来”，程序员面试中，求职者不可避免地需要回答面试官提出的各种刁钻、犀利的问题，回答面试官的问题千万不能简单地回答“是”或者“不是”，而应该具体分析“是”或者“不是”的理由。

回答面试官的问题是一门很深的学问。那么，面对面试官提出的各类问题，如何才能条理清晰地回答呢？如何才能让自己的回答不至于撞上枪口呢？如何才能让自己的回答结果令面试官满意呢？

谈话是一种艺术，回答问题也是一种艺术，同样的话，不同的回答方式，往往会产生不同的效果，甚至是截然不同的效果。在此，编者提出以下几点建议，供读者参考。首先，回答问题务必谦虚谨慎。既不能让面试官觉得自己很自卑，唯唯诺诺，也不能让面试官觉得自己清高自负，而应该通过问题的回答表现出自己自信从容、不卑不亢的一面。例如，当面试官提出“你在项目中起到了什么作用”的问题时，如果求职者回答：我完成了团队中最难的工作，此时就会给面试官一种居功自傲的感觉，而如果回答：我完成了文件系统的构建工作，这个工作被认为是整个项目中最具有挑战性的一部分内容，因为它几乎无法重用以前的框架，需要重新设计。这种回答不仅不傲慢，反而有理有据，更能打动面试官。

其次，回答面试官的问题时，不要什么都说，要适当地留有悬念。人一般都有猎奇的心理，面试官自然也不例外，而且，人们往往对好奇的事情更有兴趣、更加偏爱，也更加记忆深刻。所以，在回答面试官问题时，切记说关键点，而非细节，说重点，而非和盘托出，通过关键点，吸引面试官的注意力，等待他们继续“刨根问底”。例如，当面试官对你的简历中一个算法问题有兴趣，希望了解时，可以如下回答：我设计的这种查找算法，对于 80% 以上的情况，都可以将时间复杂度从 $O(n)$ 降低到 $O(\log n)$ ，如果您有兴趣，我可以详细给您分析具体的细节。

最后，回答问题要条理清晰、简单明了，最好使用“三段式”方式。所谓“三段式”，有点类似于中学作文中的写作风格，包括“场景/任务”“行动”和“结果”三部分内容。以面试官提的问题“你在团队建设中，遇到的最大挑战是什么”为例，第一步，分析场景/任务：在我参与的一个 ERP 项目中，我们团队一共四个人，除了我以外的其他三个人中，两个人能力很强，人也比较好相处，但有一个人却不太好相处，每次我们小组讨论问题的时候，他都不太爱说话，也很少发言，分配给他的任务也很难完成。第二步，分析行动：为了提高团队的综合实力，我决定找时间和他单独谈一谈。于是我利用周末时间，约他一起吃饭，吃饭的时候，顺便讨论了一下我们的项目，我询问了一些项目中他遇到的问题，通过他的回答，我发现他并不懒，也不糊涂，只是对项目不太了解，缺乏经验，缺乏自信而已，所以越来越孤立，越来越不愿意讨论问题。为了解决这个问题，我尝试着把问题细化到他可以完成的程度，从而建立起他的自信心。第三步，分析结果：他是小组中能力最弱的人，但是，慢慢地，他的技术变得越来越厉害了，也能够按时完成安排给他的工作了，人也越来越自信了，也越来越喜欢参与我们的讨论，并发表自己的看法，我们也都愿意与他合作了。“三段式”回答的一个最明显的好处就是条理清晰，既有描述，也有结果，有根有据，让面试官一目了然。

回答问题的技巧，是一门大的学问。求职者完全可以在平时的生活中加以练习，提高自己与人沟通的技能，等到面试时，自然就得心应手了。

经验技巧 2 如何回答技术性的问题

程序员面试中，面试官会经常询问一些技术性的问题，有的问题可能比较简单，都是历年的笔试面试真题，求职者在平时的复习中会经常遇到，应对自然不在话下。但有的题目可能比较难，来源于 Google、Microsoft 等大企业的题库或是企业自己为了招聘需要设计的题库，求职者可能从来没见过或者从来都不能完整地、独立地想到解决方案，而这些题目往往又是企业比较关注的。

如何能够回答好这些技术性的问题呢？编者建议：会做的一定要拿满分，不会做的一定要拿部分分。即对于简单的题目，求职者要努力做到完全正确，毕竟这些题目，只要复习得当，完全回答正确一点问题都没有（编者认识的一个朋友据说把《编程之美》、《编程珠玑》、《程序员面试笔试宝典》上面的技术性题目与答案全都背得滚瓜烂熟了，后来找工作简直成了“offer 杀器”，完全就是一个 Bug，无解了）；对于难度比较大的题目，不要惊慌，也不要害怕，即使无法完全做出来，也要努力思考问题，哪怕是半成品也要写出来，至少要把自己的思路表达给面试官，让面试官知道你的想法，而不是完全回答不会或者放弃，因为面试官很多时候除了关注你的独立思考问题的能力以外，还会关注你技术能力的可塑性，观察求职者是否能够在别人的引导下去正确地解决问题，所以，对于你不会的问题，他们很有可能会循序渐进地启发你去思考，通过这个过程，让他们更加了解你。

一般而言，在回答技术性问题时，求职者大可不必胆战心惊，除非是没学过的新知识，否则，一般都可以采用以下 6 个步骤来分析解决。

（1）勇于提问

面试官提出的问题，有时候可能过于抽象，让求职者不知所措，或者无从下手，所以，对于面试中的疑惑，求职者要勇敢地提出来，多向面试官提问，把不明确或二义性的情况都问清楚。不用担心你的问题会让面试官烦恼，影响你的面试成绩，相反，还对面试结果产生积极影响：一方面，提问可以让面试官知道你在思考，也可以给面试官一个心思缜密的好印象；另一方面，方便后续自己对问题的解答。

例如，面试官提出一个问题：设计一个高效的排序算法。求职者可能丈二和尚摸不到头脑，排序对象是链表，还是数组？数据类型是整型、浮点型、字符型，还是结构体类型？数据基本有序，还是杂乱无序？数据量有多大，1000 以内，还是百万以上个数？此时，求职者大可以将自己的疑问提出来，问题清楚了，解决方案自然也就出来了。

（2）高效设计

对于技术性问题，如何才能打动面试官？完成基本功能是必需的，仅此而已吗？显然不是，完成基本功能顶多只能算及格水平，要想达到优秀水平，至少还应该考虑更多的内容，以排序算法为例：时间是否高效？空间是否高效？数据量不大时也许没有问题，如果是海量数据呢？是否考虑了相关环节，例如数据的“增删改查”？是否考虑了代码的可扩展性、安全性、完整性以及鲁棒性？如果是网站设计，是否考虑了大规模数据访问的情况？是否需要考虑分布式系统架构？是否考虑了开源框架的使用？

(3) 伪代码先行

有时候实际代码会比较复杂，上手就写很有可能会漏洞百出、条理混乱，所以，求职者可以首先征求面试官的同意，在编写实际代码前，写一个伪代码或者画好流程图，这样做往往会让思路更加清晰明了。

切记在写伪代码前要告诉面试官，他们很有可能对你产生误解，认为你只会纸上谈兵，实际编码能力却不行。只有征得了他们的允许，方可先写伪代码。

(4) 控制节奏

如果是算法设计题，面试官都会给求职者一个时间限制用以完成设计，一般为 20min 左右。完成得太慢，会给面试官留下能力不行的印象，但完成得太快，如果不能保证百分百正确，也会给面试官留下毛手毛脚的印象，速度快当然是好事情，但只有速度，没有质量，速度快根本不会给面试加分。所以，编者建议，回答问题的节奏最好不要太慢，也不要太快，如果实在是完成得比较快，也不要急于提交给面试官，最好能够利用剩余的时间，认真仔细地检查一些边界情况、异常情况及极性情况等，看是否也能满足要求。

(5) 规范编码

回答技术性问题时，多数都是在纸上写代码，离开了编译器的帮助，求职者要想让面试官对自己的代码一看即懂，除了字迹要工整，不能龙飞凤舞以外，最好是能够严格遵循编码规范：函数变量命名、换行缩进、语句嵌套和代码布局等，同时，代码设计应该具有完整性，保证代码能够完成基本功能、输入边界值能够得到正确地输出、对各种不合规范的非法输入能够做出合理的错误处理，否则，写出的代码即使无比高效，面试官也不一定看得懂或者看起来非常费劲，这些对面试成功都是非常不利的。

(6) 精心测试

在软件界有一个真理：任何软件都有 Bug。但不能因为如此就纵容自己的代码，允许错误百出。尤其是在面试过程中，实现功能也许并不十分困难，困难的是在有限的时间内设计出的算法，各种异常是否都得到了有效的处理，各种边界值是否都在算法设计的范围内。

测试代码是让代码变得完备的高效方式之一，也是一名优秀程序员必备的素质之一。所以，在编写代码前，求职者最好能够了解一些基本的测试知识，做一些基本的单元测试、功能测试、边界测试以及异常测试。

在回答技术性问题时，注意在思考问题的时候，千万别一句话都不说，面试官面试的时间是有限的，他们希望在有限的时间内尽可能地去了解求职者，如果求职者坐在那里一句话不说，不仅会让面试官觉得求职者技术水平不行，思考问题能力以及沟通能力可能都存在问题。

其实，面试时，求职者往往会存在一种思想误区，把技术性面试的结果看得太重要了。面试过程中的技术性问题，结果固然重要，但也并非最重要的内容，因为面试官看重的不仅仅是最终的结果，还包括求职者在解决问题的过程中体现出来的逻辑思维能力以及分析问题的能力。所以，求职者在与面试官的博弈中，要适当地提问，通过提问获取面试官的反馈信息，并抓住这些有用的信息进行辅助思考，从而博得面试官的欢心，进而提高面试的成功率。

经验技巧 3 如何回答非技术性问题

评价一个人的能力，除了专业能力，还有一些非专业能力，如智力、沟通能力和反应能力等，所以，在 IT 企业招聘过程的笔试面试环节中，并非所有的笔试内容都是 C/C++、数据结构与算法及操作系统等专业知识，也包括一些非技术类的知识，如智力题、推理题和作文题等。技术水平测试可以考查一个求职者的专业素养，而非技术类测试则更加强调求职者的综合素质，包括数学分析能力、反应能力、临场应变能力、思维灵活性、文字表达能力和性格特征等内容。考查的形式多种多样，但与公务员考查相似，主要包括行测（占大多数）、性格测试（大部分都有）、应用文和开放问题等内容。

每个人都有自己的答题技巧，答题方式各有不相同，以下是一些相对比较好的答题技巧（以行测为例）：

1) 合理有效的时间管理。由于题目的难易不同，所以不要对所有题目都“绝对的公平”、都“一刀切”，要有轻重缓急，最好的做法是不按顺序回答。行测中有各种题型，如数量关系、图形推理、应用题、资料分析和文字逻辑等，而不同的人擅长的题型是不一样的，因此应该首先回答自己最擅长的问题。例如，如果对数字比较敏感，那么就先答数量关系。

2) 注意时间的把握。由于题量一般都比较大大，可以先按照总时间/题数来计算每道题的平均答题时间，如 10s，如果看到某一道题 5s 后还没思路，则马上放弃。在做行测题目的时候，以在最短的时间内拿到最多分为目标。

3) 平时多关注图表类题目，培养迅速抓住图表中各个数字要素间相互逻辑关系的能力。

4) 做题要集中精力，只有集中精力、全神贯注，才能将自己的水平最大限度地发挥出来。

5) 学会关键字查找，通过关键字查找，能够提高做题效率。

6) 提高估算能力，有很多时候，估算能够极大地提高做题速度，同时保证正确率。

除了行测以外，一些企业非常相信个人性格对入职匹配的影响，所以都会引入相关的性格测试题用于测试求职者的性格特性，看其是否适合所投递的职位。大多数情况下，只要按照自己的真实想法选择就行了，不要弄巧成拙，因为测试是为了得出正确的结果，所以大多测试题前后都有相互验证的题目。如果求职者自作聪明，选择该职位可能要求的性格选项，则很可能导致测试前后不符，这样很容易让企业发现你是一个不诚实的人，从而首先予以筛选。

经验技巧 4 如何回答快速估算类问题

有些大企业的面试官，总喜欢使一些“阴招”“损招”，出一些快速估算类问题，对他们而言，这些问题只是手段，不是目的，能够得到一个满意的结果固然是他们所需要的，但更重要的是，通过这些题目他们可以考查求职者的快速反应能力以及逻辑思维能力。由于求职者平时准备的时候可能对此类问题有所遗漏，一时很难想起解决的方案。而且，这些题目乍一看确实是毫无头绪，无从下手，完全就是坑求职者的，其实，求职者只要从惊慌失措中冷