Hello candidate,

亲爱的求职者，

Congratulations that your application has successfully proceeded to interview! We believe you are expecting to demonstrate your excellence during this interview. To help you better understand Google's interview and get prepared, we'd like to share some interview tips with you.

首先恭喜你进入面试环节！我们相信你也一定期望自己能够在接下来的面试过程中一展身手。为了帮助你更好地了解和准备Google的面试，我们有一些面试小技巧要与你分享。

Before reading all detailed information below, you should clearly note that Google is not simply looking for engineers to solve the problems they are already well-trained on; we are interested in engineers who have passion and capability to work out the answers to questions they have never come across before. Although we list many specific knowledge and skills for your reference, we don't suggest treating them with higher priority than those not mentioned here, because the interview may cover but not limited to them. One more thing you should know about Google's interview is that we will regard your systematic thinking, quick learning ability, verbal communication and personality as the same as or even more important than knowledge itself.

在详细阅读本文之前，你需要明白Google并不是简单地需要你在经过良好的训练之后解决某些熟知问题，我们更看重你是否有激情和能力试着去解决一些从未见到过的问题。在本文中我们罗列了面试中可能会遇到的一些知识点，但面试的考查范围并不仅限于这些，我们强烈建议你对于那些我们在此没有提到的知识同样给予足够的重视。值得一提的是，在Google的面试中，除了针对知识本身进行评估外，我们同样会甚至更加看重对于你的系统化思维，快速学习能力，语言沟通能力以及个性的考察。

I. What you will expect:

面试流程

45 minute technical interview with a Google software engineer. The interviewers will be interested in your knowledge of computer science principles (data structures, algorithms etc.) and how they can be used in your solutions.

每轮面试持续45分钟，面试官将是一位来自Google的软件工程师。面试官会考察你对于计算机专业基础知识（如数据结构，算法等）的掌握程度以及你在解决实际问题时如何利用它们。

Interview topics may cover:

面试内容包括：

- Brief self-introduction, previous projects, major accomplishments or anything on your CV. especially if you have stated that you are an expert!),
- 简单的自我介绍，之前做过的项目，所获的主要成绩以及你列在简历上的其它任何东西，尤其是当你自诩为某方面专家的时候（面试官会重点考察）。

- Coding: During the interview, you'll need to write codes on whiteboard or paper (or Google docs if phone interview).
- 编码能力测试： 在面试中，你需要在白板上或者纸上完成代码（电话面试时需要在Google docs上完成）

- Building and developing complex algorithms and analyzing their performance characteristics.
- 设计和开发复杂算法并分析其性能特征。

- Computer Science fundamentals: Object oriented programming and design; common data structures like hash tables, stacks, arrays, etc; system-level knowledge such as operating system, network, compiler basics, etc
- 计算机科学的基础知识：面向对象的设计和编程；常见数据结构如哈希表，栈，数组等；系统级的知识如操作系统，计算机网络，编译器等。

- Real-world problem solving: solve fairly new problems with CS knowledge and skills.
- 解决实际问题的能力：利用计算机的知识解决全新的问题。

- Design questions: Sample topics: features sets, interfaces, class hierarchies, designing a system under certain constraints, simplicity and robustness, tradeoffs.
- 设计问题：例如功能集，接口设计，类的继承，约束条件下的系统设计，系统的简单性和鲁棒性，折衷设计。

- Open-ended discussion: biggest challenges faced, best/worst designs seen, performance analysis and optimization, testing, ideas for improving existing products.
- 开放性问题：面临的最大挑战，已知的最好 / 最坏设计，效率分析及优化，测试问题，如何改良一个已有产品的思路。

II. What to prepare

如何准备

Coding

编程

We expect **high quality, efficient, clear and compilable code without typing mistakes.** Because all engineers (at every level) collaborate throughout the Google code base, with an efficient code review process, it's essential that every engineer works at the same high standard.

我们期待你在面试中能写出高质量且高效的，结构清晰的，无输入错误并且可被编译的代码。这是因为，Google的所有工程师（各个级别）会在同一个代码库中协同工作，让所有人的代码遵循统一的高标准对于提升代码审阅流程的效率至关重要。

You should know at least one programming language really well, and it should preferably be C++, Java, Python or Javascript. Your need to write both logically and syntactically correct code – **COMPILABLE, NOT PSEUDOCODE**. So perhaps to practice, try writing code on paper, then put it through a compiler to make sure that your code compiles (and if it doesn't, then practice some more). Sample topics include: construct / traverse data structures, implement system routines, distill large data sets to single values, transform one data set to another.

你应该至少十分熟悉一种编程语言，如果熟悉C++， Java，Python或者Javascript更好，你需要在面试中写出逻辑和语法完全正确的代码。**请注意，仅仅完成伪代码是不够的，你的代码还需要可以被成功编译执行。**请尝试如此练习：你先在纸上写出完整代码，然后将其输入到电脑中试着编译（如果不能编译，你可能需要多加练习）。可以练习的题目包括：构建 / 遍历数据结构，实现系统流程，从大型数据集合中提取单个值，完成不同数据集合间的转换等。

Data structure and algorithm

数据结构和算法

Basic big-O complexity of space/time analysis is a must. Due to the size of the products you'll be building, it's imperative you're comfortable with big O notation. It is important to understand the basic concepts including List, Array, Trees, Hash tables, Graphs, sorting algorithm, NPC problems, etc. In interviews, you need to demonstrate the capability to apply all these knowledge to solve problems. Sample topics include big-O analysis, sorting and hashing, handling obscenely large amounts of data. Also see topics listed under 'Coding'.

用"大O表示法"分析算法的时空间复杂度是必须掌握的。由于你将来可能参与构建非常复杂的产品，用"大O表示法"分析执行效率非常重要。你同样需要理解如链表，数组，树，哈希表，图，排序算法，NPC问题等基本概念。并且在面试中，你需要展现出自己有能力应用上述所有知识来解决碰到的问题，包括：利用"大O表示法"进行复杂度分析，排序和哈希算法，超大规模数据的处理以及在"编程"条目下列出的知识点。

- **Sorting**: Know how to sort. Don't do bubble-sort. You should know the details of at least one nlog(n) sorting algorithm, preferably two (say, quicksort and merge sort). Merge sort can be highly useful in situations where quicksort is impractical, so take a look at it.

- **排序：**你要懂得如何实现数据的排序，除了冒泡排序之外，你需要至少了解一种时间复杂度为nlog(n)的排序算法的细节，了解两种更佳（例如快速排序算法和归并排序算法）。另外，归并排序算法在一些快速排序算法不好用的情形下会非常有效，建议你对此进行深入了解。

- **Hashtable:** Arguably the single most important data structure known to mankind. You absolutely should know how they work. Be able to implement one using only arrays in your favorite language, in about the space of one interview.
- **哈希表：**可以认为，哈希表是所有已有数据结构中最重要的一种。你需要彻底了解它如何工作，并且能够在面试中用擅长的语言用一个数组来完整实现它。

- **Trees:** Know about trees; basic tree construction, traversal and manipulation algorithms. Familiarize yourself with binary trees, n-ary trees, and trie-trees. Be familiar with at least one type of balanced binary tree, whether it's a red/black tree, a splay tree or an AVL tree, and know how it's implemented. Understand tree traversal algorithms: BFS and DFS, and know the difference between inorder, postorder and preorder.
- **树：**你需要了解树的构造，遍历和树的处理算法等基础知识，熟悉二叉树，n叉树以及字典树。你需要至少熟悉红黑树，伸展树和AVL树这三种平衡二叉树的一种，并知道如何实现它们。你需要了解树的两种遍历算法：BFS和DFS，并且知道中序，后序和前序三种遍历算法的区别。

- **Graphs:** Graphs are really important at Google. There are 3 basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list); familiarize yourself with each representation and its pros & cons. You should know the basic graph traversal algorithms: breadth-first search and depth-first search. Know their computational complexity, their tradeoffs, and how to implement them in real code. If you get a chance, try to study up on fancier algorithms, such as Dijkstra.
- **图：**图结构在Google中非常重要，你需要熟悉图的三种基本表示方法（对象箭头表示，邻接矩阵和邻接表），并能说出它们各自的优缺点。你需要了解两种基本的图遍历算法：广度优先搜索和深度优先搜索，知道它们的算法复杂度，它们各自的适用情形，以及如何在实际代码中应用它们。如果有机会的话，你可以试着去学一些更加巧妙的算法，如Dijkstra算法。

- **Other data structures:** You should study up on as many other data structures and algorithms as possible. You should especially know about the most famous classes of NP-complete problems, such as traveling salesman and the knapsack problem, and be able to recognize them when an interviewer asks you them in disguise. Find out what NP-complete means.
- **其它数据结构：**你应该尽可能地多学一些其它数据结构和算法，了解一些著名的NP完全问题，如旅行商问题和背包问题，并且能够在面试官在提及它们的时候认出它们，还要知道

“NP完全”这个概念是什么意思。

Mathematics

数学

Some interviewers may ask basic discrete math, linear algebra or some other math questions, so you can spend some time before the interview refreshing your memory on (or teaching yourself) it. the essentials of combinatorics and probability. You should be familiar with n-choose-k problems and their ilk - the more the better.

有的面试官会问到基础的离散数学、线性代数或其他一些数学问题，所以你可以在面试前花点时间复习/自学一下。请准备基本的组合学和概率论知识。你应该熟悉“n选k”问题（n-choose-k problems）及类似问题——越多越好。

Systems Design

系统设计

You need to know powers of 2, and be good and back-of- the-envelope calculations. e.g. to estimate the required number of machines for a given design. Know Google's products, and think about how you would design the back-end and also front-end. System design questions are a test of your problem solving. Sample topics include: features sets, interfaces, class hierarchies, distributed systems, designing a system under certain constraints, simplicity and robustness, tradeoffs.

你需要熟悉2的各次幂并能以此来进行估算。例如，估计一个给定的设计需要多少台计算机。你需要了解Google的产品，并思考换做你会如何设计后端与前端。系统设计题考察的是你解决问题的能力。可能的主题包括：功能集、接口设计、类继承体系、分布式系统、约束条件下的系统设计、系统的简单性和鲁棒性、折衷设计。

Operating Systems

操作系统

In fact the interview may cover but not limited to operating systems, also there may be other topics like network, parallel computing, compiler theory, etc. Here let’s just take operating systems for example: Know about processes, threads and concurrency issues. Know about locks and mutexes and semaphores and monitors and how they work. Know about deadlock and livelock and how to avoid them. Know what resources a processes needs, and a thread needs, and how context switching works, and how it's initiated by the operating system and underlying hardware. Know a little about scheduling. The world is rapidly moving towards multi-core, so know the fundamentals of "modern" concurrency constructs.

实际上面试内容包括但不限于操作系统，还可能包括其他内容，如网络、并行计算、编译原理等。这里仅以操作系统为例：你需要了解进程、线程和并发问题，熟悉锁（locks）、互斥量（mutexes）、信号量（semaphores）、监视器（monitors）以及它们的工作方式，理解死锁（deadlock）、活锁（livelock）以及如何避免。你需要知道进程和线程所需的系统资源，清楚上下文切换（context switching）的工作方式，并了解操作系统和底层硬件如何创建它们。你还需要对任务调度（scheduling）要有一些认识。世界正在迅速向多核迈进，所以请你了解一下“现代”并行架构的基础知识。

Real-world problems

实际问题

Some interviewers may ask some real-world problems, they are looking for process of thought, creative solutions and being able to work out more than one way to solve a problem and talk through your rationale for choosing a certain way to approach solving the problem. So you could perhaps recommend an algorithm, code up a solution using that algorithm, analyze the runtime of your code and then optimize your solution.

有的面试官可能会问一些实际问题，以考察以下几个方面：思维过程是否清晰，能否提出创造性的解决方案并用多种方法解决问题，以及如何在探讨中选定解决问题的途径。因此你可以尝试提出一种算法，用这个算法写成解决方案的代码，分析代码的运行时性能，然后优化你的方案。

III. Where you can warm up and practice

在哪里热身和练习

You’re strongly suggested to PRACTICE your coding and algorithm before the interview, below resources may be helpful:

我们强烈建议你在面试前**练习**编码和算法，以下是一些有用的资源：

- Coding: Google's APIs, Code Style Guide
- 编码：Google's APIs, 代码风格指南
- Data Structure and Algorithm: List of Algorithm general topics，List of Algorithms，List of Data Structures

- 数据结构与算法：算法常规主题列表，算法列表，数据结构列表
- Sharing by Interviewer: Video
- 面试官分享：视频
- Practice: You may want to visit the website www.topcoder.com . If you launch the "Arena" widget and then go to the practice rooms, you can play with the problems in the first/second division as a warm up. Another site recommended highly by engineers is projecteuler.net. These sites will expose you to programming problems that you would not normally come across in a standard day.
- 练习：你也许会需要访问网站www.topcoder.com。如果你加载了“竞技场”（Arena）控件并进入练习室，你可以用一区或二区的题目做热身。另一个工程师们强烈推荐的网站是projecteuler.net。这些网站有一些你平时无法遇到的编程难题。

IV. How to succeed the interview

如何成功的面试

Speak out and communicate

大胆说出你的想法和主动交流

Most of the time, you'll need more information from the interviewer to analyze & answer the question to its full extent. The process of your thoughts is as important as final solution, so you are encouraged to ask more questions and speak out your thoughts.

多数情况下，你需要从面试官那里得到更多信息来分析和解答完整的问题。你思考的过程与最终的方案同样重要，因此我们鼓励你问更多的问题并说出你的想法。

- When asked to provide a solution, first define and frame the problem as you see it.
- 当需要作答的时候，请首先定义和框定你所听到的问题。
- If you don't understand the question- ask interviewer for help or clarification.
- 当你不理解问题的时候——请让面试官帮忙说明它。
- If you need to assume something - check it’s a correct assumption.
- 如果你需要做一些假设——请确保它是正确的假设。
- Describe how you want to tackle solving each part of the question.
- 请描述你打算如何解决问题的每个部分。
- Always let your interviewer know what you are thinking as he/she will be as interested in your process of thought as your solution.
- 请让面试官随时知道你的想法，因为对他/她来说，你的思考过程与最终答案同样重要。
- Listen carefully - don't miss a hint if your interviewer is trying to assist you!
- 请认真倾听——在面试官试图帮你的时候不要遗漏掉提示。

Know interviewer's expectation:

了解面试官的期望

Interviewers will be looking at the approach to questions as much as the answer:

面试官会像关注答案一样地关注你解决问题的过程：

- Does the candidate listen carefully and comprehend the question?
- 求职者是否认真听并理解了问题？
- Are the correct questions asked before proceeding? (important!)
- 求职者在开始解题前是否询问了正确的问题？（重要！）
- Is brute force used to solve a problem? (not good!)
- 求职者解题时是否使用了暴力解法？（不好！）
- Are things assumed without first checking? (not good!)
- 求职者是否使用了未经检测的假设？（不好！）
- Are hints heard and heeded?
- 求职者是否听到并留意到了提示？
- Is the candidate slow to comprehend / solve problems? (not good!)
- 求职者理解/解决问题是否缓慢？（不好！）
- Does the candidate enjoy finding multiple solutions before choosing the best one?
- 求职者在选择最优方案前，是否乐于寻找更多解决方案？
- Are new ideas and methods of tackling a problem sought?
- 求职者是否探寻过新思路和新方法来解决问题？
- Is the candidate inventive and flexible in their solutions and open to new ideas?
- 求职者在解决方案中是否有创造性和灵活性，能否接纳新的思路？
- Can questioning move up to more complex problem solving?
- 面试问题能否推广到更复杂的情形？（译注：前提是求职者很好地解决了当前问题）

Ask Questions!

提问环节

Think about what you care the most and prepare some questions for each interview. Note: please direct to your recruiter if you have questions about the interview process, remuneration or your interview performance. We highly recommend you to do some homework about Google and our products: Google at a Glance, Google's Wikipedia Page, A Google Engineer's Career.

在每次面试之前请准备一些你关心的问题。注：如果你有关于面试流程、薪酬或你的面试表现的问题，请直接咨询你的招聘HR。我们强烈推荐你做些关于Google和我们产品的功课：Google简介，Google维基主页，一位Google工程师的职业生涯。

Thanks for your reading. In the end, there is no special requirement for your dressing code, just wear comfortably and confidently. We wish you good luck in the coming interview!

感谢你的阅读。最后，我们对你的面试着装没有特别要求，穿着舒适和自信就好。祝你在接下来的面试中好运！