親愛的讀者，

讓我們這樣開場吧。

我不是招聘人員，而是一名軟體工程師。正因為如此，我懂被要求當場拿出出色的演算法，然後在白板上寫出完美的程式碼是什麼感覺。我之所以會懂，是因為不論是在 Google、Microsoft、Apple、Amazon 或其他公司的面試中，我被要求做過同樣的事情。

我懂，也是因為我一直坐在桌子的另一邊，要求面試者這樣做。我在成堆的履歷表中尋找那些我認為有可能通過面試的工程師。我在他們解決或試圖解決具有挑戰性的問題時對他們進行評估。我曾在 Google 的招聘委員會辯論過，一位面試者的表現是否足以獲得工作機會。我瞭解整個招聘過程，因為我經歷過很多次。

而您，親愛的讀者，您可能正在為面試做準備，也許是明天，也許是下週，也許是明年。本書的目標在於幫助您鞏固對電腦科學基礎的理解，然後學習如何應用這些基礎來提升程式設計師的面試力。

《提升程式設計師的面試力》第 5 版更新到第 6 版時，增加了 70% 的內容：包括附加問題、修改的解決方案，新的章節介紹，更多的演算法策略，所有問題的提示，和其他內容。請您一定要造訪我們的網站 CrackingTheCodingInterview.com，與其他面試者聯絡，發現新的資源。

我為您和您將要發展的技能感到興奮。充分的準備將使您獲得廣泛的技術和溝通技巧。不管您花了多少力氣，這都是值得的！

我鼓勵您們仔細閱讀這些入門章節。它們包含一些重要的見解，這些見解可能會造成「雇用」和「不雇用」的差別。

請記得，面試是很難的！我在 Google 當面試官的那些年裡，我看到一些面試官問的是「簡單」的問題，而另一些則問了比較難的問題。但您知道嗎？回答簡單的問題並不會讓您更容易得到這份工作。不是您完美地解決問題（也很少有求職者能做到這一點！）才會收到錄用通知。相反地，重點是要把問題回答的**優於其他面試者**。所以，當您遇到棘手的問題時請不要緊張，因為別人可能也認為這很難，不完美也沒關係。

好好學習，多練習，祝您好運！

Gayle L . McDowell
Founder/CEO, CareerCup.com
另著有《*Cracking the PM Interview*》與《*Cracking the Tech Career*》

哪裡出了點問題

我們再次沮喪地走出招聘會議。當天我們審核的 10 位面試者中，沒有一位會收到錄用通知。我們在想，我們是不是太苛刻了？

我尤其感到失望，因為我們拒絕了一個**我的**面試者，是我以前的學生也是由我引薦的人。他在華盛頓大學（University of Washington）的平均績點為 3.73，這是世界上最好的電腦科學學院之一。他精力充沛，他很有創造力，他很機靈，他努力工作，他在所有面向都是一個真正的極客。

但我不得不同意招聘委員會的其他成員：資料沒有顯示他適合。即使我強而有力的推薦能說服他們重新考慮，也肯定會在招聘過程的後期階段被拒絕，他的危險信號太多了。

雖然他很聰明，但他在解決面試中的問題時還是很吃力。大多數成功的應試者都能很快地回答第一個問題，這個問題是一個眾所周知的問題的一個變體，但他在開發演算法時遇到了麻煩。當他提出一個解決方案時，他也沒有考慮為其他場景優化的解決方案。最後，當他開始撰寫程式碼時，他用最初的解決方案快速地撰寫完程式碼，但程式碼中充滿了他未能捕捉到的錯誤。儘管他並不是我們所見過的最差的面試者，但他遠遠不及「合格標準」。

幾週後，當他在電話裡詢問面試情況時，我掙扎著不知該告訴他什麼。請更聰明些嗎？不，我知道他很聰明。做一個更好的程式設計師？不，他的技術和我所見過的一些最好的不相上下。

像許多積極的面試者一樣，他做了充分的準備。他讀過 K&R 的經典 C 語言書籍，也複習過 CLRS 著名的演算法教科書。他可以詳細地描述平衡樹的無數種方法，而且他可以用 C 語言做任何理智的程式設計師都不會想做的事情。

我不得不告訴他一個不幸的事實：那些書還不夠。學術書籍為您的研究做準備，它們可能會讓您成為一個更好的軟體工程師，但它們不足以讓您通過面試。為什麼？我給您一個提示：您的面試官還在學校求學的時候，沒見過紅黑樹。

要提升程式設計師的面試力，您需要為**真正**的面試問題做準備。您必須針對實際的問題進行練習，並學習它們的模式。重點是關於開發一個新的演算法，而不是記住現有的問題。

《**提升程式設計師的面試力**》是我在頂尖公司面試的親身經驗，以及後來透過這些面試來指導面試者的結果。這是與面試者數百次交談得到的結果，是由面試者和面試

官提出的成千上萬個問題的結果，這是看到這麼多公司提出這麼多面試問題的結果。這本書從成千上萬的可能問題中挑選出來 189 個最好的面試問題。

我的方法

本書的重點是演算法、寫程式以及設計問題。為什麼把重點放在這些上呢？因為雖然您可能也會被問到行為方面的問題，但是那些問題的答案會和您的履歷一樣五花八門。同樣地，雖然許多公司會問所謂的「瑣碎」問題（例如，「什麼是虛擬函式（virtual function）？」），但是練習這些瑣碎問題所能培養出的技能也僅限於非常特定的知識。本書將簡要介紹其中的一些問題，藉以向您展示這類問題大概是什麼樣子，但我選擇將篇幅分配給更多要學習的領域。

我的愛好

教學是我的愛好，我喜歡幫助人們理解新概念，並為他們提供工具，以幫助他們在熱愛的領域中脫穎而出。

我的第一次正式教學經歷是在 University of Pennsylvania 的一所學院中，當我成為一門電腦科學本科課程助教的第二年，我又去當了其他幾門課程的助教，最後我在那裡開設了自己的電腦科學課程，側重於實作技能。

在作為 Google 工程師時，培訓和指導新工程師是我最喜歡做的事情之一。我甚至用我 20% 的時間在 University of Washington 教授兩門電腦科學課程。

多年後的今天，我繼續教授電腦科學的概念，但這次的目標是幫助新創公司的工程師為他們的收購面試做準備。我看到了他們的錯誤和掙扎，而我開發了一些技巧和策略來幫助他們解決這些問題。

《提升程式設計師的面試力》、《*Cracking the PM Interview*》、《*Cracking the Tech Career*》以及《*CareerCup*》實現了我對教學的熱情。即使是現在，您也可以經常在 CareerCup 網站上看到我在幫助那些需要幫助的使用者。

請加入我們的行列。

Gayle L . McDowelll

面試流程

在大多數頂尖科技公司（以及許多其他公司），演算法和程式碼問題是面試流程中最重要的部分。請把這些當成是測試您解決問題的能力，面試官想要評估您解決以前沒見過的演算法問題的能力。

一般情況下，您在面試中可能只會回答一個問題。45 分鐘並不算長，在這段時間內解決幾個不同的問題是很困難的。

在整個答題過程中，您應該勇敢地儘量大聲地說出來，並解釋您的思考過程。您的面試官有時會跳出來幫您；請讓他們幫您。這是一件很正常的事，但並不代表著您做得很差（當然，不需要提示就更好了）。

面試結束時，面試官會憑感覺決定您的表現。雖然可能會依您的表現作出評分，但它實際上不是一個定量的評估，不會有圖表顯示您在每個評估項目上得到了幾分。

相反地，面試官會根據以下幾點來評估您的表現：

- 分析能力：您需要很多協助才能解決問題嗎？您的解決方案有多理想？您花了多長時間才找到解決辦法？如果您必須設計 / 架構一個新的解決方案，您是否能很好地把這個問題架構建出來，並考慮了不同決策的優缺得失？
- 撰寫程式技能：您能成功地將您的演算法轉換成合理的程式碼嗎？程式碼乾淨有序嗎？您考慮過潛在的錯誤嗎？您的風格好嗎？
- 技術知識 / 電腦科學基礎：您在電腦科學和相關技術方面有很強的基礎嗎？
- 經驗：您在過去做過好的技術決定嗎？您做過有趣的、有挑戰性的專案嗎？您是否表現出充滿熱忱與精力、積極主動，和其他重要的特質？
- 文化適應 / 溝通技巧：您的個性和價值觀是否適合公司和團隊？您和面試官溝通得好嗎？

這些點的權重將根據問題、面試官、角色、團隊和公司而有所不同。對於一個標準的演算法問題來說，幾乎完全偏重在前三點上。

▶ 為什麼？

在一個面試者開始這個面試訓練流程時，一個最常會有的問題是到底為什麼要做這個訓練？畢竟：

1. 很多優秀的面試者在這類面試中表現不佳。
2. 如果確實有過答案，您可以去查答案。
3. 在現實世界中，很少需要使用諸如二元搜尋樹之類的資料結構。如果您確實需要用到，您當然會去學習它。
4. 在白板上寫程式碼是一種人為刻意的環境。顯然，在現實世界中，您永遠不會在白板上寫程式碼。

這些抱怨不無道理。事實上，我同意以上所有的觀點，至少部分同意。

與此同時，我們有理由對某些職位（非所有職位）採取這種考核方式。您是否同意這個邏輯並不重要，但理解為什麼要問這些問題是一個好主意，這有助於您深入瞭解面試官的心態。

錯過了一些優秀的人是可以接受的

這讓人難過（也讓面試者沮喪），但卻是事實。

從公司的角度來看，錯過一些優秀的面試者其實是可以接受的。這家公司打算培養一大批優秀的員工。他們可以接受他們錯過了一些優秀的人。當然，他們不希望這樣做，因為這樣會增加他們的招聘成本。不過，這是一個可以接受的折衷方案，只要他們仍然能夠聘用足夠多的優秀人才就行了。

他們更關心的是誤判了優秀的人：那些在面試中表現出色，實際上沒那麼優秀的人。

解決問題的技巧是有價值的

如果您能夠解決幾個難題（可能在一些幫助下），那麼您可能非常擅長開發最優的演算法，因為您夠聰明。

聰明的人傾向於把事情做好，這在公司是很有價值的。當然，這不是唯一重要的事情，但它確實是一件好事。

基本的資料結構和演算法知識是有用的

許多面試官會認為，基本的電腦科學知識實際上是有用的。需要理解樹、圖、串列、排序和其他知識的時刻確實會週期性地出現。而當它出現的時候，這種理解就真的很有價值。

您能在需要用時才學習嗎？當然可以。但是如果您不知道二元搜尋樹的存在，您就很難知道您應該使用它。如果您知道它的存在，那麼您大概就已經懂得它的一些基礎知識了。

有些面試官認為資料結構和演算法是一種很好的「間接能力」，即使這些技能本身並不難學，但他們認為這與成為一名優秀的開發人員有相當大的關係。它代表著您要麼掌握了一個電腦科學程式（要掌握這種程式，代表您已經學習並累積了相當廣泛的技術知識），要麼就是自學了這些東西。不管怎樣，這都是一個好跡象。

出現資料結構和演算法知識還有另一個原因：因為若要問出一個考驗解決問題的能力的問題，很難**不講到**它們。事實證明，絕大多數能解決問題的解答都涉及這些基本知識。當足夠多的面試者都瞭解這些基本知識時，就很容易形成一些典型的問題。

白板讓您專注於重要的事情

在白板上寫出完美的程式碼是非常困難的。幸運的是，您的面試官並沒有期待要看到完美的程式碼。實際上每個人都會寫出一些 bug 或小的語法錯誤。

白板的好處在於，在某種程度上，您可以專注於大局。您沒有編譯器，所以不需要編譯程式碼。您不需要撰寫整個類別定義和樣板程式碼。您可以關注程式碼中有趣的、「有肉的」部分：與問題真正相關的函式。

這並不是說您應該只寫虛擬碼或者正確性不重要。大多數面試官不接受虛擬碼，也期待錯誤越少越好。

白板也傾向於鼓勵面試者多發言，解釋他們的思考過程。當面試者使用電腦時，與面試官之間的交流就會大大減少。

每個人、每間公司或每種情況下皆有不同

以上部分是為了幫助您瞭解公司方的思維。

您好奇我個人的想法嗎？在正確的情況下，如果做得好，這是可以拿來合理判斷一個人解決問題的能力，因為做得好的人往往相當聰明。

然而，它往往做得不是很好。您的面試官可能很糟糕，或者他們問的問題很糟糕。

它也不適合所有的公司。有些公司應該更重視員工以前的經驗，或者需要特定技術的技能，所以這類問題並不能評量這些面向。

它也不能衡量一個人的職業道德或專注能力。然而，幾乎沒有面試流程能真正評估這一點。

無論如何，這都不是一個完美的流程，但什麼才是一個完美的流程呢？所有的面試流程都有其缺點。

最後我想說的是：它就是這樣，所以讓我們盡我們所能吧。

▶ 問題是怎麼選出來的？

面試者經常會想四處問某一家特定的公司「最近」面試的問題是什麼。光是問這個問題就顯露出面試者根本搞錯了為什麼要問這些面試問題。

在絕大多數的公司裡，沒有規定面試官應該問什麼問題。相反地，每個面試官都會選擇自己的問題。

由於要問什麼問題是可以自由選擇的，所以沒有什麼叫做「Google 最近的面試問題」，除非是問到某位在 Google 工作的面試官最近問過什麼問題。

今年 Google 問過的問題與三年前的問題並沒有太大不同。事實上，Google 會提出的問題通常與類似公司（Amazon、Facebook 等）提出的問題沒有什麼不同。

不同公司之間存在一些廣泛的差異。一些公司專注於演算法（通常使用一些系統設計），而另一些公司則非常喜歡需要知識才能回答的問題。但是對於一個給定的問題類別來說，幾乎沒有什麼能讓它「屬於」一家公司，而不是另一家公司。Google 演算法問題本質上和 Facebook 演算法問題是一樣的。

▶ 一切都是相對的

如果沒有絕對的評分系統，您是如何被評分的？面試官想在您身上看到什麼？

好問題。當你能理解這個問題時，這個答案實際上就很有意義了。

面試官透過您在同一問題上相對於其他求職者的表現進行評分，這是一個相對的比較。

例如，假設您想出了一個又新又酷的智力題或數學問題。您問您的朋友 Alex 這個問題，他花了 30 分鐘來解決它。您問 Bella，她花了 50 分鐘解決它。Chris 解不出來。Dexter 需要 15 分鐘，但您必須給他一些重要的提示，如果沒有那些提示，他可能會花更長的時間。Ellie 花了 10 分鐘，並提出了一個您甚至沒有意識到的替代方法，Fred 需要 35 分鐘。

您結束的時候會說，「哇，Ellie 做得真不錯。我敢打賭她數學一定很好。」也許 Chris 運氣不好。您可以多問幾個問題，以確定這是不是運氣造成的。

面試時的問題也是如此。面試官是透過比較您和其他人的表現來瞭解您。其他人指的不是她本週面試的那幾位面試者，而是她曾經問過同一個問題的所有面試者。

因此，遇到難題並不是一件壞事。當您覺得遇到難題時，其他人一定也這麼覺得。這並不會降低您做得更好的可能性。

▶ 常見問題

我面試後沒有立即得到回覆，我是被拒絕了嗎？

不，公司的決策可能會被延遲，原因有很多。一種非常簡單的情況是，您的其中一位面試官還沒有提供他們的回饋。很少有公司的政策是不回應被拒絕的面試者。

如果您在面試後的 3-5 個工作日內沒有收到公司的回覆，一定要（禮貌地）和招聘人員聯繫。

我被拒絕後可以重新再面試同一間公司嗎？

基本上是可以的，但您通常需要等待一段時間（6 個月到 1 年）。您搞砸的第一次的面試通常不會對您有太大影響，很多人曾被 Google 或微軟拒絕，但後來又被錄用。

II

幕後

大多數公司的面試方式都非常相似。我們將提供一個概述，說明公司如何面試和他們在尋找什麼。這些資訊將能指引您的面試準備工作，以及您在面試中和面試後的反應。

一旦您被選中參加面試，通常您會經過一次篩選面試，這一般是透過電話進行的。頂尖學校畢業的面試者可能會親自參加這些面試。

不要被這個名字給騙了；「篩選」面試通常會含有程式碼和演算法方面的問題，面試的門檻可以和當面面試一樣高。如果您不確定面試是否是技術性的面試，問問您的招募人員您的面試官是什麼職位（或者面試可能包括哪些內容）。一般來說工程師會進行技術面試。

許多公司已經採用了線上同步文件編輯器，但是有些其他公司會希望您在紙上撰寫程式碼並透過電話告訴他們您寫了什麼。有些面試官甚至會在通話結束後，出一些「家庭作業」給您，或者只是讓您把自己寫的程式碼透過電子郵件發給他們。

在被帶到現場面試之前，您通常會經歷一到兩次篩選面試。

在現場面試中，您通常會與 3 到 6 位面試官面試。其中一位可能會跨過午餐時間，這種午餐面試通常不是技術性的，面試官甚至可能不會送交回饋。您很適合與這位面試官討論您的興趣愛好，也很適合詢問公司文化。其他面試將主要著重於技術性，涉及程式碼、演算法、設計 / 架構和行為 / 經驗問題的組合。

在不同公司或甚至不同團隊中，由於公司的優先順序、規模或單純只是因為隨機性，造成面試問題在上述主題中呈現不同的分佈。面試官在面試問題上通常有很大的自由度。

面試結束後，面試官會針對您的面試提出一些回饋。某些公司的面試官會聚在一起討論您的表現，然後做出決定；某些公司的面試官會向招募經理或招募委員會送交一份

推薦信，以做出最後的決定；某些公司的面試官甚至不做決定，他們的回饋會被送到招募委員會去做決定。

大多數公司會在一週後進行下一步（聘雇、拒絕、進一步面試，或者只是再安排接下去的流程）。有些公司的反應比較快（有時甚至是同一天！），有些公司則需要更長的時間。

如果您已經等了一個多星期沒有消息，您應該跟招募人員聯繫。如果您的招募人員也沒有回應，這並不代表您被拒絕了（至少對任何一家大型科技公司來說這都不代表被拒絕，而且幾乎所有公司也一樣）。讓我再重複一遍：不回覆與您的狀態無關，不回覆是因為還沒有做出最終決定，一旦做出了最終決定，所有招募人員就會告訴所有求職者已經有結果了。

也確實有可能會發生延遲的情況。如果您覺得延遲了，那就跟招募人員聯繫一下，但一定要有禮貌。招募人員和您一樣，他們也會有忙碌和健忘的時候。

▶ Microsoft 的面試

Microsoft 需要聰明的人、極客和熱愛科技的人。他們可能不會考您 C++ API 的輸入和輸出是什麼，但會要求您在白板上寫程式碼。

在一場典型的 Microsoft 面試中，您會在早上的某個時間出現在 Microsoft，填寫一些初始的表格。您會有一個簡短的面試，招募人員會給您一個簡單的問題。招募人員的目的通常是幫您做準備，而不是盤問您的技術問題。如果您被問到一些基本的技術問題，可能是因為這位招募人員想讓您輕鬆地進入面試，這樣當「真正的」面試開始時，您就不會那麼緊張了。

請善待您的招募人員。您的招募人員可能是您最大的支援者，如果您在第一次面試中受挫，他們甚至會要求重新面試您。他們可以為您爭取工作，也可以不爭取！

接下來的一整天，您通常會和兩個不同的團隊做四到五次面試。許多公司是在會議室會見面試官，而 Microsoft 則是直接在面試官的辦公室面談。這是一個瞭解團隊文化的好時機。

根據團隊的不同，面試官可能會也可能不會把他們對您的回饋告訴面試流程裡的其他人。

當您完成一個團隊的面試時，您可能會與招募經理交談（通常被稱為「as app」，是「as appropriate（適任）」的縮寫）。如果是這樣，那就是一個好跡象！這可能代表著您通過了該團隊的面試。現在就看招募經理的決定了。

您有可能會在那一天得到結果，也可能是一個星期。若一週後仍沒有收到人力資源部的訊息，請發一封友好的電子郵件要求更新狀態資訊。

如果您的招募人員反應不積極，那是因為她很忙，而不是因為您被默默拒絕了。

一定要準備的問題：

「您為什麼想在 Microsoft 工作？」

在這個問題中，Microsoft 希望看到您對技術充滿熱情。一個很好的回答可能是：「自我有印象開始，我就一直在使用 Microsoft 的軟體，我對 Microsoft 如何創造出一款優秀的產品印象深刻」。例如，我最近一直在使用 Visual Studio 學習遊戲程式設計，它的 API 非常棒。

特點：

只有在您表現得很好時，才會見到招募經理，所以如果您見到招募經理的話，這是一個很好的跡象！

此外，Microsoft 傾向於給團隊更多的個別控制權，產品的種類也是多樣化的。由於不同的團隊會想尋找不同的東西，所以 Microsoft 內部工作的體驗可能也會有很大的不同。

▶ Amazon 的面試

Amazon 的招募流程通常以電話面試開始，在這樣的電話面試中，面試者要與一個特定的團隊進行面試。如果在一小段時間裡，一個求職者有兩次或兩次以上的面試，這要麼代表面試官裡有人沒有被說服，要麼代表他們在考慮讓面試者進入一個不同的團隊或職位。在少數不尋常的情況下，在少數情況下，例如當面試者是內部員工或最近才面試過另一個不同的職位，就可能只需要做一次電話篩選即可。一個面試者可能只需要做一次電話篩選。

面試您的工程師通常會要求您透過一個共用的文件編輯器來撰寫簡單的程式碼，他們還會提出一系列廣泛的問題來探索您所熟悉的技術領域。

接下來，您要飛往西雅圖（或您正在面試的辦公室），與一到兩個團隊進行四到五次面試，這些團隊是根據您的簡歷和電話面試來選擇您。您必須在白板上寫程式碼，一些面試官會看重其他技能。每位面試官都會被分配要去探索您的一個特定的領域，而且這些領域之間可能看起來非常不同。在送交自己的回饋之前，他們無法看到其他人的回饋，而且在招募會議之前，他們不被鼓勵討論這些回饋。

其中有一位「標準提高（bar raiser）」面試官負責保持面試高標準。他們參加特殊的培訓，並獨立於您要面試的團隊，以制衡團隊本身。如果一個面試看起來很困難、很不一樣，那您就很有可能遇到標準提高面試官了。這個人在面試方面有豐富的經驗，在招募決定上有否決權。但請您要記住：您在面試中表現遭遇困難的樣子，並不代表著您實際上表現得差。您的表現是相對於其他面試者來評判的，而不是在簡單的「準確百分比」基礎上評估。

一旦您的面試官輸入了他們的回饋就會開會討論，他們將是做出雇傭決定的人。

雖然 Amazon 的招募人員通常非常善於和求職者聯絡更新，但偶爾也會出現延誤。如果您一週內沒有收到 Amazon 的來信，建議您發一封友好的電子郵件過去。

一定要準備的問題：

Amazon 非常注重擴縮性問題，所以一定要為擴縮性問題做好準備。要回答這些問題，您不需要有分散式系統方面的背景知識。請參閱系統設計（System Design）和擴縮性（Scalability）那一章中的建議。

此外，Amazon 傾向於詢問許多關於物件導向設計的問題。有關示範問題和建議，請參閱物件導向設計那一章。

特點：

會從不同的團隊請來標準提高面試官，以保持高標準。您必須使這個人和招募經理都留下深刻的印象。

與其他公司相比，Amazon 傾向於在招募過程中進行更多的新嘗試。這裡描述的流程只是典型的情況，但是由於 Amazon 有著喜歡做新嘗試的習慣，所以這個流程不一定是通用的。

▶ Google 的面試

關於 Google 面試的可怕謠言有很多，但它們大多只是謠言。Google 的面試與 Microsoft 和 Amazon 並沒有太大的不同。

第一次電話篩選由 Google 的工程師執行，所以請您預期會遇到棘手的技術問題。這些問題可能與程式碼有關，有時透過共用文件進行。應徵者通常被要求遵守相同的標準，在電話視訊上被問及的問題與現場面試的問題類似。

在您的現場面試中，您會面試 4 到 6 個人，其中一個是午餐面試官。面試官的回饋不能告訴其他的面試官，所以您可以放心，每次面試都是新的開始。您的午餐面試官不會送交回饋，所以這是一個問誠實問題的好機會。

面試官通常不會給出具體的重點，也沒有「結構」或「系統」來告訴您什麼時候該問什麼。每個面試官可以按照自己的意願進行面試。

面試官將會把書面回饋送交給由工程師和經理組成的招募委員會（hiring committee，HC），以作出聘用 / 不聘用建議。回饋通常分為四類（分析能力、撰寫程式碼能力、經驗和溝通能力），總分為 1.0-4.0 分。招募委員會通常不包括您的任何面試官。如果有，那純粹是碰巧而已。

若要聘雇，面試官會希望至少有一位面試官是「熱情的支持者」，換句話說，3.6 分、3.1 分、3.1 分和 2.6 分的簡歷會比全部都是 3.1 分的要好。

您不一定要在每次面試中都表現出色，而且您做視訊面試時，手機畫面品質通常也不會是影響最終決定的重要因素。

如果招募委員會決定要聘雇，您的履歷將被送交給薪酬委員會，然後再送交給執行管理委員會。做出決定可能需要幾個星期，因為有太多的流程關卡和委員會。

一定要準備的問題：

作為一個以 web 起家的公司，Google 關心的是如何設計一個可擴縮的系統。因此，一定要準備好回答系統設計和可擴縮性方面的問題。

Google 看重的是分析（演算法）技能，而不是您的經驗。您應該對這些問題做好充分的準備，即使您認為您之前的經驗更重要。

特點：

您的面試官不是做出聘雇決定的人。相反地，他們輸入回饋資訊，並將資訊傳遞給招募委員會。招募委員會推薦的決定仍可能（雖然很少發生）被 Google 的管理層拒絕。

▶ Apple 的面試

和公司本身一樣，Apple 的面試流程也沒有什麼官僚作風。面試官看重的是優秀的技能，但是對職位和公司的熱情也很重要。雖然您有沒有在用 Mac 作業系統並不是先決條件，但至少應該熟悉這個系統。

面試流程通常從招募人員的電話視訊開始，目的是對您的技能有一個基本的瞭解，然後是一系列團隊成員的技術電話視訊。

一旦您被邀請進入公司園區，招募人員通常會向您介紹整個招募過程。接下來，您將與團隊成員以及與您的團隊一起工作的關鍵人物進行 6 到 8 次的面試。

您要有心理準備會有一對一和二對一的面試。準備好在白板上寫程式碼，並確保您所有的想法都清晰的傳達。午餐是和您未來可能的主管一起吃的，雖然看起來比較隨意，但仍然是面試。每個面試官通常專注於一個不同的面向，公司不鼓勵面試官與其他面試官分享回饋，除非他們想讓後續面試官深入瞭解一些事情。

快結束的時候，面試官會互相交換意見。如果每個人仍然覺得您是一個可行的面試者，您將與您申請的組織的主管和副總裁進行面試。雖然這個面試不會太正式，但如果您得到這次機會，這是一個很好的跡象。如果你沒有通過面試，聘雇決定也可能在幕後發生，而您就會被直接帶出大樓，（直到此時）您都不會知道自己沒有通過面試。

如果您完成了與主管和副總裁的面試後，所有的面試官都會聚集在會議室，給您一個正式的肯定或否定的評價。副總裁一般不會出席，但如果他們對招募結果不滿意，他仍然可以否決。招募人員通常會在幾天後與您聯絡，但您可以隨時聯繫他或她，以瞭解最新情況。

一定要準備的問題：

如果您知道您在面試的是哪個團隊，一定要多瞭解那個產品。您喜歡它什麼？您會改進什麼？提供具體的建議可以顯示出您對這份工作的熱情。

特點：

Apple 公司經常進行二對一的面試，但是請不要為此感到壓力，這和一對一的面試是一樣的！

此外，Apple 公司的員工都是 Apple 的死忠粉絲，您應該在面試中表現出同樣的熱情。

▶ Facebook 的面試

一旦被選中參加面試，面試者通常會進行一兩次電話篩選，電話篩選將是技術性導向的，包括撰寫程式碼，通常是使用線上文件編輯器。

在電話面試之後，您可能會被要求完成一項包括程式碼和演算法的家庭作業，在寫這個作業時請注意您的程式碼風格。如果您過去沒有在一個有完整的程式碼審查的環境中工作過，那麼最好先請人來審查您的程式碼。

您的現場面試，主要會是和其他的軟體工程師進行，但是招募經理只要有時間也會參與。所有的面試官都受過全面的面試訓練，而與您面試的人與您被錄用的機率無關。

在現場面試中，每個面試官都被賦予一個「角色」，這有助於確保沒有重複的問題，讓他們對面試者有一個全面的瞭解。這些角色是：

- 行為（絕地武士（Jedi））：這個面試將評估您在 Facebook 環境下成功的能力。您能很好地融入公司的文化和價值觀嗎？您對什麼感到興奮？您如何應對挑戰？你也要準備好談論您對 Facebook 的興趣，Facebook 需要有熱情的人。在這次面試中，您可能還會被問到一些程式碼問題。
- 程式碼和演算法（忍者（Ninja））：這些會問的是程式碼和演算法問題，很像您在這本書裡看到的。這些問題被設計成具有挑戰性的。您可以使用任何您想要的程式設計語言。
- 設計 / 架構（海盜（Pirate））：應徵後端軟體工程師的人，可能會被問到系統設計問題；應徵前端或其他專業的人，將被問及與該學科相關的設計問題。您應該以開放態度討論不同的解決方案及其折衷方案。

您可以預期會和兩個「忍者」和一個「絕地武士」面試。有經驗的面試者通常也會得到與「海盜」面試的機會。

面試結束後，面試官在與其他面試官討論您的表現之前必須送交書面回饋。這可以確保您在一次面試中的表現不會對其他面試官的回饋產生影響。

一旦每個人都送交回饋後，您的面試團隊和招募經理就會一起合作做出最後的決定。他們達成一致決定，並向招募委員會送交最終的招募建議。

一定要準備的問題：

作為「精英」科技公司中最年輕的一家，Facebook 希望開發人員具有創業精神。在面試中，您應該表現出您喜歡快速建立東西。

他們想知道的是，您可以使用任何選擇的語言拼湊出一個優雅的、可擴縮的解決方案。是不是懂 PHP 並不是特別重要，因為 Facebook 另外還使用 C++、Python、Erlang 和其他語言做大量的後端工作。

特點：

Facebook「一般」是為公司整體面試一群開發人員，而不是為某個特定的團隊。如果您被錄用了，您將參加一個為期六週的「訓練營」，這將幫助您積累大量的程式碼經驗。您將從高階開發人員那裡獲得指導，學習最佳實作，若您在面試中曾被分配到某個專案，經過這階段訓練後能更自由性地選擇專案。

▶ Palantir 的面試

不像有些公司會為公司整體面試一群開發人員（即您面試的對象是整間公司，而不是裡面的特定團隊），Palantir 的面試是針對某個特定團隊。有時，您可能會被重新導向到另一個更合適的團隊去。

Palantir 的面試流程通常從兩個電話面試開始。這些面試大約 30 到 45 分鐘，主要是偏重在技術性主題。請連帶介紹一些您之前的經驗，特別是演算法問題。

您也被要求參加 HackerRank 程式碼考試，它將評估您撰寫最優演算法和正確程式碼的能力。較沒有經驗的人（例如社會新鮮人），特別有可能會受邀參加這樣的考試。

在這之後，成功通過考試的面試者會被邀請到公司，並將與最多五人進行面試。現場面試的內容會包含您以前的經驗、相關的領域知識、資料結構和演算法，以及設計…等。

您可能還會看到有人為您示範 Palantir 的產品，此時請拋出一些好的問題以展示您對公司的熱情。

面試結束後，面試官會和招募經理討論回饋。

一定要準備的問題：

Palantir 重視聘用到優秀的工程師。許多求職者表示，Palantir 的問題比他們在 Google 和其他頂尖公司遇到的問題更難回答。但這並不一定代表更難獲得工作機會（儘管這些問題確實可以讓人得不到工作機會）；這代表著面試官更喜歡有挑戰性的問題。如果您正在面試 Palantir，您應該徹底瞭解核心資料結構和演算法，集中精力準備最難的演算法問題。

如果您正在面試的職位屬於後端，也要溫習一下系統設計。這是整個過程中很重要的一部分。

特點：

撰寫程式碼是 Palantir 面試流程中常見的一部分。雖然在面試流程中您可以坐在電腦前，根據需要查找資料，但不要因為這樣就完全毫無準備地走進辦公室。這些問題可能極具挑戰性，您的演算法的效率將會被評分，做完整的面試準備將對您有所幫助。您也可以在 HackerRank.com 網站上練習程式設計挑戰題目。