卓越的工程師與優秀的工程師之間的界線：數學

Posted on 2015/05/04 伯樂在線

成為優秀的開發人員，可以沒有數學技能，但成為卓越的開發人員，不能沒有 。

不久之前，我開始思索數學。你也知道，到目前為止，我編寫軟體也有幾年了。老實說，在我的工作當中，我還沒有發現有關數學的需求。我要學習和掌握 許多新東西，包括語言、框架、工具、流程、溝通技巧和可以用來做你想到的任何東西的庫。在我學的新東西中，數學並沒有幫助。當然了，這不足為奇，我所做的 工作，大部分都是 CRUD 類型（編註：CRUD 是 Create、Read、Update 和 Delete 的首字母縮寫）。在網路時代，這也是我們多數開發人員所做的大部分工作。如果你做顧問，你主要是在做網站；你在大公司上班，你主要是在做網站；

你做自由職 業者，你主要是在做網站。我很清楚我是在總結，但請忍耐一下，我跑偏了。

最後你對此有些厭倦了，我也如此。別誤會我，這可以是項有趣並有挑戰性的工作，有機會解決問題，並和有趣的人一起互動，在工作時間做這個，我高 興。但在我個人時間中搭建更多的網站，這種想法已經稍微失去其光澤，於是你開始尋找一些更加有趣 / 酷 / 好玩的事情，我再一次地也如此。（所以，）有些人轉移到前台和圖像技術，比如視覺反饋就比較誘人。

但我並不是其中一員(雖然我和別人一樣都喜愛前台,但它真的不能讓我興奮。)這就是當我遇到一些搜索相關的問題時，我為什麼決定深入挖掘的原因 了。這把我帶回到故事的一開始，因為一旦我抓到第一把充滿搜索的鐵鏟，一旦我「撞到」數學時，我才真正意識到我的技能惡化的程度。數學並不像騎自行車，長 期不用就會忘記。

- **拓展視野**

多對搜索的一些了解,讓我接觸到各種有趣的軟體和計算機科學相關的事情和問題（包括機器學習、自然語言處理、算法分析等）。現在，在我接觸的各方 面，我都看到了數學，所以我更加強烈地感覺到自己技能缺乏。我已經意識到，如果你想利用計算機做又酷又有趣的事，你需要達到一個像樣的數學能力水平。

除了上面說的三個，還有一些，如：密碼學、遊戲人工智能、壓縮算法、遺傳算法、3D 圖形算法等。在理解之後,如果你想要編寫我們正討論的那些庫和工具，而不是僅僅使用它們（即：做一個「消費者」，而不是「生產者」），那你需要數學（知 識）來理解這些領域背後的你能應用的理論。即便如果你不想編寫任何庫，當你真正理解事情的原理，你在構建軟體時，它能給帶來更多的成就感，絕非僅僅把它們 連起來，就希望它們去做任何它們應該能做的。

雖然大多數開發人員會告訴你，他們在工作中從來不需要數學 (就像我前面說的:))，但是經過一番沉思後，我有了個想法（突發靈感）：就是反馬斯洛的錘子理論。你知道這個吧，當你有一把錘子，你會把一切看成是釘子。（注：伯樂在線編譯的《每位開發人員都應銘記的 10 句編程諺語》中的第 7 條就是錘子理論。）

這是一個隱喻，也就是說人們樂於使用自己鍾愛的工具，即便這並不是手中工作的最好工具。數學就是我們的一個相反的錘子。我們知道有這個錘子，但並 不太子的如何使用。所以，當我們遇到問題，我們的錘子是解決問題的最佳工具時，我們卻從未認真考慮過它。對我祖父而言，螺絲刀夠用了；對我父親來說，也很好；對我來說，同樣如此。誰還需要錘子？ **數學的技巧在於，人們懼怕它，甚至大多數程序員，你認為我們不會怕，但我們確實怕** 。 所以，我們把自己的話轉變為可以自我實現的預言。這並不是我在工作中不需要數學，這只是我真的不知道，即便我知道，我也不知道如何使用它。所以我並沒有使用它，當缺少某些東西時，如果你長期將就，不久後你甚至不會察覺它的缺失，所以對其需要更少了，這是自我實現的預言。

針對思索接近我們內心世界，這裡有一些的「糧食」——學習新技術。作為一名協作世界的開發人員，你努力成為一名通才型的專才（如果你不知道我在說 什麼，可以看看這本書《The Passionate Programmer:Creating A Remarkable Career In Software Development》）。你盡力在多數事情上做的體面，並在有些事情上做的優秀。

但是你擅長什麼？一般來說，人們會選擇一兩個框架或一門語言，然後與之相伴，這樣是不錯。但是要看到，框架和較小範圍內的語言都有保質期。如果你 要做一名 Hibernate、Rails 或 Struts 專家（使用 struts 的朋友現在真的應該擔憂一下了），當新框架取代當前的框架時，你在幾年內將不得不重新洗牌。所以，這也許是你真正的最好投資，但也可能不是。

另一方面，數學是不會很快消逝的。在我們領域中所做的一切，都是建立在穩固的數學原理之上（算法和數據結構正是這樣的例證），所以用在數學上的時間絕不是浪費，這不可辯論。再重複一次，總結起來就是： **要真正理解東西，而不是非死記硬背地使用。當涉及到計算機時，數學能有助你更深入地理解你所做的。事實上，正如 Steve Yegge 所言，作為 Coder 我們所做的事很像數學，只是我們甚至都沒有意識到這一點。**

- **什麼 / 誰造就了與眾不同？**

你不相信我？那請你想想：在我們的領域中，幾乎人人普遍尊敬的卓越工程師同樣也是大數學家。我是說像 唐納德·克努斯 、 艾茲格·迪傑斯特拉 、 諾姆·喬姆斯基 、 彼得·諾維格（Google 研究院總監）這一類人。但是這些傢伙並非真正的開發人員，他們是計算機科學家，這能真正算數麼？我再一次覺得，在我們寫出的純代碼行數能達到這些人所寫的十分之一之前，也許我們不應該再去討論這些問題了。

當然，不當科學家，你也能獲得成功和名譽，大家都聽過 加文·金（Gavin King，Hibernate 創始人）或 戴維·海涅梅艾爾·漢森（DHH，Ruby on Rails 創始人）。這還挺真實的（是不是有很多人聽說過加文和戴維，雖然這還有待確認），但是「聽說過」和普遍尊敬是不同的，這種差別就如同創建一個框架，和在你 的領域中為人類知識所做出的全部重大推動兩者之間的差別。（不要誤會我，我尊重加文和戴維，他們所做的事，遠遠超過我，但是這不能影響我所說的事實）。所 有的這些相關麼？我不知道，可能不相干，但在我們反省之後，我想無論如何要把它「扔掉」。

如今的世界正充滿著數據，每日都增加更多的數據。而在以前，我們在相對少量的數據下享受工作。我們今日編寫的軟體必須高效處理海量數據。甚至在協 作世界，這也是愈加明顯的事實。這也就是說，你更不可能只「啟動東西」，就想看其如何運作，因為你要處理的數據量將困住你，除非你非常了解它。我的預測是：算法分析將對於 Lay Programmer 越來越重要，以前不僅如此，以後也更加如此。如果要成為一位體面的算法設計專家，需要什麼？你猜到了，是一些數學技能。（編註：Lay Programmer 是指那些不認為自己是程序員的程序員，詳情 請見 Martin Fowler 的解釋 。我暫未想到合適的簡短叫法，如果哪位朋友知道，請在評論中說明。）

所以，我該怎麼辦呢？嗯，我已決定一點一點地建立或恢復我的數學技能，雖然還有大量的書要看，大量的代碼要寫，但我會盡力抽時間放在數學上，這就 像鍛煉，時不常的鍛煉總聊勝於無（再次引用 Steve Yegge 的話）。說到數學，我袖中當然還藏有一張王牌，它對我有利，但很幸運，有這個部落格，我們都會受益的。（我知道你好奇，一會告訴你:)）。

- 你在 5 年內的規劃如何？

那麼，數學對所有事都有利麼？這事先很難說，我對我現在的處境十分滿意，或許你也如此，但這都和潛能有關係。如果你是協作世界的一名開發人員，你 真的不需要數學。如果你樂於你的整個職業生涯是這樣的：在工作時間中做企業 CRUD 應用，或在閒暇時間滑翔跳傘或極限水上滑板（或其他各種時髦的極客運動），也分配較多時間在 Spring 、Hibernate、Visual Studio 或其它東西上。（其實）那些特殊的職位並沒有真正限制你的潛力，你能變得極具價值，甚至可深入追求。

但是如果你想為多樣化的職業生涯而奮鬥，想要有能力嘗試幾乎所有涉及代碼的事，從信息檢索到 Linux 內核。總之，如果你想成為一個開發人員、工程師和計算機科學家的完美組合，你必須確保你的數學技能達到標準（哎，你還是可以去玩滑翔跳傘或極限水上滑 板）。長話短說，如果你在數學方面有一定天賦，

那在軟件開發領域中沒有向你關著的門，如果沒有，那一切都是 CRUD 型工作！

（本文轉載自合作夥伴《伯樂在線》，未經授權，不得轉載；圖片來源：hackNY，CC Licensed）