程式設計心得分享

中山資工95級　莊易霖　(2006/09)

在大學的時候我學了演算法和資料結構,在演算法中大部分時間可能談的都是一個可以在 polynomial 時間內可以被 solve 的問題,最明顯的例子不外乎就是 ACM 的題目了，很多看似很難的問題透過一些常見的技術：dynamic programming, divide and conquer 或是一些 tree searching 的問題，都可以得到所謂的 optimal solution，但是實際上我們要處理的問題往往不是這種問題，在研究所裡 (或是未來到工作職場上)真正接觸的都是 NP-complete 甚至是 NP-hard 的問題，這種問題的共同特性都是我們沒辦法在 polynomial time 中找到"optimal solution"，因此很多 heuristic 方法被人提出，但其實我們觀察這些解法，他們背後所根據的技術其實也是我們學過的基本技術,因此我認為熟稔這些基本的技巧其實是很重要的。

楊老師的高等程式設計透過大家一起討論 ACM 的題目可以得到腦力激盪，我覺得這在訓練演算法設計上有很大的幫助,當你聽到別人的演算法不妨去思考一下為什麼他要這麼做，這麼做有什麼好處，我是不是可以改良他的方法，當時我在修課的時候大家就會在課堂上一起討論,如果可以找出對方演算法的問題幾乎可以說這個問題就被你解了一半了,我也十分建議如果遇到經典問題應該都要自己實做一次，像是 network flow 或 shortest path 的問題，雖然他們演算法很簡單,但是自己親自實做一次你很有可能會發現實際上要寫好也不是那麼容易的，這就會促使你想更進步,因此我十分鼓勵各位學弟妹有機會可以修高等程設這堂課，其中會學到很多 programming 的方法。

其次我認為重要性不亞於演算法的就是資料結構，在 ACM 的訓練中，題目可能比較重視演算法的實作，因此資料結構幾乎都是 apply 一些內建的資料型態就已經足夠，往往我們很少考慮效率的問題，事實上"效率"這個問題是存在的，當你的程式必須反覆的對幾十萬的 element 做運算的時候，你使用 tree 和使用線性 array 在效率上絕對就有很大的差別，甚至如果資料結構選的好或是實作的很有效率，即使你的演算法時間複雜度比人家大，可能執行速度也會比別人快。

因此在資料結構上我建議除了課堂上的學習之外,應該自己練習實做這些資料型態，不但對這些型態有更深的了解，甚至自己在寫程式時也可以擁有更多的彈性。

我也建議大家不妨有空多看看其他的 coding style，每一個人寫程式習慣不同,看別人程式學習好的寫作方式我覺得是進步最快的途徑，最後祝福每位學弟妹畢業後都能成為一個優秀的 problem solver