IntriCode: 程式評鑑系統之設計與實作

何亮頡 [1]　　江晨銘 [1]　　姜柏仰 [1]　　吳宜鴻 [2*]
中原大學資訊工程系
[1]E-mail: {jason89923、gugugu1127、10927110}@cycu.org.tw
[2]E-mail: yhwu@cycu.edu.tw

摘要

本研究設計與實作一個程式評鑑系統，其動機來自於進階程式設計課程欠缺一個通用的自動化批改系統，考量到學生與教師的需求，分別提供了三個主要功能：讓學生操作範例程式以理解題目的要求、讓學生測試不同版本並記錄過程供教師參考、讓教師瀏覽所有作業的檢測結果以便評分，本系統已實施於三門不同的課程，共計 241 位學生。

關鍵字：*資料探勘、程式教育、自動化批改。*

Abstract

This study designs and implements a program evaluation system, motivated from the lack of one such system for advanced programming courses. With the consideration of the requirements from students and teachers, we respectively provide three main functionalities: letting students operate the example program to understand the problem requests, letting students test different versions and recording the entire process to be referred by teachers, letting teachers browse all the evaluation results to make scores. Our system has been used in three difference courses by 241 students in total.

Keywords: *data mining, programming education, online judge.*

1. 研究背景與目的

在程式教育領域中，自動化批改能節省教師的時間從而應對更大規模的學生群體，蒐集整個過程的資料進行分析，可觀測學生群體趨勢或個體差異，進而及時反饋以提升教學成效。初階程式設計課程已廣泛採用線上解題系統或自動批改系統，前者如 LeetCode 及 CodeFlex [1]，後者如 ArTEMiS [2]，這些系統能自動執行學生上傳的程式碼，即時判定正確性並針對錯誤給予提示，對學習有正面影響，也能幫助教師掌握學生的學習狀況。

隨著這些系統的盛行，初階程式設計累積大量的使用資料，透過資料探勘技術能夠分析學生的學習行為、常見錯誤模式等，進而提供個人化的學習建議，使智能化教學成為可能。例如：根據學生提交程式碼的執行狀態，歸納不同學生群體在特定題目上特殊的錯誤傾向，藉以預測學生可能遇到的困難，適時提供必要的協助。

資料結構、演算法和程式語言等進階程式設計課程，其作業題目不像初階程式設計課程僅處理數理邏輯問題或將問題簡化，通常允許多種解決方法，並要求學生不僅能正確處理輸入和輸出，還要能與作業系統互動，例如檔案讀寫、時間測量以及優化執行效率。這種作業形式尚未被實現於既有的線上解題系統，因其缺乏大量輸出資料或檔案的快速且有效檢測[3]，儘管程式競賽平台如 HackerRank 有支援輸出檔案的檢測，其本質與線上解題系統相同，傾向簡化問題設計，目前尚無系統是針對進階程式設計課程而且被廣泛採用的。

有鑒於此，本研究創建一個新穎而通用於不同進階程式設計課程的系統，名為 IntriCode；其功能包括提供學生即時反饋、自動編譯執行與檢測結果、為教師提供自動批改、視覺化呈現學生作業和範例程式的輸出差異、提供教師瀏覽程式碼、線上執行和查詢版本測試的變化軌跡。整體而言，系統是從標準輸出、檔案輸出和執行時間三個層面檢驗作業

*通訊作者

程式碼的正確性；應用於不同課程蒐集使用資料，運用資料探勘技術分析個別學習行為和群體趨勢，推動智能化教學工具的研究發展。

2. 研究方法與步驟

IntriCode 由三個子系統構成，如圖 1 所示，分別為範例程式展示網頁、程式自測平台以及程式執行暨管理介面。教師先上傳範例程式至範例程式展示網頁，學生在網頁上執行範例程式、輸入指令和觀看結果，系統可藉此蒐集所有學生的操作習性。

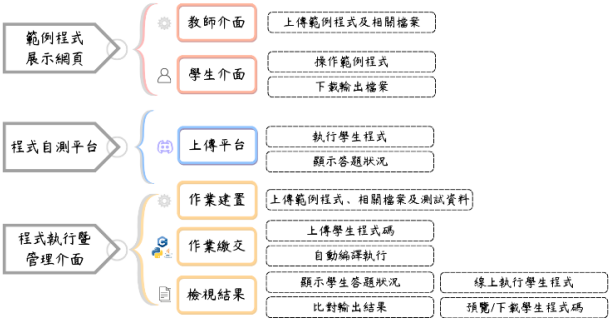

圖 1 IntriCode 系統的主要功能

學生上傳一個版本至程式自測平台後，系統會執行自動化檢測並在答案錯誤時給予提示，並顯示已通過測資的比例，系統可擷取個別學生通過測資的時間序列。程式執行暨管理介面視覺化呈現學生作業與範例程式的執行速度和輸出差異，輔助教師評分，同樣可蒐集不同學生群體的作業表現。

範例程式展示網頁前端使用 HTML、CSS 和 JavaScript，透過 WebSocket 協議與伺服端進行主動的雙向通訊，藉此在網頁中顯示虛擬終端機(terminal emulator)。後端會即時監控輸出檔案的變化，程式新生成的檔案會顯示於網頁上提供下載。

程式自測平台採用 Discord 機器人作為學生登入介面，須建立 Discord 頻道後授權機器人加入，學生可在頻道中上傳程式碼或執行命令查詢當前答題狀況，程式碼傳送至後端的伺服器會自動編譯執行並以卡片式訊息回覆結果。

程式執行暨管理介面是基於 Next.js 的 React 框架開發，Next.js 支援透過網址的查詢參數實現動態路由，能夠向後端發送請求並取得所需資料。在取得資料後透過伺服端渲染(SSR)預先處理頁面內容，能提升網頁加載速度，在「檢視結果」功能中有大量頁面對前端造成壓力，Next.js 可顯著提升頁面響應速度。為了提供更多元的登入選項，我們引入 Clerk.io 作為權限控制與身份管理工具，提供多種第三方登入選項，並具備管理介面以進行權限控制，本介面依賴正確的權限設定防止非教師身份的使用者進入系統。

系統的實作架構為多個客戶端與單一伺服端，如圖 2 所示，前述三項子系統皆是連接到同一個伺服端的客戶端，伺服端以 Node.js 實作，包含 API 模組、編譯模組與執行模組，使用 JavaScript 實作並由 Node.js 運行。共用伺服端可以降低伺服器的數量需求，也有助於共享資料並減少冗餘程式碼。

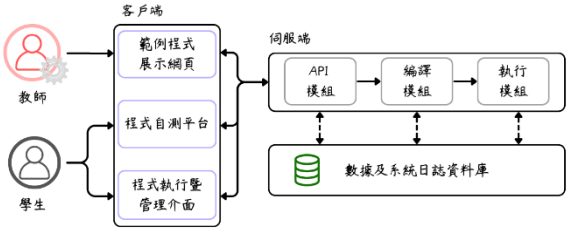

圖 2 IntriCode 系統的實作架構

API 模組負責響應來自前端的請求，編譯模組負責編譯程式碼，執行模組則運行編譯後的程式並返回結果。伺服端的三個模組皆為獨立運作的程序(process)，藉由 Redis 的 pub/sub 機制協同運作而形成管道(pipeline)，大幅降低使用者上傳後的等待時間。Redis 的 pub 模組負責發佈訊息，sub 模組負責訂閱和接收訊息；本系統的執行模組訂閱編譯模組，編譯模組訂閱 API 模組，當 API 模組接收到前端請求後，會觸發此管道進行非同步處理。

執行模組為雙層安全架構[4]，不同程式語言使用對應的 Docker 容器執行，Docker 提供輕量級的作業系統虛擬，能將容器與宿主系統隔離，容器設定阻斷網路連線和限制 CPU 與記憶體使用量，並在容器內部透過沙箱(sandbox)進一步限制程式能存取的檔案路徑；沙箱為執行中的程式提供隔離環境，即使遭遇惡意程式碼攻擊也無法損害主系統。

3. 研究結果與討論

操作圖 3 的範例程式展示網頁可透過虛擬終端機與範例程式互動，範例程式執行過程產生的檔案會顯示於右側清單，可線上預覽或下載。

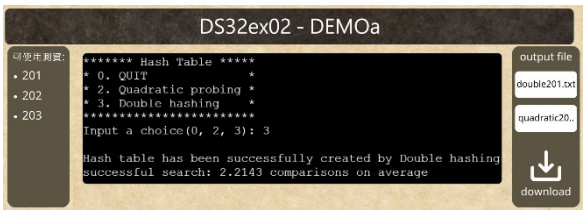

圖 3 範例程式展示網頁

傳送程式碼至圖 4 的程式碼自測平台之後，系統會依序執行既定的測試數據，並回應多則訊息，包含執行狀態、執行結果和提示，執行狀態以線上解題系統術語表示錯誤種類，例如 WA (Wrong Answer)、RE (Runtime Error)等。提示為兩個文字檔案，包含範例程式和學生作業的輸出，學生解讀這些內容即可修正其程式碼。

圖 4 程式碼自測平台

本研究假定不同測試數據之間存在相依性，如圖 5 所示，每個節點表示一組測試數據，A→B 表示若測試 A 未通過，則測試 B 不可能通過；例如印出費氏數列程式碼的二種測試，A 是印出第 3 個費氏數，B 是印出第 10 個費氏數，即可表示為 A→B。據此將所有節點串聯，可構建有向無環圖[5]，稱為相依性關聯圖；學生多次上傳所形成的相依性關聯圖狀態序列稱為學習軌跡，表示不同學生的學習進展。

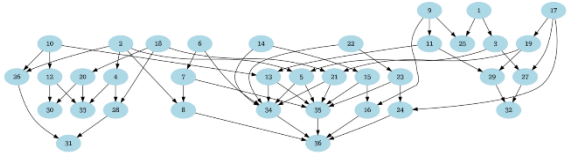

圖 5 相依性關聯圖

學生每次上傳作業時會遍歷相依性關聯圖，為了避免不同學生之間的交流干擾學習軌跡，我們基於卡恩演算法[6]設計了一個變體，首先以學生的學號作為亂數種子為每個節點生成權重，並建立一個以權重為基礎的優先佇列；接下來依據卡恩演算法的流程，尋找所有入度為 0 的節點，將其逐一加入優先佇列；反覆從佇列中取出節點並執行該節點對應的測試，若測試未通過，則記錄該筆測試；若測試通過，將該節點所指向的所有節點入度減 1，若有節點入度變為 0 則將其加入優先佇列。以上流程重複執行，直至佇列為空或累計錯誤滿 5 筆。

在相依性關聯圖中，我們將每次上傳版本通過的測試數量比率表示為該版本的狀態；而學生的學習軌跡則是由該比率組成的序列，該序列可視為一個向量。我們在人數較少的課程中進行小規模實驗，針對所有學生的學習軌跡進行兩兩距離計算，並使用層次聚類將不同學生的學習軌跡聚合為一個樹狀結構，如圖 6 所示。樹狀結構中，學號相鄰的學生顯示其學習軌跡相似。值得注意的是，從學生訪談得知，左子樹中的三位學生在過程中親自撰寫程式碼，而右子樹中的兩位學生全程使用生成式 AI。

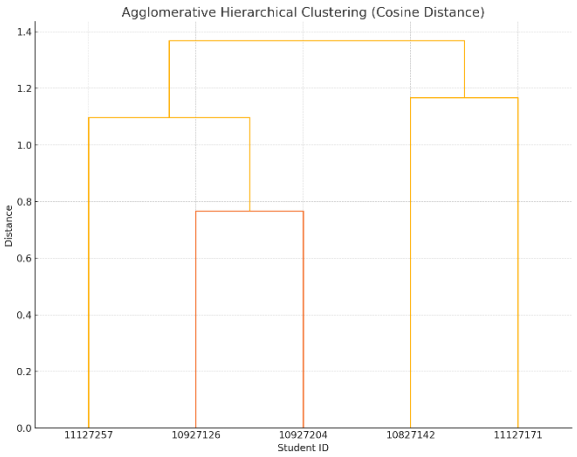

圖 6 學習軌跡之聚類分析

為了進一步探究學習軌跡、學習成效與生成式 AI 之間的關聯性，我們擴大實驗對象至 140 人，使用上述方法計算學生軌跡之間的距離，並使用 k-means 分群，再利用假說檢定方法驗證不同群學生的學習成效是否有顯著差異。學習成效分為兩個分數，程式碼評分與上機測驗評分，程式碼評分經由人工檢驗其正確性所得；上機測驗的過程教師會針對學生的程式碼提出若干問題檢驗其信心度，其分數反映程式碼是否為抄襲或由 AI 工具生成。

使用 Kruskal-Wallis H[7]檢定的實驗結果如

表 1 所示，設定顯著水準為 0.05，對於兩種學習成效，透過學習軌跡的分群皆存在顯著差異，顯示此方法不僅能區分不同程度的學生，更具有辨別程式碼是否源於生成式 AI 的潛力。

表 1 以 Kruskal-Wallis H 檢驗學習成效

學習成效	k-means 之最佳 k 值	P 值	H 值
程式碼	3	0.0005	15.3480
上機測驗	3	0.0419	6.3452

圖 7 的程式執行暨管理介面將顯示所有學生作業的執行狀態，左側的 Rank 表示此作業的執行速度排名，紅字百分比表示那份程式碼輸出結果的正確率，正確率的計算是本研究針對此問題設計的特定演算法，僅以既有的演算法直接比對輸出結果來評估正確性，基於程式碼輸出的結構性特徵，一旦出現少量錯誤就會導致相似度數值會被大幅降低，難以精確反映真實正確性；此外當輸出結果包含大量文字，既有方法難以在有限時間內完成計算。本系統目前採用的計算方式為移除空白字元後所有字元出現次數形成的分布向量，與範例程式的分布向量計算餘弦值作為相似度評估。

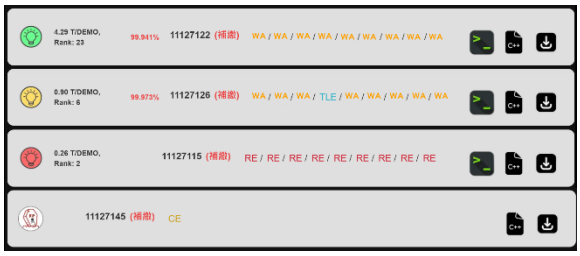

圖 7 程式碼自測平台

學號右側的執行狀態與圖 4 定義相同，學生作業與範例程式的輸出不完全相同便顯示為 WA，異常退出則顯示為 RE，編譯錯誤顯示為 CE。右側三個圖示由左至右依序為執行、預覽和下載，執行功能允許教師線上執行學生的程式碼，預覽功能可瀏覽程式碼，而下載功能會提供一個壓縮檔，內含程式碼及其他執行所需的相關檔案，方便教師下載於本機執行。

本系統已實際上線用於三門不同的程式設計課程，合計 241 位同學使用，總計 11 次不同作業一共上傳 2212 次，課程助教和教師均透過本系統批改作業，相較於手動批改的效率有明顯提升。

針對其中一門課程，程式碼自測平台進行了系統易用性量表(System Usability Scale, SUS)調查[8]，35 位學生中有 19 人填答，占 54%。調查結果顯示，該系統的平均 SUS 分數為 69.5，標準差為 14，略高於平均水準(68 分)，顯示受測學生對系統評價尚可。質化研究中發現，少數學生因缺席系統說明而遭遇困難，後續將於系統增加 help 指令，以降低使用門檻。

4. 結論與未來研究

本研究創建一個不同進階程式設計課程通用的系統，並且實施於三門不同的課程，一共 241 位同學使用過本系統。範例程式展示網頁提供學生在線上操作範例程式，幫助學生理解題目的要求，並檢視程式運行的詳細過程；程式自測平台提供自動化的程式碼測試環境，能即時回應執行情況並給予錯誤提示，協助學生偵錯；程式執行暨管理介面則呈現學生作業的檢測結果，便於教師檢視及批改。

未來對於本系統在不同課程蒐集到的資料，將以資料探勘技術進行分析，作為智能化教學工具的核心。範例程式展示網頁可分析所有學生測試範例程式的習性，預測學生可能遇到的困難。程式自測平台可分析個別學生通過測資的時間序列，在偵測到軌跡符合特定模式時提供預警，並辨識程式碼是否源於生成式 AI。程式執行暨管理介面可蒐集與分析不同學生群體的學習表現，藉此找出在特定題目上的錯誤傾向。

參考文獻

[1] M. Brito and C. Gonçalves, Codeflex: A Web-based Platform for Competitive Programming, CISTI, 14: 1-6, 2019.

[2] S. Krusche and A. Seitz, ArTEMiS: An Automatic Assessment Management System for Interactive Learning, SIGCSE, pp. 284-289, 2018.

[3] S. Wasik, M. Antczak, J. Badura, A. Laskowski,

and T. Sternal, A Survey on Online Judge Systems and Their Applications, ACM Comput. Surv., 51(1): 1-34, 2018.

[4] J. Carlos Paiva, J. Paulo Leal, and Á Figueira, Automated Assessment in Computer Science Education: A State-of-the-Art Review, ACM Trans. Comput. Educ. 22(3): 1-40, 2022.

[5] S. Haidry, and T. Miller, Using Dependency Structures for Prioritization of Functional Test Suites, IEEE Trans. Software Eng. 39(2): 258-275, 2013.

[6] A. B. Kahn "Topological sorting of large networks." Communications of the ACM, 5(11), 558-562, 1962.

[7] W. H. Kruskal and W. A. Wallis, "Use of Ranks in One-Criterion Variance Analysis," Journal of the American Statistical Association, 47(260), 583-621, 1952.

[8] J. Brooke, SUS: A Quick and Dirty Usability Scale. Usability Evaluation In Industry, 189: 4–7, 1996.