**HTML5期末專題報告**

**《圖論實作》**

**學生：柯允祈**

**學號：19103105**

**指導教授：蘇怡仁教授**

**目錄**

[一、 專題開發動機 3](#_Toc123419694)

[二、 系統架構及流程 4](#_Toc123419695)

[三、 使用套件及其他軟體 5](#_Toc123419696)

[四、 功能介紹 6](#_Toc123419697)

[乙、 鄰接矩陣 7](#_Toc123419698)

[丙、 功能列表 8](#_Toc123419699)

[丁、 路徑演算法 9](#_Toc123419700)

[i. 最短路徑：戴克斯特拉演算法 9](#_Toc123419701)

[ii. 圖遍歷演算法 9](#_Toc123419702)

[iii. 圖遍歷：最近距離搜索演算法 9](#_Toc123419703)

[iv. 圖遍歷：混合演算法 10](#_Toc123419704)

[v. 兩種圖遍歷演算法的統計 11](#_Toc123419705)

[vi. 最小權重生成樹：普林演算法 12](#_Toc123419706)

[五、 參考資料及靈感來源 13](#_Toc123419707)

[六、 心得 14](#_Toc123419708)

# 專題開發動機

在思考專題該寫甚麼時，我想起了我在 Youtube 上訂閱的一個名叫「Reducible\*」的頻道，他的影片多是圖像化數學與程式邏輯的教學影片，我有一些演算法的知識都是來自他的頻道。其中，有一部影片令我感到好奇，「The Traveling Salesman Problem: When Good Enough Beats Perfect」，那是一部有關圖遍歷演算法的科普影片。在看了整部影片後，我想，何不就以此來作為我的專題題目？

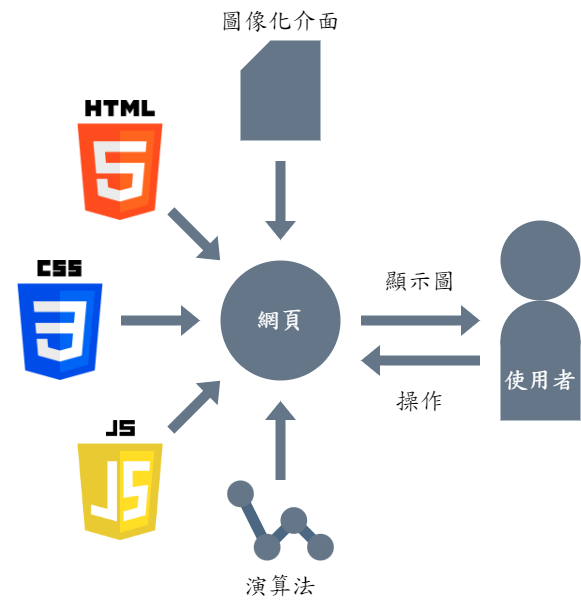
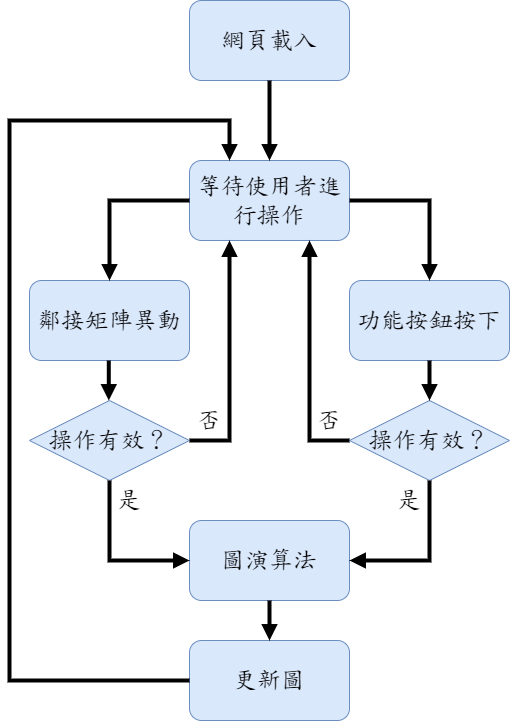
然而在實作圖遍歷演算法的過程中，我發現不只是在該影片當中，在網路上的各處幾乎都是以「完全無向圖」作為圖遍歷演算法的對象，我想或許我能不靠網路，自己寫出「不完全有向圖」的圖遍歷演算法，這是本專題一大核心主軸。

當然，以我目前的技術與知識仍無法寫出能夠在短時間內得出圖遍歷最佳解的演算法，對此我設計了一套規則，用於比對我自己的演算法與「最鄰近搜索」，哪個演算法能算出總距離較短的圖遍歷路徑。

於是，在專題方向確定後我便開始設計版面並思考程式的流程與架構，最後完成了這份期末專題《圖論實作》。

# 系統架構及流程

本專題是結合 HTML5、Javascript及CSS 為主的網頁作品，以圖像化的功能按鈕供使用者實際操作由節點（Node）、邊（Edge）及演算法所計算而出的圖。



圖一　系統架構圖

圖二　流程圖

# 使用套件及其他軟體

本專題共有使用以下套件：

* html2canvas：能將HTML元素轉為圖檔的套件。

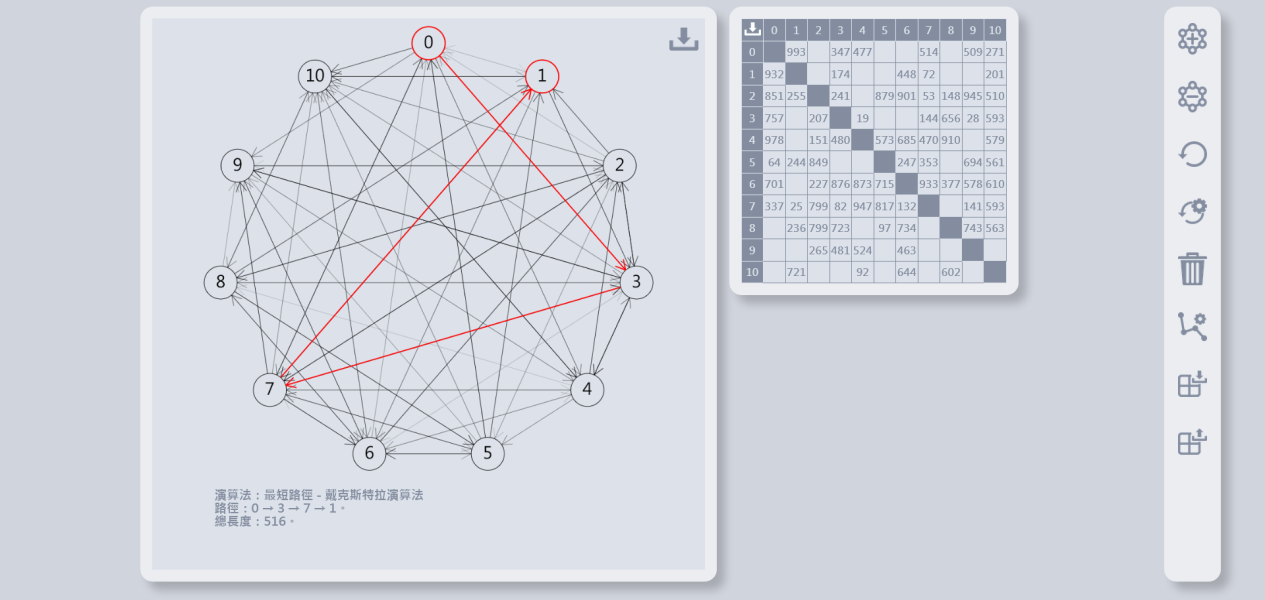
本專題開發過程中，共有使用以下軟體：

* Visual Studio Code：用於開發程式碼。
* Firefox Browser：用於運行網頁。
* Microsoft Excel：用於統計圖遍歷演算法數據。
* Adobe XD：用於設計版面。
* Adobe Illustrator：用於繪製功能按鈕圖示。
* Diagrams：用於繪製系統架構及流程圖。

# 功能介紹

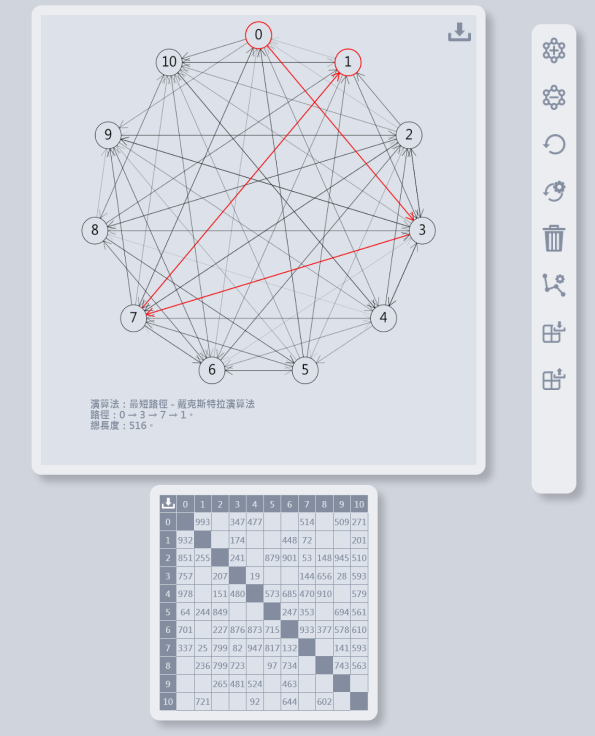
網頁主畫面分為三大區塊，從左到右分別為：

1. 顯示圖：包含圖像化的圖、文字方塊以及圖檔下載按鈕。
2. 鄰接矩陣：欄位皆代表邊的權重；左上角是圖檔下載按鈕。
3. 功能列表：圖像化的各類按鈕，以便於操作圖的變化。

正常的畫面排版如下：

圖三

畫面的排版會根據視窗的寬度而異動，若寬度過小則會以右圖呈現。



圖四

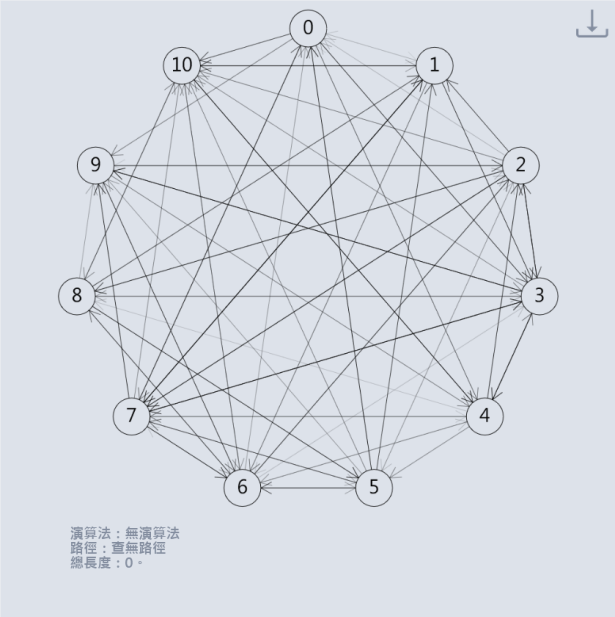
顯示圖的位置會維持在左上角，功能列表也會維持在右側，隨畫面移動\*，鄰接矩陣則會被推至畫面下方，需要向下滾動畫面才能看見。

以下將個別解釋區塊的功能及細節。

\*註：圖四是全頁截圖，因此功能列表看似是在右上角。以實際操作而言功能列表會一直維持在同樣的右側位置。

* 1. **顯示圖**

本專題的圖是以Canvas元素實現。圖為「有向圖」，由多個節點及邊所構成。節點呈環形排列，以圓及數字表示。邊則以箭頭表示其方向，灰階度會以其權重大小而異，權重小則愈深（即愈近）、權重大則愈淺（即愈遠）。



圖四

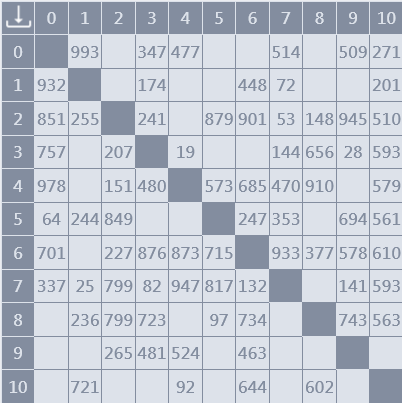
節點的排列位置、箭頭的起訖位置、圖形皆是以三角函數來計算，以確保箭頭的起迄位置皆在節點的圓上。

圖的底部是文字方塊，用於化解圖片無法一眼看清的複雜度。右上角是下載圖檔的按鈕，會將圖與文字方塊一同下載為 .png檔（不含下載按鈕圖示）。

有關如何將HTML元素轉換為圖檔請詳見**參考資料及靈感來源**章節。

## 鄰接矩陣

鄰接矩陣是以Table及Input元素實現。標題欄位的數字代表其對應節點，顏色較淺的則是邊的權重欄位。



圖五

本專題限制節點不可與自身鄰接，且兩節點間同一方向只允許存在一條邊。

權重欄位內置Input元素，可直接輸入數字。輸入數字後（onkeyup事件）圖就會被重新繪製。Input元素的極限值為0~999。

左上角的下載按鈕與顯示圖一樣，能夠將鄰接矩陣轉化為 .png檔並進行下載。

## 功能列表

每當按鈕被按下時，圖會根據功能而更新，按鈕皆配置了快捷鍵以便操作。滑鼠移至按鈕上方（onhover事件）及按下（onmousedown事件）時，按鈕圖像會有顏色改變的效果。部分的設定會被儲存至Local Storage。

按鈕功能由上到下分別為：

|  |  |  |  |
| --- | --- | --- | --- |
| **名稱** | **功能** | **圖像** | **快捷鍵** |
| 新增節點 | 於圖及鄰接矩陣動態新增一個節點。 |  | + |
| 刪除節點 | 於圖及鄰接矩陣動態刪除一個節點。 |  | - |
| 隨機鄰接矩陣 | 隨機賦予圖及鄰接矩陣的邊。 |  | Ｒ |
| 隨機鄰接矩陣設定 | 設定「隨機鄰接矩陣」功能被按下時，邊的極限值與密集度\*。（圖六） |  | Ａ |
| 清空鄰接矩陣 | 刪除圖及鄰接矩陣所有的邊。 |  | Ｃ |
| 路徑設定 | 即路徑演算法。路徑有多個選項，每種路徑皆是以演算法實現（圖七）。路徑演算法請詳見**路徑演算法**章節。 |  | Ｐ |
| 下載鄰接矩陣 | 下載鄰接矩陣的Json檔。 |  | D |
| 上傳鄰接矩陣 | 上傳鄰接矩陣的Json檔。上傳後會複寫鄰接矩陣。 |  | U |

鄰接矩陣的新增、刪除節點是以動態更新的計算方法實現，只會局部更新需要異動的部分。

\*註：邊的密集度是指當「隨機鄰接矩陣」功能被按下時，會有多少比例的節點被建立。



圖七

圖六

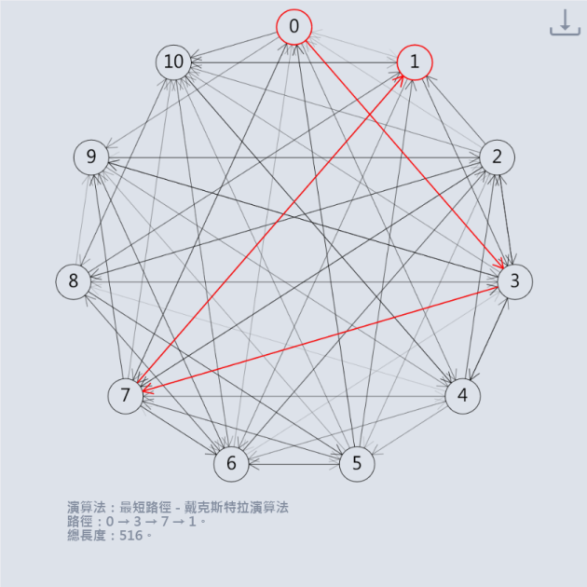
## 路徑演算法

路徑演算法共有以下三類：

1. 最短路徑：戴克斯特拉演算法。
2. 圖遍歷（自動取較優者）：
   * 最近距離搜索演算法。
   * 混合演算法。
3. 最小生成樹：貪婪演算法。

### 最短路徑：戴克斯特拉演算法

如右圖，戴克斯特拉演算法會根據功能列表中的設定來繪製最短路徑。起、終點及路徑皆以紅線顯示。



圖八

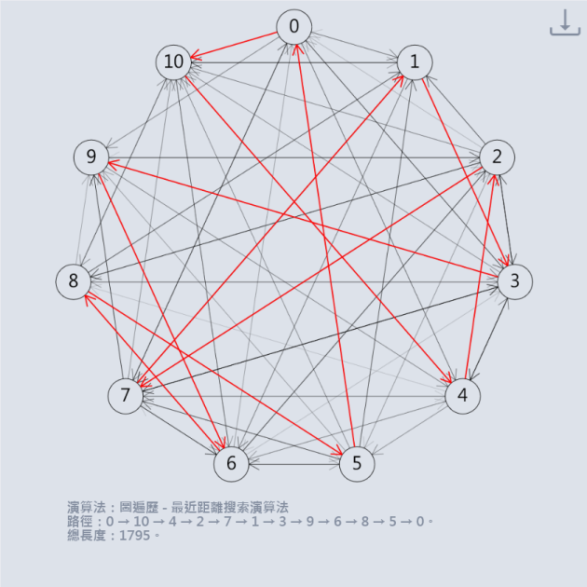
若查無路徑，則會在文字方塊中顯示「查無路徑」。

### 圖遍歷演算法

本專題的圖遍歷演算法共有兩種：「最近距離搜索法演算法」及「混合演算法」。同一節點及邊都被行經無限次。

### 圖遍歷：最近距離搜索演算法

我在網路上參考了許多圖遍歷演算法，很可惜絕大部分都是以「完全圖」作為對象，無法直接套用於本專題的非完全圖。

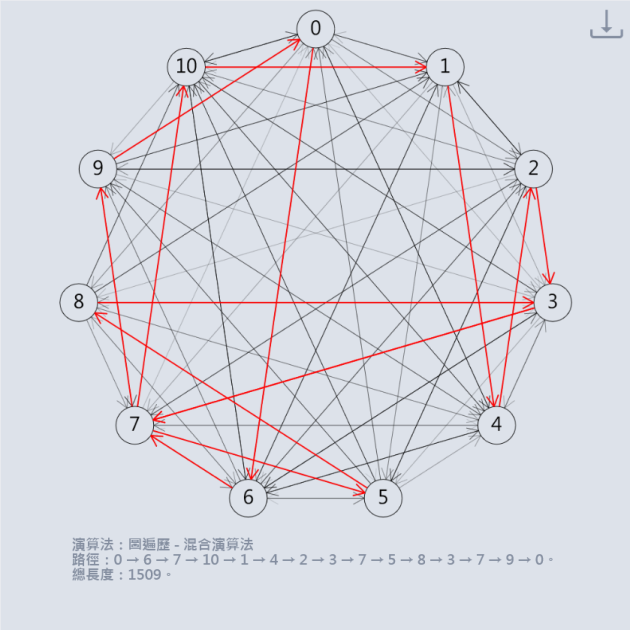


圖九

其中一個勉強能使用的演算法就是最近距離搜索演算法―不斷的查詢離陣列末端最近的未造訪節點並與之建立邊。

### 圖遍歷：混合演算法

另一個圖遍歷演算法是我在網路上查閱資料時，偶然看見某論壇上的[一則貼文](https://cs.stackexchange.com/questions/1749/dijsktras-algorithm-applied-to-travelling-salesman-problem)，其中探討如果將最短路徑演算法套用於圖遍歷之上，會擦出甚麼火花？不過底下的留言一面倒地反對此提案，並提出了實例來證明其不可靠性。



圖十

然而，留言是使用完全圖來做反證，而他的點子正好能讓我套用於非完全圖的圖遍歷算法。於是乎，我遍重新設計了此演算法，使其瘋狂程度更上一層樓。

由於其名未知，在此暫稱其為**混合演算法**（**Hybrid Algorithm**）。

其邏輯為：

1. 從0 ~ N，依序選擇節點作為起點。
2. 初始化路徑P為空陣列。
3. 將起點放入路徑P。
4. 取得路徑P的末項至所有未造訪節點的最短路徑（使用了快取減少重複計算）。
5. 在所有最短路徑當中，取得「行經未造訪節點最多」或「行經未造訪節點一樣多，但距離最短」之路徑，並將其放入路徑P。
6. 回到步驟4，直到所有節點都被造訪為止。
7. 取得路徑P末項至起點的最短距離並放入路徑P。
8. 記錄此次圖遍歷。
9. 回到步驟1，直到所有節點都當過起點為止。
10. 取得所有圖遍歷路徑當中總距離最短者。

根據其計算度的複雜程度或許我該改稱其為「**混沌演算法**（**~~High~~** **Chaos Algorithm**）」。不過在最大節點數為18的本專題當中尚不會導致卡頓。

### 兩種圖遍歷演算法的統計

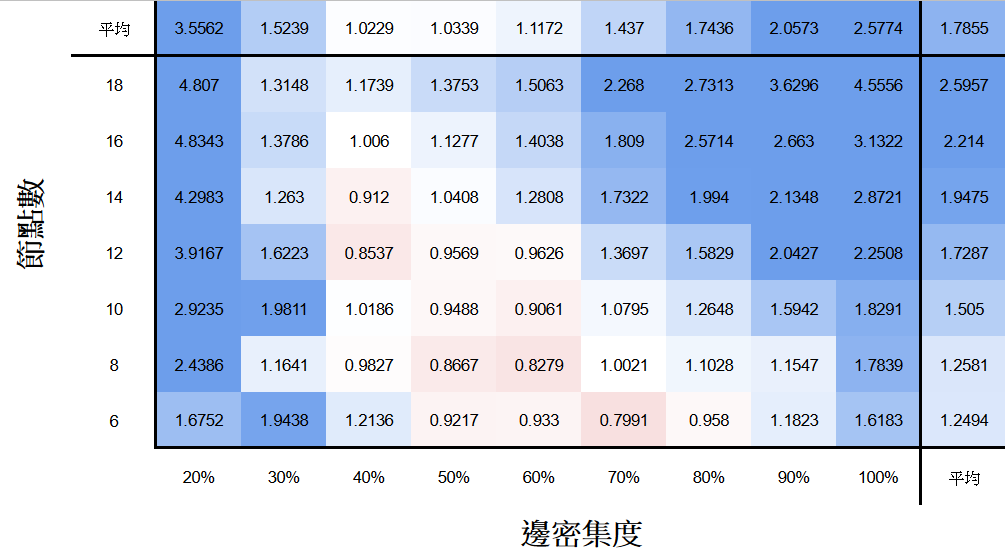
以下將統計混合演算法與最近距離搜索演算法的勝敗比。統計樣本共有63000份。節點樣本數由6到18共有7份；邊密集度樣本數由20%到100%共有9份。交叉統計節點數與邊密集度，統計區間共有63個，每區間共計算1000次，並取「混合演算法勝數 / 最近距離搜索演算法勝數」。

「勝方」判定方法如下：

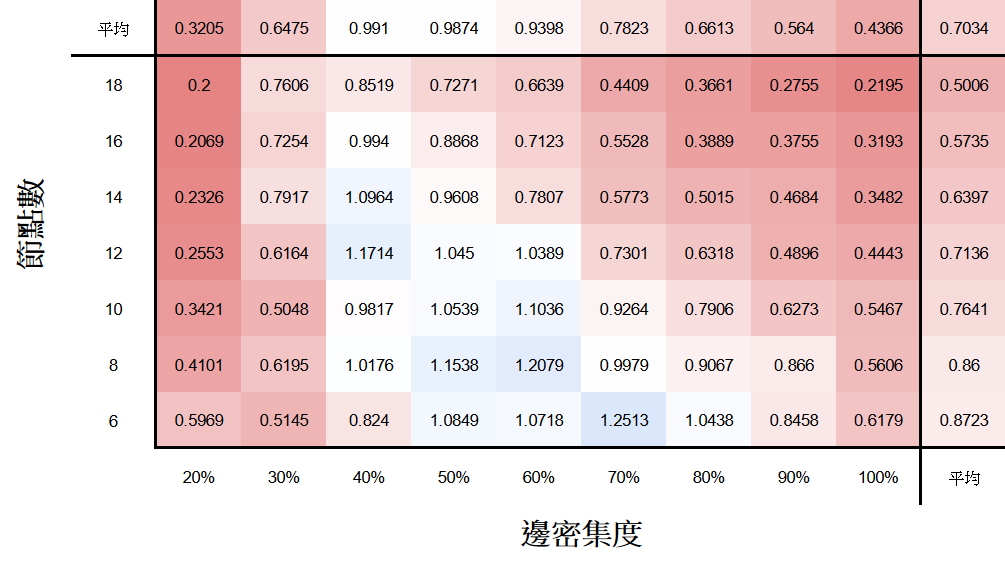
* 在另一路徑不完整的情況下路徑完整者（指成功遍歷圖）。
* 兩路徑皆完整但總長度較短者。
* 兩路徑皆完整、總長度一致但不重複的行經節點較少者。
* 兩路徑皆不完整，但不重複行經節點較多者。

勝比較高的區間顯示藍色，敗比較高顯示紅色，差距越小則越白。由於計算方法會導致勝比無上限，藍色深度最多只有2.0000，其以上皆顯示同一深藍色；紅色深度最少是0。

例：在混合演算法勝敗比統計圖中可見，節點數為6、邊密集度為20%的區間（左下角）當中，混合演算法的勝數比最近距離搜索演算法的勝數還要多1.6752倍。

****混合演算法勝敗比：

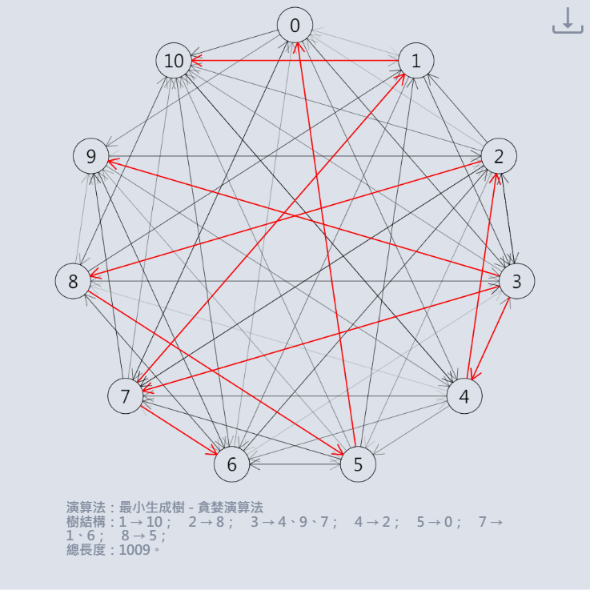
圖十一

最近距離搜索演算法勝敗比：

根據以上統計的平均數據可知，混合演算法在邊密集度20~30%、80~100%的區間較有優勢，且節點數愈高就愈有優勢。最近搜尋演算法則在節點數6~14且邊密集度40~70%較有優勢。

不過若是論及演算法的複雜度的話，基於混合演算法使用了大量的最短路徑演算法，在複雜度上恐怕會遠輸於最近距離搜索演算法。

### 最小權重生成樹：普林演算法

普林演算法將計算出一棵能將所有節點連結在一起的最小權重樹狀結構。

其計算步驟請詳見[這裡](https://zh.wikipedia.org/zh-tw/%E6%99%AE%E6%9E%97%E5%A7%86%E7%AE%97%E6%B3%95#.E4.BE.8B.E7.A4.BA)。

# 參考資料及靈感來源

本專題參考資料如下：

* [如何隱藏滾動條](https://blog.hubspot.com/website/hide-scrollbar-css)
* [如何將圖片轉白](https://stackoverflow.com/questions/24224)
* [圖片顏色過濾生成器](https://codepen.io/sosuke/pen/Pjoqqp)
* [如何產生HTML物件陰影](https://www.w3schools.com/css/css3_shadows.asp)
* [如何繪製文字於Canvas元素上](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/fillText)
* [如何取得兩座標之間的弧度](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/atan2)
* [如何取得圓周上一點的座標](https://www.mathsisfun.com/geometry/images/unit-circle-sin-cos-tan.svg)
* [如何將圖片繪製於Canvas元素上](https://stackoverflow.com/questions/6011378/how-to-add-image-to-canvas)
* [如何將HTML元素轉為圖片](https://stackoverflow.com/questions/8126623/downloading-canvas-element-to-an-image)
* [如何下載Canvas為圖檔](https://stackoverflow.com/questions/17311645/download-image-with-javascript)
* [如何上傳Json檔](https://stackoverflow.com/questions/36127648/uploading-a-json-file-and-using-it)
* [如何下載指定檔名及文字的文字檔](https://stackoverflow.com/questions/3665115/how-to-create-a-file-in-memory-for-user-to-download-but-not-through-server)
* [套件html2canvas的設定值](https://html2canvas.hertzen.com/configuration)
* [如何使用html2canvas的參數](https://stackoverflow.com/questions/71881661/html2canvas-black-background-after-background-color-is-specified)
* [如何將度數轉為弧度](https://www.w3resource.com/javascript-exercises/javascript-math-exercise-33.php)
* [如何取得指定最大及最小值的隨機整數](https://stackoverflow.com/questions/4959975/generate-random-number-between-two-numbers-in-javascript)
* [如何以一行程式碼建立陣列並以索引值填滿陣列](https://stackoverflow.com/questions/3746725/how-to-create-an-array-containing-1-n)

靈感來源：

* [戴克斯特拉演算法](https://www.youtube.com/watch?v=pVfj6mxhdMw)
* [圖遍歷及最小生成樹演算法](https://www.youtube.com/watch?v=GiDsjIBOVoA&t=1491s)
* [混合演算法討論貼文](https://cs.stackexchange.com/questions/1749/dijsktras-algorithm-applied-to-travelling-salesman-problem)
* 鄰接矩陣按鈕圖示：靈感來自Adobe Illustrator的介面設計：

# 心得

《圖論實作》是從蘇老師說明了期末專題的當天晚上我就開始計畫的專題作業，自此之後我便持續投入心力於它。它不僅是個專題作業，而更是我用來訓練自己在前端工程、使用者體驗及演算法技術的練金石。

專題開發途中偶爾會有難關，如在嘗試把HTML元素畫進Canvas元素並下載成圖片時，我發現了一個工具程式叫做「html2canvas」。它非常的方便，但它的文件寫得太不清不楚，單就「如何給它的函式輸入參數」這個問題上處處碰壁，查詢了各種關鍵字及可能的解決辦法，我甚至把Google搜尋頁面翻到了第二頁都找不到我想要的答案。前前後後大約花了我三天的時間，最終，在網路的某個角落我終於發現了答案―竟然是在毫不相關的[貼文](https://stackoverflow.com/questions/71881661/html2canvas-black-background-after-background-color-is-specified)底下找到的，想起來就覺得好笑。

對於從小就喜歡畫圖的我，在設計圖示及版面時我其實挺開心的。按鈕圖示被我重新設計不少次，因為我想盡可能地讓圖案及版面達到既精簡又能一目瞭然的境界，不過可能還有得學了。順帶一提，報告封面的葡萄紋是我最感到自豪的作品之一，它代表著豐饒與多福。

從開始接觸路徑演算法到寫出這個專題，一路上我收穫滿滿，在將腦中的創意化作成果時的滿足感讓我所花費的心力與時間都值得了。我很高興我踏入了程式設計的世界。