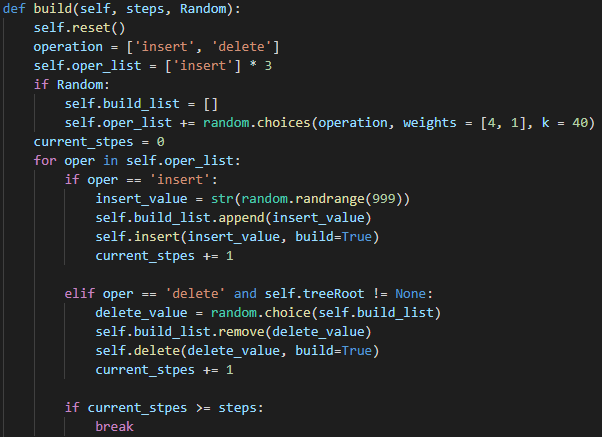
**Team 8 技術報告**

組員:丁冠廷、許書維、郭尚睿

1. **動機**  
   在學習各種資料結構時，透過文字建立起的各種規則有時會令人很難直觀的理解，而我們認為理解各種資料結構最好的方式就是透過動畫去觀察並進一步的了解實際運行的情況。因此，我們希望夠建立一個資料結構視覺化的網站，將資料結構的各種操作分解，透過動畫說明每一步的原因，令學習者能透過親手操作來理解各種資料結構。另外也希望能提供一些測試的功能，讓學習者能透過測是找出自己的盲點。
2. **題目描述**  
   (架構圖)
3. **實作細節**
   1. **網頁外觀及實作動畫**
      * 利用Bootstrap框架，建立各頁面。在首頁index.html中，透過 Carousel ，建立各資料結構的選單，讓使用者可以點選想要的資料結構進入DIY頁面。
      * 向後端 API 發送訊息的部分，使用Jquery ajax功能[[1]](#Ref1)，向server發送Http Request。  
        Ex: 圖一中的JavaScript function會在按下頁面內的Insert bottom後啟動，將目前Insert input區(id=“insert-value”)內的數字，利用”/bst/insert/<value>” url傳送至flask後端server。後端會進行資料結構操作後以JSON格式回傳相應的animation commands，並將animation commands丟入AM (Animation Manager)後依序畫出動畫。

(圖一) function in BST.js

* + - 由於原本Open Source Code年代久遠，是使用canvas基本畫法，較複雜且不易管理，因此JavaScript動畫部分決定參考[此網站](https://www.cs.usfca.edu/~galles/visualization/source.html)[[2]](#Ref2)的Open Source Code的work flow與分工架構，而創造物件和產生動畫則分別使用 Easel.JS Library 和 Tween.JS Library[[3]](#Ref3) 來達成。
      * **Managers:**  
        **AnimationManager.js:**  
        Define Animation Manager物件，負責接收並parse來自後端的animation commands，依照不同指令做出動畫。例如：收到Create Circle / objectID / objectLabel / x / y 指令，就創建一個AnimatedNode物件，顯示文字為objectLabel，在canvas上的(x,y)位置、收到Move指令就移動某物件到新座標。  
        **ObejctManager.js:**  
        Define Object Manager物件，負責管理及儲存現有的objects如Nodes、Edges，以及對object做操作如設定新(x,y)位置、從canvas上移除此object。
      * **Object**:  
        **AnimatedNode.js:**  
        Define Node物件，設定Node的properties(顏色、Label、位置...)。  
        **Line.js:**  
        Define Edge物件，設定Edge的properties。  
        **AnimatedLabel.js:**  
        Define Status物件，設定properties，顯示在Canvas下方，用來說明動畫正在執行什麼操作。  
          
        **AnimatedHighlightCircle.js:**  
        Define Highlight Circle物件，用來將Node用黃色圓圈匡起來，提示現在動畫正在操作的位置。
  1. **資料結構演算法**  
     大致上參考source code[[4]](#Ref4)演算法，但由於source code 是以JavaScript撰寫，因此需將其修改為python file，並加上一些我們所需的功能。  
     後端部分主要功能是記錄整個資料結構的狀態，並在接收前端的各種操作時，將動畫所需的每個步驟放進一個List (AnimationCommands) 回傳給前端。
     + **Build function:**  
       在min heap 的source code中有一個build的功能，該功能是將min heap清空，並在heap array中填上由大到小的數，再依min heap 的規則將其整理為一個正確的min heap，我們認為此功能不是project所需功能，因此將build改為在test中出題所使用的function，也在binary search tree、max heap、red black tree中加入build function以用於test。  
       Ex:下圖以red black tree的build function 為例，目的是在test時能以適當的比例隨機insert、delete，以供使用者判斷該動畫是哪種資料結構。  
       
     + **Reset function:**  
       由於在前端的瀏覽器重新整理時，我們希望能同時將後端的資料清空，以免造成前後端不同步的問題，因此新增了reset function。

1. **問題及解決方法**
   1. **專案框架選擇**
      * 純Javascript 全部寫在前端
      * 純Python
      * JS Frontend + Python + Flask Backend
   2. **JavaScript前端動畫**
      * 使用html canvas element當作畫布，利用Easel.js library創造物件（ex: Node, Line, Label, Highlight Circle）, 利用Tween.js 製作動畫效果 (ex:可以在指定的時間內移動某件物件到指定的位置)。
      * javascript不同步特性讓動畫不會按照順序出現。  
        解法：使用 async function / await 指令，讓javascript在執行時能按照順序，會等待前一個function做完，才能繼續往下做。
   3. **Flask api module**
      * 管理不同資料結構的api
      * 前後端分離，狀態的紀錄要如何維持一致：  
        我們使用一個變數叫做objectID來代表畫布上每個物件，而後端也有相對應的ID叫做index，來代表接下來將要建立的Node，其所屬的ObjectID，但當前端重整時，objectID會重置，但是在後端相對應的index卻不會知道，造成狀態上的mismatch  
        解法一：重整前端時，重啟Flask server -> problem  
        解法二：重整時送出api給後端重置，確定前端重整時，同時發送重置資訊給後端
   4. **Warning Card**
      * Bootstrap Modal
      * Bootstrap Card
   5. **Different navigation types might produce coherency issues** **between frontend and backend**
      * 了解現在網址是哪一種型態進入的，主要根據navigation.performance.type分成三種：  
        navigate(0), reload(1), back\_forward(2)  
        當偵測到這些活動時，就進行後端server的重置
   6. **Deployment**
      * Heroku: Free 方案沒辦法scale
2. **未來方向**  
   目前我們的project僅有min heap、max heap、BSTree、RBTree四種，但由於已建立起一套完整的架構，因此擴充性相當好，未來加入多種其他資料結構並不成問題。
3. 分工表
4. Reference

* 1. <https://www.youtube.com/watch?app=desktop&v=2OYkhatUZmQ>

* 1. <https://www.cs.usfca.edu/~galles/visualization/source.html>

* 1. <https://createjs.com/>
  2. Open Source Code: <https://www.cs.usfca.edu/~galles/visualization/source.html>
  3. Library: [https://createjs.com/getting-started/easeljs](%20https:/createjs.com/getting-started/easeljs)
  4. Async / await 教學: <https://zellwk.com/blog/js-promises/>
  5. Async / await 教學: <https://zellwk.com/blog/async-await/>
  6. Async / await 教學: <https://zellwk.com/blog/async-await-in-loops/>