Data Structure Report

B07502089
許書維

- 工作內容
  1. 建立首頁(index.html)和視覺化資料結構的各頁面(BST.html, MinHeap.html …)，並美化各頁面，需用到 html / css 技術。
  2. 使用JavaScript，串起前後端，建立 event listener ，使網頁可互動 。例如：在 BST 頁面，按下insert buttom，就傳送目前輸入的insert value到flask後端，指令如下：
     Http Request: ”GET /bst/insert/<value> HTTP/1.1”。
  3. 使用JavaScript，接收python後端資料結構變化過程中發出的指令(commands)，在前端 html canvas element（畫布），做出與指令相對應的動畫。
- 實作細節
  1. 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，讓使用者可以選擇想要的資料結構 (Binary Search Tree, Min Heap, Max Heap, Red Black Tree)，點選即可進入DIY頁面。
  2. 向後端 API 發送訊息的部分，使用Jquery ajax功能 （教學），向server發送Http Request。
     Ex: 下面這個JavaScript function會在按下(click event)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(done)回傳相應的commands（以JSON格式），此時將commands丟入AM (Animation Manager)創造對應動畫。

```html
<!-- insert -->
<input type="text" size="5" id="insert-value" />
<button id="insert" class="btn btn-dark">Insert</button>
```

(<— in BST.html)

```javascript
// ---- insert buttom event listener ----
$(function () {
  $("#insert").click(function () {
    var appdir = "/bst/insert/";
    var n = parseInt($("#insert-value").val());
    $.ajax({
      type: "GET",
      url: server + appdir + n,
    }).done(function (data) {
      //console.log(data);
      $("#insert-value").val("");
      var commands = JSON.parse(data);
      AM.StartNewAnimation(commands);
    });
  });
});
```

(<— in BST.js)

3. JavaScript動畫部分，參考 https://www.cs.usfca.edu/~galles/visualization/source.html 網站的Open Source Code的work flow與分工架構：

   Managers:
   - AnimationManager.js
     Define Animation Manager物件，負責接收來自後端的指令，並一個一個parse指令，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(數字=Label)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 (現有的Nodes, Edges)，以及對現在的object做操作(ex: 設定新(x,y)位置、從canvas上移除此object)。

   Objects:
   - 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（顯示在Canvas下方，顯示現在操作是什麼），設定properties。
   - AnimatedHighlightCircle.js
     Define Highlight Circle物件（用來將Node用黃色圓圈匡起來，提示現在位置）。

   創造物件和產生動畫分別使用 Easel.JS Library 和 Tween.JS Library 來達成（原本Open Source Code年代久遠，那時候還沒有這些Library，使用canvas基本畫法，較複雜且不易管理）。

   註：以上所有檔案僅參考Open Source Code的“作業流程”和“分工架構”，所有操作function和動畫都是自己研究Library之後創造出。

• 遇到的困難 & 解法

- 困難 1：
  基本Canvas API無法物件式管理畫布上的物件，畫上去就無法再修改（除非全部清除再重新畫上去）。（像是在畫布上畫Node，並不能移動畫上去的Node）
  解法 1：
  使用CreateJS Library (內含 Easel.JS & Tween.JS)對canvas進行操作，可以物件式管理畫過的Objects（對某物件進行移動）。（像是先畫Node在便利貼上，再貼到畫布上，之後可以任意移動）

- 困難 2：
  JavaScript非同步特性，並不會按照Code順序執行，在產生動畫時會產生許多error（ex: 物件還沒完成init，就已經收到對物件進行改變的指令）。

解法 2：

使用Javascript 特殊 function async 和特殊指令 await ，強制Javascript在某些地方按照順序進行。（教學參考Reference 3,4,5）

- **Reference**

1. Open Source Code: https://www.cs.usfca.edu/~galles/visualization/source.html
2. Library: https://createjs.com/getting-started/easeljs
3. Async / await 教學: https://zellwk.com/blog/js-promises/
4. Async / await 教學: https://zellwk.com/blog/async-await/
5. Async / await 教學: https://zellwk.com/blog/async-await-in-loops/