資料結構

HW2

系級：電機系大三

姓名：林凡皓

學號：B103012002

一、 Design of my program

- __init__ :

  SparseMatrixLL 有三個 attributes，分別為_nrows、_ncols 和_row_list。_nrows 和_ncols 直接去接 initialize 時所輸入的參數即可。_row_list 則是使用 for loop 去迭代_nrows 次，產生一個 list 裡面有_nrows 個 UnorderedList。

- __setitem__ :

  先利用 check_row 和 check_col 來確認說要設定的值的位置是否有在矩陣內。接著去產生一個 MatrixEntry，然後去看要存入的值是否為 0。如果不為 0，就用 linked list 的 add 將 MatrixEntry 接到 linked list 中，否則利用 search 去查看 row list 中是否有 MatrixEntry，如果有就利用 remove 將他移除，這麼做是為了避免不小心存到 value 為 0 的 matrix entry。

- __getitem__ :

  先利用 check_row 和 check_col 來確認說要設定的值的位置是否有在矩陣內。接著去產生一個 MatrixEntry，並用 search 去檢查 MatrixEntry 是否存在要取得的 row 對應的 row list 中。如果存在就用 while loop 遍歷整個 linked list，找到存剛要取得的 column 對應的 matrix entry，並回傳該 matrix entry 的 value。如果不存在，代表說該元素在矩陣中為 0，因此值接回傳 0。

- __add__ :

  矩陣可以相加的條件為兩個矩陣的形狀要一樣，因此要先去檢查準備相加的兩個矩陣的形狀是否一樣。一樣的話，先去初始化一個結果矩陣，然後利用巢狀 for loop 去遍歷兩個矩陣中所有元素，並將元素相加後 assign 給結果矩陣。

- __sub__ :

  矩陣可以相減的條件為兩個矩陣的形狀要一樣，因此要先去檢查準備相減的兩個矩陣的形狀是否一樣。一樣的話，先去初始化一個結果矩陣，然後利用巢狀 for loop(兩層)去遍歷兩個矩陣中所有元素，並將元素相減後 assign 給結果矩陣。

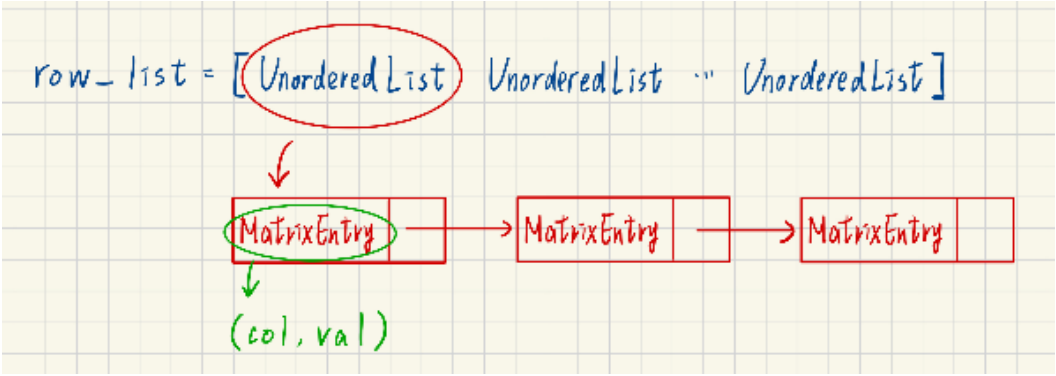
- __mul__ :

  矩陣可以相乘的條件為，放在前面的矩陣的 column 數目要等於放在後面的矩陣的 row 數目，因此要先檢查準備相乘的兩個矩陣是否滿足矩陣相乘的條件。一樣的話，先去初始化一個結果矩陣，接著透過巢狀 for loop(三層)去做矩陣乘法。其中兩層 for loop 分別去遍歷 row 和 column，最後一層用來計算元素相乘後的總合。計算完總和後將結果 assign 給結果矩陣。

- 討論 :

  1. 使用的資料結構 :

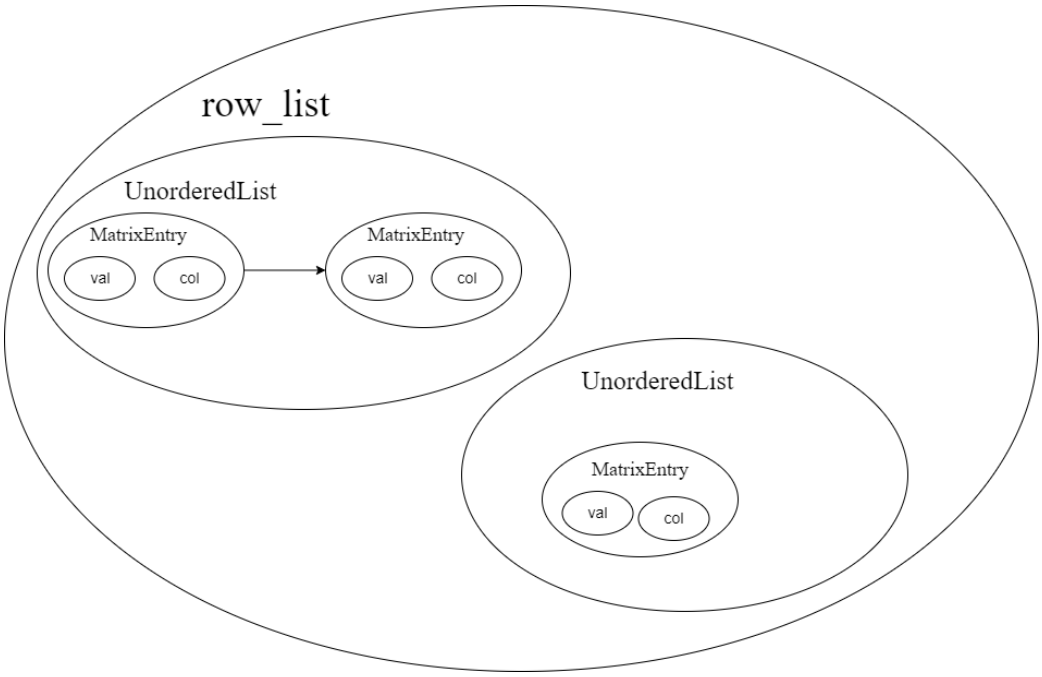
     這次作業所使用的資料結構主要為 linked list 中的 unordered list。在這次作中，我有 import 教授給的 source code 中的 UnorderedList 來幫助我的實作。資料結構如下圖，



  2. 遇到的問題與學習到的事物 :

     這次作業中遇到最大的問題就是對於資料結構不熟悉。在實作的過程中，特別是在__setitem__和__getitem__中，發生了很多問題像是要如何加入一個 matrix entry、要如何去找到一個 column 對應到的 matrix entry 等。對於這些問題，我都會很直覺的想說用 list 來做，例如，要加入一個 matrix entry 就使用 list.append( )。但是這是不可行的，雖然說 linked list 在 python 中是使用 list 做實現，但是實際上 linked list 跟 python list 是有差距的，像是 python list 是不會有指標的，但是 linked list 中的連接都是使用指標。這些不同就會造成要對這些資料結構做一些操作會有不同處，例如 python list 的 append 和 linked list 的 add 雖然說都是加入一個東西進去，但是 linked list 中還有指標的改變的問題要考慮。因此在使用 linked list 的時候，我們會需要重新去寫一些專屬於他的方法，之後要實做 linked list 的時候也要根據屬於他的方法做延伸應用。

除了方法上的問題，整個資料結構的架構在一開始做作業的時候也是不明確的。List of list 中的 list 是甚麼、unordered list 中要放甚麼，在一開始讓我花了很多的時間去理解。現在我會將他想成是一包一包的東西，有點類似機率中集合的概念。舉例來說，對於這次作業的架構，我會理解為最外層有一個 list，list 中包含很多個 unordered list，unordered list 的 node 會是存放 column 和 value 的一個結構，也就是 matrix entry，所謂的類似集合的概念如下圖



這樣的理解方式就能夠很好的去看出整個資料結構中，每個物件之間的關係。

二、 Array v.s. linked list

- Array
  1. 優點 ：
     - 在 array 中存取資料或是查詢資料，只需要用 index 即可對特定位置的資料作存取與查詢，時間複雜度為 O(1)。
     - 相較於 linked list，array 更加節省記憶體空間。Linked list 需要額外的記憶體來存指標，這樣會多花費記憶體空間來存非真正要處理的資料。
  2. 缺點 ：
     - Array 的元素在記憶體中是連續的，因此要新增一個元素在

array 的第一個位置，就需要將後面所有元素一一搬動，時間複雜度為 O(n)。

- Array 的長度不可以改變，宣告太長會浪費記憶體空間，宣告太短會不夠用。

3. 使用時機：
   - 希望能夠快速存取與查詢資料。
   - 已知欲處理資料的大小。
   - 要求記憶體很小。

- Linked list

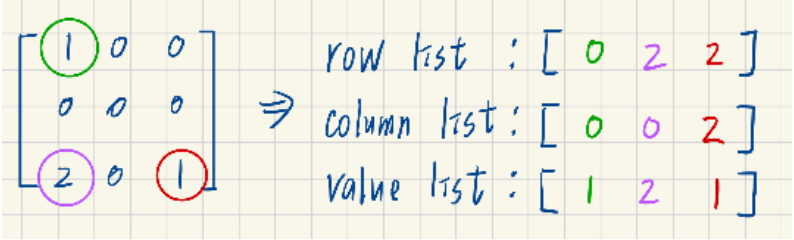
1. 優點：
   - 增加或是刪除資料比 array 簡單，只需要去調整指標指向的節點即可。如果是在 linked list 的最前端新增節點，只需要 O(1)的時間複雜度。
   - Linked list 的長度可以不固定，不需要預先分配記憶體給 linked list。
2. 缺點：
   - 要存取或查詢特定節點的資料需要從頭開始找，因此時間複雜度為 O(n)。
   - 需要額外分配記憶體給指標。
3. 使用時機：
   - 無法預先知道資料數量。
   - 需要頻繁的新增或是刪除資料。
   - 對於存取與查詢資料的速度沒有特別要求。

三、 SparseMatrixLL v.s. SparseMatrix v.s. coordinate format

- Sparse matrix in coordinate format

1. 概念：

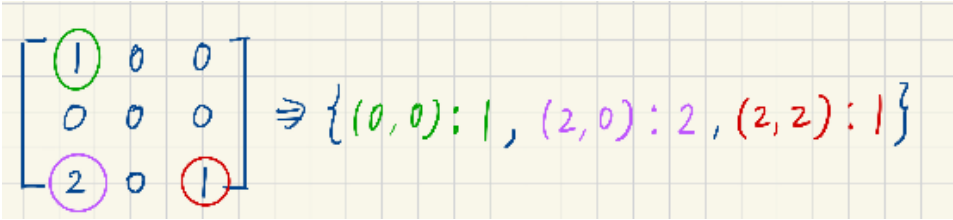
   透過三個 list 分別存矩陣中非零元素的 row、column、value。 架構如下圖



2. 優點：
   - 最簡單和最好理解。
   - 對於逐一元素的操作，COO 可以很簡單的實現，像是矩陣乘法、加法。
3. 缺點：
   - 需要三個 list 來儲存一個矩陣，記憶體空間相對需要比較多。
   - 隨著矩陣的擴大，記憶體空間需求會增加很快，因為三個 list 都在同時做增長。

- Sparse matrix using dictionary of keys
  1. 概念：

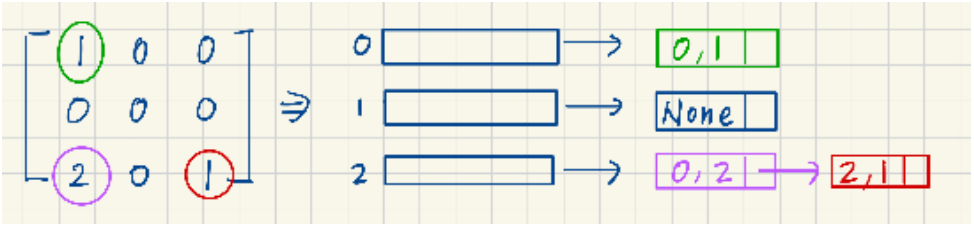
     使用 dictionary 來存放矩陣中非零元素。Dictionary 的 key 為儲存 row、column 的 tuple，dictionary 的 value 為元素值。架構如下圖

  2. 優點：
     - 查詢給定位置的數值很快，時間複雜度為 O(1)
     - 可以靈活地進行切片操作
     - 要添加、刪除、改變矩陣中的元素值都具有高效性。
  3. 缺點：
     - 進行矩陣運算會比較慢
     - 由於 DOK 是使用 hash table 來儲存，因此會需要占用大量記憶體中間。

- Sparse matrix using list of list
  1. 概念：

     使用 list 中包含 linked list 的架構來儲存矩陣中非零元素。整體架構如下圖

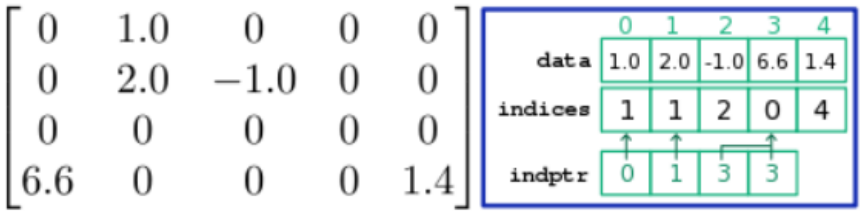
     

2. 優點：
   - 可以動態添加或是刪除元素。要添加或刪除元素只需要修改 column list 即可。
   - 可以靈活地進行切片操作。
   - 記憶體使用量較少，因為 row list 的長度只需要跟該 row 非零元素一樣就好。
3. 缺點：
   - 存取矩陣中特定元素比較慢。

四、 Other sparse matrix formats

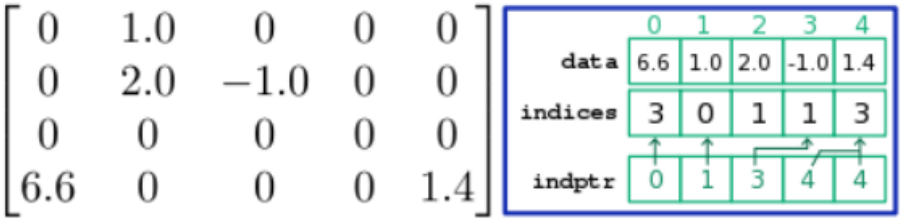
- Compressed sparse row (CSR)
  1. 概念：

     架構如下圖



     利用三個 array 分別是 data、indices、indptr 來儲存非零元素。Data 沿著矩陣，由左到右、由上到下儲存非零元素，indices 會儲存 data 元素所在的 column。Indprt 會儲存每個 row 中，第一個非零元素在 data 中儲存的 index，如果整個 row 都是零，則存下一個 row 第一個非零元素在 data 中儲存的 index。
  2. 優點：
     - 很容易取得矩陣中特定元素。
     - Row slicing 很容易完成。
     - 矩陣-向量乘法可以很有效地完成。
  3. 缺點：
     - Column slicing 很麻煩。
     - 要在矩陣中添加或刪除元素會很複雜。
- Compressed sparse column (CSC)
  1. 概念：

     概念和 CSR 類似，差別在於將 CSR 中的 row 與 column 互換。架構如下圖

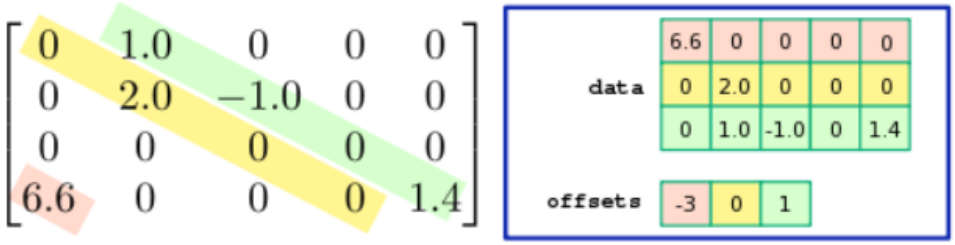


2. 優點：
   - 很容易取得矩陣中特定元素。
   - Column slicing 很容易完成。
   - 向量-矩陣乘法可以很有效地完成。
3. 缺點：
   - Row slicing 很麻煩。
   - 要在矩陣中添加或刪除元素會很複雜。

- Diagonal storage (DIA)

1. 概念：

   架構如下圖



   data 所儲存的是矩陣中的對角線元素，offsets 儲存的是 data 中的 row 所對應到的對角線，其中主對角線為 0。

2. 優點：
   - 對於有很少有非零元素的對角線的矩陣，像是對角矩陣或是三角矩陣，DIA 可以提供高速的運算。
3. 缺點：
   - 對於有很多有非零元素的對角線的矩陣，要存取矩陣中的元素會變得很複雜。

五、 心得總結

這次作業主要分成三個部分，分別是實作稀疏矩陣、array v.s. linked list、Sparse matrix format 比較。

第一部分是使用 linked list 中的 unordered list 實作稀疏矩陣。經過這次的實作，除了更加熟悉 Python class 的語法，我覺得更重要的是學會如何在腦袋

中自己建立資料結構視覺化後的樣子。在腦袋中有資料結構整體架構對於 coding 或是分析自己寫的 code 都會帶來很大的幫助，因為這樣更容易去思考整個運算的過程以及找尋哪裡出問題。

第二部分是比較 array 和 linked list。透過自己去網路上搜尋關於 array 和 linked list 的資料，對於上課教授教的內容又有了一次複習，也透過比較兩者之間的優劣以及操作的時間複雜度，讓我更加知道甚麼時候要選擇 array，甚麼時候要選擇 linked list。

最後一部分是比較 sparse matrix format。上網搜尋資料的時候，除了看到許多教授有教的內容，像是 COO、DOK 等，還看到許多針對一些特殊矩陣所開發的 format，例如 DIA。對於矩陣的表示法，我覺得對於不同的使用情境真的都有不一樣的選擇，並沒有哪一個方法是最差的，哪一個方法是最好的這樣的說法。

六、 Reference

[1] yt.liao “資料結構與演算法筆記(1) – linked list 與 array 於 O(n)之差異比較”
https://medium.com/@maggieliao.cm04g/%E8%B3%87%E7%B5%90%E8%88%87%E6%BC%94%E7%AE%97%E6%B3%95%E7%AD%86%E8%A8%98-1-linked-list-%E8%88%87-array-%E6%96%BCo-n-%E4%B9%8B%E5%B7%AE%E7%95%B0%E6%AF%94%E8%BC%83-badbf08b17ce

[2] Jimmy 的架站筆記 “資料結構筆記 1 - Array(陣列), Linked List(鏈結串列)”
https://jimmyswebnote.com/%E8%B3%87%E6%96%99%E7%B5%90%E6%A7%8B%E7%AD%86%E8%A8%98-1-array-linked-list/

[3] Y.D. Chong “Sparse Matrix Formats”
https://phys.libretexts.org/Bookshelves/Mathematical_Physics_and_Pedagogy/Computational_Physics_(Chong)/08%3A_Sparse_Matrices/8.02%3A_Sparse_Matrix_Formats

[4] Scipy lecture notes “ List of List Format (LIL)” https://scipy-lectures.org/advanced/scipy_sparse/lil_matrix.html

[5] Scipy lecture notes “Dictionary of Keys Format (DOK)” https://scipy-

lectures.org/advanced/scipy_sparse/dok_matrix.html
[6] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.openai.com/
[7] Matt Eding Python & Data Science Blog “Sparse Matrix” https://matteding.github.io/2019/04/25/sparse-matrices/#coordinate-matrix