資料結構

HW2

系級 : 電機系大三

姓名 : 林凡皓

學號 : B103012002

1. 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 :

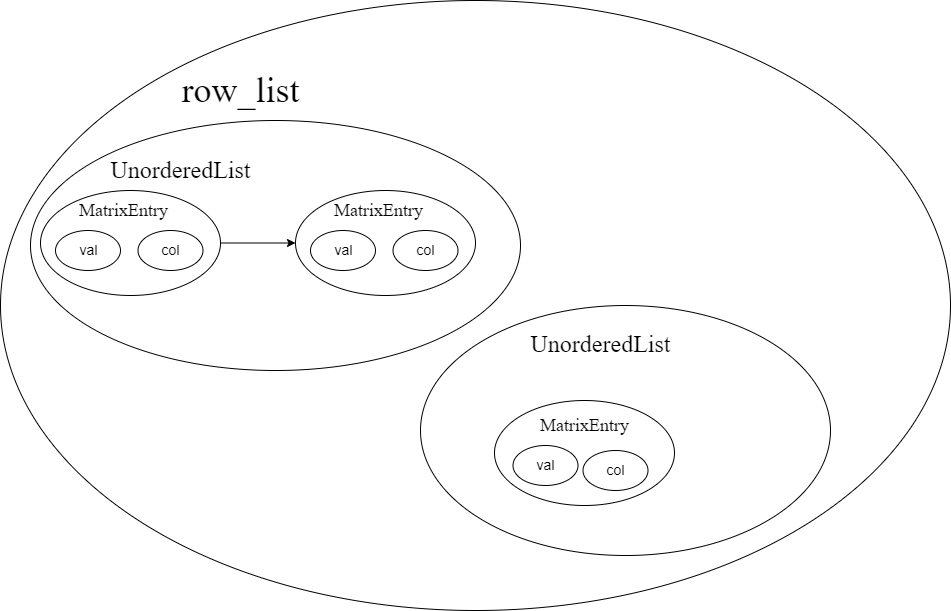
這次作業所使用的資料結構主要為linked list中的unordered list。在這次作中，我有import教授給的source code中的UnorderedList來幫助我的實作。資料結構如下圖，

一張含有 文字, 字型, 筆跡, 行 的圖片

自動產生的描述

1. 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。

1. 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。

1. 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。

1. 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。

1. 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。

1. 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。

1. 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

一張含有 文字, 筆跡, 字型, 行 的圖片

自動產生的描述

1. 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。

1. 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

一張含有 文字, 筆跡, 字型, 行 的圖片

自動產生的描述

1. 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。

1. 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

一張含有 文字, 行, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

1. 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。

1. 缺點 :

* 存取矩陣中特定元素比較慢。

1. Other sparse matrix formats

* Compressed sparse row (CSR)

1. 概念 :

架構如下圖

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

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。

1. 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。

1. 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

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

1. 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。

1. 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

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

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。

1. 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。

1. 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。

1. 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。

1. 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>