2023 Data Structure - Final Project - Group 3

Members:

|  |  |  |
| --- | --- | --- |
| A1105505  林彧頎 | Prob 2 – 5 (a)(b)(c),Merged Code(oop、DS),  Design(OOP、DS)，  Word報告撰寫(+統整),PPT(製作+報告) | |
| A1105521  黎子崴 | Prob 1 - Task A&B-(7)(8)(9),  Word報告撰寫,PPT(製作) | |
| A1105523  巫柔筠 | Prob 1 - Task A&B-(1)(5)(6)  Word報告撰寫,PPT(製作) | |
| A1105524  吳雨宣 | Prob 2 - (1)(2)(3)(4)  Word報告撰寫,PPT(製作+報告) | |
| A1105545  潘妤揚 | Prob 1 - Task A&B- (10),Prob 2 – 5 (d),效能分析  Word報告撰寫,PPT(製作) | |
| A1105549  杜佩真 | Prob 1 - Task A&B-(2)(3)(4),  Word報告撰寫,PPT(製作) |

Catelog

Problem 1 - Task A……………… 1

Problem 1 - Task B……………… 4

Problem 2………………………… 7

專題開發過程............................... 10

效能分析...................................... 21

Problem 1 - Task A

(1) 4571 dates

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (2)   |  |  | | --- | --- | | date | price | | 20011003 | 3446.26 | | 20011002 | 3492.12 | | 20011004 | 3493.66 | | 20010925 | 3493.78 | | 20011008 | 3520.35 | | 20010924 | 3533.51 | | 20010927 | 3567.63 | | 20011005 | 3585.46 | | 20010921 | 3591.85 | | 20011009 | 3618.93 | | (3)   |  |  | | --- | --- | | date | price | | 20000217 | 10202.20 | | 20000405 | 10186.17 | | 20000219 | 10161.05 | | 20000211 | 10128.67 | | 20000410 | 10127.48 | | 20000218 | 10096.38 | | 20000411 | 10068.05 | | 20000216 | 10064.49 | | 20000210 | 10057.67 | | 20000401 | 10050.43 | |

(4) date:20100706, median price:7548.48

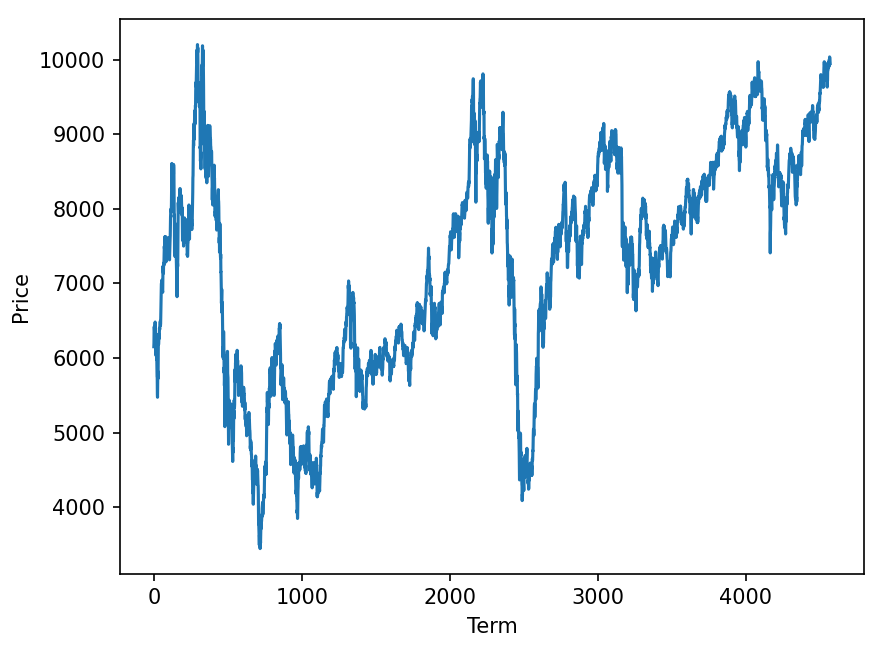
(5) date: 20090429, maximum return 6.74218%

date: 20040319, minimum return -6.67886%

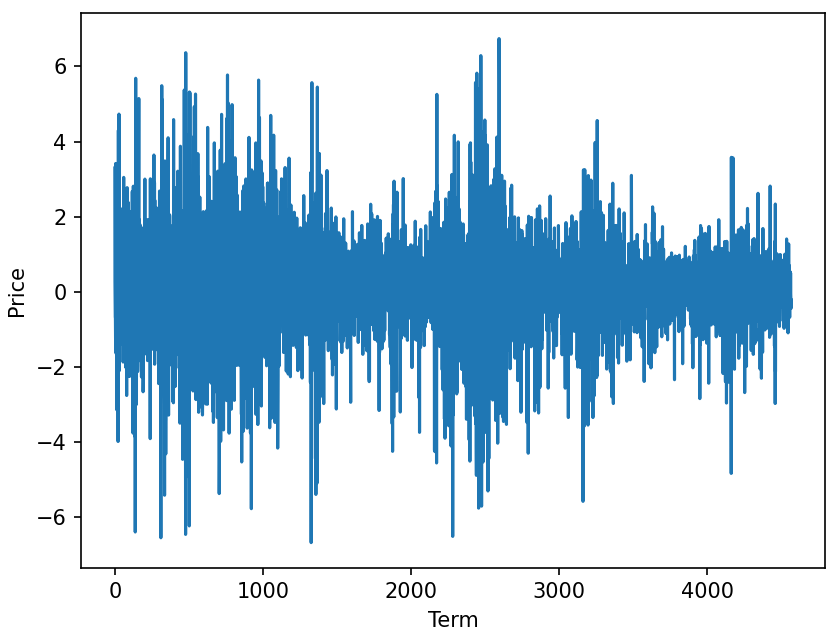
(6) date: 20001121, maximum return 6.96364%

date: 19990716, minimum return -7.20748%

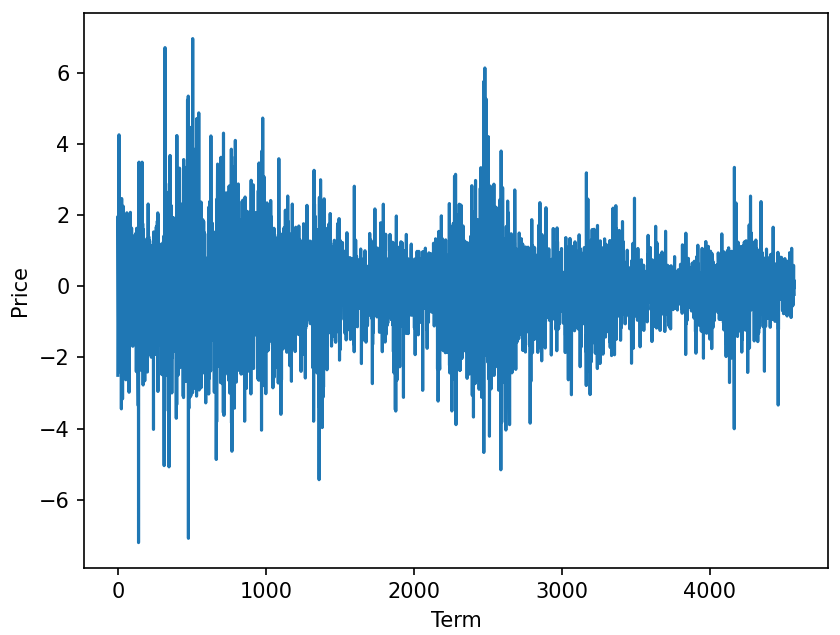
(7)



(8)



(9)



(10)

Maximum price = 10393.59000, occurring date: 20000218

Minimum price = 3411.68000, occurring date: 20010926

Median price = 7550.23000, occurring date: 20110823 and 7550.13000, occurring date: 19990427

Problem 1 - Task B

(1) 915 dates

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (2)   |  |  | | --- | --- | | date | price | | 20010927 | 3567.63 | | 20011005 | 3585.46 | | 20010920 | 3698.84 | | 20011015 | 3712.82 | | 20011022 | 3900.62 | | 20021009 | 3947.61 | | 20011029 | 4065.10 | | 20011105 | 4080.51 | | 20081121 | 4171.10 | | 20021002 | 4171.76 | | (3)   |  |  | | --- | --- | | date | price | | 20000217 | 10202.20 | | 20000210 | 10057.67 | | 20170512 | 9986.82 | | 20170321 | 9972.49 | | 20000406 | 9969.28 | | 20170519 | 9947.62 | | 20000412 | 9911.39 | | 20170505 | 9899.94 | | 20170406 | 9897.80 | | 20170328 | 9876.45 | |

(4) date: 20080130, median price:7543.50

(5) date: 20001019, maximum return 18.54848 %

date: 20001114, minimum return -16.06407 %

(6) date: 20081107 , maximum return 6.13798 %

date: 19990716, minimum return -7.20748 %

(7)

一張含有 文字, 螢幕擷取畫面, 行, 繪圖 的圖片

自動產生的描述

(8)

一張含有 文字, 螢幕擷取畫面, 陳列, 繪圖 的圖片

自動產生的描述

(9)

一張含有 螢幕擷取畫面, 文字, 繪圖, 陳列 的圖片

自動產生的描述

(10)

Maximum price = 10328.98000, occurring date: 20000406

Minimum price = 3467.94000, occurring date: 20011005

Median price = 7538.04000, occurring date: 20110913 and 7535.82000, occurring date: 20111026

Problem 2

(1) \_1029\_ unique products

(2) \_\_NO\_\_ (Yes or No)

(3) \_\_Yes \_ (Yes or No)

(4) \_\_NO\_ (Yes or No)

(5)

|  |  |  |
| --- | --- | --- |
| Date | Time | Price |
| 20170517 | 114907 | 76 |
| 20170517 | 114907 | 76 |
| 20170517 | 114914 | 76 |
| 20170517 | 114914 | 76 |
| 20170517 | 114914 | 76 |
| 20170517 | 114914 | 76 |
| 20170517 | 114914 | 76 |
| 20170517 | 114914 | 76 |
| 20170517 | 114907 | 77 |
| 20170517 | 114907 | 77 |

|  |  |  |
| --- | --- | --- |
| Date | Time | Price |
| 20170516 | 090502 | 154 |
| 20170516 | 090502 | 154 |
| 20170516 | 090502 | 154 |
| 20170516 | 090502 | 154 |
| 20170516 | 090502 | 154 |
| 20170516 | 090502 | 154 |
| 20170516 | 090503 | 154 |
| 20170516 | 090503 | 154 |
| 20170516 | 090503 | 154 |
| 20170516 | 090502 | 153 |

1. Median Price = \_\_\_112\_\_\_\_
2. Maximum return: 6.03448 %

|  |  |  |
| --- | --- | --- |
| Date | Time | Price |
| 20170516 | 234424 | 116 |
| 20170516 | 235135 | 123 |

So, (123 - 116) / 116 \* 100% = 6.03448%

Minimum return: -25.8065%

|  |  |  |
| --- | --- | --- |
| Date | Time | Price |
| 20170516 | 235446 | 124 |
| 20170517 | 084500 | 92 |

So, (92 - 124) / 124 = -25.8065%

專題開發過程

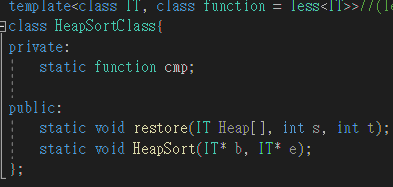
一、專題開發流程前要

我們這組一開始是先分配工作，我們的分配方式是依照分數配比盡量均分，再來是由於我們的heapsort會需要一直重複使用到，所以我們是一起先寫一個heapsort，再讓各組員後續是需要自行增加需要的部分，並於最後統整。

(以下為主要用於本專題資料分析的資料結構)

a.Heapsort

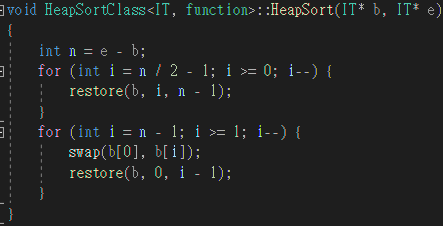
本次專題中強制要求我們使用HeapSort進行排序，這種排序法有好有壞，其中的壞處就是cpu的使用，使得我們在過程中有時候會使用g++進行編譯(因為MSVC的受限)，heap sort最重要就是需要使用到二元樹，並依照資料要分析的需求進行小到大或大到小的排序，也因此我們的實作中因為會需要使用到很多種的排序方式，因此我們用template存入資料與選擇排序的方式(function=cmp)，而細節的排序比較方式各題會說明(cmp)。



我們的heapsort中最重要的兩個function為 :

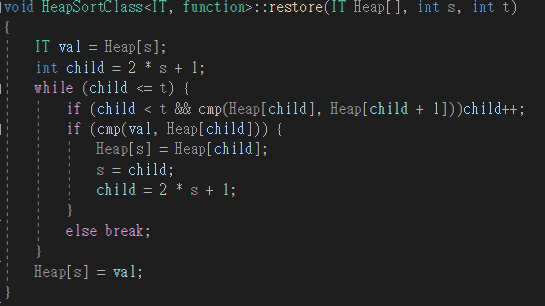
1. HeapSort

HeapSort的實際邏輯，首先將整個陣列轉化成heap以後。再將最小的元素分別放到陣列的arr[0],arr[1],arr[2]...。



2. restore

HeapSortClass當中，讓最小的東西放在Heap的最頂端，待回HeapSort需要用for迴圈做n次從後面往前將最小的東西分別放到Heap的頂端。其中比較的方式會依照使用者於main中呼叫的排序法進行比較，並排序儲存，排序的方式的function會依照題目要求於後續詳細說明。



b.DataFrame

我們是建立一個二元樹，可以找出有哪些資料有出現過。

1. 第一題的DataFrame

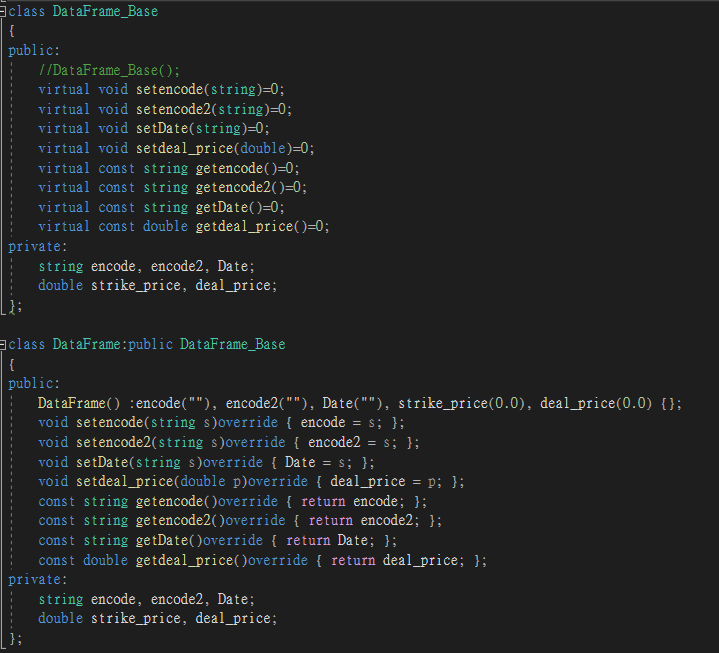
DataFrame是一個資料表型別 (二維表格)，用於存儲具有相關性的數據，是基於表格型數據結構，將數據組織成行和列的形式，每一列代表一個變量或屬性，每一行代表一個觀察或記錄。

如圖，第一題的DataFrame結構表示包含日期、開盤價格、最高價格、最低價格和收盤價格等資料的資料表。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

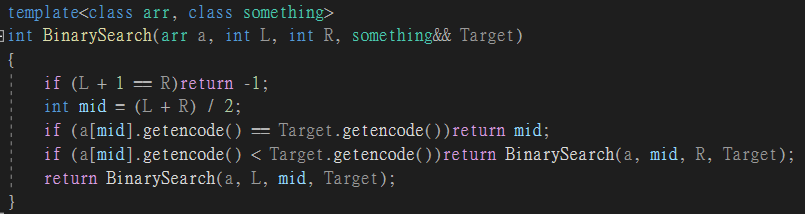
自動產生的描述

2. 第二題的DataFrame



第2題定義了一個名為DataFrame的類別。類別中包含了私有成員變數encode、encode2、Date、strike\_price和deal\_price，以及公有成員函數和建構函式。encode為根據CSV檔2\_3\_4\_5欄位的編碼；encode2為根據 CSV檔1\_6欄位的編碼；Date為日期；deal\_price為成交價格。

c.Binary Search



template<class arr, class something>有兩個模板參數 arr和 something，用於接受不同的類型。int BinarySearch(arr a, int L, int R, something&& Target)是二元搜尋函式的定義，接受一個陣列 a、兩個整數 L 和 R（表示搜尋範圍的左右邊界），以及一個 Target（目標值）。

接著判斷，如果 L + 1 等於 R，表示範圍內只有一個元素，且該元素不等於目標值，則返回 -1，表示未找到目標值。再用int mid = (L + R) / 2來計算搜尋範圍的中間索引，並將其保存在 mid 變數中。若中間元素等於目標值，則返回中間索引，表示已找到目標值。若中間元素小於目標值，則對右半部分進行遞迴搜尋。將 L 更新為 mid，表示新的搜尋範圍是右半部分。若不符合上述兩種判斷條件，則代表中間元素大於目標值，並對左半部分進行遞迴搜尋。將 R 更新為 mid，表示新的搜尋範圍是左半部分。

二、Problem1

Task A

(1) 沒重複(日期)的資料 2%

先將原始資料根據日期從小排到大，使用HeapSort演算法實現排序。

利用DataFrame 來去除重複日期的資料



將排序後的資料只保留第一個發現的日期，後面的都不放，儲存在arr2陣列代表沒有重複資料的筆數。

一張含有 文字, 字型, 螢幕擷取畫面 的圖片

自動產生的描述

(2)在Close\_price中找前十小的價格和包含這些價格的日期 2%

先用我們定義好的heapsort把過濾過的資料由小到大排序，然後再從陣列中印出前面十筆。

(3) 在Close\_price中找前十大的價格和包含這些價格的日期 2%

在排序好的陣列中印出後面十筆。

(4) 在Close\_price中找中位數的價格和日期 2%

在排序好的陣列中印出陣列大小除2那格的價格。

(5)計算每天的daily return 2%

使用堆疊排序存放，對不重複日期的arr2用Date排序



利用公式 [P(t+1)-P(t)]/P(t) \*100%計算出每天的 daily return，並利用判斷式，計算出最大最小值的數據以及日期。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

(6) 計算每天的intraday return 2%

用Date排序，利用公式 [Close\_price(t)-Open\_price(t)]/Open\_price(t) \*100% 計算出每天的 intraday return，並利用判斷式，計算出最大最小值的數據以及日期。

一張含有 字型, 圖形, 螢幕擷取畫面, 印刷術 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

(7) 繪製收盤價隨時間變化的圖2%

Outfortest 為ofstream型態變數向檔案寫內容，呼叫open方法使其與test1.txt關聯。使用for迴圈將陣列arr2之Close\_price寫入ss。當迴圈結束，取得當前串流ss中的字串形式後再轉換為C-style字元陣列，釋放系統內存。最後初始化stringstream ss並關閉test1.txt。

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述**

(8) 繪製日收益隨時間變化的圖2%

Outfortest 為ofstream型態變數向檔案寫內容，呼叫open方法使其與test2.txt關聯。使用for迴圈將陣列arrDayilyreturn寫入ss。當迴圈結束，取得當前串流ss中的字串形式後再轉換為C-style字元陣列，釋放系統內存。最後初始化stringstream ss並關閉test2.txt。

(9) 繪製日內收益隨時間變化的圖2%

Outfortest 為ofstream型態變數向檔案寫內容，呼叫open方法使其與test3.txt關聯。使用for迴圈將陣列arrIntradayreturn寫入ss。當迴圈結束，取得當前串流ss中的字串形式後再轉換為C-style字元陣列，釋放系統內存。最後初始化stringstream ss並關閉test3.txt。

Python環境導入sys和matplotlib函式庫，其中使用matplotlib函式庫在圖表中繪製數據。

定義PLOT的函數，該函數接受兩個參數分別為filename和title。函數內部sys.stdin以utf-8編碼方式打開的filename文件。調用input()函數，從文件中讀取的一行數據。使用split()函數對數據進行拆分，並使用列表推導方式將每個數值轉換為浮點數，最終得到一個data數值列表。

plt.plot(x, y)函數被調用，將x和y的值傳入函數後，繪製數值在圖表上。

title為plt.title 的值，' Term '為plt.xlabel，' Price '為plt.ylabel， 三者分別為圖表的標題、x軸標籤和y軸標籤。呼叫show()顯示繪製的圖表。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

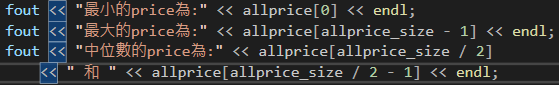
自動產生的描述

(10)根據有數字的四個欄位找出最小數字、最大數字、中位數 2%

解題思路: 利用前面所得到的不重複的資料，利用HeapSort和DataFrame把4個price集中到一個allprice的dataframe2去做整合比較，其中成員有date與price，並且用運算子多載定義好該型別的輸出，最後用cmpforprice的Struct進行價格排序找到最小、最大、中位數，最後output出正確輸出(也是運用運算子多載)。

使用自己的堆疊存放:



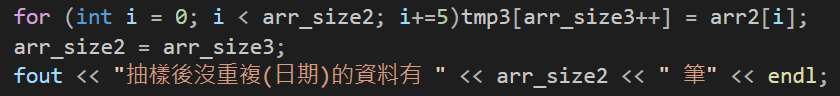


Task B

(1) 沒重複(日期)的資料 2%

同 Task A 利用HeapSort演算法實現排序，並利用DataFrame 來去除重複日期的資料，且arr2 代表沒有重複資料的筆數。

每5筆資料抽一次保留到tmp3中，且將arr2覆蓋成tmp3，最後得到每五筆資料取第一筆資料後沒有重複日期的資料筆數。



(2) 在Close\_price中找前十小的價格和包含這些價格的日期 2%

同Task A ，把每五筆取出的資料由小到大排序，印出前十筆資料。

(3) 在Close\_price中找前十大的價格和包含這些價格的日期 2%

在排序好的陣列中印出後面十筆。

(4) 在Close\_price中找中位數的價格和日期 2%

在排序好的陣列中印出陣列大小除2那格的價格。

(5) 計算每天的daily return 2%

同Task A，使用堆疊排序存放，對不重複日期的arr2用Date排序，並利用公式計算出 daily return 和判斷式計算最大與最小值的數據及日期。

(6) 計算每天的intraday return 2%

同Task A，用Date排序，利用公式計算 intraday return，並利用判斷式計算最大與最小值的數據和日期。

(7) 繪製收盤價隨時間變化的圖 2%

同Task A 方法， 將arr2之Close\_price寫入test4.txt。

(8) 繪製日收益隨時間變化的圖2%

同Task A 方法， 將陣列arrDayilyreturn寫入test5.txt。

(9) 繪製日內收益隨時間變化的圖2%

同Task A，將陣列arrDayilyreturn寫入test6.txt。

在Python環境導入sys和matplotlib函式庫，並使用matplotlib函式庫在圖表中繪製數據的方法亦與Task A相同。差異之處僅在於PLOT函數之filename參數。(7)(8)(9)小題分別為text4.txt、text5.txt、text6.txt。

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述

(10)根據有數字的四個欄位找出最小數字、最大數字、中位數 2%

解題思路同TaskA: 利用前面所得到的不重複的資料，利用HeapSort和DataFrame把4個price集中到一個allprice的dataframe2去做整合比較，其中成員有date與price，並且用運算子多載定義好該型別的輸出，最後用cmpforprice的Struct進行價格排序找到最小、最大、中位數，最後output出正確輸出(也是運用運算子多載)。

三、Problem2

主要變數分析:

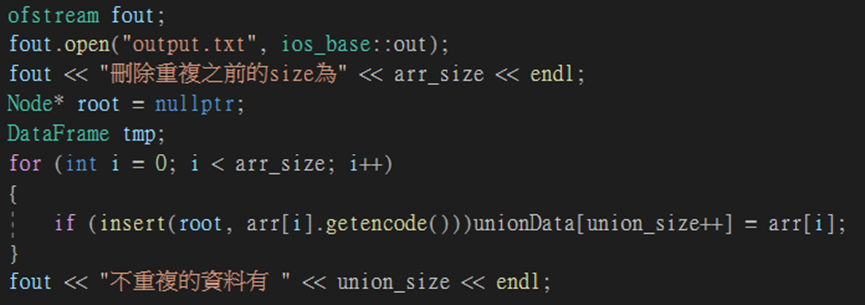
在第二題中由於依照題目我們其實從頭到尾都不需要分析column8、column9的內容，因此我們僅存第1欄到第7欄的內容。並依照需求，我們又額外建立陣列存取5種資料分別為1. 原始資料(arr)2. 不重複資料(unionData)3. TXO\_9900\_201705\_C的資料(unionData2)4. 未排序重複資料(unionData3)5. TXO\_9900\_201705\_C 的重複資料(unionData4)

分析資料流程:

我們是透過c++中讀檔的方式，透過直接寫好檔案的路徑抓取資料(由於資料總共有5份因此會依照資料順序從15存到19號的資料)，而我們發現問題2的資料會有2欄是不必要的，因此會先讀完兩行，接著正常讀取，我們再這裡利用auto的變數型態的方式存取資料，並依照逗號分割文件欄位的資料，並在這裡使用到 「istringstream」的c++風格的資料串流輸入分析，以存入個欄的資料，並將資料存入DataFrame的function之中，以利後續的資料分析。

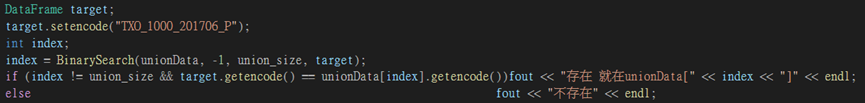
問題分析:

1.這5個dataset裡面有多少個不重複的product(3%)



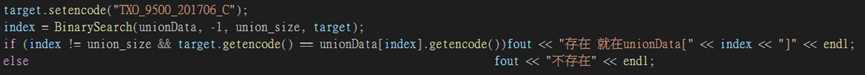
在第1題當中，我們使用到二元搜尋樹（Binary Search Tree），並用此資料結構建立根節點 root。將排序後的陣列 arr 中的元素依次插入二元搜尋樹中，如果插入成功（即該元素在樹中是第一次出現），則將該元素存儲到 unionData 陣列中，並將 union\_size +1。因此，第1題使用了二元搜尋樹和動態陣列作為主要的資料結構。

2. TXO\_1000\_201706\_P 是否存在於這些datasets中。 (3%)



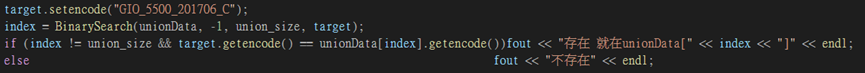
第2題使用了二元搜尋（Binary Search）建立了一個名為 target 的 DataFrame 物件，並設置其 encode 屬性為 "TXO\_1000\_201706\_P"，表示要搜尋的目標。接著，調用 BinarySearch 函式，在 unionData 陣列中進行二元搜尋，尋找目標元素。如果找到了目標元素，則輸出該元素存在於 unionData 中的索引位置；否則，輸出該元素不存在。因此，第2題使用了二元搜尋（Binary Search）來確定特定商品是否存在於資料集 unionData 中。

3. TXO\_9500\_201706\_C 是否存在於這些datasets中。 (3%)



將 target 要搜尋的目標改成 "TXO\_9500\_201706\_C"，其餘步驟與第2題相同。

4. GIO\_5500\_201706\_C 是否存在於這些datasets中。 (3%)



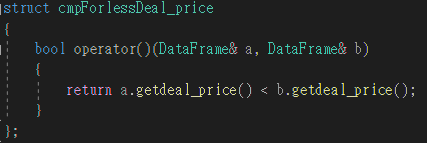
將 target 要搜尋的目標改成 "GIO\_5500\_201706\_C"，其餘步驟與第2題相同。

5.

在第5題中，由於資料只需要分析 TXO\_9900\_201705\_C的資料，因此我們於HeapSortClass中是放入DataFrame的資料集，並因為題目需要找日期，因此我在這裡是讓它依照日期由小到大(使用到cmpForDate(這裡使用到c++中的bool operator的方法比較大小的方式來排序)

(a) 5% 找到10個最低的價格



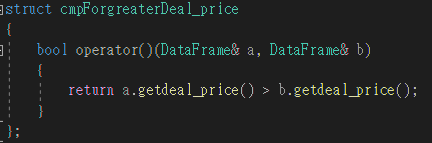


我們透過heapsort的方法，將題目要求的 TXO\_9900\_201705\_C 資料傳入heap sort class中，並呼叫我們自己寫的cmp function中使用到c++中operator的方式將價格由小排到大而其中DataFrame用來表示排序時的比較方法，將HeapSortClass的用這個表示就是根據成交價格從小到大排序。

最後我們會得到由小排到大的資料儲存，並取前10個最小的結果之日期與價格。

(b) 5% 找出10個最大的價格，以及這些最大價格出現的時間。





我們透過heapsort的方法，將題目要求的 TXO\_9900\_201705\_C 資料傳入heap sort class中，並並呼叫我們自己寫的cmp function中使用到c++中operator的方式將價格由大排到小而其中DataFrame用來表示排序時的比較方法，將HeapSortClass的用這個表示就是根據成交價格從大到小排序。

最後我們會得到由大排到小的資料儲存，並取前10個最大的結果之日期與價格。

(c) 5% 尋找該產品的中位數價格。

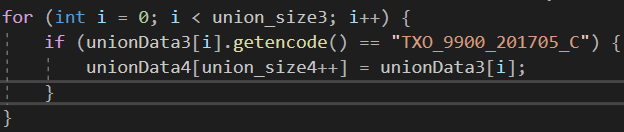


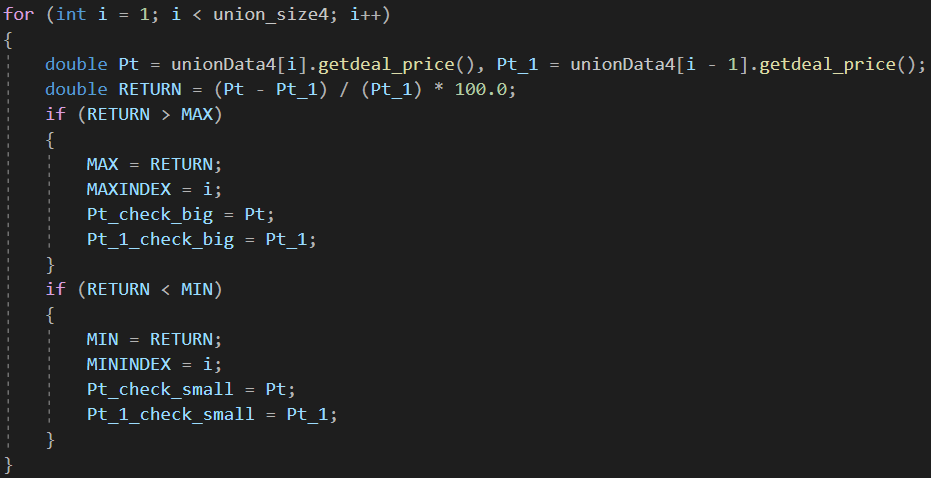
找中位數的方法比較特別，因為不管是從小到大排或是從大到小排最終數都會是中間那個，因此我們在這裡先讓他從小排到大並最後取中間值之價格。

(d) 5% 最大和最小回報率是多少

Note:這裡我們訂某一筆和他的前一筆為excel相鄰的兩筆資料。

解題思路: 這裡使用在一開始讀檔時與arr一起存取資料的unionData3，然後利用unionData3找出指定的product存到unionData4，但不進行HeapSort排序，原因是因為excel是照順序排的，接著再比較前一筆以及這一筆的價格差距，並帶入公式計算出 tick-based return ，找出最大差以及最小差和發生的時間。





四、效能分析

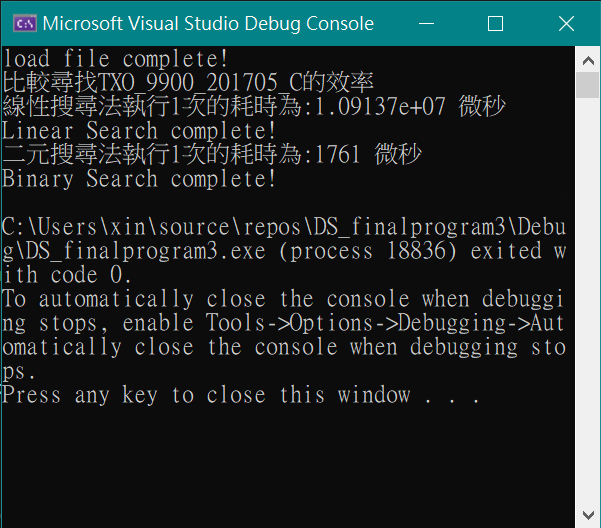
選擇使用Linear Search & Binary Search來做分析

以下為效能分析的步驟:

1.讀檔

2.尋找TXO\_9900\_201705\_C的效率(Linear Search)

3.尋找TXO\_9900\_201705\_C的效率(Binary Search)

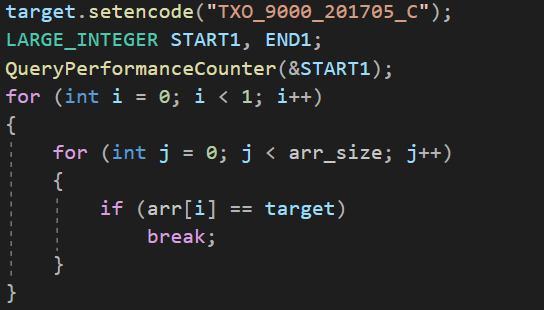


Linear Search: 是指在資料列中從頭開始一一做比對，直到找到相符合的資料為止，下面圖示實作方法。優點: 不用先進行任何排序。缺點: 不適合資料量過大的搜尋。

Note:假設有陣列中有n個元素

Best Case:O(1)，第一個index就是被搜索的元素。

Worst Case:O(n)，目標元素在陣列最後面，因此需要執行n次比對的步驟。



Binary Search: 拿排好順序的資料列從中間砍一半，不斷地遞迴，直到尋找到正確的資料，下面圖示實作方法。優點: 比較快速。缺點: 資料須事先排好。

Note:假設有陣列中有n個元素

Best Case:O(1)，中間的值就是目標數值。

Worst Case:O(lg(n))，最慘需要一直折半，比對lg(n)次。

