**a. pseudo code:**

* 排序副程式**merge sort()**:

step 1. 取出陣列midpoint，根據此midpoint取出左右兩條sub array

step 2. merge sort(起點, midpoint)

step 3. merge sort(midpoint + 1, 終點)

step 4. merge兩條sub array，確保排序正確

* Divide and Conquer副程式**DC():**

step 1. 建立二維座標系，將integrity設為X軸座標，而score設為Y軸座標

step 2. 取出目前按照integrity排序好的陣列之midpoint，以之為垂直線劃分出左、右區塊

step 3. DC(左區塊)

step 4. DC(右區塊)

step 5. 陣列按照score大小進行**merge sort()**

step 6. 若「左區塊」有公司資料且薪水比「右區塊」的學生的desire\_salary更高，則更新此位學生的desire\_salary和decision

* 主程式：

step 1. 定義一個structure，可以包含公司或是學生的資訊：

* isCom(是否為company)
* id(編號)
* integrity(品行)
* score(學期成績)
* salary(公司可提供薪資)
* desire\_salary(學生可獲得的薪資)
* decision(最後學生抉擇的公司)
* side(後續Divide and Conquer標記用)

step 2. 建立overall的structure陣列並匯入student.txt跟company.txt的資料

step 3. 針對integrity(品性成績)作**merge sort()**

step 4. Divide and Conquer來得出學生們的decision --> 使用**DC()** function

step 5. 針對編號(id)作**merge sort()**

step 6. 將學生的編號(id)及最終選擇(decision)循序寫入output.txt

**b. Eexperimental results and analysis (Time complexity)**

* Time Complexity預測：

(ps) student number: n1, company number: n2, n = n1 + n2

* Divide and Conquer副程式**DC()**

根據上頁的步驟分析：

step. 3含有merge sort --> (c1\*nlgn)

step. 4含有merge sort --> (c2\*nlgn)

T(n) = 2T(n/2) + c1\*nlgn + c2\*nlgn = 2T(n/2) + c\*nlgn

by Master theorem,

T(n) = O(nn)

* 主程式：

step 1. 定義一個structure，可以包含公司或是學生的資訊

step 2. 匯入student.txt跟company.txt的資料 --> O(n)

step 3. integrity **merge sort()** --> O(nlgn)

step 4. Divide & Conquer function --> O(nn)

step 5. id **merge sort()** --> O(nlgn)

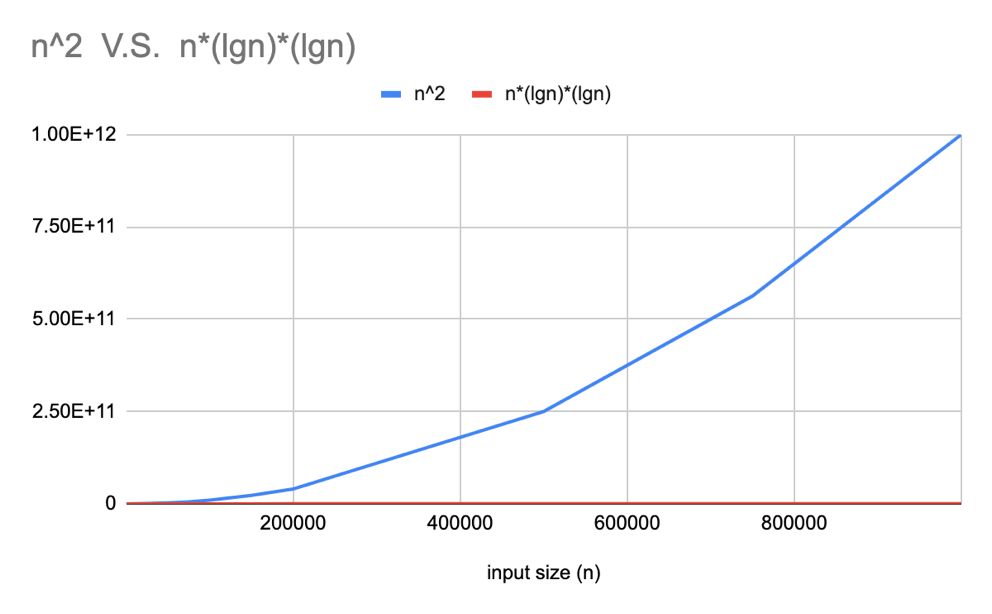
step 6. write in output.txt --> O(n)

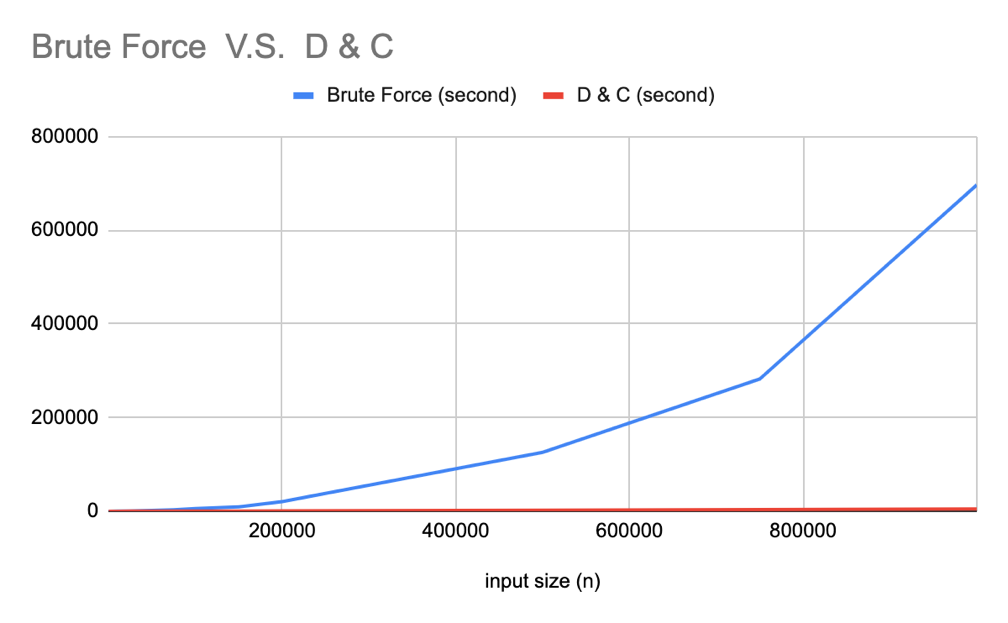
Time Complexity = O(n) + O(nlgn)+ O(nn) + O(nlgn) + O(n) = O(nn)

* Time Complexity 驗證
* 此次設計的Divide-and-Conquer Algorithm主要的bottleneck是在DC()副程式中針對學期成績(score)作merge sort的步驟，因此我設計了一個測試程式來自動生成學生和公司的資料共n筆，其中會將score設計成merge sort的worst case排序方式。
* 另外我也寫了一份Brute Force方法的程式來比對驗證此Algo的正確性。Brute Force O(): 先將公司資料按照薪資大小排序，將著每位學生都檢查所有的公司一遍，看是否符合公司標準。
* 由下圖可以觀察兩種方法對應input size呈現的趨勢，驗證優化後的時間複雜度為O(nn)

* V.S. nn 對應input size的產出值
* Brute Force V.S. Divide-and-Conquer 對應input size的程式執行時間

(ps) 我是在程式頭尾利用兩個變數接收clock()回傳值，相減後得出執行時間。





|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| input size (n) | n^2 | n\*(lgn)\*(lgn) | Brute Force (second) | D & C (second) |
| 1000 | 1000000 | 99316.85641 | 0 | 0 |
| 10000 | 100000000 | 1765633.003 | 80 | 25 |
| 25000 | 625000000 | 5336039.87 | 460 | 60 |
| 50000 | 2500000000 | 12183043.79 | 1490 | 120 |
| 75000 | 5625000000 | 19669887.41 | 2980 | 180 |
| 100000 | 10000000000 | 27588015.67 | 5580 | 250 |
| 150000 | 22500000000 | 44348155.72 | 9260 | 320 |
| 200000 | 40000000000 | 62019887.53 | 20310 | 900 |
| 500000 | 250000000000 | 179202144.2 | 125610 | 2280 |
| 750000 | 562500000000 | 285671238.8 | 282310 | 3420 |