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(nlg^2n)

➢ 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(nlg^2n)

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

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

Time Complexity = O(n) + O(nlgn)+ O($nlog^2n$) + O(nlgn) + O(n) = O(nlg^2n)

- Time Complexity 驗證

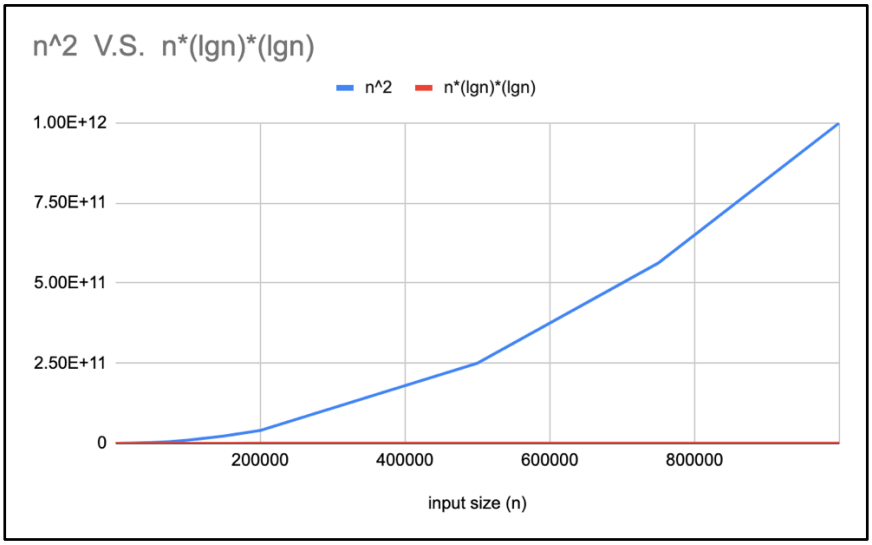
➢ 此次設計的 Divide-and-Conquer Algorithm 主要的 bottleneck 是在 DC()副程式中針對學期成績(score)作 merge sort 的步驟，因此我設計了一個測試程式來自動生成學生和公司的資料共 n 筆，其中會將 score 設計成 merge sort 的 worst case 排序方式。

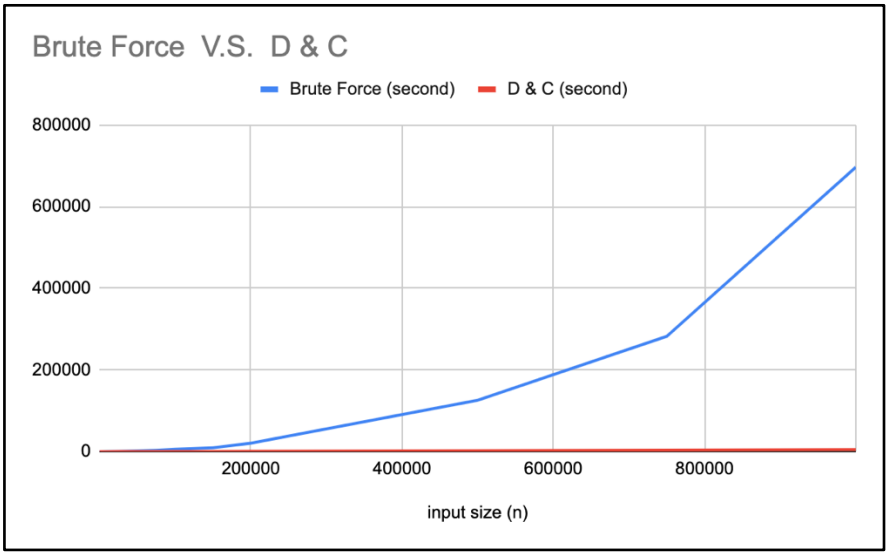
➢ 另外我也寫了一份 Brute Force 方法的程式來比對驗證此 Algo 的正確性。Brute Force O(n^2): 先將公司資料按照薪資大小排序，將著每位學生都檢查所有的公司一遍，看是否符合公司標準。

➢ 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(nlg^2n)

- n^2 V.S. nlg^2n 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