程式設計（109-1）
期中專案

題目設計：孔令傑
國立臺灣大學資訊管理學系

說明：本專案題目的類似題目曾經出現在之前幾份作業。為了閱讀上的方便，在這份文件中我們完整地描述了整個問題。你可以利用你之前的想法或程式碼[1]，不過除非你想多看一些範例，否則你不需要回去看那兩個題目的敘述。

1 題目敘述

1.1 機台、良率與維修

你有一個工廠，裡面有 m 台機台，第 i 台的理論上的生產速率是每小時 a_i 單位，但隨著時間流逝，機台會因為使用時間變長而使生產效率變差，因此在從現在往後數的第 t 小時，機台 i 會有其良率 p_{it}，是一個介於 0 到 1 之間的機率（或比率），例如機台 2 的第 1 小時（此刻起算的 60 分鐘內）的良率為 p_{21}。機台 i 的良率隨著時間以每小時 b_i 的速率下降，例如若 $p_{21} = 95\%$ 且 $b_2 = 2\%$，則 $p_{22} = 93\%$、$p_{23} = 91\%$，依此類推。我們假設機台 i 的良率不會降到 0%，最低只會降到 $L_i > 0$，例如若延續前例且 $L_2 = 90\%$，則 p_{24} 將是 90% 而非 89%，且 $p_{25} = p_{26} = \cdots = 90\%$。題目會給定 $p_{i,0}$ 為初始良率，則給定 $p_{i,0}$ 與 b_i，我們知道 $p_{i,1} = p_{i,0} - b_i$。機台 i 在第 t 小時的真正生產速率為理論速率 a_i 乘上良率 p_{it}，因此以前例而言，機台 2 在第 1 小時的生產量將是 $p_{21}a_2 = 0.95a_2$、第二小時是 $p_{22}a_2 = 0.93a_2$，依此類推。

如果一直放著不管，機台的良率將愈來愈低，如此勢必難以完成工廠被交付的生產任務，因此工廠可以選擇合適的時機做機台維修。當機台進行維修時，維修中時段的良率為 0%，而維修完成後的下一個時間段良率將為 100%（之後再以每小時 b_i 的速率下降）。舉例來說，若機台 1 在第 10 小時時維修完畢，則 $p_{1,11} = 100\%$，$p_{1,12} = 100\% - b_1$，依此類推。在機台 i 進行維修所需的小時數為 c_i，在這些時間段中良率都是 0%。

[1]或許你應該感謝當時你有好好模組化，讓你的某些程式（特別是函數）可以被重新利用。如果你正在咒罵當時的你沒有好好模組化，也請不要介意，因為大部分人都是如此。從今天起好好模組化吧！

1.2 訂單、需求與交期

你接了 n 張訂單（都訂購同一個產品），需要靠你的 m 個機台去完成生產。第 j 張訂單有其訂購量 q_j，因此我們在排定此訂單的生產程序時，需要給它一段至少能生產出 q_j 單位產品的時間（可以多不能少）。

舉例來說，若針對機台 3 我們知道 $a_3 = 100$、$b_3 = 2$、$c_3 = 1$、$p_{30} = 94$、$L_3 = 70$，而我們預計把訂單 1 和訂單 2 放在機台 3 上完成，且 $q_1 = 120$、$q_2 = 180$，則在無維修的情況下，機台 3 的前 5 小時的每小時最大可生產量依序為 $100 \times 92\% = 92$、90、88、86、84。若我們先做訂單 1，則由於第 1 期僅有 92 的生產量，少於其所需生產量 120，因此我們再多安排 1 小時給訂單 1，到第 2 期時累積的最大可生產量為 $92 + 90 = 182$ 單位，超過訂單 1 所需生產量 120，則訂單 1 的生產排程結束。很自然地，訂單 2 便可以從第 3 期開始排程，發現需要到第 5 期時累積的最大可生產量為 $88 + 86 + 84 = 258$，才超過所需生產量 180，因此我們可以把訂單 2 排在第 3 期到第 5 期。

此外，訂單 i 的交期（到期時間）為 d_i，也就是理想上應該要在第 d_i 小時前交貨的時間，若未能在給定的時限內完成，則會有延遲交貨、賠償違約金的情況，造成工廠虧損（提前交貨則沒有獎勵）。以前例來說，若 $d_1 = 3$、$d_2 = 4$，則若我們按照前一段的方式處理訂單 1 與訂單 2，將會在訂單 2 發生 1 小時的延遲，訂單 1 則沒有延遲，因此我們總共有了 1 小時的延遲。

1.3 可行維修方案

現在讓我們來考慮維修。我們對於安排的維修時間有個條件是「不可中斷任一訂單的生產過程」，意思是假設現在有某個訂單在某個機台從第 2 小時開始生產，需要到第 4 小時才做完，則若要插入一個維修，該維修的起始時間只能選擇在第 2 小時該訂單還沒開始生產時即先進行維修（因此該訂單的起始時間會往後延），或是第 4 小時等此訂單整個生產完畢再進行維修（後面如果有之前排定的訂單，則也往後延）。

舉例來說，考慮機台 5，給定 $a_5 = 100$、$b_5 = 10$、$c_5 = 1$、$p_{50} = 80$、$L_5 = 20$，而我們預計在機台 5 上處理訂單 1 到 3，且 $q_1 = 100$、$q_2 = 140$、$q_3 = 60$，$d_1 = 3$、$d_2 = 5$、$d_3 = 7$。假設我們已經決定要先做訂單 1 再做訂單 2，最後做訂單 3，則我們首先可按照上題步驟得知，在沒有維修情況下，三個訂單的開始時間將依序為第 1、3、7 小時，結束時間依序為 2、6、9，總延遲時間為 $0 + 1 + 2 = 3$ 小時。若我們選擇在 $t = 1$ 時進行維修，則 $t = 2$ 時機臺良率將回到 100%，進而我們可以算出三個訂單開始時間會變成 2、3、5，且所有訂單都有在時限前生產完畢，因此總延遲時間為 0。換言之，在 $t = 1$ 維修是最佳策略。

在單一機台上，你可以安排多次維修；各機台的維修時間無須相同也無須錯開，但由於維修人員有限，在同一個小時內最多只能有 h 個機台處在維修狀態。

1.4 生產與維修計畫

在本專案中，你會被給定 m 個機台與 n 張訂單，我們希望你能排定一個生產與維修計畫，以最小化延遲交貨時間。你可以任意地將訂單分配到機台上，並且在單一機台上安排訂單的處理順序。你更可以在合適的時機在各機台上插入維修，以提升機台良率並且增加生產效率。你的目標是最小化總延遲時間。這次的期中專案，就是希望大家可以發想並且嘗試不同的演算法，看看能否讓你的演算法可以得到盡量好的計畫。

2 一個簡單（愚蠢）的演算法

為了協助你進入狀況，這裡我們提供一個簡單（愚蠢？）的演算法。首先，我們把訂單按照編號用類似輪流的方式分給機台，也就是機台 1 負責訂單 1、$m+1$、$2m+1$ 等，機台 2 負責訂單 2、$m+2$、$2m+2$ 等，依此類推。接著針對各機台，我們將交貨日期較早的訂單排在前面，若有交貨日期相同的訂單則將訂購量較少的排在前面，若訂購量也相同則將訂單編號較小的排在前面。我們完全不考慮維修。

以下舉一個簡單的例子。假設 $m = 3$、$n = 8$，且機台資訊與訂單資訊如表 1 與表 2 所述。則排出來的行程將如圖 1 所示，總延遲時間將是 4 小時（訂單 3 延遲 3 小時、訂單 4 延遲 1 小時）。顯而易見地，在機台 3 安排維修應該有幫助，例如若在第 5 小時做維修，則排出來的行程將如圖 2 所示，總延遲時間減為 3 小時（訂單 3 延遲 2 小時、訂單 4 延遲 1 小時）。

機台編號 i	a_i	b_i	c_i	$p_{i,0}$	L_i
1	100	5	3	100	70
2	100	10	2	100	70
3	100	1	1	61	30

表 1: 機台資訊

訂單編號 j	q_j	d_j
1	140	7
2	200	7
3	300	7
4	280	3
5	240	4
6	200	4
7	90	10
8	250	10

表 2: 訂單資訊

小時	1	2	3	4	5	6	7	8	9	10
機台 1 最大生產量	95	90	85	80	75	70	70	70	70	70
機台 1 處理之訂單	4				1		7		idle	
機台 2 最大生產量	90	80	70	70	70	70	70	70	70	70
機台 2 處理之訂單	5			2			8			
機台 3 最大生產量	60	59	58	57	56	55	54	53	52	51
機台 3 處理之訂單	6				3					

圖 1: 使用簡單演算法的生產計畫（無維修）

小時	1	2	3	4	5	6	7	8	9	10
機台 1 最大生產量	95	90	85	80	75	70	70	70	70	70
機台 1 處理之訂單	4				1		7		idle	
機台 2 最大生產量	90	80	70	70	70	70	70	70	70	70
機台 2 處理之訂單	5			2			8			
機台 3 最大生產量	60	59	58	57	0	100	99	98	97	96
機台 3 處理之訂單	6				fix	3				idle

圖 2: 更好的生產與維修計畫

3 輸入輸出格式

系統會提供一共 25 組測試資料，每組測試資料裝在一個檔案裡。每個檔案中會有 $m+n+1$ 行：

- 第一行包含三個整數 m、n 和 h。
- 第二行至第 $m+1$ 行皆包含五個整數，其中第 $i+1$ 行依序為機台 i 的 a_i、b_i、c_i、$p_{i,0}$、L_i。
- 第 $m+2$ 行至第 $m+n+2$ 行皆包含兩個整數，其中第 $m+1+j$ 行依序為訂單 j 的 q_j 與 d_j。

以上每一行中，兩兩整數之間皆被一個空白隔開。其中 $1 \le n \le 1000$、$1 \le m \le 50$、$1 \le h \le n$、$1 \le a_i \le 1000$、$1 \le b_i \le 10$、$1 \le c_i \le 5$、$1 \le p_{i,0} \le 100$、$1 \le L_i \le p_{i,0}$、$1 \le q_j \le 500$、$1 \le d_i$。

讀入資料後，請用 m 行輸出你的計畫，其中第 i 行中包含的數字或字母「M」代表要被依序處理的訂單編號或維修，合起來表示在機台 i 上的計畫。一行中的兩個數字間以一個逗點隔開。舉例來說，如果輸入是

```
3 8 1
100 5 3 100 70
100 10 2 100 70
100 1 1 60 30
150 7
200 7
300 7
280 3
280 4
200 4
90 10
250 10
```

則採用上述的簡單演算法，輸出應該是

```
4,1,7
5,2,8
6,3
```

PDOGS 會讀取你的輸出、檢查可行性、判定為可行，並且計算你得到的總延遲時間是 4。如果你發明了一個更好的演算法，讓你得到如圖 2 的計畫，則你應該輸出

```
4,1,7
5,2,8
6,M,3
```

PDOGS 會讀取你的輸出、檢查可行性、判定為可行，並且計算你得到的總成本是 3。如果你輸出比如說

```
4,1,7
5,2,8
6
```

或

```
4,1,7
5,2,8
6,3,9
```

則會因為不能代表一組合理正確的計畫，在這筆測試資料中將會得到 0 分。

你的.cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你***可以***使用任何方法。

4 評分原則

這一題的其中 75 分會根據程式運算的結果給分。你的程式不需要找出真的能最小化總延遲時間的最佳方案（optimal solution）。只要你的輸出符合規定，且確實是一組可行方案（feasible solution，滿足所有限制式），就會得到分數。對於每一組輸入，PDOGS 會檢查你的輸出，如果輸出格式不合乎要求或方案不可行，則在該筆測試資料會得到零分；如果合乎要求，則對每筆測資，我們依下列公式計分：假設 z 是這組的總延遲時間、z_1 是上述「一個簡單的演算法」的總延遲時間、z_0 是所有組的總延遲時間中最小的，則在這筆測資的得分就是

$$3\left(\frac{z_1 - z}{z_1 - z_0}\right)。$$

寫程式之外，每組還需要合力用中文或英文寫一份書面報告（所謂「寫」，就是用電腦打的意思），以組為單位上傳 PDF 檔至 PDOGS。在報告裡請用文字描述你的演算法（可以用 pseudocode 但不能直接貼 code）、系統的設計（哪個函數做什麼、程式執行的流程等等）、分工方式（誰寫哪個函數、誰負責指揮、誰負責寫書面報告、誰負責買便當等等；當然一個人可以又買便當又寫程式），以及每個人的簡單心得感想。報告***不可以超過八面 A4 紙***。書面報告佔 25 分。這份專案截止後，書面報告才會被批改。

此外，授課團隊會考慮 PDOGS 上的得分以及書面報告上的解法，邀請若干組在合適的上課時間，每組用 10 分鐘跟全班同學介紹自己的解法。被邀請的組可以婉拒上台報告，但上台報告的組可以獲得專案總成績 5 分的額外加分。

5 繳交方式

請修課的同學們自行組成每組四至五人的小組，以組為單位繳交你們的程式和報告。

有兩件事需要注意。首先，系統會以該組內任意一位同學的最後一次上傳得到的分數，做為該組所有人的分數，所以愈傳愈低分是有可能的。其次，原則上 PDOGS 不限制兩次上傳間的時間間隔，但如果有許多組在同個時間大量地上傳執行時間很長的程式，導致 PDOGS 大塞車，屆時我們會對兩次上傳的時間間隔做出限制。

程式的截止時間是 ***12 月 8 日早上八點***，書面報告的截止時間是 ***12 月 10 日早上八點***。