程式設計（113-1）
作業六

作業設計：孔令傑
國立臺灣大學資訊管理學系

繳交作業時，請至 PDOGS（`http://pdogs.ntu.im/`）為唯一的一題上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳），並且到 NTU COOL 上傳一份 PDF 檔。每位學生都要上傳自己寫的解答。不接受紙本繳交；不接受遲交。這份作業的程式碼的截止時間是 ***10 月 18 日早上八點***，書面報告的截止時間則是 ***10 月 20 日早上八點***。為這份作業設計測試資料並且提供解答的助教是黃芷榆和宋凱翔。

本次作業滿分基本上是 110 分，另外有額外加分最多可以加 25 分，得幾分就算幾分。若整學期有 n 份作業，則學期的作業總成績即為 n 份作業的總分除以 n（不論超過 100 與否）。

1 題目敘述

1.1 概述

在本題中，我們要幫一個由多臺電梯組成的電梯組寫一個決定由哪臺電梯去接哪個乘客的演算法，這在業界被稱為「電梯派車演算法」。這個電梯派車演算法會在發生各種事件的時候被執行，例如有乘客在等待區或電梯裡按下按鈕的時候，或者有乘客抵達目的地樓層的時候等等。為了簡單起見，在本題中我們只考慮某個乘客在等待區按下按鈕的瞬間。以下我們詳述這個要被最佳化的電梯派車問題。

在某層樓有一個乘客抵達等待區時，他會按下往上或往下的按鈕呼叫電梯，這種呼叫我們稱之為「外部呼叫」（outer call），此時電梯派車演算法就要分配一臺電梯去接這位乘客。隨著外部呼叫陸續出現，每臺電梯就都會被陸續分配要去接等待中的乘客，因此每臺電梯身上就都會掛著零至數個「已被分配但尚未被服務」的外部呼叫。當一臺電梯到某層樓讓乘客進電梯，我們就說這個外部呼叫被服務了，那麼這臺電梯的「已被分配但尚未服務」的外部呼叫就會減少一個。當這個乘客進入電梯後，如果他要去的樓層還沒有人要去，他就會按下一個按鈕告知電梯他的目的地樓層，此時我們說該電梯獲得了一個「內部呼叫」（inner call）。每臺電梯身上也都會掛著零至數個「已被分配但尚未完成服務」的內部呼叫。等這臺電梯把一個乘客送達目的地，「已被分配但尚未完成服務」的內部呼叫就會減少一個。

截至目前為止，上面的敘述跟作業五第三題的都一樣，但在本題中，我們考慮一

個更實際的情況：在某處新增一個外部呼叫、電梯派車演算法被呼叫的瞬間，可能有很多掛在各電梯上的「已被分配但尚未被服務」的外部呼叫，而因為啟動這些外部呼叫的乘客從來就不知道是哪臺電梯要去接他們[1]，所以電梯派車演算法可以視情況修改之前的分配，同時考慮所有「尚未被分配」和「已被分配但尚未被服務」的外部呼叫，幫這所有的外部呼叫一一分配電梯[2]。在這一題中，讓我們來試著處理這個問題！

1.2 電梯與呼叫的資訊

這棟大樓內共有 n 臺電梯，且大樓共有 K 層樓，分別是一樓、二樓直到第 K 樓。對於這些電梯和這棟大樓，我們知道以下資訊：

- 電梯 i 目前在 f_i^E 樓往方向 d_i^E 前進，其中 $d_i^E = -1$ 表示電梯正在往下走、$d_i^E = 1$ 表示電梯正在往上走，而 $d_i^E = 0$ 表示電梯正靜止不動。

- 在電梯 i 裡面有 m_i^I 個內部呼叫，分別要前往 $t_{i,1}^I$、$t_{i,2}^I$ 直到 $t_{i,m_i^I}^I$ 樓。這些內部呼叫的目的地不會是電梯目前所在的樓層，但可能在電梯目前移動方向的反方向（例如電梯正在六樓往上走，但有一個內部呼叫是要去三樓）。在實務上，一個內部呼叫可能來自多位乘客，但就結果來說就是一個內部呼叫而不是多個，我們也無從得知是幾個乘客要去同一層樓，所以為了簡單起見，我們就會假設每個內部呼叫都恰好對應到一個乘客。一臺電梯的內部呼叫彼此不會重複。如果電梯 i 現在靜止中，它身上不會有任何內部呼叫。

- 如前所述，雖然有些外部呼叫是已經被分配的，但基於它們都尚未被服務，所以都可以被重新分配。因此，我們不用特別區分一個外部呼叫是已分配但尚未被服務還是未分配，總之整棟大樓有 m^O 個外部呼叫，其中第 k 個外部呼叫是要從第 f_k^O 樓往方向 d_k^O 移動，其中 $d_k^O = -1$ 表示相對應的乘客想要往下、$d_k^O = 1$ 表示相對應的乘客想要往上。所有外部呼叫都不會重複，且外部呼叫不會從一樓要往下或從第 K 樓要往上。為了簡單起見，我們假設一個外部呼叫對應到恰好一個乘客，而且這個乘客必須也一定會搭已經被分配好要去接他的電梯了，其他電梯到那一層樓的時候是不會停的（除非剛好有乘客要出電梯，而且即使如此，在等候區等待的乘客還是會繼續等待要接他的電梯抵達才進入那臺電梯）。

給定上述資訊，你的任務是為所有的外部呼叫各分配一臺電梯，基本目標是使得所有呼叫（包含外部和內部呼叫）的總等待時間最小，其中外部呼叫的等待時間被定義為

[1]如果你還是覺得很怪，就請想像應用場域是等待區的乘客看不到各電梯現在哪一樓的大樓吧。

[2]當然，「已被分配但尚未完成服務」的內部呼叫就不能被重新分配了。

從現在起到該乘客走進電梯的時間差，而內部呼叫的等待時間被定義為從現在起到該乘客離開電梯的時間差。

在計算各呼叫的等待時間時，我們需要知道電梯移動和開關門所需的時間。我們假設電梯移動一個樓層（例如從三樓到四樓）需要 s^F 秒，且我們忽略電梯開始移動和準備停靠時會有不同速度，亦即在任何情況下移動一個樓層都需要同樣的秒數。此外，電梯在某層樓停下來開關門一次（不論多少人進出）需要 s^S 秒。在電梯抵達一個樓層時，我們假設要離開電梯的乘客會在電梯抵達該樓層的瞬間就離開電梯，要進電梯的乘客也會在電梯抵達該樓層的瞬間就進入電梯；換句話說，所有的等待時間都不包含在所屬樓層開關門的秒數。

1.3 電梯的移動規則

若要完成給定的演算任務，在我們將若干外部呼叫分配給 n 臺電梯中的任何一臺，我們必須有辦法知道該電梯會以什麼方式移動、處理其身上的內部呼叫和分配給它的外部呼叫，才能計算這臺電梯即將服務的所有乘客各需要等多久。為了讓題目簡單一點，在本題中電梯的運作方式是受限的，或者說有點單純。

首先，我們已經知道第 i 臺電梯裡面有 m_i^I 個內部呼叫。假設第 i 臺電梯被我們分配了 m_i^O 個外部呼叫，則它可以知道這 $m_i^I + m_i^O$ 個呼叫都發生在哪些樓層，以及自己所在的樓層 f_i^E，進而知道這些樓層中的最低樓層 L_i（未必是一樓）與最高樓層 H_i（未必是第 K 樓）。在算出 L_i 和 H_i 後：

- 如果電梯 i 現在正在往上，它就會先從目前樓層 f_i^E 往上移動到第 H_i 樓，接著折返往下到 L_i 樓，然後折返往上回到第 f_i^E 樓。這些折返都不花費任何時間。如果在第 H_i 樓有個往下的外部呼叫，電梯折返時會轉為往下，讓該外部呼叫對應到的乘客在電梯一抵達該樓層時就進電梯。

- 如果電梯 i 現在正在往下，它就會先從目前樓層 f_i^E 往下移動到第 L_i 樓，接著折返往上到 H_i 樓，然後折返往下回到第 f_i^E 樓。這些折返都不花費任何時間。如果在第 L_i 樓有個往上的外部呼叫，電梯折返時會轉為往上，讓該外部呼叫對應到的乘客在電梯一抵達該樓層時就進電梯。

- 如果電梯 i 現在正在靜止中，它會馬上考慮往上或往下然後順著走一圈，試算哪一個會讓自己身上的所有呼叫的等待時間總和最小，然後選擇往該方向前進。

在給定如上的電梯移動演算法後，每臺電梯都會（預期自己會）照這個方式移動，然後沿路服務內部呼叫（讓電梯內的乘客離開電梯）和外部呼叫（讓等候區的乘客進入電梯），直到服務完所有呼叫為止。

為了簡單起見，我們在計算時只需要考慮當下已知的所有呼叫，不考慮未來可能出現的新呼叫，包含乘客走進電梯後新增的內部呼叫。基於這個原則，每個呼叫的「等待時間」只是一個預計的、估計的等待時間，不一定是該呼叫真正的等待時間，但本題中我們不要求大家估計或計算真正的等待時間。

1.4 同樓層分配與容量上限

在分配電梯時，有兩個議題需要留意，分別是同樓層分配與容量限制。

首先，如果一臺電梯正在某一層樓移動中（往上或往下），則若在這個瞬間在該樓層有一個跟電梯移動方向同向的外部呼叫，而且我們將此電梯分配給該外部呼叫，那麼也會因為電梯無法突然停下，該外部呼叫會等到電梯順著走一圈又回到該樓層時才服務到該乘客。但如果電梯是在該樓層靜止，則若我們將此電梯分配給該外部呼叫，電梯就會馬上開門讓該乘客進入電梯，該乘客的等待時間就是 0。

其次，我們都知道在真實世界中每臺電梯的容量都是有限的，我們用 c 表示一臺電梯原則上應該載最多 c 個人。演算法分配了外部呼叫給電梯後，每臺電梯就會去執行各自的任務；如果分配得不好，就有可能發生太多人要擠進一臺電梯的情況，那自然是不理想的分配。為了簡單起見，在本題中我們假設不論超過 c 多少，所有人都一定會擠進電梯，但電梯裡的人數超過 c 愈多，就是愈糟糕的分配。如果在同一個樓層同時有乘客要進和出電梯，所有人都會遵守先出後進原則（這也會讓計算出的超載程度相對於先進後出比較低）。

1.5 最佳化目標

每當給定了一組大樓、電梯與呼叫的資訊，就相當於獲得了一個電梯分配問題的實例，此時不同的演算法可能會給出不同的分配，而不同的分配可能有好壞之別。在本題中，我們關心所有外部呼叫的等待時間、所有內部呼叫的等待時間，以及電梯超載的程度，將這三個數值綜合計分。具體來說，令 α、β、γ 為三個做為權重的正整數、w_k^O 為第 k 個外部呼叫的等待時間、w_{ih}^I 為電梯 i 的第 h 個內部呼叫等待時間、u_k^O 為外部呼叫 k 進入電梯後該電梯內的人數，則我們的最佳化目標為

$$\min\ \alpha\sum_{k=1}^{m^O} w_k^O + \beta\sum_{i=1}^{n}\sum_{h=1}^{m_i^I} w_{ih}^I + \gamma\sum_{k=1}^{m^O}\max\{u_k^O - c, 0\}\text{，}$$

亦即最小化所有外部呼叫的等待時間、所有內部呼叫的等待時間、所有外部呼叫進電梯後的超載程度三者的加權和。

1.6 計算範例

我們用以下這個例子說明相關計算的過程。假設 $n = 2$、$K = 8$、$c = 1$、電梯移動一層需要 $s^F = 5$ 秒，開關門一次需要 $s^S = 10$ 秒。假設電梯 1 現在正在三樓靜止，電梯 2 正在八樓往下；電梯 1 因為靜止，身上自然沒有任何呼叫，而電梯 2 身上有兩個內部呼叫，分別要前往六樓和三樓，還有一個已分配的外部呼叫，要從七樓往下。

如果有一個待分配的外部呼叫要從六樓往下，則加上剛剛已被分配的外部呼叫，可以被（重新）分配的外部呼叫共有兩個，包含已分配但尚未被服務的「七樓往下」，以及未分配的「六樓往下」。把兩個外部呼叫分配給兩臺電梯，共有四個分配方案：

- 分配「六樓往下」給電梯 1、「七樓往下」給電梯 2：

  電梯 1 會發現自己往上或往下沒有差別（因為往下的話也會瞬間折返），都相當於往上，在往上抵達六樓後開門服務此外部呼叫，因此這個待分配的外部呼叫的等待時間是 $5 \times |6 - 3| = 15$ 秒。請注意當電梯抵達六樓開門的瞬間，該外部呼叫就算是被服務了，因此其等待時間不包含電梯開關門所需的時間。

  電梯 2 要服務完自己身上已經被分配的任務，為此它會持續往下，在七樓、六樓和三樓分別完成服務外部呼叫、內部呼叫和內部呼叫，三者的等待時間分別是 $5 \times |8 - 7| = 5$ 秒、$5 \times (|8 - 7| + |7 - 6|) + 10 \times 1 = 20$ 秒（要包含電梯在七樓開關門一次所花的時間，但不包含在六樓開關門的時間）、$5 \times (|8 - 7| + |7 - 6| + |6 - 3|) + 10 \times 2 = 45$ 秒。

  綜合以上，所有外部呼叫的總等待時間是 $15 + 5 = 20$ 秒、所有內部呼叫的等待時間是 $20 + 45 = 65$ 秒。最後，電梯 1 沒有發生超載，但電梯 2 在「七樓往下」的外部呼叫乘客進電梯後共有 3 個乘客在電梯裡，基於電梯理想容量是 $c = 1$，發生了 2 單位超載，所以所有外部呼叫進電梯後的總超載量是 $0 + 2 = 2$ 人。

- 分配「六樓往下」給電梯 2、「七樓往下」給電梯 1：

  電梯 1 會發現自己往上或往下沒有差別（因為往下的話也會瞬間折返），都相當於往上，在往上抵達七樓後開門服務此外部呼叫，因此這個待分配的外部呼叫的等待時間是 $5 \times |7 - 3| = 20$ 秒。

  電梯 2 要服務完自己身上已經被分配的任務，為此它會持續往下，在六樓同時完成一個內部呼叫和開始服務一個外部呼叫，然後在三樓完成服務一個內部呼叫，前二者的等待時間都是 $5 \times |8 - 6| = 10$ 秒，最後一個的則是 $5 \times (|8 - 6| + |6 - 3|) + 10 \times 1 = 35$ 秒。

  綜合以上，所有外部呼叫的總等待時間是 $20 + 10 = 30$ 秒、所有內部呼叫的等待時間是 $10 + 35 = 45$ 秒。最後，電梯 1 沒有發生超載，而電梯 2 在抵達六樓時

會先有一位乘客離開電梯，然後「六樓往下」的外部呼叫乘客進入電梯後共有 2 個乘客在電梯裡，發生了 1 單位超載，所以所有外部呼叫進電梯後的總超載量是 $0+1=1$ 人。

- 把「六樓往下」跟「七樓往下」都分配給電梯 2：

  電梯 1 可以繼續維持靜止。

  電梯 2 會持續往下，在七樓、六樓、六樓和三樓分別完成服務外部呼叫、內部呼叫、外部呼叫和內部呼叫，四者的等待時間分別是 $5\times|8-7|=5$ 秒、$5\times(|8-7|+|7-6|)+10\times 1=20$ 秒、20 秒（同前一個呼叫的等待時間）、$5\times(|8-7|+|7-6|+|6-3|)+10\times 2=45$ 秒。

  綜合以上，所有外部呼叫的總等待時間是 $5+20=25$ 秒、所有內部呼叫的等待時間是 $20+45=65$ 秒。最後，電梯 1 沒有發生超載，而電梯 2 在抵達七樓時會發生 2 單位超載，抵達六樓時又發生一次 2 單位超載，所以所有外部呼叫進電梯後的總超載量是 $0+4=4$ 人。

- 把「六樓往下」跟「七樓往下」都分配給電梯 1：

  電梯 1 會發現自己往上或往下沒有差別（因為往下的話也會瞬間折返），都相當於往上，在往上抵達七樓後開門服務一個外部呼叫，再折返往下到六樓服務另一個外部呼叫，兩者的等待時間分別是 $5\times|7-3|=20$ 秒以及 $5\times(|7-3|+|7-6|)+10\times 1=35$ 秒。

  電梯 2 會持續往下，在六樓和三樓分別完成服務兩個內部呼叫，兩者的等待時間分別是 $5\times|8-6|=10$ 秒、$5\times(|8-6|+|6-3|)+10\times 1=35$ 秒。

  綜合以上，所有外部呼叫的總等待時間是 $20+35=55$ 秒、所有內部呼叫的等待時間是 $10+35=45$ 秒。最後，電梯 1 在抵達六樓時會發生 1 單位超載，而電梯 2 因為沒有服務任何外部呼叫，自然沒有超載，所以所有外部呼叫進電梯後的總超載量是 $1+0=1$ 人。請注意雖然電梯 2 一開始就載了超過 $c=1$ 人，但因為本題中超載程度的計算規則是只計算「外部呼叫進入電梯後」的超載情況，所以在這種分配方案下應該被視為沒有超載。

綜合以上分析，各種方案在三種指標上各有好壞。給定 α、β 和 γ 的值之後，就可以比較出四個方案的好壞了。

2 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。在每個檔案中會有 $2n+2$ 列：

- 整體資訊：第 1 列裝著八個整數，依序是 n、K、c、s^F、s^S、α、β 和 γ。已知 $2 \le n \le 10$、$2 \le K \le 100$、$1 \le c \le 15$、$5 \le s^F \le 20$、$5 \le s^S \le 20$、$0 \le \alpha \le 100$、$0 \le \beta \le 100$、$0 \le \gamma \le 100$。
- 電梯資訊：第 2 列到第 $n+1$ 列中，由第 2 列起算的第 i 列裝著兩個整數，依序是 f_i^E 和 d_i^E。已知 $f_i^E \in \{1, ..., K\}$、$d_i^E \in \{-1, 0, 1\}$。
- 內部呼叫資訊：第 $n+2$ 列到第 $2n+1$ 列中，由第 $n+2$ 列起算的第 i 列裝著 $m_i^I + 1$ 個整數，第一個整數是 m_i^I，接著依序是 $t_{i,1}^I$、$t_{i,2}^I$ 直到 $t_{i,m_i^I}^I$。已知 $0 \le m_i^I \le K$、$t_{ij}^I \in \{1, ..., K\}$。
- 外部呼叫資訊：第 $2n+2$ 列裝著 $2m^O+1$ 個整數，第一個整數是 m^O，接著依序是 f_1^O、d_1^O、f_2^O、d_2^O 直到 $f_{m^O}^O$ 和 $d_{m^O}^O$。已知 $1 \le m^O \le 2(K-1)$、$f_k^O \in \{1, ..., K\}$、$d_k^O \in \{-1, 1\}$。

每一列的任兩個數字之間被一個空白字元隔開；所有參數的值會遵循題目敘述中提到的不重複、不相同等等規則。

讀入上述資訊後，請決定要分配哪臺電梯給外部呼叫 1、外部呼叫 2 直到外部呼叫 m^O，並且依序輸出被分配之電梯的編號，兩個數字之間以一個逗點隔開。舉例來說，如果輸入是

```
2 8 1 5 10 1 1 20
3 0
8 -1
0
2 6 3
2 7 -1 6 -1
```

其中最後一列表示有 2 個外部呼叫，第一個是「七樓往下」，第二個是「六樓往下」。則一組可行的輸出是

```
1,2
```

表示把第一個外部呼叫「七樓往下」分配給電梯 1、第二個外部呼叫「六樓往下」分配給電梯 2。由於 $\alpha = \beta = 1$ 而 $\gamma = 20$，所以這個分配方案的目標式值就會是

$$1 \times 30 + 1 \times 45 + 20 \times 1 = 95。$$

另一組可行的輸出是

```
2,2
```

表示把兩個外部呼叫都分配給電梯 2，目標式值是

$$1 \times 25 + 1 \times 65 + 20 \times 4 = 170。$$

所以單以這兩個方案來比較的話，把第一個外部呼叫「七樓往下」分配給電梯 1、第二個外部呼叫「六樓往下」分配給電梯 2 比較好。

3 評分原則

這一題的其中 75 分會根據程式運算的結果給分，共有 25 筆測試資料。你的程式不需要找出真的能最小化目標函數的最佳解（optimal solution）。只要你的程式在時間和記憶體限制內跑完、輸出符合格式規定，且確實是一組可行解（feasible solution），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（例如只有 $n = 3$ 臺電梯，但你把電梯 7 分配給某個外部呼叫），則在該筆測試資料會得到零分；如果合乎要求，且目標式值非負，則對每筆測資，我們依下一段的方式計分。

首先，助教會寫一個程式，在 PDOGS 上取名為「TA-algorithm」，這個演算法會稍微有點聰明（但不會太誇張）。更具體地說，TA-algorithm 的演算法會執行數輪，在第 k 輪中幫第 k 個外部呼叫分配在那一輪看起來能讓目標式值最低的一臺電梯（如果有平手，就從中挑編號最小的），然後就固定給這個外部呼叫的分配不再更改，接著進行下一輪，直到所有外部呼叫都有被分配電梯。你可能已經發現這個演算法不一定會找到最佳解，但做為給大家的比較基準，應該還算合適。

計分時，每一筆測試資料原則上佔 3 分。假設你的輸出合乎格式，且投放方案是可行的，PDOGS 就會幫你的方案計算目標式值。假設 z 是你的解得到的目標函數值、z_{TA} 是「TA-algorithm」得到的目標函數值，則你在這筆測資的得分就是

$$1.5 + 1.5\left(\min\left\{1.4, \frac{2z_{TA} - z}{z_{TA}}\right\}\right)。$$

換言之，只要方案可行就會得到 1.5 分（所以輸出 m^O 個 1 其實也可以得分），然後你的分配方案愈好（z 愈小），就愈能得到好成績。要留意的是，如公式所示，如果

$z > 2z_{TA}$，亦即你的解比「TA-algorithm」的兩倍還要糟，那你得到的分數會比 1.5 分還低；與之類似，如果你的解比「TA-algorithm」還要好，你會得到超過 3 分，但最多就是 $1.5 + 1.5 \times 1.4 = 3.6$ 分[3]。

在 25 筆測試資料中，為了讓同學們比較容易拿到分數，前 15 筆測試資料的範圍會限縮在 $n \leq 5$、$K \leq 20$，同學們即使實作了比較無效率的演算法，也不是很容易在執行時超過 PDOGS 上設定的時間上限。最後 10 筆測試資料則是題目規模相對大，不夠有效率的演算法可能會在時限內跑不完。同學們也可以針對不同規模的題目設計不同的演算法，在讀入資料後根據讀入的參數值大小決定要呼叫哪一個演算法即可。

寫程式之外，你還要寫一份書面報告（所謂「寫」，就是用電腦打的意思），並且上傳書面報告 PDF 檔至 NTU COOL。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），以及簡單心得感想。報告***不可以超過四面 A4 紙***。書面報告佔 35 分。

此外，授課團隊會考慮 PDOGS 上的得分以及書面報告上的解法，可能邀請若干同學在合適的上課時間，每位同學用 10 分鐘跟全班同學介紹自己的解法。被邀請的同學可以婉拒上台報告，但上台報告的同學可以獲得本次作業成績最多 10 分的額外加分。

4 繳交方式

請修課的同學們以個人為單位繳交你們的程式和報告。程式方面只要是 C++ 就可以，換言之，可以使用任何語法。書面報告方面可以用中文或英文，並且應該盡量讓自己的報告清爽、好閱讀；NTU COOL 上的「PD_reportFormatGuideline.pdf」有提供最基本的寫作與排版指引，強烈建議所有同學寫報告前先讀過一遍。

有兩件事需要注意。首先，系統會以每位同學的最後一次上傳得到的分數，做為該同學的分數，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，但如果有許多同學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。

[3]大家可能有發現，我們已經把「TA-algorithm」的演算法告訴大家了。所以這次作業一方面是鼓勵有興趣的同學們積極挑戰，多拿分數的同時也看看自己能做得多好，但另一方面也不會為難同學們，只要能照著 TA-algorithm 的演算法正確實作，就幾乎可以得到滿分。