2023 國際運算思維挑戰賽

編輯團隊：李忠謀、柯佳伶、林于立
黃子容、葉宜珊、梅宜琇

教育部 國立臺灣師範大學 資訊工程系

Bebras
國際運算思維挑戰賽

International Challenge
on Informatics and Computational Thinking

2023任務解析

目錄

挑戰賽介紹

國際運算思維挑戰賽 (International Challenge on Informatics and Computational Thinking，簡稱 Bebras Challenge) 自 2004 年開始，每年於 11 月的國際 Bebras 週 (World-Wide Bebras Week) 在全球各國同步舉行。

Bebras Challenge 採用淺顯易懂且生活化的情境式任務，參與學生需運用抽象化、演算法設計、問題拆解、模式辨識、樣式一般化、自動化等運算思維 (Computational Thinking) 來解決問題挑戰。

Bebras Challenge 可以讓任課教師了解學生的運算思維知能，發掘具備資訊科學性向的學生，亦希望透過問題思考過程激起學生對資訊科學的學習興趣。

挑戰賽目標

激發學生對資訊科學之學習興趣

Bebras Challenge 藉由情境式的任務，在挑戰的問題中融入資訊科學基本概念；目的是讓學生了解生活中隨處可見資訊科學概念的運用，認識相關概念具廣泛的應用性，進而激發學生對資訊科學的學習興趣。

提升學生運用運算思維解決問題之能力

Bebras Challenge 的任務以家庭生活、團體合作、工作安排等生活情境，引導學生思考進而解決問題。解題過程運用的是運算思維及問題解決能力，學生僅需該年齡層的基本知識即可作答。從任務敘述中推理出問題重點及解題方向，亦可引發學生高層次思考，提升運用運算思維解決問題的能力。

降低學生對資訊科學之恐懼

Bebras Challenge 將抽象的資訊科學知識具體化，以日常生活中會遇到的情境或故事呈現，題組內容有趣生動，有助於降低學生對資訊科學的學習焦慮。未曾受過資訊科學正式課程的學生亦能運用邏輯、歸納、推理、運算等能力進行解題，讓學生對資訊科學的學習具有信心。

實施地點

Bebras Challenge 為了讓學生在熟悉的學習環境中進行，全球皆由學校教師於課堂中利用一節課的時間（國小挑戰時間 36 分鐘，國高中皆為 45 分鐘）於線上實施。由於需在課堂中進行，故僅能由教師為學生報名。挑戰賽需透過瀏覽器登入系統進行，使用電腦或是行動裝置皆可（為了瀏覽任務之視覺舒適度，行動裝置建議使用平版電腦）。

參與對象

Bebras Challenge 參與學生並無特定資格限制。除每年 11 月與全球同步舉行，4 月亦增辦一場；只要是在學學生，皆可透過教師整班報名參加對應年級之挑戰賽組別。挑戰賽依學生年級共分 6 個組別，詳細分組情形請見計分方式。我國於 2016 年起每年舉辦 Benjamin、Cadet、Junior、及 Senior 4 個組別，未來將視參與情形向下推廣。

計分方式

Bebras Challenge 依年級分組，各組別的任務分成 3 種難度（易、中、難），根據任務難度有不同計分標準：答對給分、答錯扣分，略過未答則不給分亦不扣分，各組別分數計算分式如下表所示。Benjamin 組別的題數為 12 題，每個難度各 4 題；其他組別的題數為 15 題，每個難度各 5 題。為了避免總分有負分，各組別的預設總分為 60 分，各題皆答錯扣分得到最低 0 分，全部皆答對得到最高分 300 分。

		難度						題數
		易		中		難		
		正確	錯誤	正確	錯誤	正確	錯誤	
Pre-Primary	一、二年級	尚未開放						
Primary	三、四年級	尚未開放						
Benjamin	五、六年級	16	-4	20	-5	24	-6	12
Cadet	七、八年級	12	-3	16	-4	20	-5	15
Junior	九、十年級	12	-3	16	-4	20	-5	15
Senior	十一、十二年級	12	-3	16	-4	20	-5	15

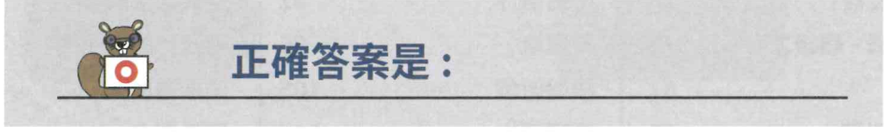
本書說明

運算思維挑戰賽題組介紹頁 (P.6) 列出本年度本國在各個組別中不同難度的選用任務。

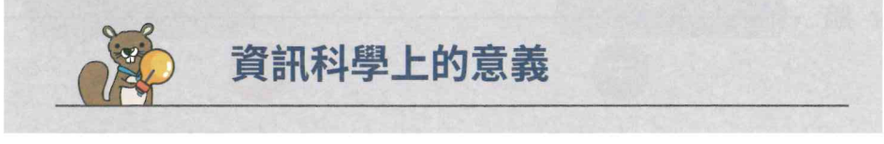
本書中的每一題任務，除了任務背景敘述及問題，會提供**命題國家**及**任務對不同組別學生的難度**、**參考詳解**、**此任務運用的資訊科學概念**、及**進階學習關鍵字**等說明，如以下四個圖所標示。

頁首難度

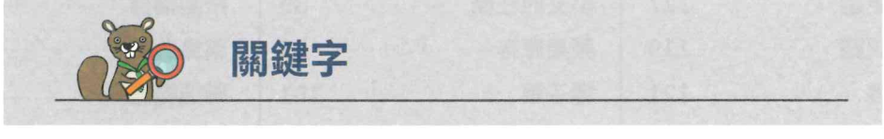
參考詳解



此任務運用之資訊科學概念



進階學習的關鍵字



2023 運算思維挑戰賽題組

Benjamin 組

Cadet 組

Junior 組

Senior 組

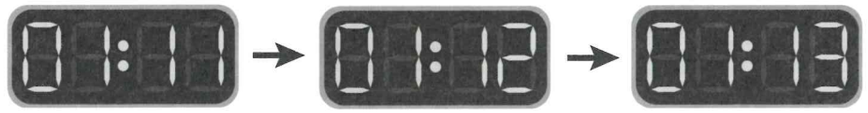
1. 壞掉的電子鐘

電子鐘上顯示小時和分鐘的每個數字都由 7 條 LED 組成，每條 LED 可設為「亮」或「暗」，來組合顯示一個數字。

下圖分別為數字 0 和 2 所顯示的樣子。

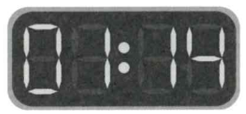
小狸的電子鐘，在顯示分鐘的個位數字有一條 LED 故障無法亮起。

下面的圖像是這個電子鐘從 1 點 11 分到 1 點 13 分顯示的樣子。

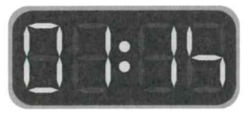


請問這個電子鐘在 1 點 14 分會顯示什麼樣子？

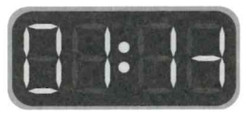
A.



C.



B.



D.

正確答案是：B

任務中的圖像顯示了電子鐘連續三分鐘的狀態，其中分鐘的個位數字 (分別是 1、2 和 3) 皆正常顯示。可以從中發現以下 6 條 LED 皆可正常亮起。

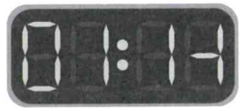
因此，故障的 LED 條一定在左上位置，如圖一。

1 點 14 分時，分鐘個位數 4 因為故障的左上位置 LED 條不會亮，因此顯示的樣子將如圖二。

圖一

圖二

因此，電子鐘呈現的樣子會如選項 B 所示：



選項 A 是錯誤的，若電子鐘會顯示此結果，表示分鐘個位數字的每個 LED 條都沒有故障。

選項 C 和 選項 D 也是錯誤的，因為它們故障的 LED 條在前面幾分鐘是可正常顯示的。

資訊科學上的意義

運用 **邏輯推論 logical inference** 和 **除錯 debugging** 可以找出本任務的電子鐘內壞掉的 LED 燈。邏輯推論是進一步從一些已知的事實進行條件判斷，並推論出新的資訊。而除錯則是透過分析執行錯誤或是不符合預期的程式，進而找出並改善程式中的錯誤指令。

資訊科學家在運用電腦程式解決問題時，需要運用邏輯推理的能力來設計程式，有時也需要反覆驗證並對程式進行除錯，才能確保程式可以有效的解決問題。

日常生活中，可將解決某項問題的檢查項目整理成清單，再依清單中的每個項目逐一進行檢查。例如：當遙控汽車無法動作時，會逐一檢查是汽車的開關沒有打開，遙控器的開關沒有打開，汽車本身的電池沒電了還是遙控器的電池沒電，或是有某些零件損壞影響運作，一一檢查進行故障排除，找出問題讓功能恢復正常，這就是除錯的過程。

關鍵字

邏輯推論、除錯

2. 海狸的寶藏

如下圖所示，某座島上有 3 個空的寶箱，分別放在山腳、棕櫚樹下和海邊。

有一天，海狸小寶把寶藏裝進其中一個寶箱。

三位尋寶人分別拍了一張寶箱的照片，如下所示：

		
小錢拍到海邊的寶箱	小帥拍到海邊和棕櫚樹下的寶箱	小卡拍到山腳和棕櫚樹下的寶箱

其中一張照片是在寶藏裝進寶箱之前拍的，另外兩張則是之後才拍的；但這三張照片都只拍到空的寶箱。

請問小寶把寶藏放在哪個寶箱裡呢？

A. 棕櫚樹下的寶箱

B. 海邊的寶箱

C. 山腳的寶箱

D. 不能確定在哪個寶箱

正確答案是：C

由於只有一張照片是在寶藏裝進寶箱之前拍攝的，因此可以分別檢驗三個寶箱中，哪一個裝有寶藏才符合拍攝時間的條件，來確認寶藏可能在哪一個寶箱中：

- 假設寶藏在棕櫚樹下的寶箱裡，那麼小帥和小卡的照片都必須在裝寶藏前拍攝，不符合拍攝時間的條件；故寶藏不可能在棕櫚樹下的寶箱中。
- 假設寶藏在海邊的寶箱裡，那小錢和小帥的照片都必須在裝寶藏前拍攝，不符合拍攝時間的條件；故寶藏不可能在海邊的寶箱中。
- 假設寶藏在山腳下的寶箱裡，那只有小卡的照片是在裝寶藏前拍攝，符合線索的特徵；故寶藏可能在山腳下的寶箱中。

綜合以上三點，小寶只可能將寶藏裝在山腳下的寶箱中，因此正確答案是 C。

資訊科學上的意義

邏輯推論 logical inference 是進一步從一些已知的事實進行條件判斷，並推論出新的資訊。此過程中，若某條件成立，則稱該條件為 真 true，反之則稱該條件為 假 false。

在本任務中，已知的事實包含三張寶箱的照，和其中有兩張是在寶藏放入寶箱之後拍攝的。透過驗證不同條件假設，再將驗證為真的條件敘述整合後，進而找到正確答案的過程，就是邏輯推輪。

邏輯推論是資訊科學領域中很重要的能力，程式中經常會運用條件的邏輯組合來判斷程式執行的狀況，並依判斷的結果決定接下來相對應的執行指令。

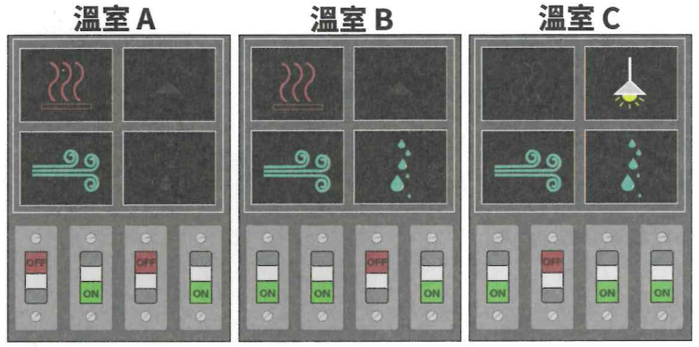
關鍵字

邏輯推論

3. 控制桿

太空站有 3 間溫室（A、B、C），每間溫室都設有一樣的控制台，上面有 4 組開關及燈號；開關分別控制溫室的暖氣機、抽風機、電燈和加濕機，燈號顯示正在運轉中的電器。若是太空人操作錯誤，會讓那間溫室中的植物枯萎。

因為控制台老舊，上面的標籤都脫落了，大空人只知道每間溫室控制台上的開關位置相同，太空人可以透過觀察控制台面板來確認每個開關是控制哪個電器。



上圖是各溫室控制台現在的狀態，請問控制台上的開關由左到右分別控制什麼電器？

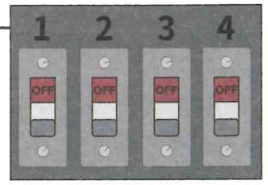
A. 

B. 

C. 

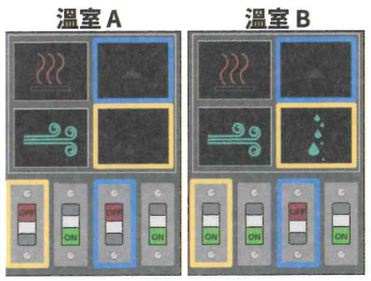
D. 

正確答案是：C

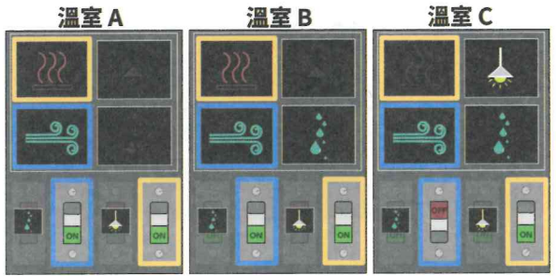


假設每個控制台上的開關由左至右分別編號 1 到 4，要找出控制台上的每個開關分別控制什麼電器，最直接的方法是比較 3 間溫室內控制台上的開關狀態、與面板上的電器燈號。

- 溫室 A 和 B 的燈號只有加濕器不同，且這兩間溫室只有 1 號開關狀態不同，由此推論 1 號開關控制著加濕器。
- 溫室 A 和 B 的電燈都是關著的，且這兩間溫室只有 3 號開關同樣關閉著，由此推論 3 號開關控制著電燈。



- 觀察 3 個控制台上的燈號，可以看出 3 間溫室的暖氣機狀態分別是「開、開、關」，而開關狀態同樣是「開、開、關」的只有 2 號開關，由此推論 2 號開關控制著暖氣機。
- 剩下的 4 號開關在 3 台控制台上都開著，對應 3 溫室的抽風機的確都開著，因此 4 號開關控制著抽風機。



資訊科學上的意義

邏輯運算 logical operation 透過運算子，對布林值（真 true 或假 false）進行運算；常見的運算子包括 且 AND、或 OR、非 NOT 等。

邏輯運算常運用在邏輯推論的過程中：運用邏輯運算，將某些已知的條件訊息結合後，推論出更多訊息。如本任務中，若用布林值來表示電器開關狀態，即可將開啟視為真、關閉視為假；再進一步利用三間溫室的電器開關狀態，推論出所有開關控制的電器分別為何。

邏輯運算常被運用在推理遊戲中，如數獨、1A2B 猜數字等，都是從某些已知條件，運用邏輯運算推論出更多的訊息，再進一步從所有訊息中，找出正確的解答或可能解決問題的方法。

邏輯運算與邏輯推論是電腦科學的基礎，也是資訊科學家透過程式解決問題時，經常用來檢驗程式正確性的方法。

關鍵字

邏輯運算

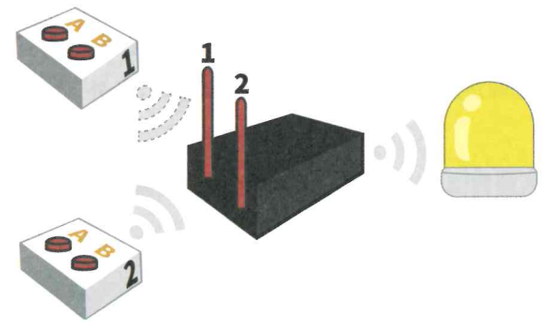
4. 天線遙控器

小海狸找到兩支遙控器（1 號和 2 號）上面各有兩個按鈕（A 和 B），且會透過下表的方式發送訊號：

按鈕 A	按鈕 B	發送的訊號
關	關	開
開	關	開
關	開	開
開	開	關

小海狸又找到一支控制花園燈的天線型遙控器，小海狸還發現它的功能與 1 號和 2 號遙控器相同，只是按鈕 A 和 B 換成天線 1 及 2。

1 號遙控器的訊號會傳到天線 1，2 號遙控器的訊號會傳到天線 2。



如果 1 號遙控器的按鈕都關著，2 號遙控器應該打開哪些按鈕才能點亮花園燈？

A. 按鈕 A

B. 按鈕 B

C. 兩個按鈕都不要打開

D. 兩個按鈕都要打開

正確答案是：D

由於 1 號遙控器的按鈕都關著，表示 1 號遙控器發送出開的訊號；因此，為了開啟花園燈，2 號遙控器必須發送出關的訊號。而只有 2 號遙控器的 2 個按鈕都打開時，2 號遙控器才會發送出關的訊號，故正確答案是選項 D；其餘 3 個選項都會使 2 號遙控器發送出開的訊號。

資訊科學上的意義

NAND 運算 Not AND operation （或稱作 NAND operation）是邏輯運算 logical operation 中的一種運算子。邏輯運算透過運算子，對布林值（真 true 或假 false）進行運算；常見的運算子包括且 AND、或 OR、非 NOT 等。而 NAND 運算就是結合了 NOT 和 AND 的運算：

P NAND Q = NOT (P AND Q)

NAND 運算的真值表如下所示：

P	Q	P AND Q	P NAND Q
假	假	假	真
真	假	假	真
假	真	假	真
真	真	真	假

如果將本任務中的開視為真、關視為假，就可以看出任務中的遙控器或天線的運作方式就是 NAND 運算。資訊科學家透過程式解決問題時，經常運用到邏輯運算；此外，許多電器中使用的邏輯電路，也是邏輯運算應用之一。

關鍵字

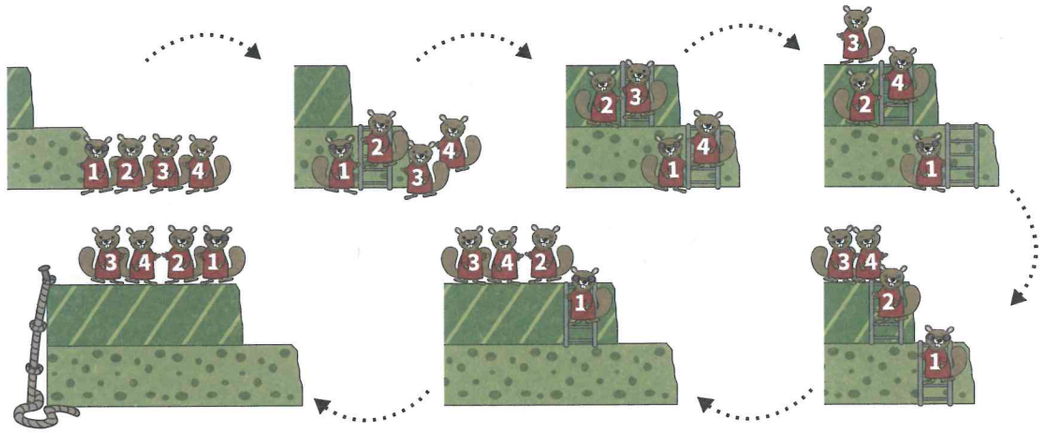
NAND 運算

5. 爬山訓練 - 題組一

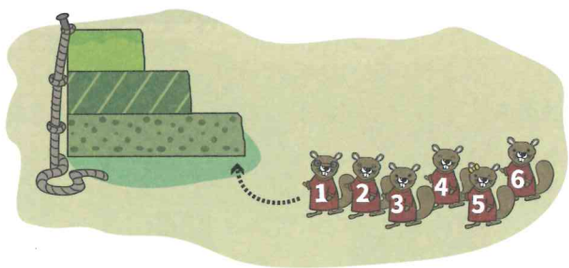
海狸們在協力登山時，每隻海狸都會帶著自己的梯子，排成一排依序行進。

每當需要向上爬的時候，在最前面的海狸便會扶著自己的梯子，讓其他海狸依序向上爬；等沒扶著梯子的海狸都爬上山頂後，扶梯子的海狸才會一同用自己的梯子依序爬到山頂，重新加入隊伍的最尾端繼續前行。要離開山頂時，他們會依相同的順序用繩子爬下山。

以下圖為例，4 隻海狸依 1234 的順序出發後，1 號先扶著牠的梯子，接著 2 號爬上山後也扶著牠的梯子，讓 3 號及 4 號一路爬到山頂。最後 2 號和 1 號再同時往上爬，加入山頂的其他海狸。因此，最後海狸的順序是 3421。



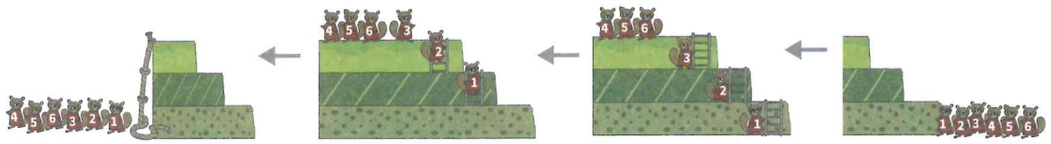
6 隻海狸（1~6 號）依 123456 的順序前往下圖的山丘登山，請問幾號海狸會最後一個下山？



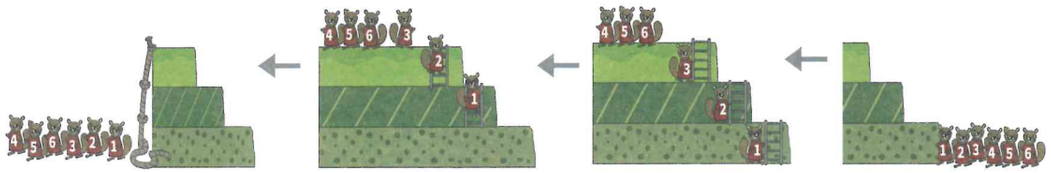
題組一的正確答案是：1

海狸們出發時的順序是 1, 2, 3, 4, 5, 6，而他們的目標山丘有 3 層。

在爬第 1 層時，由 1 號海狸扶梯子讓 2, 3, 4, 5, 6 號依序向上爬。

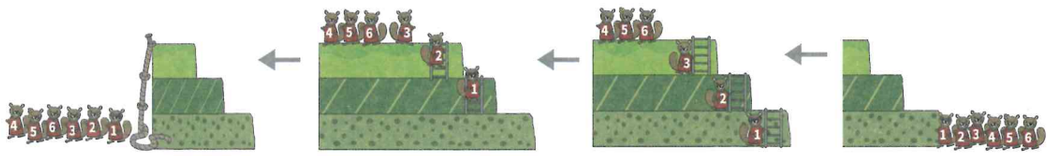


在爬第 2 層時，由 2 號扶梯子讓 3, 4, 5, 6 號依序向上爬。

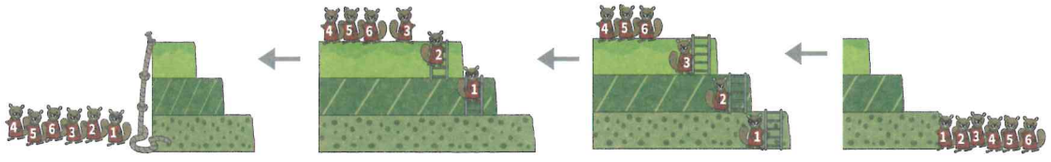


在爬第 3 層時，由 3 號在扶梯子，讓 4, 5, 6 號依序爬到山頂。

等 4, 5, 6 號都抵達山頂之後，3 號、2 號、1 號再依序爬到山頂，加入隊伍的最尾端。



因此，海狸們抵達山頂的順序是 4, 5, 6, 3, 2, 1。而因為海狸會依抵達山頂相同的順序用繩子爬下山，所以最後一個下山的是 1 號海狸。

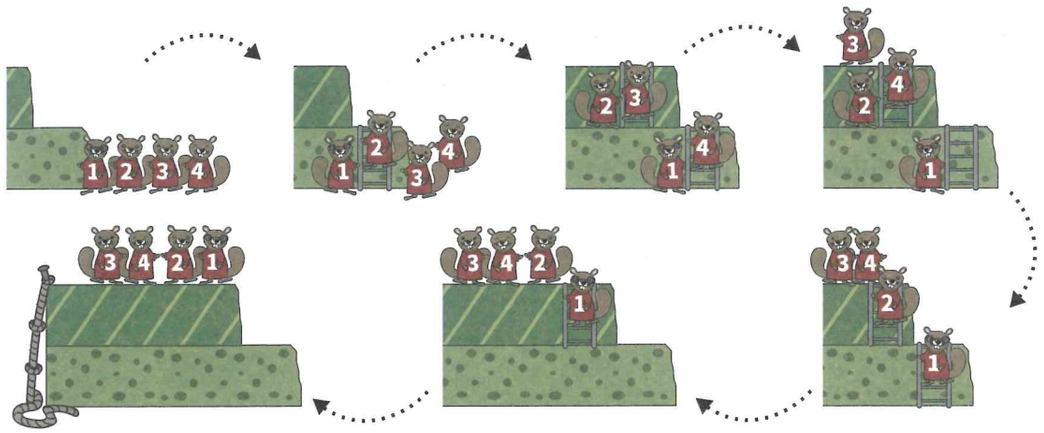


6. 爬山訓練 - 題組二

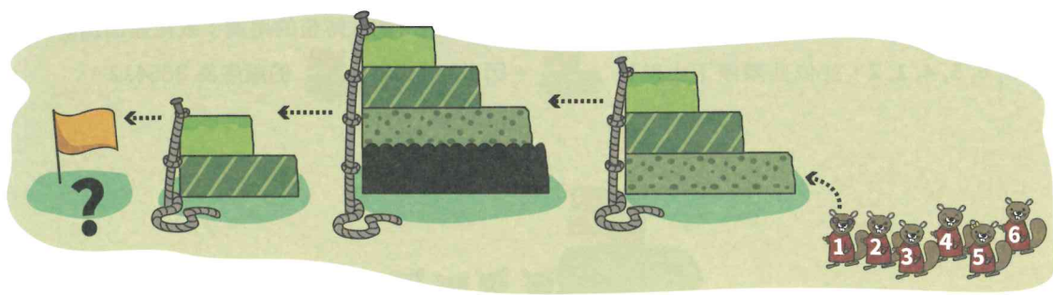
海狸們在協力登山時，每隻海狸都會帶著自己的梯子，排成一排依序行進。

每當需要向上爬的時候，在最前面的海狸便會扶著自己的梯子，讓其他海狸依序向上爬；等沒扶著梯子的海狸都爬上山頂後，扶梯子的海狸才會一同用自己的梯子依序爬到山頂，重新加入隊伍的最尾端繼續前行。要離開山頂時，他們會依相同的順序用繩子爬下山。

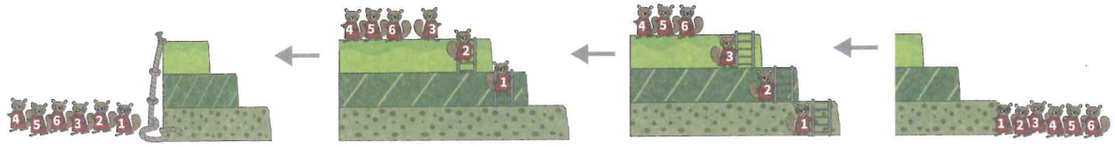
以下圖為例，4 隻海狸依 1234 的順序出發後，1 號先扶著牠的梯子，接著 2 號爬上山後也扶著牠的梯子，讓 3 號及 4 號一路爬到山頂。最後 2 號和 1 號再同時往上爬，加入山頂的其他海狸。因此，最後海狸的順序是 3421。



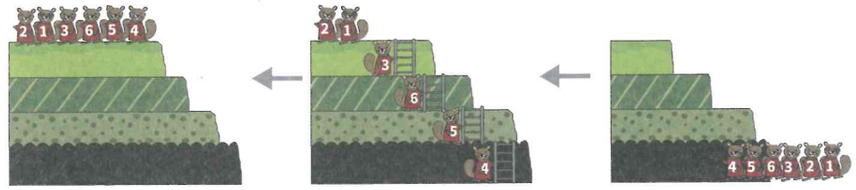
6 隻海狸（1~6 號）依 123456 的順序出發，並沿著箭頭的順序登 3 座山，請問他們抵達的順序為何？（請排列 6 位數字，代表 6 隻海狸的順序（如：123456））



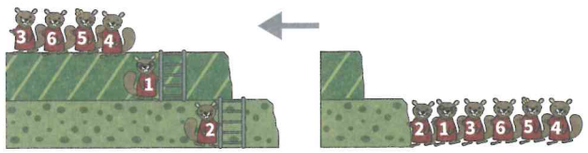
題組二的正確答案是：365412



上圖由右至左分別表示海狸們登第 1 座山的過程，下山後海狸隊伍的順序是 **456321**（詳細過程如題組一），並依此順序前往第 2 座山。



如上圖，第 2 座山共 4 層，故隊伍的前 4 隻海狸（4 號、5 號、6 號、3 號）依序扶著梯子，讓 2 號及 1 號依序登上山頂。之後 3 號、6 號、5 號、4 號再依序爬到山頂，加入隊伍的尾端；故抵達山頂的順序是 **2, 1, 3, 6, 5, 4**，並依此順序下山前往第 3 座山。



最後如上圖，第 3 座山共 2 層，故隊伍的前 2 集海狸（2 號、1 號）依序扶著梯子，讓 3 號、6 號、5 號及 4 號依序登上山頂。之後 1 號及 2 號再依序爬到山頂，加入隊伍的尾端；故抵達山頂的順序是 **3, 6, 5, 4, 1, 2**，並依此順序下山前往  。因此，抵達  的順序為 **365412**。

資訊科學上的意義

堆疊 stack 與 **佇列 queue** 是資訊科學領域中常見的資料結構，它們分別具有不同的特性：

- 如同疊盤子只能從上方放入或取出一樣，在堆疊中的資料只能從一側新增或移除；因此，資料進出堆疊時必定依循 後進先出 **last in first out (LIFO)** 的特性。任務中負責扶梯子的海狸們要等後方的海狸都爬山上之後，才可以爬自己的梯子到山頂，所以最先開始扶梯子的海狸會最後爬到山頂；如同堆疊中的資料一樣有後進先出的特性。
- 像是結帳的隊伍一樣，在佇列中的資料會從一側的入口進入，並依序從另一側的出口取出；因此，資料進出佇列時必定依循 先進先出 **first in first out (FIFO)** 的特性。任務中不需扶梯子的海狸只要隊伍順序一路前行上山頂，如同佇列中的資料一樣有先進先出的特性。

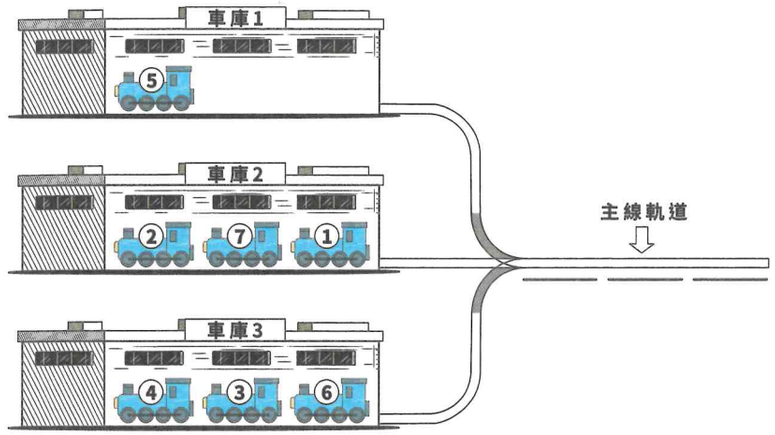
堆疊和佇列是很常運用的資料結構，例如應用堆疊儲存機器人探索迷宮的路徑，後進先出的特性有利於取出最後探索的地方，讓機器人原路返回到之前走過的地方；而佇列常應用在資訊的傳輸，例如印表機中待列印的資料就被儲存在佇列中，待最先送印的資料列印完成後，才會列印下一筆送印的資料。

關鍵字

佇列、堆疊

7. 列車

小狸有 7 輛玩具火車（編號從 1 到 7）和 3 個玩具火車停車庫，目前火車停放的位置如下圖所示：



小狸可以進行以下操作：

- **OUT(X)**：表示將車庫 X 右側的第一輛火車移到主線軌道上，成為主線軌道左側的第一輛火車。
- **IN(X)**：表示將主線軌道左側的第一輛火車移進車庫 X，成為車庫 X 右側的第一輛火車。
- **主線軌道上或車庫中的火車數量都不能超過 3 輛。**

下列哪一組選項的操作可以將火車編號 1、2、3 從左至右依序停放在車庫 1 中？

A. **OUT(1) → OUT(2) → IN(1) → OUT(2) → OUT(2) → IN(1) → OUT(3)**

B. **OUT(1) → OUT(2) → IN(1) → OUT(2) → OUT(2) → IN(1) → OUT(3) → IN(2) → OUT(3) → IN(1)**

C. **OUT(1) → OUT(2) → IN(1) → OUT(2) → IN(1) → OUT(3) → IN(2) → OUT(3) → IN(1)**

D. **OUT(1) → OUT(2) → IN(1) → OUT(2) → OUT(2) → IN(1) → OUT(3) → OUT(3) → IN(1)**

正確答案是：B

依選項 B 所列步驟執行，以下是每個車庫和主線軌道上火車（從左至右）的狀態：

步驟	車庫 1	車庫 2	車庫 3	主線軌道
	5	2 7 1	4 3 6	
OUT(1)		2 7 1	4 3 6	5
OUT(2)		2 7	4 3 6	1 5
IN(1)	1	2 7	4 3 6	5
OUT(2)	1	2	4 3 6	7 5
OUT(2)	1		4 3 6	2 7 5
IN(1)	1 2		4 3 6	7 5
OUT(3)	1 2		4 3	6 7 5
IN(2)	1 2	6	4 3	7 5
OUT(3)	1 2	6	4	3 7 5
IN(1)	1 2 3	6	4	7 5

最後結果達成任務要求「火車編號 1、2、3 從左至右依序停放在車庫 1」。

選項 A 不正確，因為執行結果為車庫 1 停放火車編號 1、2。

選項 C 不正確，因為執行結果為車庫 1 停放火車編號 1、7、3。

選項 D 不正確，因為在 OUT(1) → OUT(2) → IN(1) → OUT(2) → OUT(2) → IN(1) → OUT(3) → OUT(3) → IN(1) 的操作中，第八步驟主線軌道上的火車數量就會超過 3 輛。

資訊科學上的意義

堆疊 stack 是資訊科學領域中常見的資料結構。如同疊盤子只能從上方放入或取出一樣，在堆疊中的資料只能從一側 **新增 push** 或 **移除 pop**；因此，資料進出堆疊時必定依循 後進先出 **last in first out (LIFO)** 的特性。

任務中車庫只有右側一個進出口，主線軌道也只有左側一個進出口，因此火車進出車庫和主線軌道的順序就像在堆疊中的資料一樣，遵循著後進先出的特性。移動玩具火車或是是否達成容納上限而無法再放入玩具火車，也是解題時重要的考量與限制。

OUT(X) 的操作相當於把堆疊 X（車庫 X）的最後一筆資料（右側第一輛火車）從堆疊中移除，並將該資料新增至另一個堆疊（主線軌道）中。相反的，**IN(X)** 就相當於把堆疊中的最後一筆資料移除，並將該資料新增至堆疊 X。

堆疊是在資訊科學領域中很常運用的資料結構，例如應用堆疊儲存機器人探索迷宮的路徑，後進先出的特性有利於取出最後探索的地方，讓機器人原路返回到之前走過的地方。

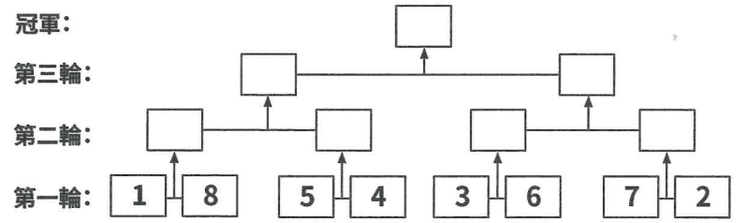
關鍵字

堆疊

8. 淘汰賽

有八支球隊編號 1 至 8 號參加單淘汰賽。每一輪比賽的勝者都會晉級到下一輪，而輸掉比賽的球隊則被淘汰。賽程表及比賽規則如下：

1. 一開始，所有球隊（以方塊中的編號表示）顯示在底部一行。
2. 兩個相鄰方塊之間的水平線表示這兩支球隊為一組要進行比賽。例如，第 1 隊對第 8 隊。
3. 然後在每組比賽球隊的上方空格中寫入勝者的編號。這樣重複進行，直到最後的冠軍產生。



比賽結束後，四隻海狸（Andy、Beth、Cece 和 Daniel）觀看了比賽，並記錄了每支球隊的編號在最終賽程表中出現的次數，如下表。

	Andy	Beth	Cece	Daniel
第 1 隊	1	2	1	1
第 2 隊	4	1	2	1
第 3 隊	3	1	3	1
第 4 隊	1	1	1	1
第 5 隊	2	3	2	3
第 6 隊	1	4	4	2
第 7 隊	2	2	1	2
第 8 隊	1	1	2	4

啊～糟糕！只有其中一隻海狸的記錄是正確的！

請問哪隻海狸的記錄是正確的？

A. Andy

B. Beth

C. Cece

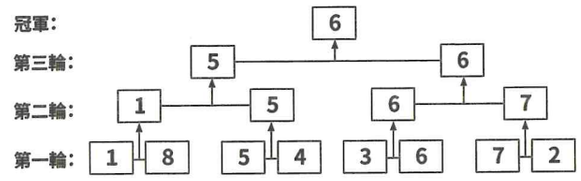
D. Daniel

正確答案是：B

經過檢查初始的賽程表及比賽規則，我們可以排除一些選項中的不可能情況。

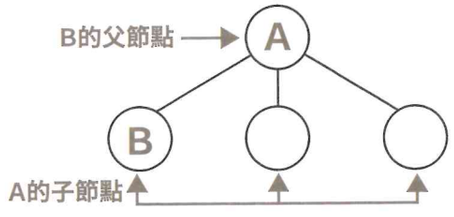
- **選項 (A) 是錯誤的。**根據 Andy 的記錄 ，因為第一輪第 1 隊和第 8 隊比賽，其中一隊應該晉級到下一輪至少會有二次出賽，但它們的出賽次數記錄都是 1；而且，第一輪中第 2 隊和第 7 隊比賽後會有勝隊晉級，所以應該有其中一隊的出賽次數是 1，但卻是 4 和 2。
- **選項 (C) 是錯誤的。**根據 Cece 的記錄，因為第一輪比賽後應該有 4 隊淘汰，也就是應該有 4 隊的出賽次數是 1，但出賽次數記錄為 1 的隊伍只有 3 隊；而且，第二輪應該有 2 支隊伍被淘汰，但記錄中卻有 3 隊的出賽次數都是 2。
- **選項 (D) 是錯誤的。**根據 Daniel 的記錄，第一輪比賽的獲勝是第 5 隊、第 6 隊、第 7 隊和第 8 隊，而第 6 隊和第 7 隊各只進行了 2 場比賽，這代表他們都輸掉了第二輪的比賽，但在第二輪中他們兩隊是對陣比賽，其中一隊必須晉級到下一輪，出賽次數記錄不該都是 2。
- **選項 (B) 是正確的。**根據 Beth 的記錄顯示第 1 隊、第 5 隊、第 6 隊和第 7 隊在第一輪中獲勝；第 5 隊和第 6 隊在第二輪獲勝；第 6 隊最終獲得到冠軍。

右圖是根據 Beth 記錄產生的最終賽程表：



資訊科學上的意義

資訊科學領域中許多種 **資料結構 data structure**，泛指不同種儲存、組織資料的方式：每種資料結構都透過特定的方式來表示各筆資料之間的關聯性。常見的資料結構有：推疊、佇列、樹…等等。本任務中的賽程表除了紀錄了參與每一輪競賽的隊伍之外，還包含了對戰、晉級、淘汰等關聯性。帶有階層性質的資料經常運用 **樹 tree** 的結構來表示；其中每個 **節點 node** 代表一個項目，節點之間的連結稱為 **分支 branch**。如下圖，上端的節點稱作 **父節點 parent node**，與之相連的節點稱作它的 **子節點 child node**；而最上層的父節點又稱作 **根節點 root**，最下層的子節點又稱作 **葉節點 leaf**。



電腦在儲存資料夾及檔案時，就是應用到樹的結構：磁碟機可視作根節點，存在磁碟機中的所有資料夾及檔案，都是該磁碟機的子節點。而上述資料夾除了是磁碟機的子節點，更是資料夾中其他資料夾及檔案的父節點，所有資料夾裡面都可以再存放檔案（子節點），一層層的往下延伸。而所有的檔案，因為沒有向下延伸的檔案，所以都是葉節點。

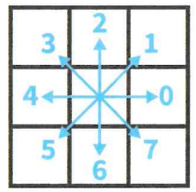
關鍵字

資料結構

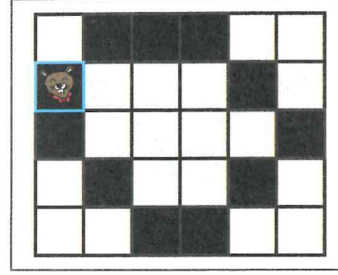
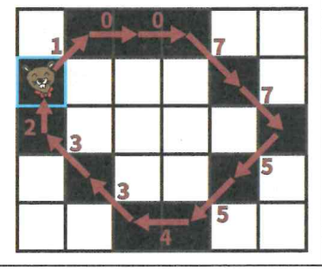
9. 環狀密碼

地圖上的黑色方塊表示海狸村築巢的位置。小狸使用一種「環狀密碼」來記錄這些繞成一圈的築巢位置。

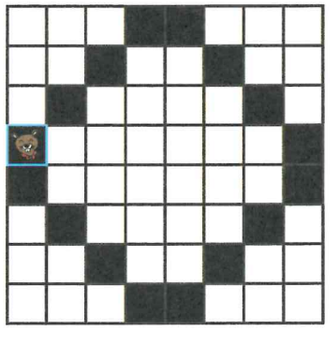
小狸從他家所在的方塊開始，順時鐘依序記錄移動到下個方塊的方向，直到回到他家。每個移動方向的代碼如下圖所示：



範例：

海狸村地圖	順時針方向移動一圈	環狀密碼
		1, 0, 0, 7, 7, 5, 5, 4, 3, 3, 2

海狸村進行擴建，新的築巢位置如下圖，請問哪個選項是小狸新記錄的「環狀密碼」？



A. 1, 1, 0, 0, 7, 7, 6, 6, 5, 5, 4, 4, 3, 3, 2, 2

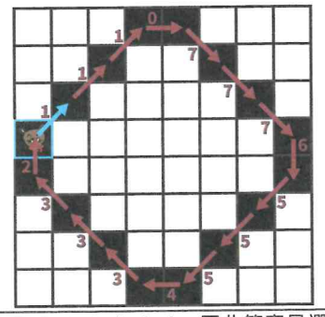
B. 6, 7, 7, 7, 0, 1, 1, 1, 2, 3, 3, 3, 4, 5, 5, 5

C. 1, 1, 1, 0, 7, 7, 7, 6, 5, 5, 5, 4, 3, 3, 3, 2

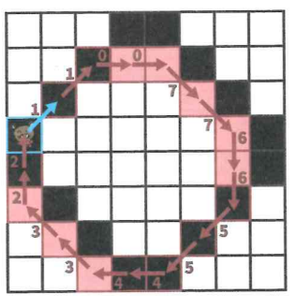
D. 0, 1, 1, 0, 7, 7, 7, 6, 5, 5, 5, 4, 3, 3, 3, 0

正確答案是：C

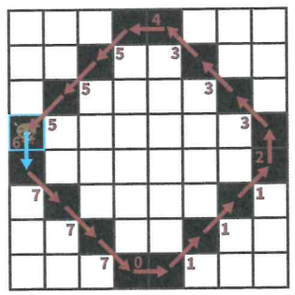
依照「環狀密碼」記錄規則，從小狸家方塊開始，順時鐘依序記錄移動到下個方塊的方向，直到回到他家。如下所示：



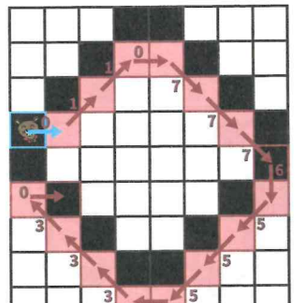
應該得到 1, 1, 1, 0, 7, 7, 7, 6, 5, 5, 5, 4, 3, 3, 3, 2，因此答案是選項 C。



選項 A 的記錄密碼移動方向如下圖，對應到黑色方塊以外區域，因此不是答案。



選項 B 則以逆時針方向記錄方向，因此錯誤。



選項 D 在第一個跟最後一個紀錄的方向錯誤，對應到黑色方塊以外區域。

資訊科學上的意義

在資訊科學中，有許多不同的 **影像壓縮 image compression** 方法，目的是節省影像儲存或傳送的資料量。**鍊碼 chain code** 是一種以輪廓為基礎的影像壓縮法。如同本任務中的規則，從起始**像素 pixel**，即任務中的方塊，沿著輪廓紀錄移動到下一個像素的向量。 鍊碼的優點是它是一種不會損失資訊的壓縮方式——**無損壓縮 lossless compression**，可還原原圖，不會降低影像的品質。

有的壓縮法會改變原始影像的部分資訊，例如去除部分人眼無法辨識的色彩資訊；調整影像的亮度、彩度等，使影像與原圖不完全相同。在日常生活中，我們在傳輸影像或發佈到社群媒體時，會感覺影像沒有原始檔案畫質清晰，那是因為影像經過壓縮後再傳送，雖然犧牲一些品質，卻可以提高儲存與傳輸效率。

關鍵字

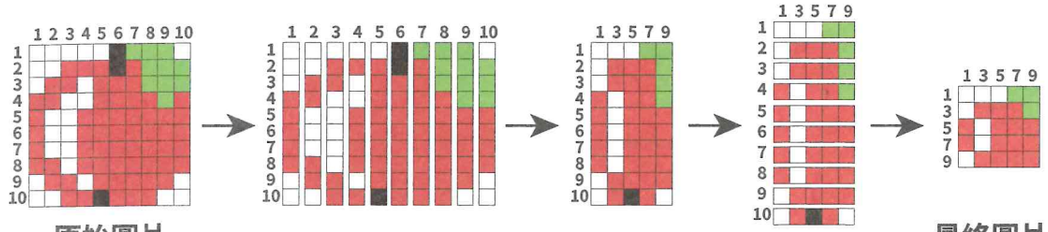
影像壓縮、鍊碼

10. 海狸壓縮法

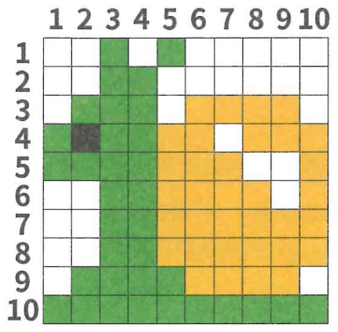
海狸使用一種特殊技術來壓縮圖片。

首先，將原始圖片垂直分成 10 個大小相等的行，留下奇數行，組合起來得到一張新的圖片。

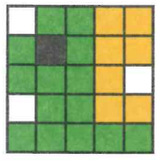
接著，再將新圖片水平分成 10 個大小相等的列，留下奇數列，組合起來得到最終的壓縮圖片。



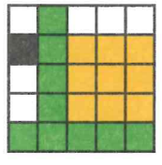
若使用海狸的技術來壓縮下面這張圖片，會得到什麼結果？



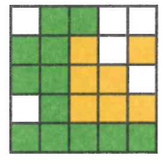
A.



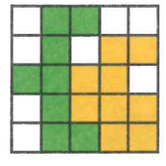
B.



C.

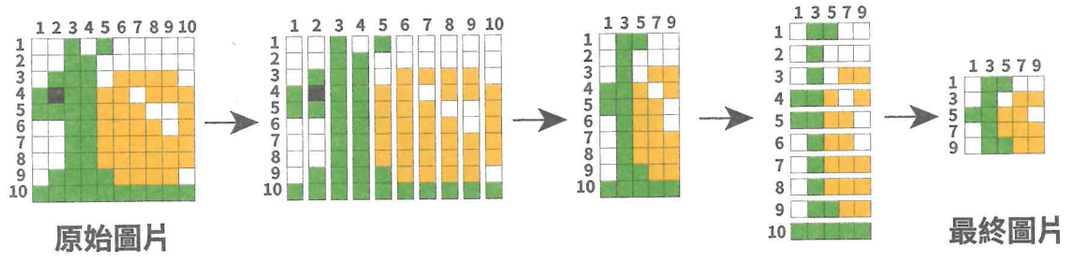


D.



正確答案是：D

如下圖所示，根據任務描述的壓縮技術，先留下原始圖片中的所有奇數行，刪去所有偶數行；之後再刪去所有偶數列，剩下的奇數列就是最終的圖片，也就是選項 D 的圖片。



資訊科學上的意義

資料壓縮 data compression 是透過減少資料中的次要資訊，使它們減少儲存空間或縮短傳輸時間。對於文章、圖像、影像或聲音等不同類型的資料，都有不同方式可以進行壓縮。

本任務是對影像資料進行壓縮，此壓縮方式是一種 **有損壓縮 lossy compression** ，即壓縮後影像的品質會下降，並且無法復原。

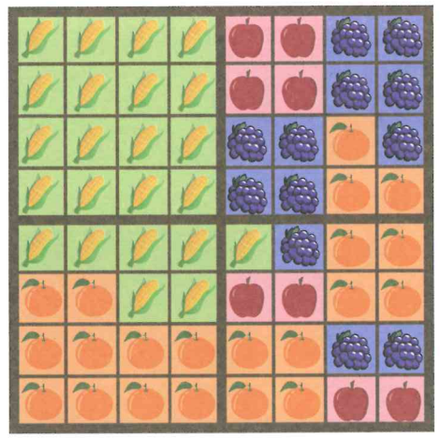
生活中我們也可以運用資料壓縮的觀念，例如同學的學號有很多碼，但是同班同學的學號大多前幾碼都相同，如果要寫多個班上同學的學號，共同的前幾碼只要第一個同學寫完整，後面同班的同學就只要寫後面不同的幾碼，就可以少記一些數字。

關鍵字

資料壓縮、有損壓縮

11. 灑水器

阿達計畫在他 8x8 大小的農場安裝灑水器來灌溉農作物，農場裡種植了四種農作物：蘋果、玉米、葡萄和橘子，各種農作物種植的區域如下圖所示。



附近商店出售三種灑水器，灌溉範圍分別是 4×4、2×2 和 1×1 大小的土地。

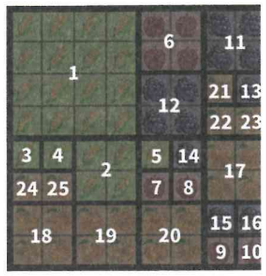
為了讓農作物達到最佳生長，一個灑水器只能灌溉同一種農作物，且每個區域的農作物只能被一個灑水器灌溉到。

請問阿達最少需要買幾個灑水器來灌溉農場裡的所有農作物？

正確答案是：25

安排灑水器時必須確保：

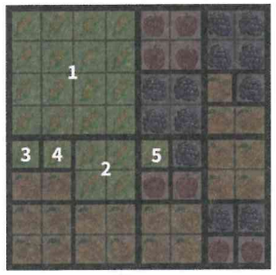
1. 同一個灑水器灌溉範圍內皆為相同類型的農作物
2. 任 2 個灑水器灌溉的範圍沒有重疊



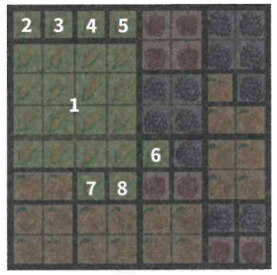
為了盡可能減少灑水器的數量，同一種農作物的種植範圍，應該優先採用灌溉範圍 4×4 的灑水器，未能涵蓋到的範圍再依序考慮採用灌溉範圍 2×2 和 1×1 土地的灑水器。左圖展示其中一種可能的灌溉安排方法。

以玉米種植範圍為例，採用一個灌溉範圍 4×4 的灑水器 (編號 1)，先沿著邊界涵蓋左上角 4×4 的範圍，剩下的範圍可用 1 個灌溉範圍 2×2 的灑水器 (編號 2)，以及涵蓋 3 個灌溉範圍 1×1 的灑水器 (編號 3、4、5)，只需要 5 個灑水器。如下圖一。

若將灌溉範圍 4×4 的灑水器，涵蓋從第二排開始的 4×4 範圍，則剩上面一排 1×4 的、下面一排 1×2、及右邊 1×1 的種植範圍，共需要用 8 個灌溉範圍 1×1 的灑水器。如下圖二。



圖一



圖二

因此玉米種植範圍採用目前這種安排，所需灑水器數目 5 個為最少。以相同方式檢查，可發現蘋果、葡萄、橘子種植範圍最少所需灑水器數目分別為 5、6、及 9。因此總共最少需要 25 個灑水器。

資訊科學上的意義

4 分樹 quad tree 是一種透過將二維空間資料垂直與水平中分，劃分為 4 個象限面積相等的區域來表示空間範圍的方法。對於一個地圖涵蓋的範圍，重複採用 4 分樹將範圍均分為四個區塊，直到一個區塊對應到相同的地理區域，如同本任務中一個區塊中皆種植同一種水果。

4 分樹可以將圖像資料的每個像素儲存為一些區塊的集合，從而減少儲存量，並提高運算速度。

4 分樹也可以用來建構地圖中不同目標地點的索引，用來有效率空間資料。也就是如本任務，將大範圍地圖重複切分為 4 塊較小的區域，直到區域中涵蓋的地點數很少 (在本任務中是同一種水果)，記下這個地點每次在切分過程中被分到第幾象限。當想搜尋某個區域有哪些地點，就可以透過指定區域範圍跟 4 分樹的比對，比較快搜尋出答案。

關鍵字

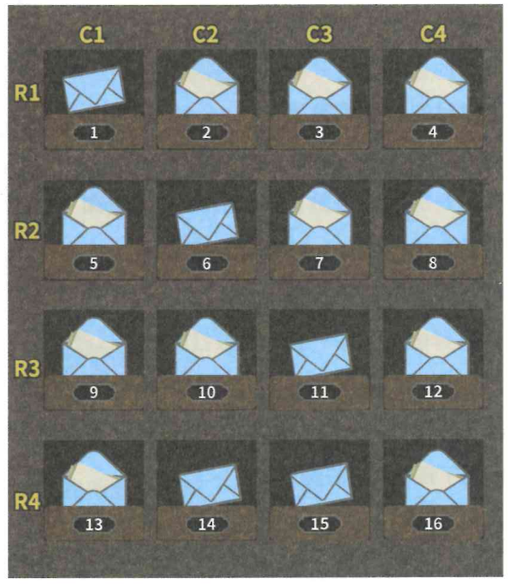
4 分樹

12. 密封信件

海狸國將 16 封機密信件（編號 1 至 16）分別放在櫃子中。

其中有 10 封已拆開 、6 封未拆開 。

某夜，敵國間諜潛入並拆開其中一封信件，第二天警衛發現已拆開的信變成 11 封，如下圖所示：



警衛不記得原本 10 封已拆開的信放在哪，只記得以下幾點：

- C2 與 C4 兩行合計，拆開的信應該是偶數封
- C3 與 C4 兩行合計，拆開的信應該是偶數封
- R2 與 R4 兩列合計，拆開的信應該是偶數封
- R3 與 R4 兩列合計，拆開的信應該是偶數封

請問敵國間諜拆開編號幾的信件？

A. 編號 5

B. 編號 9

C. 編號 10

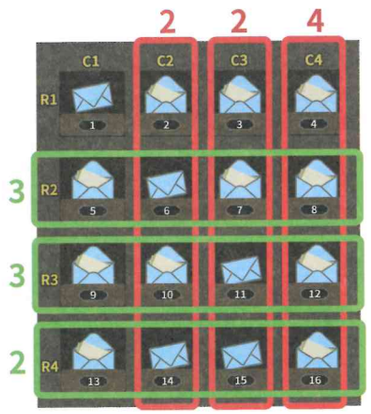
D. 編號 13

E. 資訊不足，無法判斷

正確答案是：D

首先，間諜只打開一封信，因此那封信所在的那一行或列中，拆開信件的總數一定會由偶數變為奇數，或由奇數變為偶數。間諜潛入過後，櫃子中的信件狀態如下圖所示，其中 C2、C3 中分別有 2 封信被拆開，而 C4 中有 4 封信被拆開，以此類推。

只要遂一檢驗現在拆開信件的數量，和警衛記憶中的數量有何不同，就可以找出間諜拆開的信件放在哪裡。



- C2 與 C4 中所有已開封的信件總數是 2+4=6 封，與警衛記憶相符，表示這兩行的信件都沒有被間諜打開。
- C3 與 C4 中所有已開封的信件總數是 2+4=6 封，與警衛記憶相符，表示這兩行的信件都沒有被間諜打開。
- R2 與 R4 中所有已開封的信件總數是 3+2=5 封，與警衛記憶不相符，表示這兩列信件中有一封被間諜打開。
- R3 與 R4 中所有已開封的信件總數是 3+2=5 封，與警衛記憶不相符，表示這兩列信件中有一封被間諜打開。

由於只有一封信被打開，且 R2 與 R4 中有一封信件被拆開，而 R3 與 R4 中有一封信件被打開，表示被打開的信一定在 R4。此外，C2、C3 與 C4 中所有信件都沒有被打開，表示這封被拆開的信一定在 C1。因此，間諜拆開了編號 13 的信件。

資訊科學上的意義

本任務運用了**錯誤檢測 / 錯誤更正**的概念，在資訊科學中，資料是由一連串的 1 和 0 組成。在網路傳輸的過程中，可能會遇到一些雜訊或遺失而使資料錯誤；有些儲存裝置例如 DVD ，也可能因為有刮痕而造成儲存資料的部分損壞。因此，設計一種能夠檢測有損壞發生（錯誤檢測），並能夠校正損壞位元的方法是很重要的。

本任務中警衛所記憶的就相當於一種檢查碼，信封的開與關可以對應為 1 與 0，藉由檢查 1 與 0 的數量可以找出錯誤的位元 (被間諜拆開的信封)。

生活中身分證字號的最後一碼是檢查碼，是由身分證字號的其他碼代入一個數學公式算出來的，這樣就可以很容易的檢查出是否有不小心輸入錯誤的狀況，也就是一種錯誤檢測的方法。

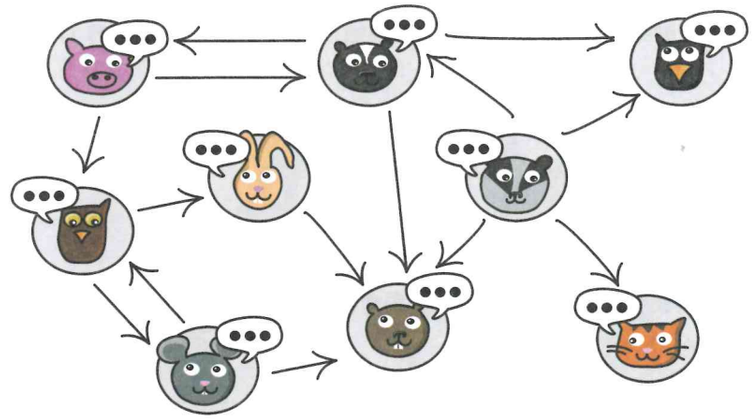
關鍵字

錯誤校正碼、錯誤檢測

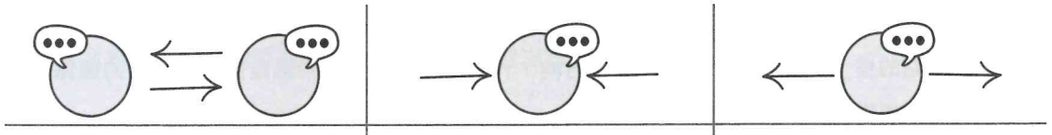
13. 狸書追蹤

「海狸書」是一個提供動物村上網發佈訊息的服務。

動物間追蹤訊息的關係可用下圖表示。



動物間的互動，可透過觀察這個關係圖來了解。舉例來說：



表示兩隻動物彼此是朋友，會互相追蹤。	表示這隻動物可能很有名，都是別的動物單向追蹤牠。	表示這隻動物是新手，還沒有其他動物追蹤牠。

請問，依據上面動物間追蹤訊息的關係圖，以下哪個敘述<u>錯誤</u>？

A.  與  **是朋友。**

B.  **可能很有名。**

C.  **可能是新手。**

D.  與  **都只有兩隻動物追蹤牠。**

正確答案是：D

根據關係圖

- 選項 A 的敘述正確。
   與  互相追蹤，彼此是朋友的關係。
- 選項 B 的敘述正確。
   有 4 隻動物追蹤牠（有四個箭頭指向牠），但牠並沒有追蹤牠們，所以牠可能很有名。
- 選項 C 的敘述正確。
   沒有任何動物追蹤牠，這表示牠可能是新手。
- 選項 D 的敘述錯誤。
   有兩隻動物追蹤牠（有兩個箭頭指向牠）。但是  只有一隻動物追蹤牠（有一個箭頭指向牠）。

資訊科學上的意義

圖 graph 是一種具有節點和邊的資料結構，當一個圖的邊有特定的方向，就被稱作 **有向圖 directed graphs**。此任務中的圖用來表示動物間的社交關係，每隻動物用一個節點代表，動物追蹤的情況對應到連接節點的邊；比較特別的是邊的箭頭方向，代表由其中一隻動物追蹤另一隻動物的關係，因此這個追蹤圖相當於一個有向圖。

當某個節點有一些連接邊，箭頭是指向此節點，這樣的連接邊數稱為這個節點的 **入度數 (in-degree)** ；當節點有一些連接邊，箭頭是指向其他節點時，這樣的連接邊數稱為這個節點的 **外度數 (out-degree)** 。在此任務中，當節點的入度數較大時，這個節點表示的動物可能很有名；而外度數較大時，這個節點表示的動物可能是新手。

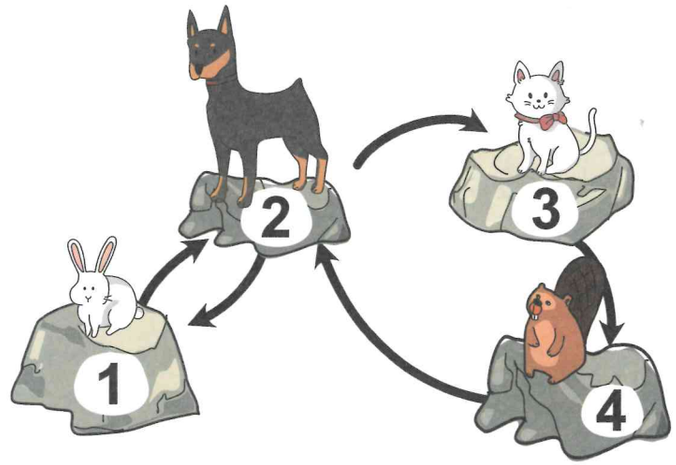
在日常生活中，捷運地圖或公車路線圖就形成圖，每個捷運站 / 公車站都是一個節點，而可換車到其他路線的節點 (站)，就會有較多的連接邊。而社群網站上的朋友關係也可用圖來表示，連接較多邊的節點，通常代表人緣較好或較具有影響力的人，若運用有向圖，則可以更清楚呈現朋友間互動「方向」的資訊。

關鍵字

有向圖、輸入度、外度數

14. 岩石上的派對

兔子、小狗、小貓和海狸分別生活在 1 號、2 號、3 號和 4 號岩石上，而且牠們只會依照下圖中箭頭的方向在岩石之間跳躍移動。例如：動物在 2 號岩石上時，只能選擇往 1 號或 3 號跳。



有一天動物們決定一起運動完之後，在 4 號岩石上開派對；牠們約定從一塊岩石跳到另一塊算跳躍一次，且每隻動物必須跳躍相同次數後停在 4 號岩石上。

這些動物最少須跳躍幾次，才能在 4 號岩石上開派對？(範圍為 [1~20] 的整數。)

正確答案是：6

分別觀察從不同起點開始跳躍至 4 號岩的可能次數：

- 從 3 號岩開始（小貓）可能跳躍：1、4、6、…次
- 從 2 號岩開始（小狗）可能跳躍：2、4、5、6、…次
- 從 1 號岩或 4 號岩開始（兔子或海狸）必定會跳到 2 號岩，故可能跳躍次數即如從 2 號岩開始多 1 次：3、5、6、7、…次

綜合以上的分析，所有動物最小的共同跳躍次數即是 6 次。

進一步找出每隻動物跳躍 6 次抵達 4 號岩的路徑為：

- 兔子：1 → 2 → 3 → 4 → 2 → 3 → 4
- 小狗：2 → 1 → 2 → 1 → 2 → 3 → 4
- 小貓：3 → 4 → 2 → 1 → 2 → 3 → 4
- 海狸：4 → 2 → 3 → 4 → 2 → 3 → 4

故每隻動物最少要跳躍 6 次，才能同在 4 號岩上開派隊。

資訊科學上的意義

圖 graph 是一種具有節點和邊的資料結構，當一個圖的邊有特定的方向，就被稱作 **有向圖 directed graphs**。此任務中的每個岩石相當於一個節點，可以跳躍移動的方向對應到連接節點的邊，形成一個有向圖。

在生活中，如果把公車站視為節點，而兩個車站間有公車到達，就可以畫出一個邊。因為公車路線有的去程跟回程經過的站是不同的，所以邊是有方向的。多個公車路線就可形成一個有向圖。

路徑 就像你從家到學校搭公車的路線，中間經過一些車站，也可能換車，最終到達學校鄰近的車站，這就是你到學校的搭公車路徑。**迴圈** 是一種特別的路徑，就像有些公車的路線，中途可能經過不同的車站，但終點又回到跟起點是同一個車站，這樣的路徑就稱為迴圈。**路徑長度**：就像你從家附近公車站上車後，到學校鄰近車站共經過多少個車站。任務中要觀察動物所在岩石到 4 號岩石的路徑長度，以及中間用迴圈繞回某個岩石，對不同動物找出一個相同長度的路徑跳到 4 號岩石。

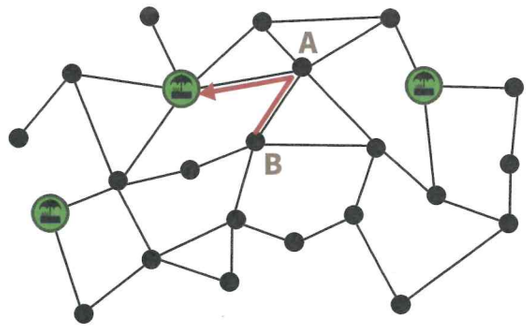
運用有向圖可以清楚呈現以「方向」為重要資訊的資料，例如城市中有單行道的道路地圖、飛機航線圖、公車路線圖等。

關鍵字

有向圖

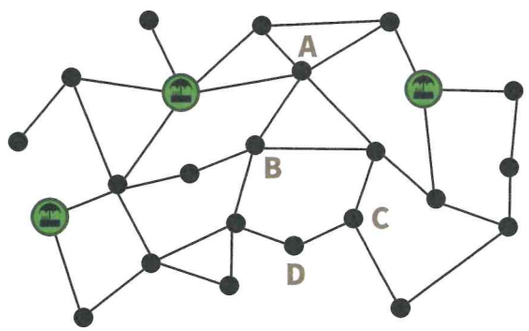
15. 果汁車

海狸市的某些交叉路口設有果汁車；下圖是海狸市的街道圖，黑色圓圈 表示交叉路口，綠色圓圈 表示該交叉路口有果汁車，而連接圓圈的線表示街道。



市政府準備新增 1 台果汁車，好讓市民從任何 1 個交叉路口出發，**最多**只要經過 1 個交叉路口，就能遇到果汁車。例如：從上圖的 交叉路口 B 出發，只要經過 交叉路口 A 就遇到一台果汁車。

請問新的果汁車應該設置在下圖中哪 1 個叉路口，才能達成市政府的目標？



A. 交叉路口 A

B. 交叉路口 B

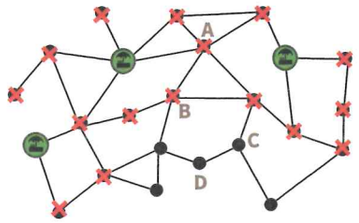
C. 交叉路口 C

D. 交叉路口 D

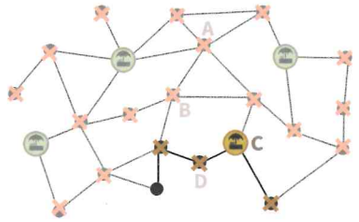
正確答案是：D

市政府的目標是要讓市民從任何 1 個叉路口出發，最多只要經過 1 個叉路口，就能遇到果汁車；因此，若要達成目標，每個交叉路口必須與果汁車最多相隔 2 條街。

如右圖，先在海狸市街道圖上，排除與現有果汁車相隔 2 條街道以下的交叉路口後（排除了交叉路口 A 及 B），剩下 5 個交叉路口。

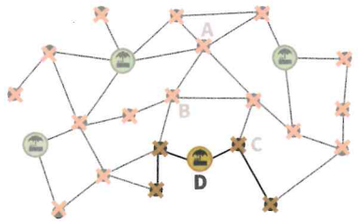


剩下的 5 個交叉路口中，交叉路口 D 位於正中間；若如右圖將果汁車設於該路口，則剩下的 5 個交叉路口皆於果汁車最多相隔 2 條街。



若如右圖將果汁車放在交叉路口 C，則會有 1 個交叉路口與最近的果汁車相隔 3 個交叉路口。

由此可知，應將新增的果汁車設置在交叉路口 D。



資訊科學上的意義

一個圖 **graph** 由**節點 node**（通常以點表示）以及連接節點的**邊 edge**（通常以線段表示）所組成；圖 可用於表示物件間的關係，若兩個節點間有邊相連，就表示這兩個節點對應的物件間存在某種關聯。在一個圖中選擇一些節點，使其餘節點皆與所選的某個節點有邊連接，則所選的這些節點稱作這個圖的一個**支配集 dominating set**；其中節點數量最少的支配集稱作 **最小支配集 minimum dominating set (MDS)**。

在此任務中的交叉路口可視為節點，交叉路口間連接的道路即是邊，要尋找果汁車設置位置，接近於尋找支配集，不過必須改成其他節點到選取節點的路徑長度限制在 2 以內。

許多現實生活中的問題以圖表示後，就能透過程式實作一個演算法，有系統的找出某個任務的解答。例如：找出在哪些位置架設基地台，能以最少的基地台將網路訊號達到最大化利用。在規劃公共設施時，也可運用圖來尋找適當的建設位置，例如規劃醫院地點時，會考慮讓城市內不同區域的民眾都能在一定時間內抵達醫院。

關鍵字

最小支配集

16. 小艾的任務

以下是小艾家的村莊地圖，地圖上的數字代表各路徑行走所需的時間（以分鐘為單位）。

單位:分鐘

有一天，小艾需要完成以下任務：

- 到五金行買開罐器。
- 到藥房買藥品。
- 到市場買水果。

這些任務的執行不限順序，行走路徑可以重覆，每個任務在各個店鋪完成採買需要 1 分鐘。小艾從她家出發，在完成所有任務後直接回家。

舉例來說，如果小艾按照以下路線走，她需要 41 分鐘：

小艾的家 → 市場（水果）→ 麵包店 → 五金行（開罐器）→ 公園 → 藥房（藥品）→ 學校 → 小艾的家

（說明：4+6+3+9+7+3+6 為 38 分鐘，再加上 3 分鐘在店鋪完成採買，共需要 41 分鐘。）

請問小艾從她家出發，在完成所有任務後直接回家，所需要的最短時間為何？

A. 37 分鐘

B. 38 分鐘

C. 39 分鐘

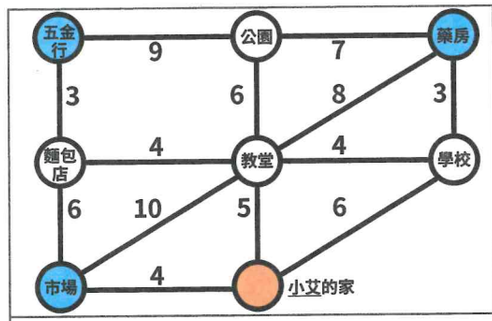
D. 40 分鐘

正確答案是：C

最短時間的路線是：

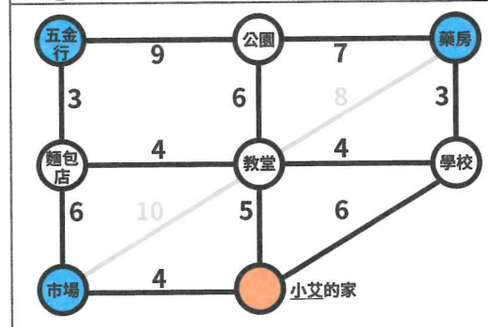
小艾的家 → 學校 → 藥房 → 學校 → 教堂 → 麵包店 → 五金行 → 麵包店 → 市場 → 小艾的家

她需要 6 + 3 + 3 + 4 + 4 + 3 + 3 + 6 + 4 = 36 分鐘 + 3 分鐘的任務時間 = 39 分鐘。她也可以反向走這個路線，都是需要相同時間。

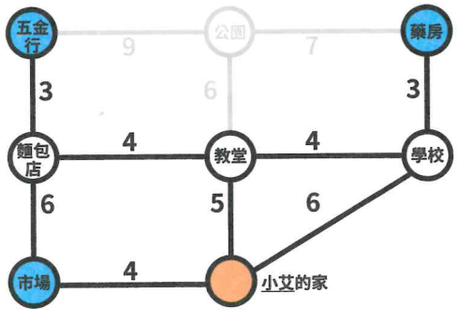


為了確認是否沒有更短的路徑，我們使用簡化的地圖（左圖）。

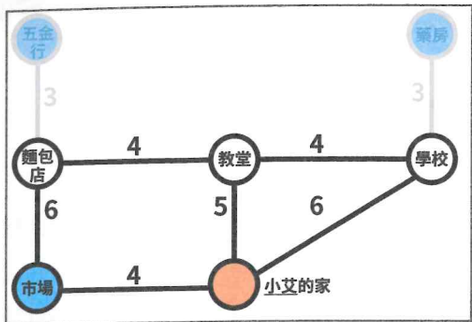
圓圈（以下稱為節點）代表村莊中不同的地方，而線（以下稱為邊）代表它們之間的路徑。如同在地圖上一樣，每條邊旁的數字表示沿著這條路徑行走所需的時間。



我們可以忽略灰色的邊（如左圖）。因為可以透過其他的節點找到更快的路徑。

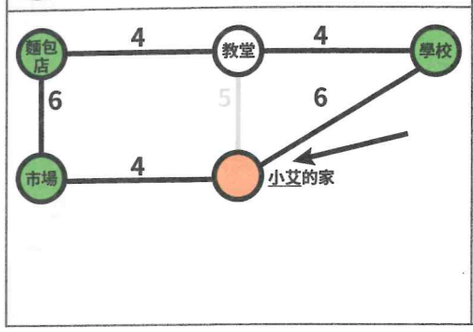


我們還可以忽略「公園」節點。小艾在公園沒有任何任務，而且對於每一條通過公園的路徑，都有更快的替代路徑。



在左圖中，小艾必須拜訪上面的兩個節點（五金行和藥房），都必須通過其下方的節點，每一段路徑都需要 3 分鐘，來回各需要 6 分鐘。

我們先移除這些節點，而且最後結果記得要將此必須花費的 12 分鐘加進去。



最後得到左圖。我們需要從下方右側的節點（有箭頭指向）開始並結束，並拜訪所有三個有顏色的節點。最短路徑是通過所有五個節點，但略過灰色的邊。

這個路徑需要 6 + 4 + 4 +6 + 4 = 24 分鐘。再加上上一個步驟需加上的 12 分鐘，以及在每個拜訪地方完成任務所需的 3 分鐘，最後得到的總時間是 39 分鐘。

資訊科學上的意義

在資訊科學中，評估路徑選擇的問題，常用 **圖 graph** 來表示各點間互通的情形。圖通常由**節點 node** 和若干連接這些節點的**邊 edge** 所組成，可以用來表示物件與物件之間的關係，邊也可以加上權重值 weight，用來表示兩個節點間的「距離」或是「成本」。

在此任務中，給定每一個邊的權重值（兩商店間的步行時間），目標是找到最快通過所有指定店鋪並回到家中的迴路 curcuit；相當於經過所有指定節點的 **最短路徑問題 shortest path problem**（或說是最短迴路問題 shortest circuit problem）。

資訊科學中 **旅行推銷員問題 traveling Salesman Problem** 就是這類的問題，旅行推銷員問題是指：給定一系列城市和城市之間的距離，嘗試找出走過每一座城市一次並回到起始城市的最短迴路。

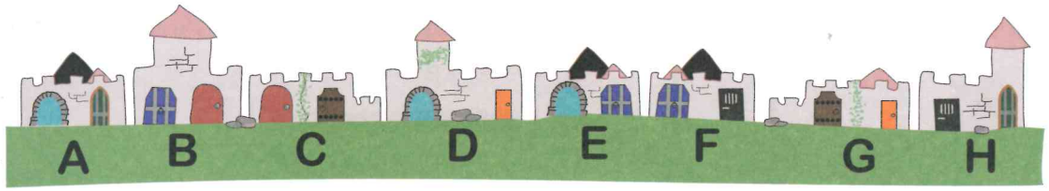
現實生活中也常會需要規劃最短迴路的狀況，例如垃圾車清運路線或郵差送信路線等。

關鍵字

圖、最短路徑問題

17. 奇幻城堡

奇幻城堡裡有 8 個房間和 7 種不同顏色的門，每個房間有 2 個不同顏色的門。一開始都在房間裡，如果向外打開一個門，可以瞬間移動到另一個具有相同顏色門的房間裡，例如：在房間 A 裡打開藍色的門，可以瞬移到有藍色門的房間 D 或 E 裡。

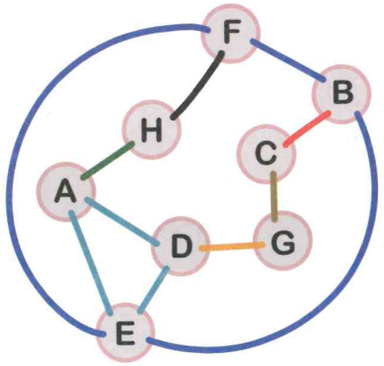


請問你要在房間 A 裡移動到房間 C 裡，在經過最少房間的情況下，需要打開多少個門？（範圍為 [1~8] 的整數。）

正確答案是：3

畫一個圖，把房間當作節點，用邊連接具有相同顏色門的房間。

如果從 A 打開其中一個門，你可以進入房間 D、E 或 H 裡。如果再打開第二個門，你可以從 A 進入房間 B 裡（透過房間 E 的深藍色門）、F（透過房間 E 的深藍色門或房間 H 的黑色門）或 G 裡（透過房間 D 的橘色門）。如果你打開房間 B 的紅色門或房間 G 的棕色門，你可以到達房間 C 裡。因此，從房間 A 裡到房間 C 裡，最少通過 3 個門。



從左圖可知，從房間 A 裡移動到房間 C 裡，有下列幾種方式：

1. A → H → F → B → C
2. A → D → G → C
3. A → E → B → C
4. A → E → F → B → C
5. A → E → D → G → C

因此，從房間 A 裡到房間 C 裡，在經過最少房間的情況下，至少需要打開 3 個門。可以由 A → D → G → C，也可以由 A → E → B → C。

資訊科學上的意義

圖 graph 是資訊科學中非常重要的資料結構，通常由**節點 node** 和若干連接這些節點的**邊 edge**（通常以線段表示）所組成，可以用來表示物件與物件之間的關係；在日常生活中，例如人與人之間互動所形成的社交網絡、或是不同地點間的交通網絡，都可以使用圖來表示。

在此任務中，每個房間都是一個節點，連接相同顏色的門就形成了邊，目標是找出從起點 (A 房間) 到目標 (C 房間) 經過最少節點的路徑，也就是 **最短路徑問題 shortest path problem**。

然而在現實生活中，當有成千上萬條路可選擇時，我們可利用電腦技術輔助，例如 GPS 導航系統。系統內具備良好的路徑規劃演算法，用以找到最短路徑，將能事半功倍，為生活帶來莫大的便利。

關鍵字

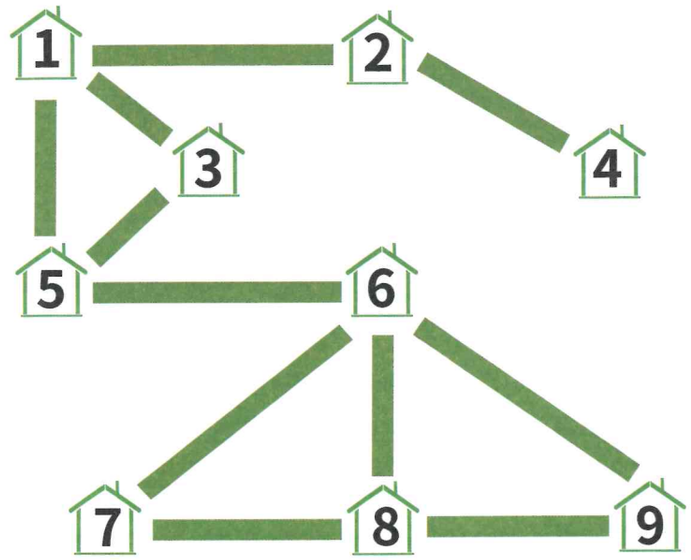
圖、最短路徑

18. 海狸村的彩色門

海狸國進行基礎建設，共建造 9 棟新房，分屬 3 個不同部落。每個部落有 3 戶家庭，每戶家庭可分得一棟房子。

同一部落的房子需使用相同顏色油漆，油漆共有黃、藍、紅三種顏色，街道相鄰的兩棟房子不能使用同色油漆。

下圖標示 9 棟新房的位置及相連的街道。

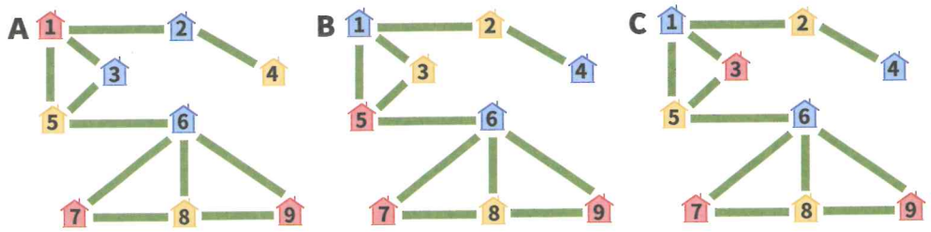


哪一棟房子一定跟 7 號房漆成相同顏色？（範圍 [1~9] 的整數）

正確答案是：9

6 號房和 8 號房相連，所以必須使用不同顏色油漆，而 7 號房和 9 號房都與 6 號房和 8 號房相連，所以 7 號房和 9 號房必須使用剩下的第三種顏色。

現在我們來證明除了 9 號房以外的其他房子與 7 號房不一定會漆成同色。以下考慮三種著色方案 A、B 及 C：



首先，拆成兩個部分，先考慮上半部編號為 1~5 號房的配對組合。在方案 A 中，兩對 {2; 3} 和 {4; 5} 各自使用同色油漆，而在方案 C 中，這兩對也可以各自使用不同色油漆，並不一定有房子須漆成同色的規律存在。

接著，考慮下半部編號為 6~9 號房的配對組合。在 3 種著色方案中，除了 {7; 9} 外，所有的配對組合都使用不同的顏色，6 號和 8 號一定跟 7 號漆成不同顏色。

最後，上下部分各取一棟房子進行配對組合。1~5 號房在方案 B 中的顏色與方案 A 完全不同，而 6~9 號房在方案 A 和 B 中的顏色完全相同。因此，1~5 號房中的任一棟不一定與 7 號房同色。

資訊科學上的意義

資訊科學中的 **圖 graph** 是種具有節點和邊的資料結構，此任務中的房屋代表節點，兩間房屋相連的街道代表邊。任務要求有邊連接的兩個節點不能同色，而討論圖最少需要幾種顏色來對節點著色，這就是著名的 **著色問題 graph coloring problem**。

資訊科學家發現至多只需要四種顏色，就可以讓地圖上相鄰的區塊都著上不同顏色，這稱為四色定理 four color theorem。由於此任務中的圖較為簡單，只要用三種顏色就能做到。

在現實中有許多問題與著色問題相似，例如考試考生座位安排讓同班同學分開坐，或是運動賽程中，避免讓同一隊短時間內有太多賽程等。

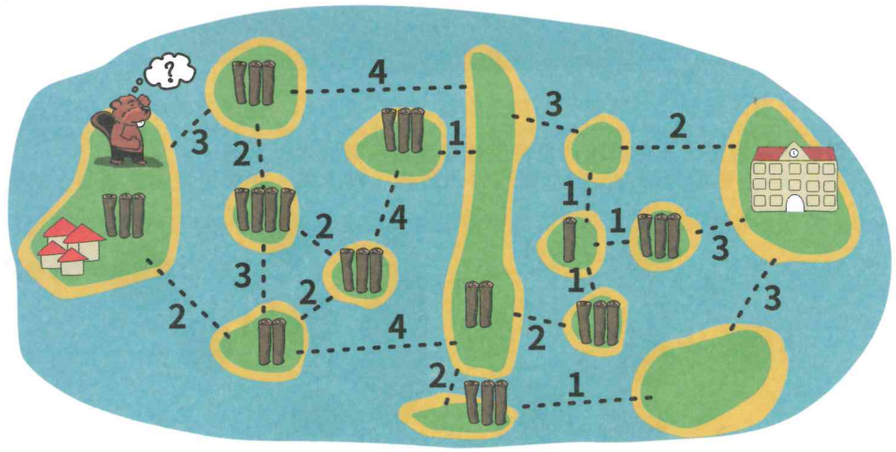
關鍵字

圖、著色問題

19. 搭建橋樑

阿布要幫小鎮建造橋樑，讓小鎮學生可以順利前往學校。

下圖是阿布規劃的地圖，圖中小島上的木材數量代表島上可取得的木材總量，黑色虛線代表每座橋樑建造所需的木材數量。



阿布從小鎮出發時，可以先攜帶三根木材，然後在任何相鄰小島之間過橋後取得其他島上額外的木材，每根木材只能使用一次，但不一定要用完所有木材。由於阿布不會游泳，他必須先建造一座橋樑，以確保能前往下個島嶼。如果木材不夠就無法搭橋，另外建好的橋樑，可重複使用，不用重建。

請問阿布由小鎮到學校要完成建橋工作至少需要多少根木材？

A. 11 根

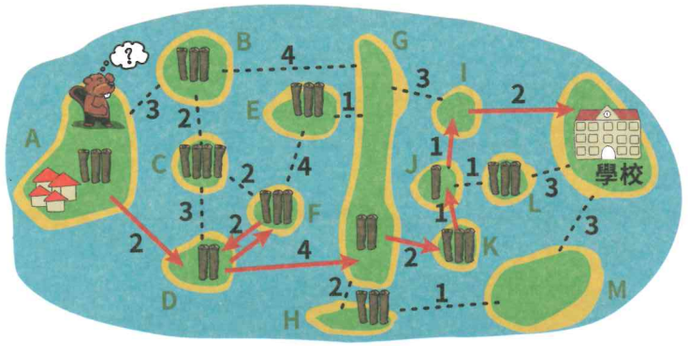
B. 12 根

C. 13 根

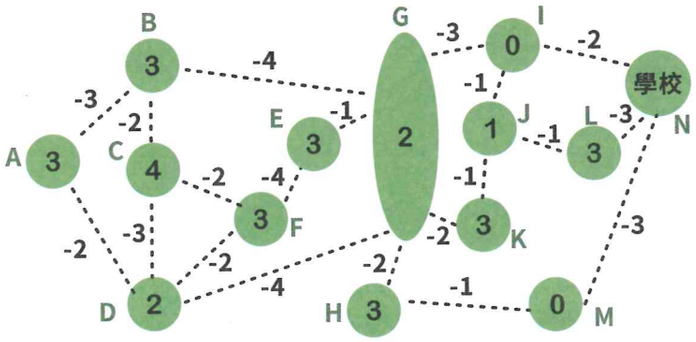
D. 14 根

E. 17 根

正確答案是：D



我們可以先將阿布畫的圖簡化並將每個島加上編號，如下圖所示，圓形代表小島，圓形內的數字代表島上可取得的木材總量，黑色虛線代表每座橋樑建造所需的木材數量。



將每完成一個橋樑建造抵達小島視為一個步驟，我們可以記錄每一個步驟阿布擁有的所有木材總數量和建造橋樑使用的木材總數量，用來評估正確的步驟是否可行（每個步驟累積的總木材數量須≧建橋使用的總木材數量）：

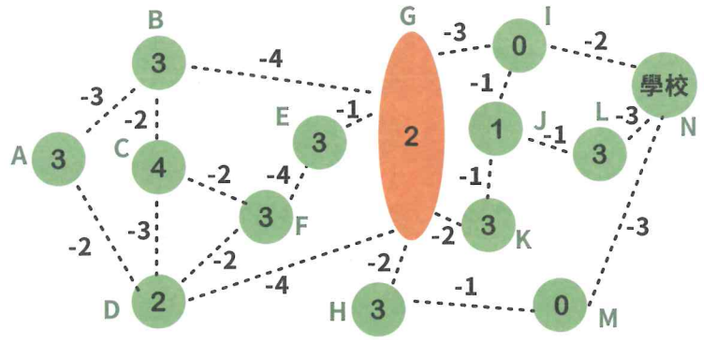
例如：

1. $\overline{AD}$ (3,2) 表示過橋前阿布擁有的木材總數量為 3，建造 $\overline{AD}$ 之間的橋樑須用掉 2 根木材。
2. 若下個步驟是前往 F 島，須紀錄為 $\overline{DF}$ (5,4)，表示抵達 F 島時，已將事先收集的 3+2=5 根木材，用於搭建 $\overline{AD}$、$\overline{DF}$ 兩兩島嶼間的橋樑，共消耗 2+2=4 根木材。
3. 若下個步驟是前往 D 島，則共能收集到 3+2+3=8 根木材，由於 $\overline{FD}$ 連結的橋樑早已建造好，故不需使用剩下的木材，紀錄為 $\overline{FD}$ (8,4)。

以此類推，最佳路徑為：

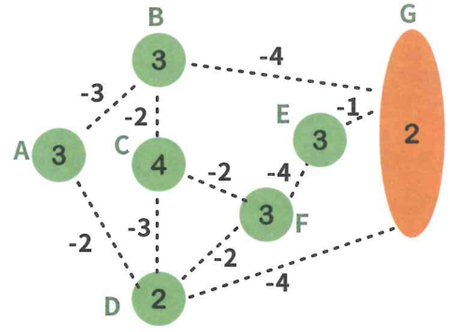
$\overline{AD}$ (3,2) → $\overline{DF}$(5,4) → $\overline{FD}$ (8,4) → $\overline{DG}$ (8,8) → $\overline{GK}$ (10,10) → $\overline{KJ}$ (13,11) → $\overline{JI}$ (14,12) → $\overline{IN}$ (14,14)

我們可以透過驗證是否有其他更好的解法，也就是是否能夠使用少於 14 根木材，若沒有就證明 14 根是最佳解。簡化圖後可以發現，從 A 前往 N 的所有路線都必須經過中間的 G 島，所以我們可以將簡圖以 G 島為準拆分為左側和右側兩個圖。



接著，從 A 島到 G 島（左側）的可行步驟至少需要 6 根木材，從 G 島到學校 N（右側）的可行步驟需要至少 5 根木材（我們稱這些數字為下限，左側下限為 6，右側下限為 5）。

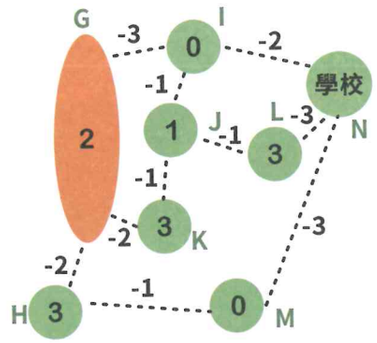
我們先使用右側下限 5 來進行左側部分的驗證，由於我們預測的最佳解至少需要使用 14 根木材完成所有橋樑。基於右側至少需要 5 根木材，這表示我們須找到左側部分使用 ≦ 9 根木材的所有可行步驟（因為右側部分至少須使用 5 根木材才能抵達學校 N，所以若左側使用任何 ≧ 10 的解決方案，則最終須使用至少 15 根木材）。



在左側部分，我們有以下抵達 G 島且使用 ≦ 9 根木材的步驟（以每個步驟累積的總木材數量及建橋使用的總木材數量表示）：

$\overline{AB}$ + $\overline{BG}$ =(6,7)，$\overline{AB}$ + $\overline{BC}$ + $\overline{CB}$ + $\overline{BG}$ =(10,9)，AD+DG=(5,6)，$\overline{AD}$ + $\overline{DC}$ + $\overline{CD}$ + $\overline{DG}$ =(9,9)，$\overline{AD}$ + $\overline{DF}$ + $\overline{FD}$ + $\overline{DG}$ =(8,8)，$\overline{AD}$ + $\overline{DF}$ + $\overline{FE}$ + $\overline{EG}$ =(11,9)。刪除不可行的解決方案（即每個步驟累積的總木材數量 < 建橋使用的總木材數量），剩下的方案如下：$\overline{AB}$ + $\overline{BC}$ + $\overline{CB}$ + $\overline{BG}$ =(10,9)，$\overline{AD}$ + $\overline{DC}$ + $\overline{CD}$ + $\overline{DG}$ =(9,9)，$\overline{AD}$ + $\overline{DF}$ + $\overline{FD}$ + $\overline{DG}$ =(8,8)，$\overline{AD}$ + $\overline{DF}$ + $\overline{FE}$ + $\overline{EG}$ =(11,9)。

以上四個方案中，$\overline{AD}$ + $\overline{DF}$ + $\overline{FD}$ + $\overline{DG}$ =(8,8) 為最佳解。表示左側至少須使用 8 根木材。



我們透過左側使用 8 根木材的下限來解決右側部分，計算所有使用 ≦ 6 根木材的方案如下：$\overline{GI} + \overline{IN}$ =(2,5)，$\overline{GI} + \overline{IJ} + \overline{JI} + \overline{IN}$ =(3,6)，$\overline{GK} + \overline{KJ} + \overline{JI} + \overline{IN}$ =(6,6)，$\overline{GH} + \overline{HM} + \overline{MN}$ =(5,6)。最後，我們將左側和右側的解決方案結合，確認所有可行配對方案，讓左右側總共所須使用的木材數≦ 14，其中只有 $\overline{AD} + \overline{DF} + \overline{FD} + \overline{DG}$ =(8,8) 加 $\overline{GK} + \overline{KJ} + \overline{JI} + \overline{IN}$ =(6,6) → (14,14) 可行，其他配對方案都會造成每個步驟累積的總木材數量 < 建橋使用的總木材數量的情況。

資訊科學上的意義

在資訊科學中評估路徑選擇及所費資源常用 **圖 graph** 來表示。本任務中把每座島表示成**節點 node**，每座潛在的橋（黑色虛線）表示為**邊 edge**，造橋所需的木材量作為**權重值 weight**，並將島上可取得的木材量紀錄到每個節點上，如此一來阿布建橋的路徑就能以建橋順序列表來表示。目標是在資源有限的情況下找到一條路徑可以取得足夠木材到達終點，這個任務結合了路徑演算法及**限制最佳化 constraint optimization**。

在此任務中，可以利用一般路徑演算法來找到路徑，但隨著圖的規模變大，所需複雜度也會增加，因此常會利用演算法來降低複雜度，例如**分而治之法 divide and conquer**。限制最佳化也常被應用在資源的分配或謎題，例如八皇后問題或數獨等。

關鍵字

圖、限制最佳化問題

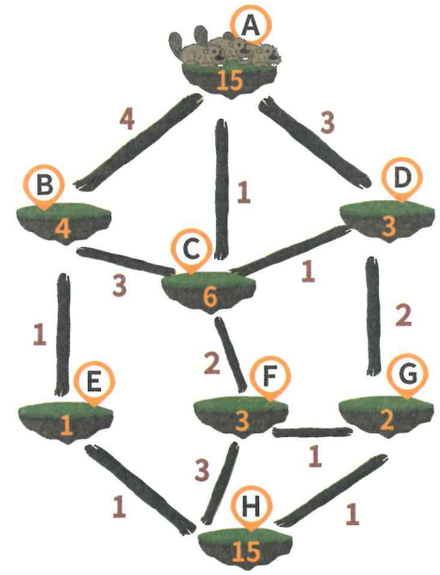
20. 跳島遊戲

15 隻海狸在玩跳島遊戲，需要從 A 島前往 H 島。

遊戲地圖如下所示：島上的數字代表每個島上可以同時存在的海狸數量上限；島跟島之間有原木連結，原木上的數字代表可以同時通過它的海狸數量上限。

每回合每隻海狸都要決定是否移動；若要移動，每隻海狸一次只能通過一塊原木；若不移動就留在原本的島上。當所有海狸都決定後，才能開始下一回合。

例如，在第一回合結束時，A 島上的 4 隻海狸可以走到 B 島，1 隻走到 C 島，3 隻走到 D 島，剩下的 7 隻則留在 A 島。等大家都到達位置後， 才能再繼續第二回合，直到 15 隻海狸都到達 H 島為止，遊戲就結束。



在第五回合結束時，最多有幾隻海狸可以到達 H 島？

A. 11

B. 12

C. 13

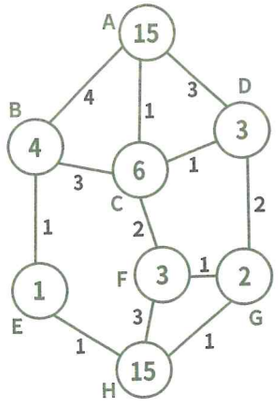
D. 14

E. 15

正確答案是：C

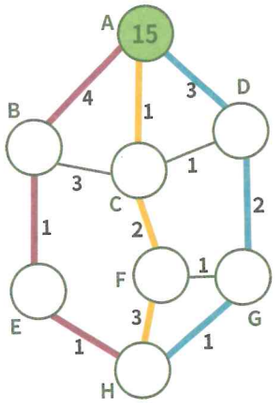
我們以圖來協助解答此問題，其中綠色節點表示該島上的海狸數量，白色節點表示空島。由於所有原木的承載數量都等於或小於它們連接島嶼的容納海狸數量，所以我們可以忽略島上的數字。

旅程開始時如右圖所示，我們可以看到有海狸到達H島至少需要三個回合。只有以下三條路徑可行：



- 如果海狸遵循左邊的路徑（紅色），第一回合有四隻海狸移動到B島上，其中一隻在第二回合時移動到E島上，再在第三回合到達H島。
- 如果海狸遵循中間的路徑（黃色），第一回合有一隻海狸移動到C島上，第二回合這隻海狸移動到F島上，再在第三回合到達H島。
- 如果海狸遵循右邊的路徑（藍色），有三隻海狸移動到D島上，其中一隻海狸移動到G島上，再在第三回合到達H島。

因此，在第三回合結束時，最多只能有 3 隻海狸到達H島。

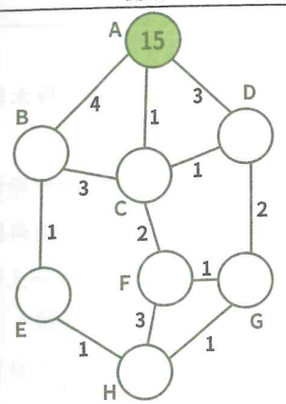
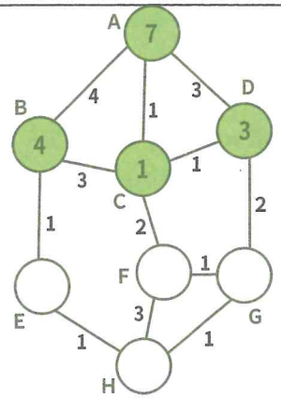
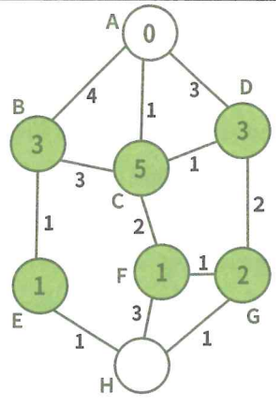
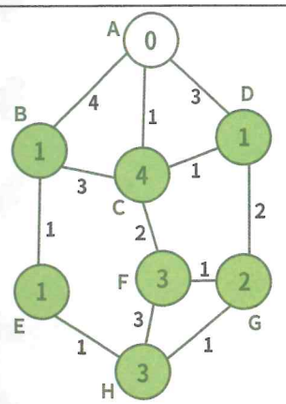
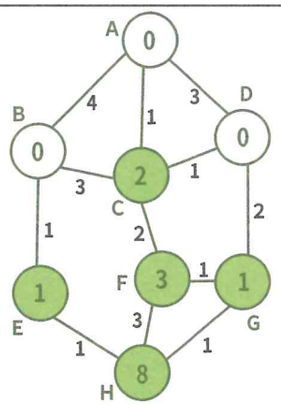
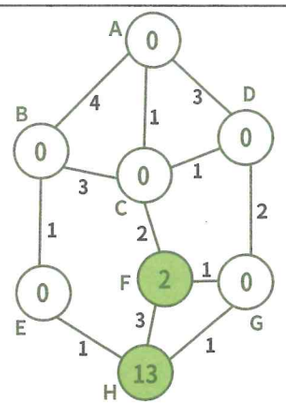


由於連接到H島的原木每次可以運送最多 1+3+1=5 隻海狸，因此在第五回合結束時，H島上的最多海狸數量必須小於或等於 13（即 3+5+5）。因此，可以排除 D）和 E）的選項。

但剩下的選項中哪個是最大值？

其實只要找到在第四回合及第五回合開始前，E 島一定有 1 隻海狸以上，F 島一定有 3 隻海狸以上，G 島一定有 1 隻海狸以上的可能，就可以證明最多可以有 13 隻海狸在第五回合結束時到達H島。

如接下來的表格所示，我們可以清楚看到每回合結束後，島上的海狸數量，及每回合海狸如何移動，最後發現最多可以有 13 隻海狸在第五回合結束時到達H島。需要注意的是因為只能從 E、F、G 三島移動到H島，但 F-->H 可以同時有三隻海狸移動，因此策略是盡量讓更多的海狸移動到 F 島，但也要保持 E 島及 G 島也有海狸可以移動：

開始	第一回合	第二回合
		
	A→B :4 A→ C:1 A→ D:3 (A:15-4-1-3=7)	A→B :3 A→ C:1 A→ D:3(A:7-3-1-3=0) B→ E:1 (E:1) B→ C:3(B:4+3-1-3=3) D→ C:1 D→ G:2(D:3+3-1-2=3) C→ F:1(C:1+1+3+1-1=5) (G:2;F:1)
第三回合	**第四回合**	**第五回合**
		
B→ E:1 B→ C:1(B:3-1-1=1) D→ G:2(D:3-2=1) C→ F:2(C:5+1-2=4) E→ H:1(E:1+1-1=1) G→ H:1 G→ F:1(G:2+2-1-1=2) F→ H:1(H:1+1+1=3) (F:1+2+1-1=3)	B→ E:1(B:1-1=0) E→ H:1(E:1+1-1=1) D→ G:1(D:1-1=0) G→ H:1 G→ F:1(G:2-1-1=1) C→ F:2(C:4-2=2) F→ H:3(H:3+1+1+3=8) (F:3+1+2-3=3)	C→ F:2(C:2-2=0) E→ H:1(E:1-1=0) G→ H:1(G:1-1=0) F→ H:3(H:8+1+1+3=13) (F:3+2-3=2)

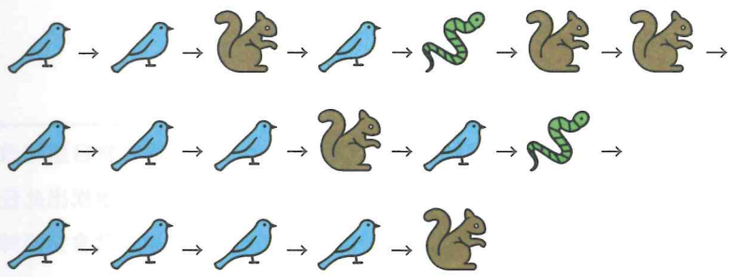
21. 神奇樹

小狸家附近有一棵神奇樹。
每當一隻鳥 停到樹上再飛走，樹上就會長出 2 顆果實。
若樹上有果實，每當一隻松鼠 爬到樹上再離開，就會掉下 1 顆果實。

每當有一條蛇 爬到樹上再爬走，樹上的果實就會全部掉落！

而且每隻動物都會等神奇樹上沒有其他動物時才會到樹上。

有一天小狸來到神奇樹，發現上面結了 25 顆果實。牠一直坐在樹下，依序紀錄每隻到過神奇樹上的小鳥、松鼠和蛇。



請問當小狸離開神奇樹時，樹上有多少顆果實？

A. 3

B. 7

C. 17

D. 31

資訊科學上的意義

我們可以把島之間的關係表示成**圖 graph**，也就是把每座島表示成**節點 node**，島之間的原木表示為**邊 edge**，原木的承載限制就是**權重值 weight**，可視作以 A 島為起點，H 島為終點的圖。
本任務為 **最大流量 max flow** 問題，可將每個邊同時間通過的海狸數目稱為其流量，但要注意流量不能超過每個邊的容量限制。這個任務的目標是要讓流量最佳化，使得在指定的時間內，能有最多的海狸到達終點。
最大流量問題可應用於對道路系統建立交通模型，評估可容納的最大車流量並找出交通瓶頸。

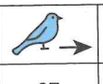
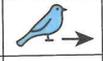
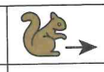
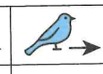
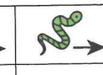
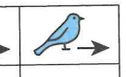
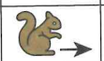
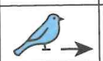
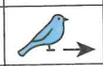
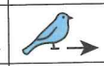
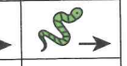
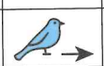
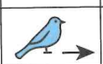
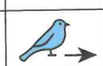
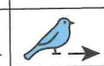
關鍵字

圖、最大流量問題

正確答案是：B

當小狸離開神奇樹時，樹上有 7 顆果實。

若一步一步追蹤樹上果實數的變化，則如下表所示：

出現動物	一開始						
果實數量	25	27	29	28	30	0	0
出現動物							
果實數量	0	2	4	6	5	7	0
出現動物							
果實數量	2	4	6	8	7		

但是可觀察到，每當有蛇爬到樹上時，所有的果實都會立刻消失，也就是果實的數目會變成 0，所以我們可以省略最後一條蛇爬到樹上前發生的改變。

在最後一條蛇出現後，接下來有四隻鳥停到樹上，這表示樹上會長出：2 顆 × 4 = 8 顆 果實。最後又有一隻松鼠爬上樹幹，使一顆果實掉下，最後剩下：8 顆 − 1 顆 = 7 顆果實。

因此，正確答案為選項 (B)。

資訊科學上的意義

這個任務主要需要觀察如何由給定條件 (何種動物來到神奇樹)，根據目前樹上的果實數目進行改變。若對果實數目直接執行固定值 (0) 的設定，則跟先前的果實數目無關。因此要有效地找出此任務的答案，只需要查看出現最後一條蛇後出現的動物。因為每當蛇出現時，樹上的果實就會全部掉落，數量直接設為 0，先前樹上的果實剩下多少個都不會影響最後結果。

任務中的樹上果實數目就像程式中的**變數 variable**，它們就像是一記錄器，可以儲存各種數字和資訊，可以記錄需要計算的東西。發生不同的狀況，可用不同的指令來更改變數所記錄的內容。

想像一下你在玩線上遊戲時，有個計數器可顯示有多少生命值可挑戰，這個值可以用一個變數存起來。當你獲得一個寶藏就會增加生命值，未通過障礙，計數的生命值會減少。中途直接按重玩一次，就會設定成固定的生命值。

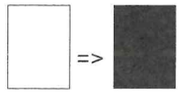
另外，電視遙控器可以按向前或向後，就會根據目前正在看的頻道換到前一個或下一個頻道，也可以直接設定頻道編號直接切換，記錄目前正在看的頻道也是一個變數。

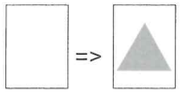
關鍵字

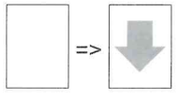
變數、條件語句

22. 印刷機

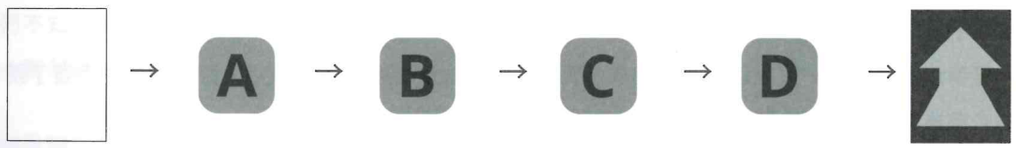
小狸有一台機器能將紙旋轉 180 度，還有三台印刷機器，功能分別如下：

整張塗黑，完全覆蓋原有圖案： 

印製不透明灰色三角形： 

印製不透明灰色向下箭頭： 

將這四台機器各安排到 ABCD 其中一個順序位置後，可依序接續運作，將一張白紙印出  。



請問將紙旋轉 180 度的機器應該安排在哪個位置？

A. 

B. 

C. 

D. 

正確答案是：C

想印製出  ，機器的的順序安排考慮如下：

- 整張塗黑的機器不能放在最後的位置 D ，因為會得到一整張都塗黑的結果。
- 將紙旋轉 180 度的機器不能放在位置 A ，因為一開始就將白紙旋轉後仍是一張白紙，而隨後印製的灰色向下箭頭方向及灰色三角形，在未進行旋轉的情況下，無法疊合印出  中的灰色圖案。
- 把紙整張塗黑的機器必須排在其他兩個印製機器運作前，否則其他機器所印製成的圖案都會被黑色蓋掉。
- 要疊合印出  中的灰色圖案，必須先在紙上印製向下箭頭後，將紙旋轉 180 度，再印製灰色三角形；或是，先在紙上印製灰色三角形，將紙旋轉 180 度，再印製向下箭頭。

根據上述分析，我們可推知印製的順序是先將紙塗黑（位置 A）、印向下箭頭（位置 B）、旋轉 180 度（位置 C）、最後印三角形（位置 D）；或是先將紙塗黑（位置 A）、印三角形（位置 B）、旋轉 180 度（位置 C）、最後印向下箭頭（位置 D）。

上述兩種安排順序，將紙旋轉 180 度的機器皆要放安排在位置 C。所以答案是選項 C。

資訊科學上的意義

在這個任務中，機器 A、B、C 和 D 都各有特定的印製功能，以不同的順序組合在一起時，可能會呈現不同的印製結果。

要得到這個任務中一張黑紙上有兩個灰色圖案，取決於對機器的安排順序。因此，學生需要了解不同的機器執行順序，會如何影響在紙上產生的圖案。

排列機器就像撰寫程式一樣。為了讓一個程式能正確解決一個任務，程式中有些指令會因為安排的先後順序不同，得到不同的執行結果，因此必須安排正確的順序。

演算法中的 **循序結構 sequential structure** 就是一步一步告訴電腦或機器應該做什麼，就像我們需要將這些機器按照正確的順序排列一樣。

日常生活中，例如在組裝玩具時，需要按照組裝說明書中的步驟進行，就是一種循序指令。如果你在安裝馬達前安裝車體和輪子，那麼你可能會遇到馬達裝不進去的問題。在進行教室地板清潔時，也通常有個順序，例如可能要先掃地，然後才拖地。如果顛倒兩個順序，地板比較不容易清潔乾淨。

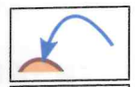
關鍵字

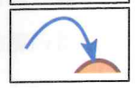
循序結構、演算法

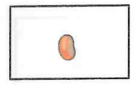
23. 種植胡蘿蔔

機器兔用來把胡蘿蔔種子種植在小土堆上。

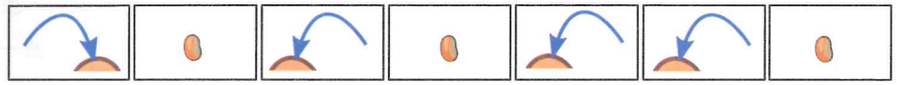
它能執行以下指令：


向**左**跳到下一個土堆


向**右**跳到下一個土堆


在目前所在的土堆種下一顆**胡蘿蔔種子**

以下由左到右為機器兔收到的一串指令：



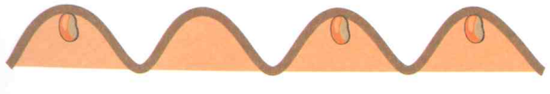
我們不知道機器兔一開始被放在哪個土堆上，只知道依照指令執行完，機器兔在其中三個土堆各種下一顆胡蘿蔔種子。

請問是哪三個土堆被種下胡蘿蔔種子？

A. 

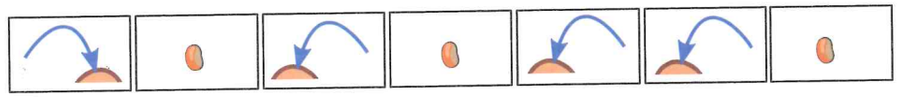
B. 

C. 

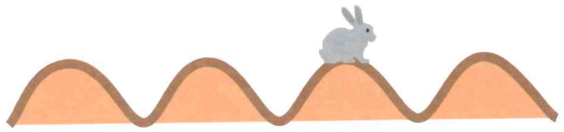
D. 

正確答案是：D

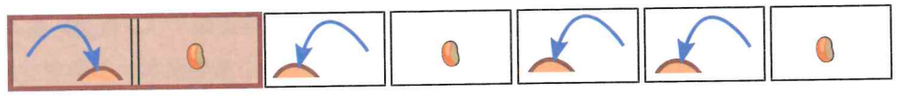
由下圖所示指令序列：



其中先向右跳躍一次，然後向左跳躍三次；因此機器兔一開始必須放在從左邊數過來的第三個土堆，才不會跳到這些土堆以外。



依序執行前兩個指令後：

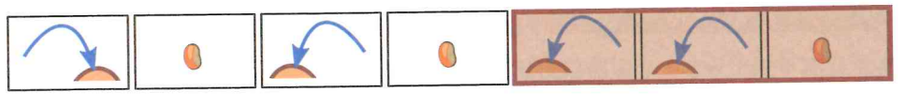


機器兔在最右邊的土堆種下種子，如下圖：

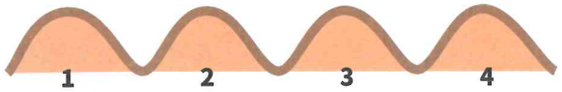
接著執行下兩個指令後：

機器兔會在右邊數過來第 2 個土堆種下種子，如下圖：

接著向左跳躍兩次並種下最後一顆種子：



胡蘿蔔種子種下的結果，如下圖，因此答案選項 D。



若機器兔一開始放在從左邊數過來的第一個土堆，
執行完前四個指令後：會在前兩個土堆各種下一顆種子，在第五個指令時跳出土堆範圍而發生錯誤。

若機器兔一開始放在從左邊數過來的第二個土堆，
執行完前四個指令後：會在中間兩個土堆各種下一顆種子，執行第五個指令時跳出土堆範圍而發生錯誤。

若機器兔一開始放在最右邊的第四個土堆，
執行第一個指令時便會跳出土堆範圍而發生錯誤。

因此不可能得到選項 A、選項 B、或選項 C 的結果，只有選 D 正確的。

資訊科學上的意義

電腦程式就像控制機器兔的指令，但它們通常更複雜，且由一長串指令（程式碼）組成。在寫程式時，如果你發現執行結果不正確，並且懷疑某一行指令可能出了問題，你可以嘗試逐步執行從那一行指令開始的後續指令，並觀察最終的結果是否正確。這樣的過程被稱為**程式碼追蹤 code tracing**，它有助於你理解程式碼的目的和作用。

在這個任務中，我們給機器兔一連串要按照順序執行的指令，這些指令用來控制機器兔在山丘上種植胡蘿蔔種子的位置和順序。要找到機器兔最初的位置，需要先設定一個初始位置，再按照給定的指令一步一步執行，觀察每一步看到的結果，就相當於在進行程式碼追蹤。程式碼追蹤的主要目的是協助＂程式除錯＂，也就是找出並修正程式中的錯誤指令，以確保程式能正確運作。

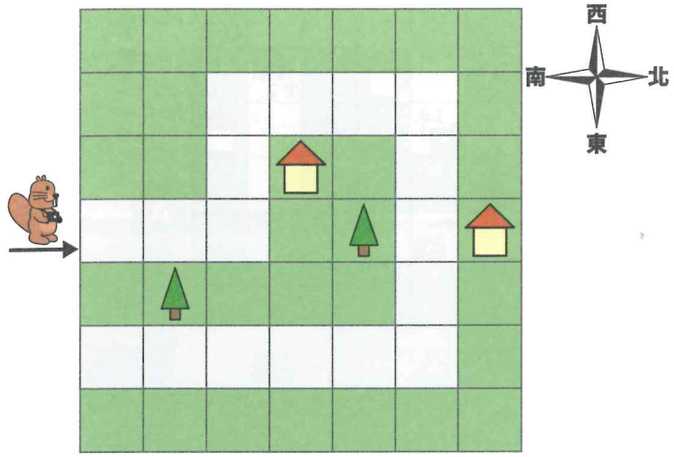
生活中，例如你在組裝一個玩具時，通常會有一份說明書，指導你按照特定步驟組裝。當你組裝完卻無法正常運作時，通常會需要拆解，回到前面的步驟，再一步一步檢查，看過程中是哪個步驟弄錯了。

關鍵字

程式碼追蹤、循序結構

24. 散步日誌

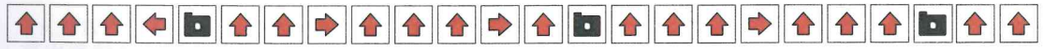
小狸每天都帶著相機沿著白色道路散步。



在散步時，小狸會在日誌上用以下三種符號記錄所走的路線，並記錄在某些位置進行拍照 。每張照片可拍到小狸前方 3 x 3 方格範圍中的景物。

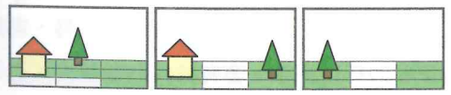
前進一步	向左轉	向右轉	拍照
⬆	⬅	➡	📷

這天，小狸的散步日誌紀錄如下。

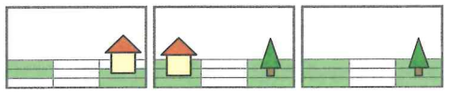


請問以下哪個選項可能是小狸所拍攝的 3 張照片？

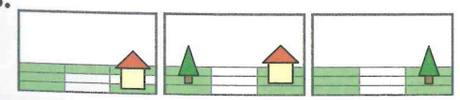
A.



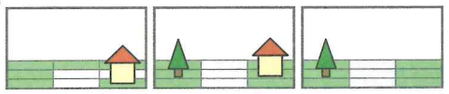
C.



B.

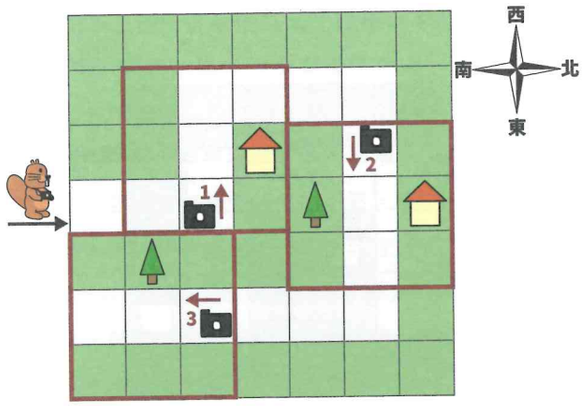


D.



正確答案是：C

根據日誌紀錄，小狸在以下標示 📷 的 3 個位置拍攝了照片 ：



為方便說明，若小狸出發時的是面對北方，

拍第一張相片時，小狸面向西方，所以照片中的景色應該在右側有一棟房子。

拍第二張相片時，小狸面向東方，所以照片中的景色應該在左側有一棟房子，且右側有一棵樹。

拍第三張相片時，小狸面向南方，所以照片中的景色應該在右側有一棵樹。

只有選項 C 完全符合。

資訊科學上的意義

海龜繪圖 turtle graphics 是一種電腦繪圖方式，採用一個相對游標的概念，在二維平面上進行向量式繪圖。游標代表畫筆，透過指定畫筆的座標位置、方向 (旋轉角度)、移動長度、畫筆顏色 (寬度)、是否下筆，決定畫出的圖案。

在本任務中，學生必須像拿著一張地圖，並照著海狸日誌中記下的指令，把海狸走過的路線畫出來，並以海狸拍照的所在位置，想看看在面對的前方能看到什麼東西。就像海龜繪圖時，一開始指定游標位置 (海狸出發起點)，給定移動長度或方向改變指令 (向前、向左轉、或是向右轉)，拍照 (下筆)，就會畫出一張畫 (拍出一些相片)。

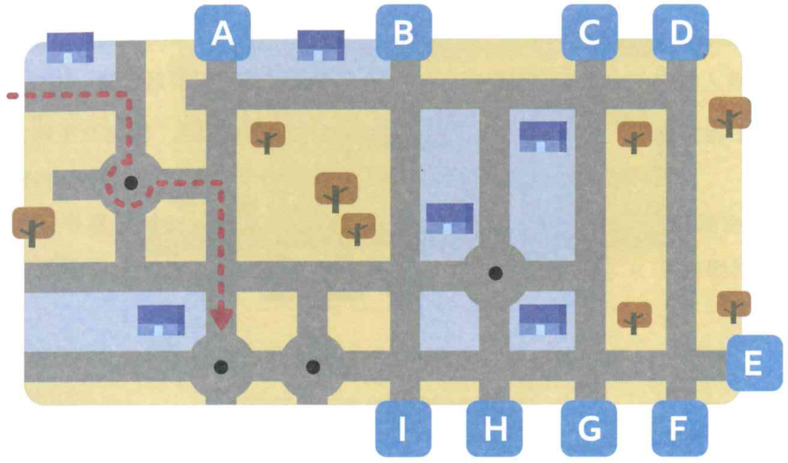
這個任務與海龜繪圖，跟我們想用程式控制機器人的方式很像，控制機器人右轉、左轉、向前、向後時，必須要知道機器人所在位置，才能給機器人正確的行走指令。要有效控制機器人，必須能判斷其位置及相對方向，才能提供正確指令到達目標。

關鍵字

海龜繪圖、機器人控制

25. 自駕車

海狸自駕車在更新之後出現了問題：只要遇到相同類型的路口時，就會往相同的方向行駛。觀察下圖可以發現自駕車遇到不同種類路口的不同行駛方式，舉例來說：車子每次遇到圓環時，都會逆時針繞著圓環行駛到第三個路口再右轉離開圓環；而遇到 T 字型路口就會右轉。



請問若是順著上圖箭頭方向繼續行駛，海狸自駕車最終會抵達哪個字母所示的地點？

A. B

B. C

C. D

D. F

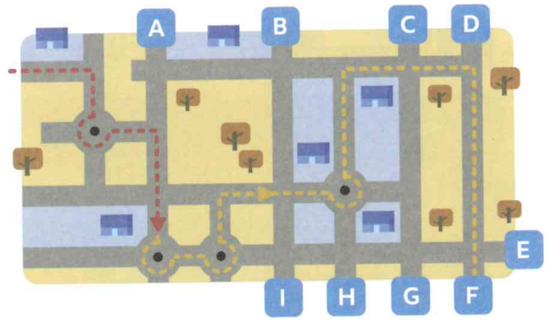
E. G

正確答案是：D

由於自駕車只要遇到相同類型的路口時，就會往相同的方向行駛；而由上圖可以歸納出以下規則：

- 遇到 T 型路口時，自駕車就會右轉。
- 遇到圓環時，自駕車會逆時針繞著圓環行駛到第三個路口，再右轉離開圓環。
- 遇到十字路口，自駕車會直行通過。

根據上述規則，自駕車依紅色虛線行駛至圓環後，就會如下圖依黃色虛線繼續行駛至 F 處，故正確答案是選項 D。



資訊科學上的意義

程式設計師寫好的程式碼可由電腦執行出結果，但是當電腦無法順利執行程式，或是執行的結果在預期之外，就可以應用 **程式碼追蹤 code tracing** 的技巧，推算出一段程式執行的結果。

程式碼追蹤可以使用人腦執行，在此任務中，我們追蹤車子行駛前幾條道路的原則，推算出之後行駛的路線。此時人腦需要確實循著演算規則一步步運算以得到結果；另外也有許多程式開發軟體可以輔助程式碼追蹤，包括除錯器 debugger、設定斷點 breakpoint 等功能。

程式碼追蹤的能力對程式設計師來說很重要，例如替程式除錯，或是與別人合作開發專案，需要閱讀別人的程式碼。善用程式碼追蹤技巧可以節省重複執行大量程式的電腦資源

關鍵字

程式碼追蹤、指令、演算法、自動化

26. 碰撞機器人

碰撞機器人遊戲方式是透過指令移動機器人，讓指定的機器人從目前位置移動，最後停在星星 的位置。

遊戲規則如下：

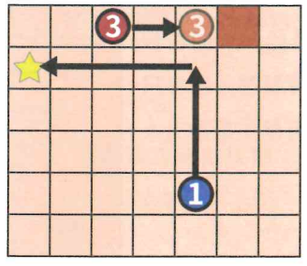
1. 棋盤上的棕色方塊 及其他機器人都是障礙物。當遇到障礙物，機器人是無法通過的。
2. 機器人只能透過指令，往上（↑）、下（↓）、左（←）、右（→）4 個方向移動。
3. 一旦機器人開始向某個方向移動，就會一直移動，直到撞到障礙物或棋盤的邊緣才停下來。
4. 當有機器人開始移動時，其他機器人會等待不動，直到移動中的機器人停下來。

我們用下面的例子來說明：目前棋盤上有機器人 1 及機器人 3 ，玩家透過 3 個指令就能讓機器人 1 移動並停在星星 的位置。

首先，第 1 個指令是讓機器人 3 向右移動（這將使它移動並停在 障礙物旁邊）。

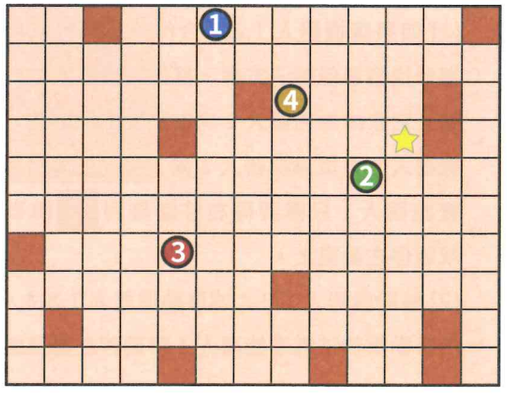
第 2 個指令是讓機器人 1 向上移動（利用剛剛停下來的機器 3 人 當障礙物，讓它停在與星星同一橫列上）。

第 3 個指令是讓機器人 1 向左移動碰到棋盤邊緣後停在星星的位置。



請問，依據下列棋盤上的情況，哪個選項可以讓機器人 1 移動停在星星 上的指令是<u>最少</u>的？

A. 只移動機器人 1
B. 移動機器人 1 2
C. 移動機器人 1 4
D. 移動機器人 1 2 3
E. 移動機器人 1 2 3 4



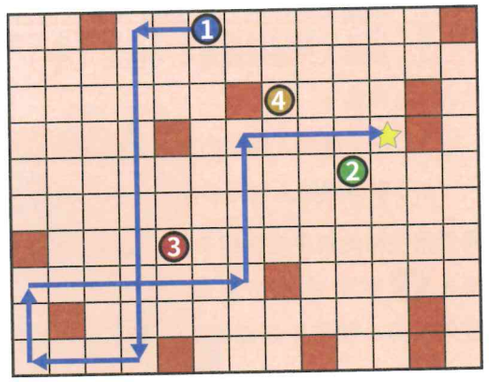
正確答案是：D

在下面的解釋圖中，每個機器人的移動指令都以箭頭標記。在解決這個任務時，機器人之間的合作非常重要。

以下說明，將分為 (I) 機器人沒有合作的解決方案、(II) 只有兩個機器人合作的解決方案及 (III) 三個機器人合作的最佳解決方案。

(I) 機器人沒有合作的解決方案

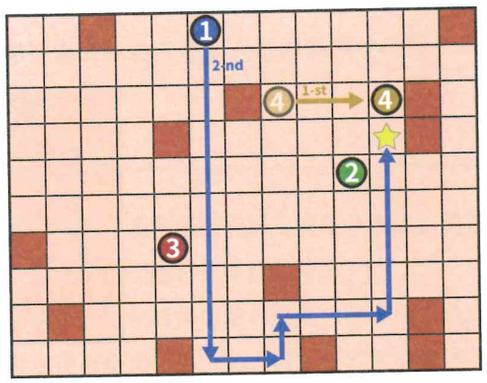
是只使用機器人 1，從起點向左移動需要 7 個移動指令，即可停在星星，如右圖所示。



(II) 只有兩個機器人合作的解決方案

(1) 如果讓機器人 1,2 或機器人 1,3 合作，他們的移動都無法幫助機器人 1 移動停在星星上。

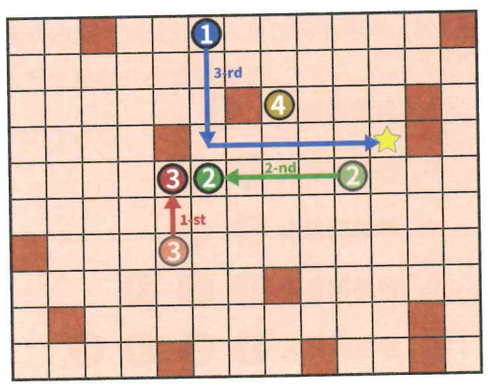
(2) 如果讓機器人 1,4 合作，讓機器人 4 向右移動可以成為機器人 1 在移動後的障礙物，這樣只需要 6 個移動指令，機器人 1 即可停在星星上，如右圖所示。



(III) 三個機器人合作的解決方案。

(1) 如果讓機器人 1,2,3 合作，我們可以用 4 個指令找到最佳解決方案。首先將機器人 3 向上移動，它會成為機器人 2 向左移動的障礙物。然後機器人 2 會成為機器人 1 向下移動的障礙物。然後機器人 1 只需要再右移動碰到 障礙物，就能停在星星上。

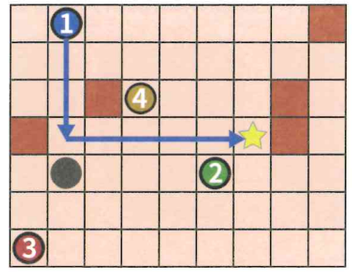
(2) 移動機器人 1,2,4 或移動機器人 1,3,4，他們的移動都無法幫助機器人 1 移動停在星星上。



如何得到正確解答：

如果我們開始考慮機器人 1 的可能移動方向，會發現有太多組合。因此，我們可以從結束的位置開始思考。一個可能的解決方案是它從左邊碰到 障礙物停在星星上；另一個選擇是先將機器人 4 向右移動碰到 障礙物停下，然後機器人 1 可以從下方往上碰到機器人 4 停在星星上。但要從下方到達星星，其他機器人需配合為障礙物來幫助機器人 1 移動，會需要更多的指令。

如果我們將從左邊到達星星作為最後一個移動，我們可以檢查機器人 1 該如何停在最後一個移動的起始方塊上。然後我們發現需要一個障礙物在 (如右圖)，作為其他機器人停下來的合適位置。這個過程將在後續步驟中重複進行，直到我們找到解決方案。

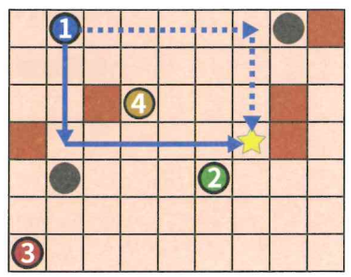


另一種策略是找出到達最終目標的最短路徑，再檢查需要在那些轉彎處設立障礙物來幫助轉彎。如果需要，我們也可以試著讓其他機器人移動到 當做障礙物來幫助機器人 1 轉彎。

為什麼 3 個指令無法完成任務：

機器人 1 沒辦法在 1 個指令內到達星星。

有兩種 2 個指令的移動方法 (如右圖)。要實現這樣的移動，我們需要先在其中一個 (灰 色 圓形) 放置一個機器人，這樣機器人 1 才能彈回並停在它上面。但我們沒辦法讓其他機器人在 1 個指令內移動到這兩個 之一。



因此，4 個指令是解決方案的最少的數量。

資訊科學上的意義

控制單台機器人 / 車執行任務是當前熱門研究主題，未來我們將遇到另一個重要議題，就是如何讓多台機器人 / 車彼此合作以完成任務。

以本任務為例，如果只有一號機器人在場上，我們可以確定它要走較多步才能抵達星星；然而如果和其他機器人彼此合作，那麼一號機器人就可以更快達到目標。

當然協調機器人彼此合作不是容易的事，它除了需要精確計算，還要考慮機器人的角色、工作順序、彼此的關係架構等，設計不良反而可能會讓機器人互相阻礙。

近代發展的協作機器人 Cobots 聚焦在機器協助人類工作，應用在醫療手術、農業、精密工程、服務業等。而研究機器人彼此之間的合作將是未來新的研究與發展方向。

關鍵字

自走車控制

27. 香蕉背包

小麥手邊有八串香蕉，各串數量如下圖左，媽媽手邊也有五串香蕉，各串數量如下圖右。

今天小麥想把所有香蕉串都在不撕開的狀況下，將他的香蕉數量平均分配，裝在三個背包中。如果有需要的話，可以請媽媽再給一串香蕉加入分配。

依任務規則，所有香蕉串都在不撕開的狀況下，小麥是否能將他手邊的香蕉數量平均分配在這些背包中？

A. 不行，即使媽媽多給小麥一些香蕉也無法平均分配。

B. 可以，媽媽必須給小麥一根的香蕉串。

C. 可以，媽媽必須給小麥兩根的香蕉串。

D. 可以，媽媽必須給小麥三根的香蕉串。

E. 可以，媽媽必須給小麥五根的香蕉串。

正確答案是：D

媽媽必須給小麥三根的香蕉串，這樣他才能將 15 根香蕉平均分配到每個背包，例如像這樣：

為了解決這個任務，我們必須找到可以被 3 整除的所有香蕉串的總數。小麥所有的香蕉數量加總是 3 + 4 + 4 + 4 + 5 + 6 + 8 + 8 = 42 根香蕉。 42 可以被 3 整除，得到 14。

我們來嘗試分配這些香蕉串。14 是一個偶數，因此必須將 3 根和 5 根的香蕉串組合放在同一個背包。我們將以下數量的香蕉分配到三個背包中：(5 + 3)、8、8、4、4、4 和 6；發現可以用 (8 + 6=14) 或 (4 + 4+ 6=14) 來填滿其中一個背包後，在不能撕開香蕉串的條件下，就沒有辦法再將其他兩個背包填滿 14 根香蕉了。無法將 (4, 4, 4, 8, 8) 中的數字或 (3, 4, 4, 4, 5, 8) 中的數字組合起來得到 14 的和，這是可以通過數學來證明的。

我們的結論是，小麥必須再跟媽媽拿一串香蕉，而且小麥的香蕉總數是 42 加上跟媽媽拿的香蕉數量的總和一定要是一個可以被 3 整除的數字，所以小麥就必須選擇 3 根的香蕉串。因此 42 + 3 = 45 根香蕉，可以被 3 整除，也就是說每個背包必須裝滿 15 根香蕉。從最大的香蕉串 (8、8、6) 開始，8 根的香蕉串，可以配合找出香蕉數量 7 的組合 (4+3)，6 根的香蕉串可以配合找出香蕉數量 9 的組合 (4+5)，就完成將香蕉平均分配在三個背包的任務了。

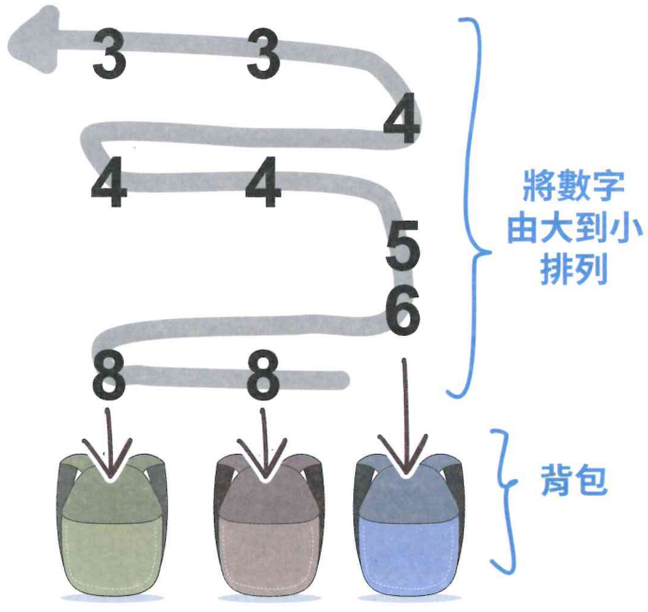
資訊科學上的意義

最佳化 optimization 是尋求一個問題最佳解答的過程，也就是在已知的條件限制下，將問題以數學方式表示，並透過運算或搜尋的方法求出最佳解的步驟。

分區問題 partition problem 即最佳化的一種，是將一個集合（此任務中所有的香蕉串）分為若干個子集合（背包內的香蕉串），且每個子集合的和（背包內香蕉數量）必須相等。這類的問題在集合很小時，很容易判斷是否有解並找出一組答案，但當集合變大時，就變得難以快速找到解。

目前為止，資訊科學家沒有找到有效率的演算法可以解決分區問題，這類問題被歸類為 **NP 完備 NP-complete**：可以快速驗證一組答案是否正確，但很難找到一組解。

在特定的條件下，分區問題有較有效率的解法，例如：在處理數字分區問題時，可運用貪心演算法，將集合中的數字先由大到小排列，接下來每次將數字加入總和最小的子集合。此任務中每個背包應該分配哪一串香蕉即可使用此方法。



分區問題可運用於文件要如何平均分給不同的印表機列印，才能讓所有文件能夠最快速列印完畢；貨櫃要如何平均的分配到多艘貨輪上；班內大隊接力競賽要如何分組才能讓各隊實力最平均。

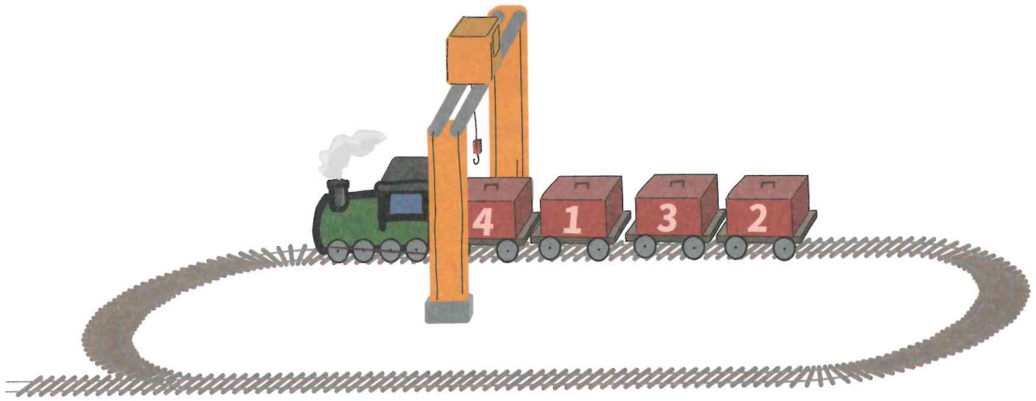
關鍵字

最佳化問題、分區問題

28. 火車卸貨 - 題組一

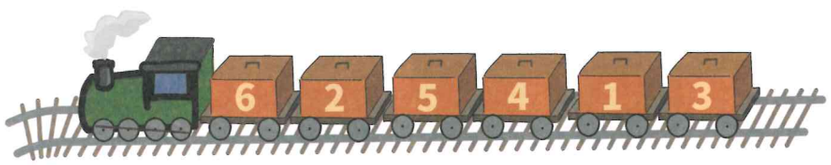
火車有數節車廂，每節車廂上有一個編號貨櫃。卸貨區有環狀軌道和一台固定的起重機。只有貨櫃移動到起重機正下方的時候，起重機才能卸下這個貨櫃。

依規定，起重機必須照貨櫃編號由小到大卸貨，因為火車只能往前行駛，如果貨櫃已經超過起重機，火車就必須多行駛一圈回到起重機下方以繼續卸貨。



以上圖為例，火車上有 4 個貨櫃。火車行駛第一圈時，起重機依序卸下貨櫃 1、貨櫃 2；跳過了貨櫃 4 及貨櫃 3。接著火車行駛第二圈卸下貨櫃 3，再行駛第三圈卸下貨櫃 4。

如下圖，請問這列貨運火車最快在行駛第幾圈時，起重機才能卸下所有貨櫃？



A. 2

B. 3

C. 4

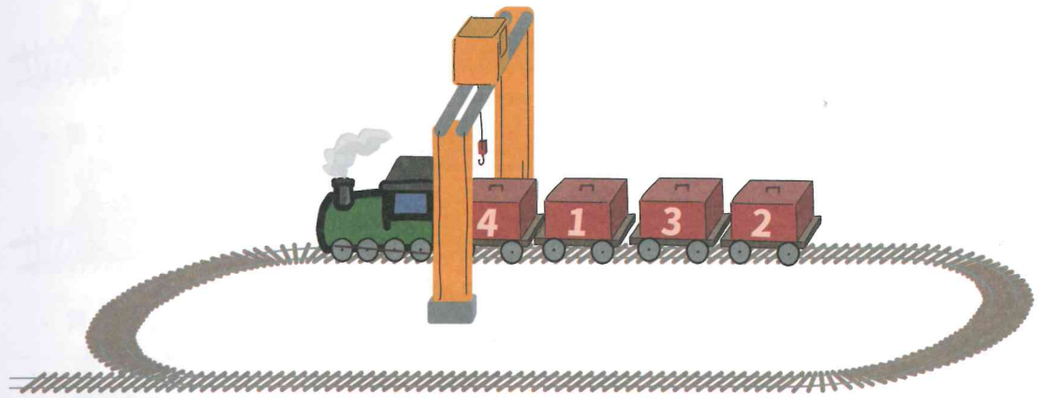
D. 5

E. 6

29. 火車卸貨 - 題組二

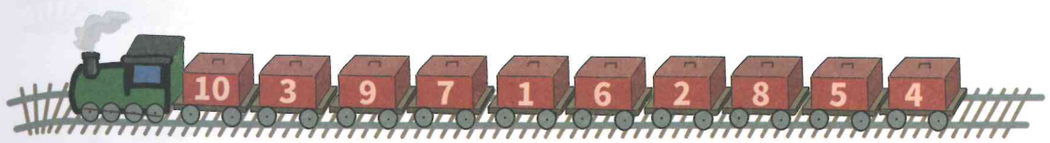
火車有數節車廂，每節車廂上有一個編號貨櫃。卸貨區有環狀軌道和一台固定的起重機。只有貨櫃移動到起重機正下方的時候，起重機才能卸下這個貨櫃。

依規定，起重機必須照貨櫃編號由小到大卸貨，因為火車只能往前行駛，如果貨櫃已經超過起重機，火車就必須多行駛一圈回到起重機下方以繼續卸貨。

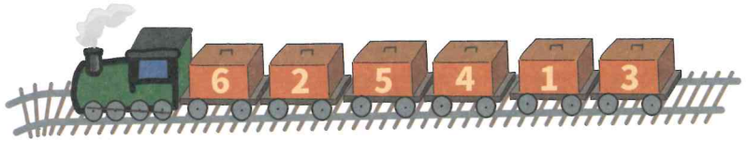


以上圖為例，火車上有 4 個貨櫃。火車行駛第一圈時，起重機依序卸下貨櫃 1、貨櫃 2；跳過了貨櫃 4 及貨櫃 3。接著火車行駛第二圈卸下貨櫃 3，再行駛第三圈卸下貨櫃 4。

如下圖，請問在這列貨運火車最快在行駛第幾圈時，起重機才能卸下所有貨櫃？
(範圍 [1~10] 的整數)

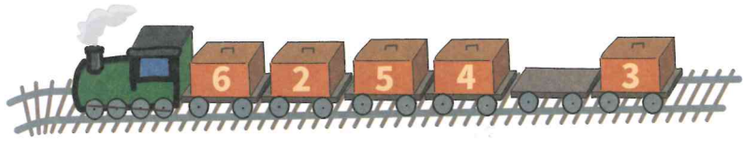


28. 題組一的正確答案是：D

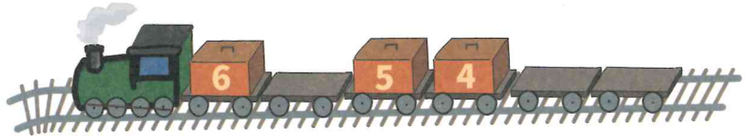


依照「起重機必須依照貨櫃編號由小到大卸貨」的規定操作，卸貨的過程如下所述：

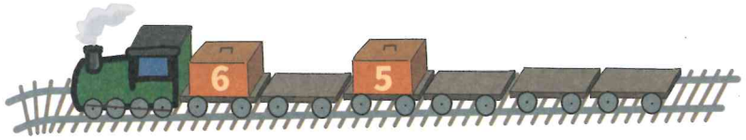
- 第 1 圈：跳過貨櫃 6、2、5 和 4，卸下貨櫃 1



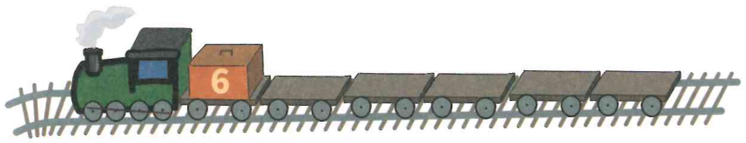
- 第 2 圈：跳過貨櫃 6，卸下貨櫃 2 和 3



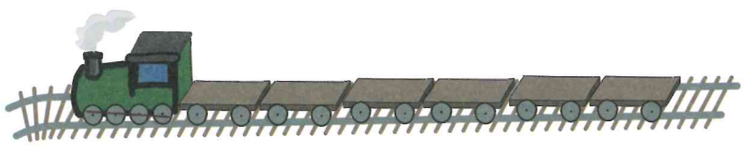
- 第 3 圈：跳過貨櫃 6 和 5，卸下貨櫃 4



- 第 4 圈：跳過貨櫃 6，卸下貨櫃 5

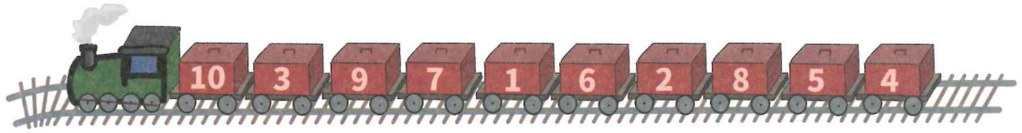


- 第 5 圈：卸下貨櫃 6，工作結束



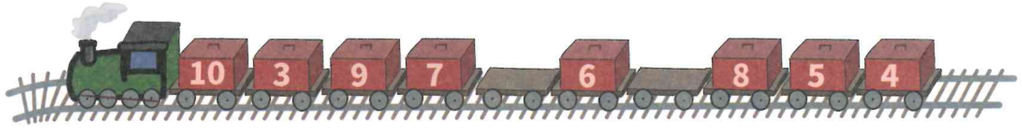
因此，正確答案是最快在第 5 圈時，起重機才能卸下所有貨櫃。

29. 題組二的正確答案是：7

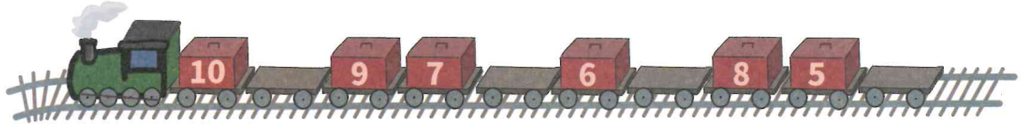


依照「起重機必須依照貨櫃編號由小到大卸貨」的規定操作，卸貨的過程如下所述：

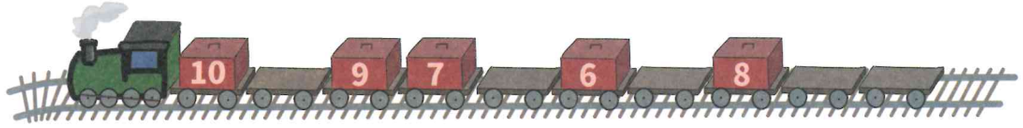
- 第 1 圈：跳過貨櫃 10、3、9 和 7，卸下貨櫃 1 和 2



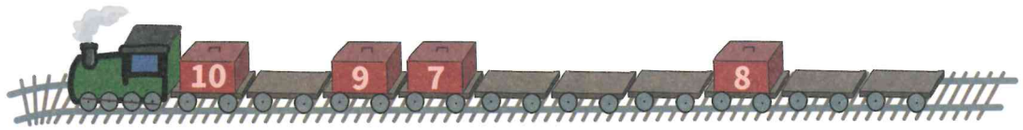
- 第 2 圈：跳過貨櫃 10，卸下貨櫃 3 和 4



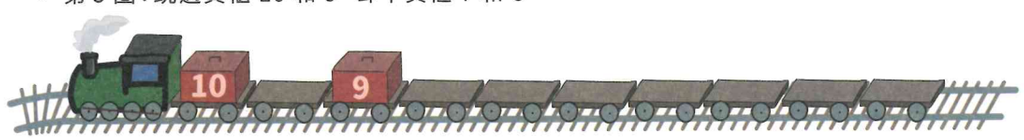
- 第 3 圈：跳過貨櫃 10、9、7、6 和 8，卸下貨櫃 5



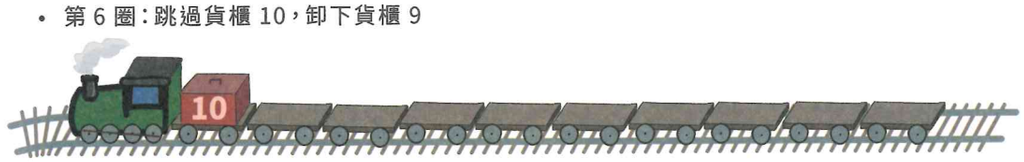
- 第 4 圈：跳過 10、9、7 號貨櫃，卸下 6 號貨櫃



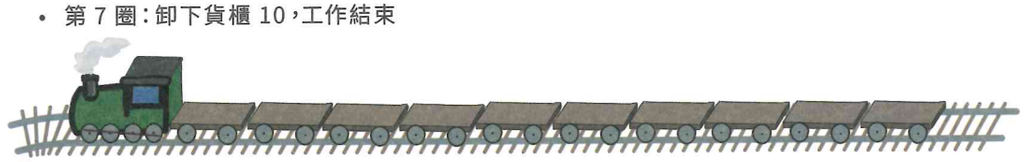
- 第 5 圈：跳過貨櫃 10 和 9，卸下貨櫃 7 和 8



- 第 6 圈：跳過貨櫃 10，卸下貨櫃 9



- 第 7 圈：卸下貨櫃 10，工作結束



因此，正確答案是最快在第 7 圈時，起重機才能卸下所有貨櫃。

要確認火車卸貨最少須行駛幾圈，除了透過按規定操作來驗證，也可以透過觀察起重機卸貨的規律來計算：依序觀察兩個連續數字（如 (1, 2)、(2, 3)、…等），只要觀察到較大的數字出現在較小的數字左側的情況，卸貨時火車就要多行駛 1 圈，也就是火車卸貨最少須行駛的圈數要加 1。因為我們必須先卸下編號較小的貨櫃，所以會跳過編號外大的貨櫃，接著火車就必須多繞一圈才能將接續著編號的貨櫃行駛到起重機下方。

以題組二為例，先觀察 (1, 2)，因為貨櫃 2 在貨櫃 1 的右側，所以貨櫃 1 和 2 在同 1 圈卸貨即可，也就是在第 1 圈可以卸完貨櫃 1 和 2；再觀察 (2, 3)，因為貨櫃 3 在貨櫃 2 的左側，所以火車要多行駛 1 圈卸下貨櫃 3，也就是在第 1+1 圈時卸完貨櫃 1~3，以此類推。

觀察此題組二中貨櫃的編號序列，符合左大右小的數對有 (2, 3)、(4, 5)、(5, 6)、(6, 7)、(8, 9) 和 (9, 10) 共 6 對；因此，第 1 圈開始卸貨後，還須多繞 6 圈，在第 1+6=7 圈卸下所有貨櫃。

資訊科學上的意義

在資訊科學中，排序是指將資料依順序排列好，例如將數值由小排到大；而透過程式排序時經常使用 **排序演算法 sorting algorithms** 。若排序的目標是由小到大排列，那麼較小的數值應該在較大的數值之前；此時，如果較小的數值在較大的數值之後，則這樣的數字關係被稱作 **逆序 inverse** 。如數列 “ 6, 7, 1” 中，就有兩對逆序的數對：(6, 1) 和 (7, 1)。

觀察本任務中每回合火車的卸貨狀態會發現，不管是幾號貨櫃，如果下一號數字的貨櫃在它的左側（在它於火車頭之間），卸貨的回合數就會增加 1。這就是因為這兩個貨櫃是逆序的。

在資訊科學中，某些排序演算法會運用到逆序的觀念，例如 **氣泡排序法 bubble sort**：其規則是在每一回合中將元素兩兩進行比較，如果順序錯誤（逆序）就進行交換。因此，氣泡排序法中完成排序所需要交換的總次數，其實就是數字列中逆序的次數，逆序的次數越少，表示需要進行互換的次數越少，排序的效率越高。以第一段中所舉 “ 6, 7, 1” 為例，若用氣泡排序法就要進行兩次交換才能將其排序為 “1, 6, 7” 的順序。

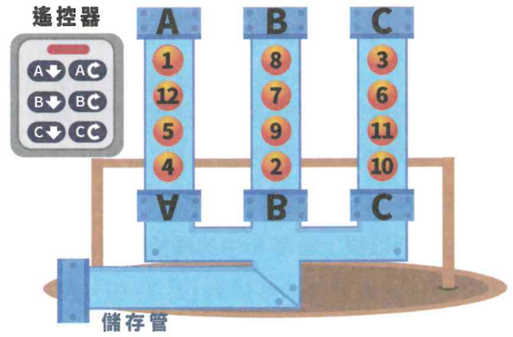
在現實生活中，人們時常需要對資料進行排序；像是依照成績高低排序來確定排名，或是字典中的單字編列也是依照字母順序來排列。這些都會運用到排序演算法。而逆序則常應用在線上商城的行銷上：首先，商城會讓使用者們依照偏好順序對同一組商品進行排名，如果使用者在排名的逆序數量越相近，表示他們的品味越一致，商城就可以依此將他們歸類為「相似」的客戶，進而推薦他們相同的產品。

關鍵字

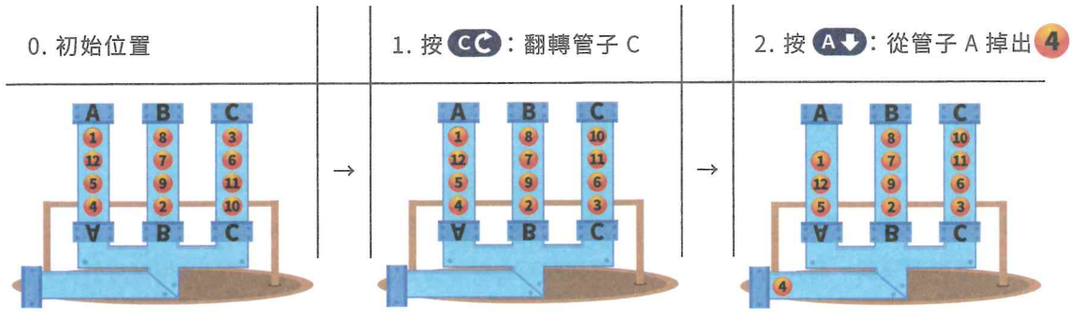
排序、逆序數

30. 旋轉球臺

如下圖所示，玩具球臺上有三根透明管子（A、B、C），每根管子裡面分別裝 4 顆數字球，另有一個遙控器可以控制球臺：按鍵 A↻、B↻、C↻ 會分別翻轉管子 A、B、C（180 度），而按鍵 A↓、B↓、C↓ 則會分別讓管子 A、B、C 最下方的一顆球掉下，再沿著管道滾至儲存管。



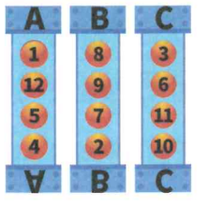
例如，在遙控器上先按 C↻ 再按 A↓，球的位置就會如下圖改變：



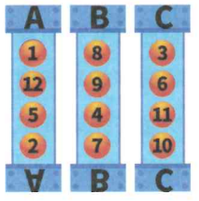
小明想要用遙控器操作球臺，將所有球依數字由小到大的順序，滾到儲存管中。

請問下列哪一組球的初始位置，無法達成小明的目標？

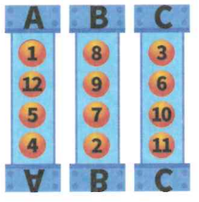
A.



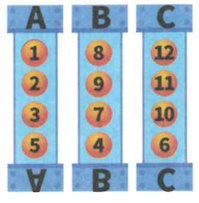
B.



C.



D.



正確答案是：B

我們知道球的正確順序為：1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12。因此，因為球 4 位於球 7 和球 9 之間，所以無法讓球 4 在球 7 或球 9 之前放至儲存區。

其他選項都是可以由小到大排列的。每個管子內的球的編號順序從底至頂必須滿足以下條件之一，小明才能按照升序的方式放下球：

1. 遞增順序
2. 遞減順序
3. 遞增順序後面跟著遞減順序

選項 A：可做到數字由小到大的排列目標。管子 A 中的球按照遞增順序排列為 [4, 5, 12]，然後遞減順序 [12, 1]。管子 B 中的球按遞增順序排列 [2, 7, 9]，然後是遞減順序 [9, 8]。管子 C 中的球按遞增順序排列 [10, 11]，然後是遞減順序 [11, 6, 3]。

選項 C：可做到數字由小到大的排列目標。管子 A 中的球按遞增順序排列 [4, 5, 12]，然後是遞減順序 [12, 1]。管子 B 中的球按遞增順序排列 [2, 7, 9]，然後是遞減順序 [9, 8]。管子 C 中的球按遞減順序排列 [11, 10, 6, 3]。

選項 D：可做到數字由小到大的排列目標。管子 A 中的球按遞減順序排列 [5, 3, 2, 1]。管子 B 中的球按遞增順序排列 [4, 7, 9]，然後是遞減順序 [9, 8]。管子 C 中的球按遞增順序排列 [6, 10, 11, 12]。

資訊科學上的意義

在資訊科學中，排序是指將資料依順序排列好，例如將數值由小排到大；而透過程式排序時經常使用的 **排序演算法 sorting algorithms** 有很多種。例如 **合併排序法 merge sort** 是先將數字陣列拆分至每個小陣列都只剩一個元素，再將這些小陣列兩兩合併；合併後陣列中的數字已依序排序好，於是再將尚須合併的陣列，從第一項開始比較大小並依序合併，並反覆合併陣列，直到所有小陣列都合併至同一個陣列便完成排序。而 **K- 路合併排序法 K-way merge sort** 和合併排序法作法相似，只是每次都將 K 個數字陣列依規則合併成一個數值依序排好的陣列，再重覆直到所有小陣列都合併到同一個陣列中。

在本任務中，比較三個管子最前端的球上的數字，透過操作讓所有的球從小排到大，就類似 K- 路合併排序法；三個管子就好比三個數字陣列，只是在 K- 路合併排序法實作中的每小個陣列都會是已排序好的陣列，而本任務中的管子裡的球並沒有先排序好。在現實生活中，人們時常需要對資料進行排序；像是依照成績高低排序來確定排名，或是字典中的單字編列也是依照字母順序來排列一樣。因此，需要穩定又有效率的排序法，以利正確又快速的完成排序。

關鍵字

排序演算法、K- 路合併排序、雙向佇列

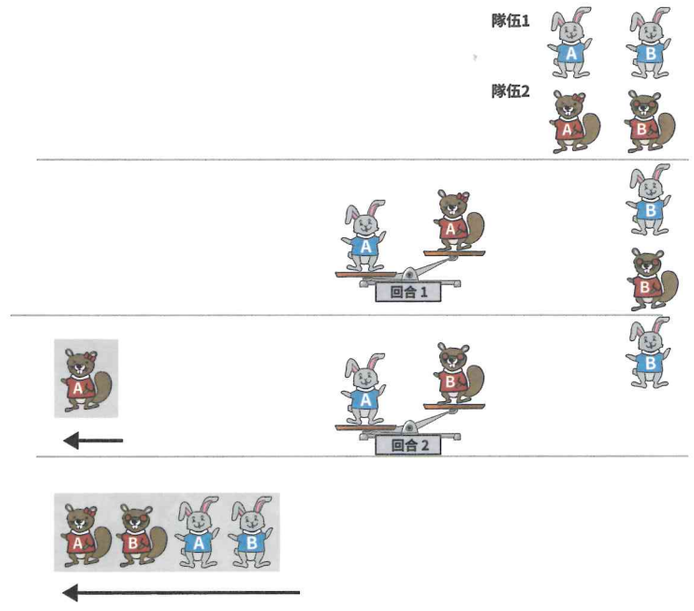
31. 排隊

動物盃舉重比賽規定所有參賽隊伍需依照體重的輕重順序參賽。

首先，每隊會由輕到重（由左到右）各排成一列，再依照下列步驟決定入場順序。

1. 兩列的第一個動物會站上天平比較體重。
2. 較輕的動物會先入場，較重的動物會回到原列的第一個。
3. 一直重複步驟 1 及 2，直到其中一列的動物都已入場。
4. 剩下未入場的動物則按照它們原本的排列順序入場。

右圖顯示兩列動物入場的過程。



這兩隊排順位需要兩次比較。

現在，有三個隊伍如右圖排列。動物上面的數字代表它們的體重。哪兩個隊伍排順位需要進行最多次的比較？

	A	B	C
兔子隊	20	25	27
青蛙隊	13	17	23
海狸隊	11	15	18

A. 海狸隊和青蛙隊

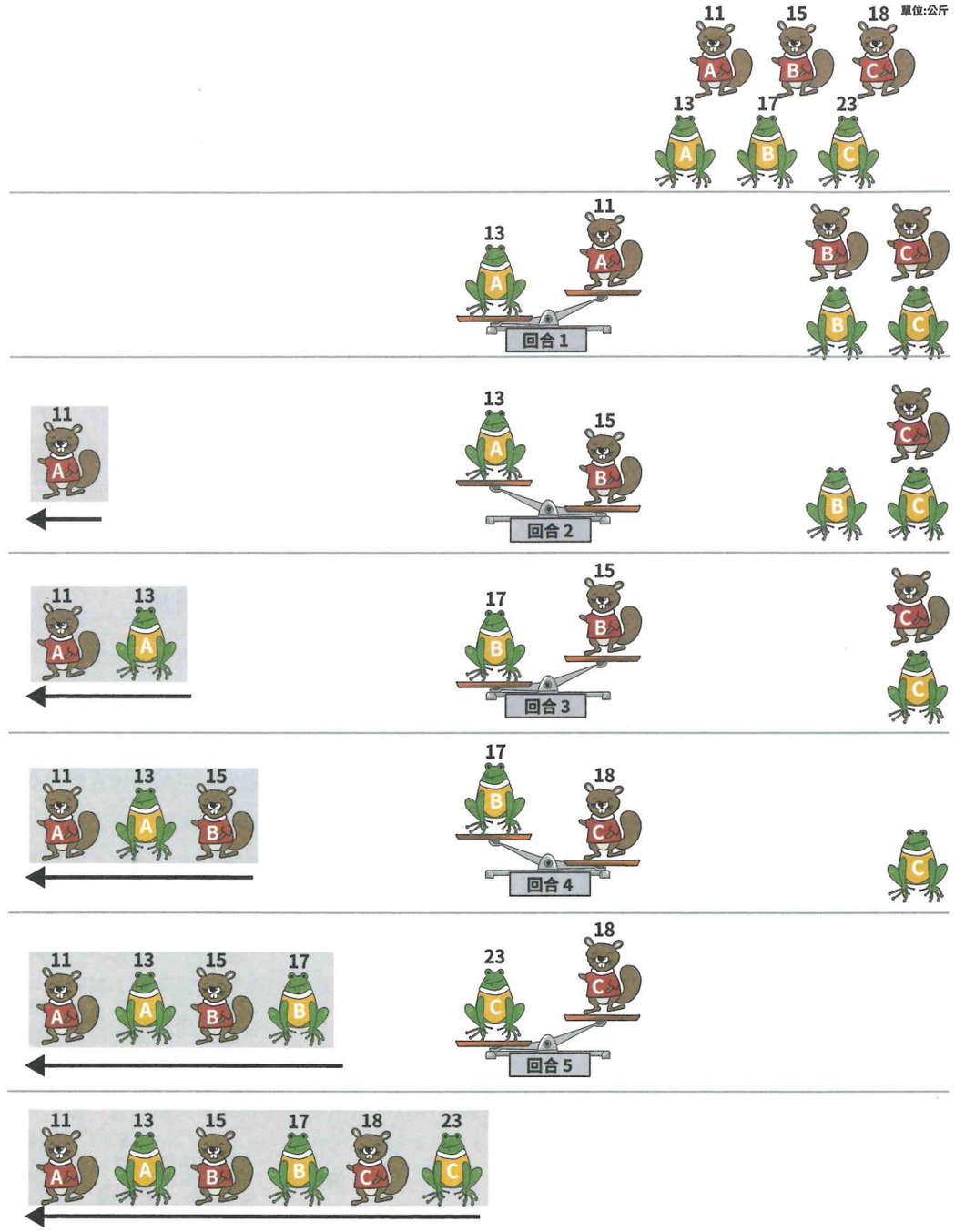
B. 青蛙隊和兔子隊

C. 海狸隊和兔子隊

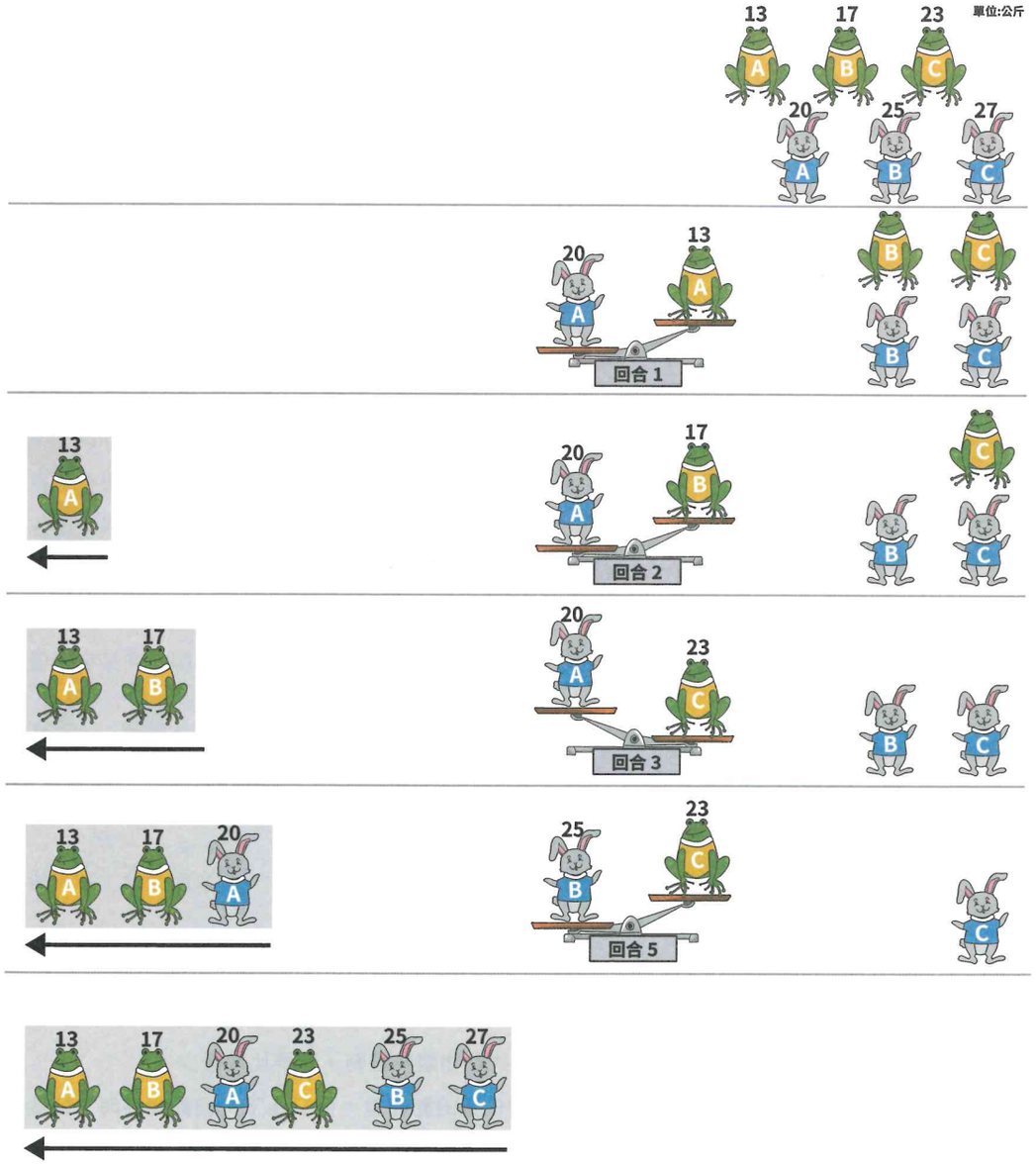
D. 比較的次數都一樣多

正確答案是：A

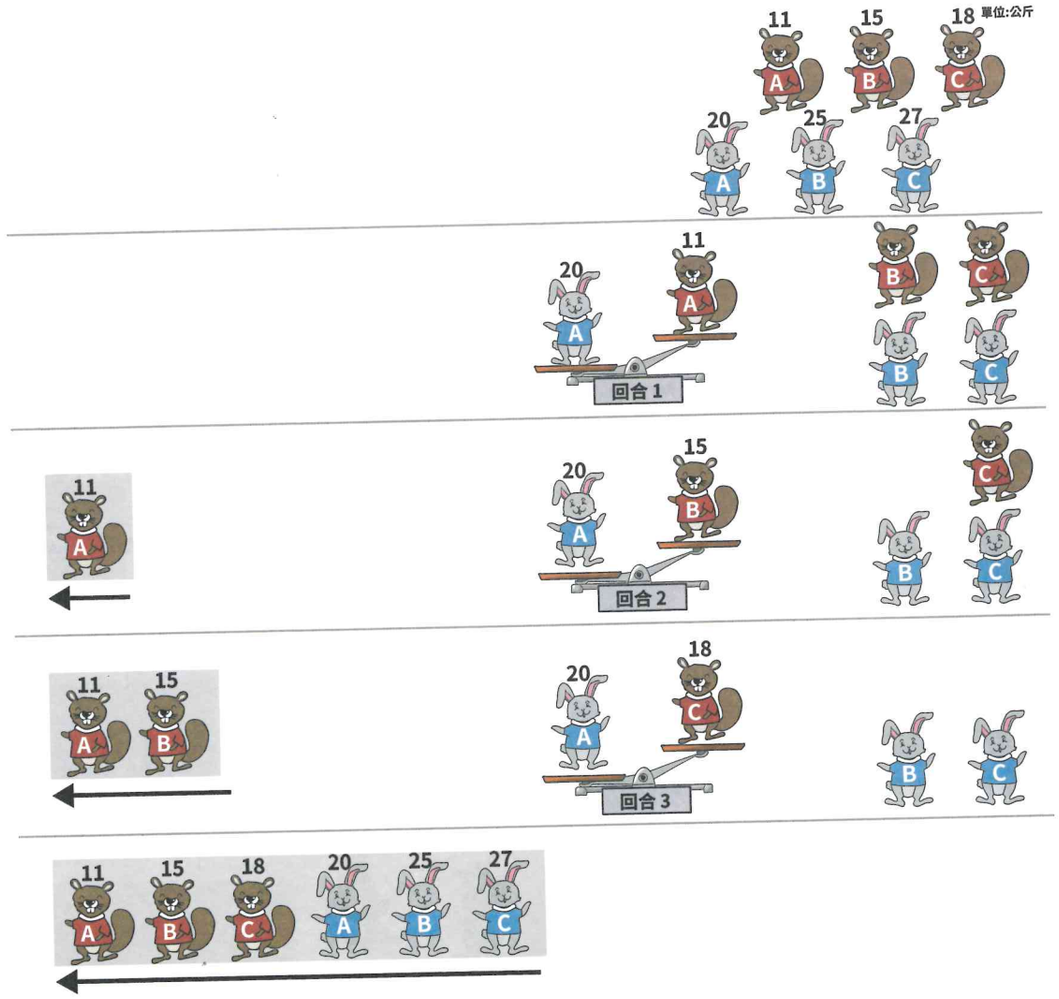
將海狸隊和青蛙隊排順位需要五次比較，如下所示：



將青蛙隊和兔子隊排順位需要四次比較，如下所示：



將海狸隊和兔子隊排順位需要三次比較，如下所示：



實際上並不需要對這三個隊伍執行合併排序演算法；只需觀察：

1. 當一個隊伍中最重的動物比另外一隊最輕的動物還要輕時，需要比較最少次；
2. 當一個隊伍中每個動物都比另外一隊相同位置的動物重，但又比下一個動物輕時，需要比較最多次；也就是說，對於兩個隊伍的動物來說，下一個比較重的動物在另外一隊中。

海狸隊和兔子隊滿足上述第 1 項的特性。

海狸隊和青蛙隊滿足上述第 2 項的特性，因此它們是需要比較最多次的兩個隊伍。

資訊科學上的意義

在資訊科學中，排序是指將資料依順序排列好，例如將數值由小排到大；而透過程式排序時經常使用的 **排序演算法 sorting algorithms** 有很多種。例如 **合併排序法 merge sort** 是先將數字陣列拆分至每個小陣列都只剩一個元素，再將這些小陣列兩兩合併；合併後陣列中的數字已依序排序好，於是再將尚須合併的陣列，從第一項開始比較大小並依序合併，並反覆合併陣列，直到所有小陣列都合併至同一個陣列便完成排序。

本任務中每一個隊伍就好比是已依序排列好的小數字陣列，透過每次比較隊伍的第一人，將所有動物合併至同一個隊伍中，就與合併排序法相似。

而本任務最終的目標其實是「將一群動物依照體重由輕到重排序」；然而，因為每一隊的動物已排序好了，因此我們可以將任務的目標拆解成較小的目標：「每次比較兩個隊伍最前面的動物」，當所有動物都比較完成後，就完成了任務最終的目標了；這樣的問題解決辦法被稱作 **分而治之法 divide and conquer**，也就是將較大的問題拆解為相同模式的子問題時，再透過解決多個小問題後，合併統整進而解決大問題。分而治之法常常運用在資訊科學中，也是設計許多重要演算法的關鍵，提高演算法的效率。

在生活中，人們也常常運用分而治之法來解決問題，提高效率。而人們時常需要對資料進行排序；像是依照成績高低排序來確定排名，或是字典中的單字編列也是依照字母順序來排列一樣。因此，需要穩定又有效率的排序法，以利正確又快速的完成排序。

關鍵字

合併排序、分而治之法

32. 動物園一日遊

小安打算從上午 9:00 到下午 16:00 都待在動物園。
動物園摺頁中介紹了許多活動，並標示出每個活動開始到結束的時間區段。

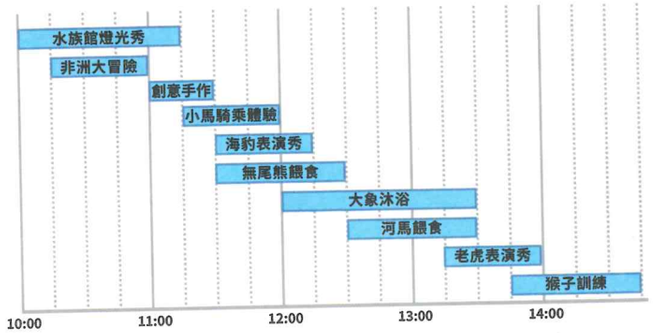
小安對每個活動都很感興趣，但動物園要求遊客不能在活動進行中離場，必須從頭到尾參與，因此時間區段有重疊的活動無法都參加。

請問小安一天最多能參與幾項活動？（範圍為 [0~10] 的整數。）

正確答案是：5

解決這個問題的一種方法，是列出所有活動組合，每個在時間上沒有重疊的活動組合都是一個可安排的時間表，然後從中找出可參加最多活動的時間表。

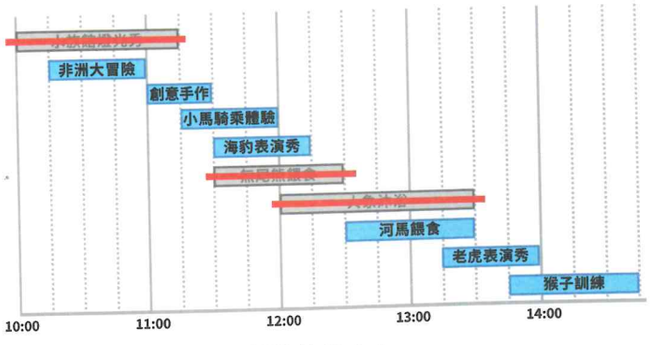
但是透過觀察，可以發現並不需要列出所有活動組合。如果有兩個活動時間上有重疊，那麼這兩個活動中只有一個可安排在時間表中。如果有一個活動的時間完全與另一個活動重疊，那麼選擇時間較短的活動比較不會與其他活動重疊，因此較可能安排更多活動。



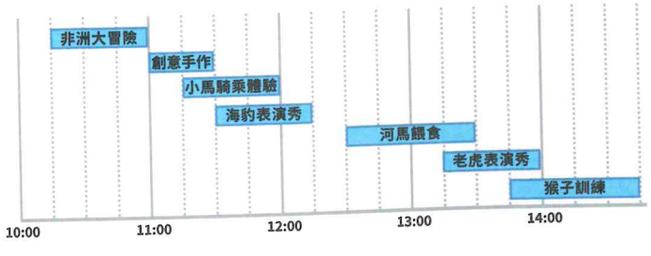
舉例來說，「非洲大冒險」與「水族館燈光秀」的時間完全重疊，且「非洲大冒險」時間較短，因此這兩者最好不要選「水族館燈光秀」，因為它會另外與「創意手作」的時間重疊。

同理可知，我們可以做以下選擇：

- 排除「無尾熊餵食」，因為「海豹表演秀」時間完全與「無尾熊餵食」重疊，且時間較短。
- 排除「大象沐浴」，因為「河馬餵食」時間完全與「大象沐浴」時間重疊，且時間較短。

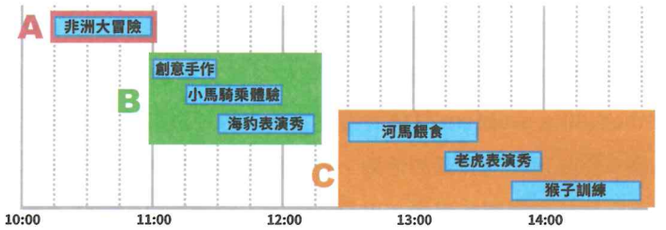


排除以上三項活動後，我們只需考慮以下活動的組合：

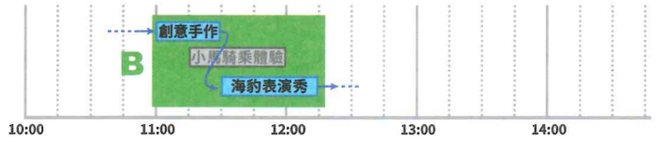


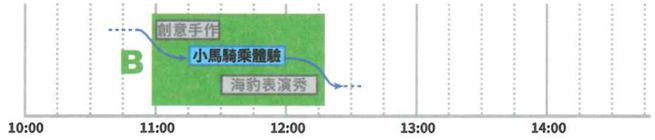
接下來可以將這些活動分成幾個在時間區間上不重疊的群組，如下圖中的 A(10:00-11:00)、B(11:00-12:15) 和 C(12:30-14:45) 三個群組。透過找到在每個群組可參與最多活動的時間安排，合起來便可找出一整天最多能參與幾項活動。

在 A 群組中只有一個活動「非洲大冒險」可參加。



在 B 群組中有三個活動。進行下述活動組合的列舉後，可找出 B 群組中最多可參與「創意手作」和「海豹表演秀」兩項活動。

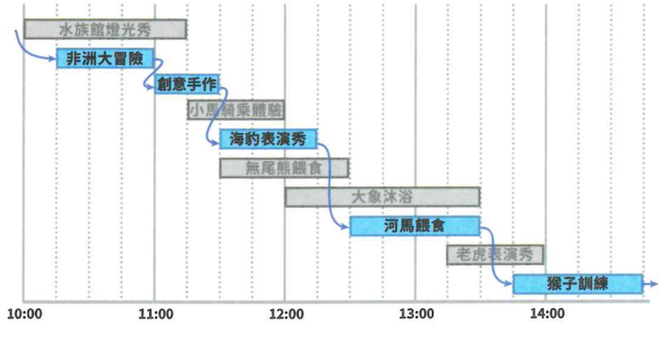




C 群組中也有三個活動，採用相同列舉方式，可找出最多可參與「河馬餵食」和「猴子訓練」兩項活動。

將各群組找出的活動合起來，我們可得到以下時間表：

在這個時間表中共可參加 5 項活動，所以答案是 5。



資訊科學上的意義

最佳化問題 optimization problem 是指在所有可行的方案中，找到最佳方案的問題；最佳化問題通常需要找出某種最大值或最小值：例如，建設的最小成本，或是銷售的最大收益等。本任務中小安在固定的時間區段中，參加最多項活動，也是一種最佳化問題。

在資訊科學領域中，有許多不同解決最佳化問題的演算法，例如透過 **窮舉法** 將所有方案執行結果羅列出來，再從中找出最佳的方案；雖然窮舉法的概念很容易理解且應用廣泛，但是執行起來非常耗時。在選擇解決問題的演算法時，應該考量問題的特性來找出最適合且有效率的演算法。例如 **區間排程問題 interval scheduling problem** 就能利用更有效率的方法來解決問題。

在區間排程問題中，有許多將佔用相同資源的活動，這個資源一次只能進行一個活動，而每項活動都有特定的進行時間；而區間排程問題的目標，就是在固定的時間區段中，找出最多項能在時間區段進行的活動。本任務中小安一次只能參與一個活動，而每個活動又有特定的進行時間；因此，找出小安參與最多活動的活動組合方式，就是在解決典型的區間排程問題。

在本任務中，利用窮舉法可以列舉出 1,024 種活動組合，若要逐一檢視哪個組合小安能參與的活動數量最多，會非常耗時且沒有效率。而透過分析不同活動的時間重疊情形，可以找出須排除或選擇的活動項目。相較於窮舉法，這樣的問題解決方式比較有效率。

關鍵字

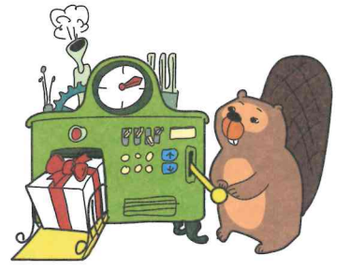
區間排程、最佳化問題、子問題

33. 聖誕老人的幫手

海狸們總是使用玩具製作機來幫聖誕老人製作玩具。

海狸抵達工作室時，會先進入等候區。玩具製作機一次只供一隻海狸使用，並由在等候區裡，製作時間最短的海狸優先使用。

下表是海狸今天抵達工作室的時間和要製作的玩具列表：



海狸	玩具	製作時間	海狸到達時間
斑斑	小汽車	5 分鐘	8:00
皮皮	泰迪熊	10 分鐘	8:00
綿綿	洋娃娃	7 分鐘	8:04
凡凡	火車	7 分鐘	8:10
東東	積木	9 分鐘	8:10

請問下列何者是海狸製作玩具的先後順序？

A. 斑斑、皮皮、綿綿、凡凡、東東

B. 斑斑、綿綿、東東、皮皮、凡凡

C. 斑斑、皮皮、綿綿、東東、凡凡

D. 斑斑、綿綿、皮皮、東東、凡凡

正確答案是：B

在 8:00 時，斑斑和皮皮在等候區中；因為斑斑所需的玩具製作時間（5 分鐘）比皮皮的（10 分鐘）短，所以由斑斑先使用玩具製作機。

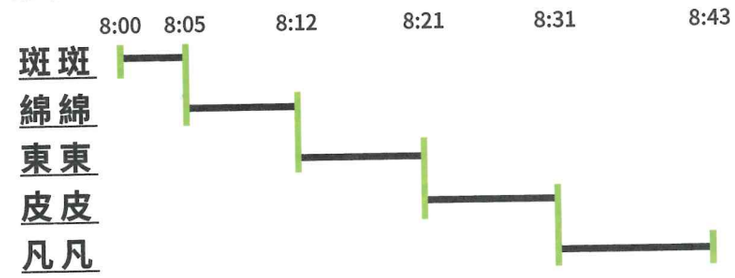
8:05 時，斑斑完成製作，此時皮皮和綿綿在等候區中；因為綿綿所需的時間（7 分鐘）比皮皮（10 分鐘）短，所以接著由綿綿使用玩具製作機。

8:12 時，綿綿完成製作，此時皮皮、凡凡和東東在等候區中；因為東東所需的時間（9 分鐘）比皮皮（10 分鐘）和凡凡的（12 分鐘）短，所以接著由東東使用玩具製作機。

8:21 時東東完成製作，此時剩皮皮和凡凡在等候區；因為皮皮所需的時間（10 分鐘）比凡凡（12 分鐘）短，所以接著由皮皮使用玩具製作機。

8:31 時皮皮完成製作，等候區中只剩凡凡，最後輪到凡凡使用玩具製作機，並於 8:43 分完成製作。

下圖是海狸們分別使用玩具製作機的時間順序。只有選項 B 是正確的，其餘的選項並不符合「製作時間最短的海狸優先使用玩具製作機」的規定。



資訊科學上的意義

程序是執行程式時的運作單位，一支程式可能包含許多程序。電腦的中央處理器 (CPU) 一次只能執行一個程序；**排程 scheduling** 就是指安排程序被 CPU 執行的順序，而決定如何排程的方法，就稱作 排程演算法。其中一種是 **最短工件優先 Shortest Job First (SJF)** 演算法，採用 SJF 的 CPU 會在所有等待執行程序中，先選擇執行「預估花費時間」最短的程序，其優點是可以減少程序平均等待被執行的時間。

若將本任務中的玩具製作機視為電腦的 CPU，每隻等待使用機器的海狸就是一個程序；製作玩具時間最短的海狸可以先使用器機，所以本任務就是最短工件優先的例子。

以班級圖書的借用來說，假設有一本書非常熱門，很多同學登記排隊借閱。通常會依登記時間的早晚，來決定同學借書的先後順序。可是如果某位順位較前的同學閱讀的速度很慢，那排在他後面的同學的平均等待時間就會拉長。如果想縮短同學們平均等待該書籍的時間，就可以讓看書速度最快的同學優先借閱。

關鍵字

排程、最短工件優先

34. 寵物造型師

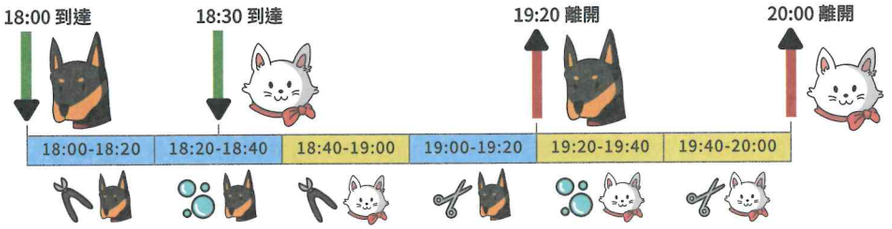
珍狸開了一間寵物美容店。

她將寵物美容分成三項服務：每隻寵物均須依序接受剪指甲、洗澡、和剪毛三項服務（各 20 分鐘）之後，才算完成美容。

為了提升服務品質，珍狸會按照以下規則服務寵物們：

1. 每隻寵物抵達店內，會排在隊伍最末端。
2. 珍狸從隊伍第一隻寵物開始，一次只為一隻寵物進行一項服務 20 分鐘不中斷。
3. 結束一項服務後，珍狸會移到隊伍的下一隻寵物，進行該寵物的下一項服務，並以此類推。
4. 當珍狸結束隊伍最後一隻寵物的一項服務後，會返回到隊伍的第一隻寵物繼續服務。
5. 若結束某隻寵物一項服務時，隊伍中已經沒有其他寵物，珍理就會繼續為同一隻寵物進行下一項服務。
6. 當某隻寵物結束 3 項服務後，就會離開隊伍。

昨天晚上有 2 隻寵物來到珍狸的店：來福 18:00 到達，白白 18:30 到達。下圖顯示珍狸服務 2 隻寵物的時間與順序：



今天早上有 3 隻寵物光臨美容店，各自到達時間如下：

10:00	10:30	11:20
		
小咪	可可	阿金

請問以下哪一個選項是正確的？

A. 小咪在 11:40 離開隊伍

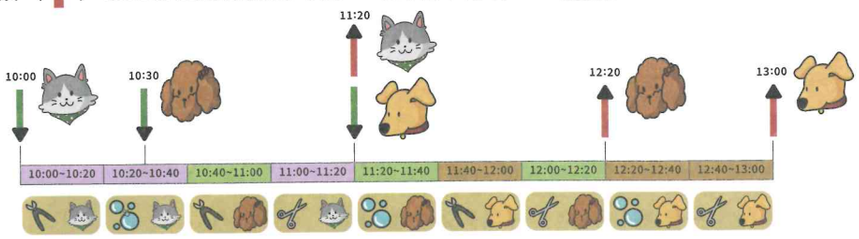
B. 可可在 12:20 離開隊伍

C. 阿金在 12:40 離開隊伍

D. 以上皆非

正確答案是：B

下圖將珍狸的服務順序標示在時間軸上；時間軸上方的綠色線段（↓）指向各寵物抵達的時間，紅色線段（↑）指向各寵物離開的時間，時間軸下方標示出珍狸按照規則為各寵物服務的內容。



從時間軸中可以看出，小咪會在 11:20 離開隊伍，阿金則是在 13:00 離開隊伍。比較需要注意的是 11:20 這一刻：此時小咪剛結束最後一項服務準備要離開，同時阿金剛到達店內。根據規則，珍狸應該為排在隊伍中的下一隻寵物（可可）服務，而阿金排在隊伍的末端，也就是可可的後面。因此，在可可完成第二項服務後，才輪到阿金開始接受第一項服務。

資訊科學上的意義

程序是執行程式時的運作單位，一支程式可能包含許多程序。電腦的中央處理器 (CPU) 一次只能執行一個程序；**排程 scheduling** 就是指安排程序被 CPU 執行的順序，而決定如何排程的方法，就稱作 排程演算法。隨著資訊科技的進步，現在的電腦中通常有多個 CPU 來執行任務。

為了使這些 CPU 發揮最高的運行效率，資訊科學家設計出許多不同的排程演算法，像是先到先服務 First Come First Served 演算法，就是讓 CPU 依進入隊伍的時間順序執行程序，等先進入隊伍的程序執行完成之後，再開始執行隊伍中的下一個程序；而採用 **循環輪流排程 Round Robin Scheduling** 演算法的 CPU 會依序輪流執行各個排隊中的程序，每次都固定執行一段時間；如果該程序在這段時間之內完成，就會離開隊伍，否則就會留在對隊中繼續等待。CPU 會持續循環輪流處理隊伍中的下一個程序，直到所有程序都執行完畢。

若將本任務中的美容師珍狸視為一個 CPU，珍狸服務寵物的順序規則，就是循環輪流排程。循環輪流排程的特別之處，是它確保每一輪的運作，不會持續花費過長的時間處理同個程序。就像在急診室中，醫生會輪流為每位病患進行一部分的療程，好讓每位病患都可以盡快開始接受治療。在這個任務中，可可只等待 10 分鐘就開始美容服務；若採用先到先服務的排程演算法，可可將需要等待 30 分鐘才會開始美容服務。

在學期初常有許多班級活動需要投入很多創意與美編，例如教室佈置、班刊製作、教師節活動策劃等，為了顧及每一個活動的完成度，此時就可以應用循環輪流排程來達成每個任務的階段性成果。

關鍵字

排程、循環輪流排程

35. 蘋果庫存

購物網站上有一個商家只賣蘋果一種商品。

這個商家希望每天一早在網站上公布的蘋果庫存量接近 10 盒。

每天早上 8 點到 8 點 30 分是系統維護時間，暫停販售。商家有 6 個管理者，在這段時間會各自到網站查詢庫存量，並根據以下規則採取行動：

查詢狀況	行動
網站上查到的庫存量 < 10 盒	進貨 1 盒蘋果加到庫存量
網站上查到庫存量 > 10 盒	將庫存量減少 1 盒
網站上查到的庫存量剛好是 10 盒	不做任何事

這 6 個人會各自查詢及採取行動，不會互相聯絡。每個人一上網站，可馬上查詢到網站上記錄的庫存量，但採取行動需要驗證手續，因此 5 分鐘後網站上記錄的庫存量才會被更新。

8 點以前，蘋果庫存量有 9 盒。

6 個管理員分別在以下時間查詢庫存量：

時間	事件
8 點 0 分	小愛查詢庫存量
8 點 4 分	小寶查詢庫存量
8 點 7 分	卡卡查詢庫存量
8 點 8 分	丹丹查詢庫存量
8 點 10 分	琳琳查詢庫存量
8 點 11 分	小蘭查詢庫存量

請問 8 點 30 分開始販售時，網站上記錄的蘋果庫存量有幾盒？

A. 8 盒

B. 9 盒

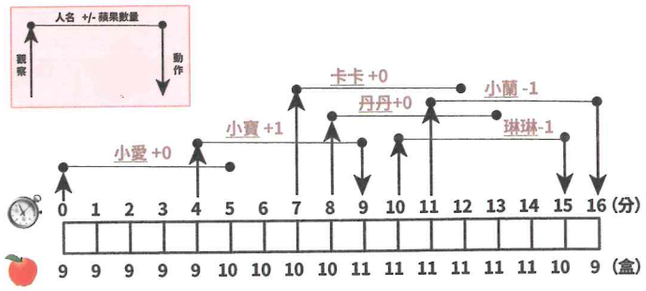
C. 10 盒

D. 11 盒

E. 12 盒

正確答案是：B

我們可以繪製如下圖所示的時間軸，進行庫存量更動計算。時間軸記錄每個時間點由誰進行查詢，以及他們根據查詢結果採取的行動，以及行動結果更新到網站的時間。



在時間軸上方列 ，可以看到 8 點以分為單位的時間。在時間軸下方 列 ，可以看到該時間點的蘋果庫存量數。時間表可以幫助我們更容易看出為什麼在 8 點 16 分後庫存量變成 9 盒。

資訊科學上的意義

分散式系統具有多個優點，這些優點使其在現代計算和網路環境中非常有用。以下是一些主要的分散式系統的優點：高可用性、容錯能力、擴展性、資源共享、更好的安全性等。這個任務涉及一個具有多個獨立參與者（管理者）和共享狀態（蘋果庫存量）的 **分散式系統 distributed system**。分散式系統在設計上如果不小心，很可能會因為某種原因造成的 **競爭條件 race condition** 而做出意料之外的行為。

這個任務很有趣，可以看到多個管理者雖然都努力實現同一個目標（蘋果庫存量接近 10 盒），但最後還是失敗的情況，原因就在於參與者無法互相聯絡形成了分散式系統中，因為時間上的差異讓參與者不同的執行順序與時機，造成了競爭條件而導致的。

要針對分散式系統中的失敗進行除錯是出了名的困難。幸好，有一些經過驗證的演算法（如選舉演算法和時間同步演算法）仍可運作良好。並行控制技術則在資料庫系統中非常重要。還有一些特殊程式語言（例如 TLA+）可以用來確保演算法在分散式系統上按照預期運作。

這個任務屬於演算法的範疇，用於介紹分散式算法概念時相當有用（儘管它是一個故意設計的錯誤演算法），它也可以被當作在分散式環境中進行除錯和評估的練習。

生活上常出現類似分散式系統與競爭條件的情況：例如兩個人想要在最短時間與對方碰面，在沒有良好溝通方式的情況下，若其中一人搭乘左側電梯上樓找對方，另一人卻搭乘右側電梯下樓而錯失碰面的時機，這就形成了所謂的競爭條件。另外諾貝爾經濟學獎得主 John Nash 提出的賽局理論中囚徒困境也是生活中常見的例子。

關鍵字

分散式系統、競爭條件

36. 尋找加密金鑰

咪咪和珠珠設計一種訊息加密法，藉由一組金鑰【數字 , (數列) 】，透過三個步驟就能完成訊息加密。

以下是加密的步驟，並以這組【3, (3, 1, 4, 2)】將 "AMORE CUPID" 訊息加密為例：

1. 依照 " 數字 "，由左至右將訊息分割成 " 數字 " 長度的詞語。若最後的詞語長度小於 " 數字 "，則在詞語尾端添加空格 (用底線來呈現)。

   例如，" 數字 " 為 3，訊息分割後會得到以下詞語：

AMO	RE_	CUP	ID_

2. 接著，反轉各詞語內的順序，將詞語依序編號 (1,2,3,4…)

OMA	_ER	PUC	_DI
1	2	3	4

3. 最後，依照 " 數列 " 將詞語重新排列後，連接詞語，得到密文。

   例如，依照 " 數列 "(3, 1, 4, 2) 重新排序並連結詞語，會得到密文是：PUCOMA DIE R

PUC	OMA	_DI	_ER
3	1	4	2

而 3 和 (3, 1, 4, 2)，這組數字被稱為金鑰。

珠珠將 "I LOVE YOU MIMI" 轉換為密文，並發送給咪咪。
如果咪咪收到的密文是 "Y EV IMIM UOOL I"，請問金鑰是什麼？

A. 2, (4, 3, 8, 7, 5, 6, 1, 2)

B. 3, (3, 2, 5, 4, 1)

C. 4, (2, 4, 3, 1)

D. 5, (2, 3, 1)

正確答案是：C

我們可以利用選項的金鑰，將訊息 "I LOVE YOU MIMI"，逐一轉換為密文，就可以得到答案。

- 選項 A　2, (4, 3, 8, 7, 5, 6, 1, 2)

  "I _", "LO", "VE", " _ Y", "OU", " _ M", "IM", "I _"

  ⇒ " _ I", "OL", "EV", "Y _", "UO", "M _", "MI", " _ I"

  ⇒ "Y _", "EV", " _ I", "MI", "UO", "M _", " _ I", "OL"

  ⇒ Y EV IMIUOM IOL

- 選項 B　3, (3, 2, 5, 4, 1)

  "I _ L", "OVE", " _ YO", "U _ M", "IMI"

  ⇒ "L _ I", "EVO", "OY _", "M _ U", "IMI"

  ⇒ "OY _", "EVO", "IMI", "M _ U", "L _ I"

  ⇒ OY EVOIMIM UL I

- 選項 C　4, (2, 4, 3, 1)

  "I _ LO", "VE _ Y", "OU _ M", "IMI _"

  ⇒ "OL _ I", "Y _ EV", "M _ UO", " _ IMI"

  ⇒ "Y _ EV", " _ IMI", "M _ UO", "OL _ I"

  ⇒ Y EV IMIM UOOL I

- 選項 D　5, (2, 3, 1)

  "I _ LOV", "E _ YOU", " _ MIMI"

  ⇒ "VOL _ I", "UOY _ E", "IMIM _"

  ⇒ "UOY _ E", "IMIM _", "VOL _ I"

  ⇒ UOY EIMIM VOL I

另一種解決方案是取密文 "Y EV IMIM UOOL I"，使用金鑰找出原始訊息。

選項 (B) 和 選項 (D) 是不正確的，因為將密文依照第一個數字切割為更短的詞語後，詞語的總個數，會比金鑰的序列個數多一個。

例如：選項 (B) 3, (3, 2, 5, 4, 1) → 序列個數 5

"Y_E", "V_I", "MIM", "_UO", "OL_", "I__" → 詞語總個數為 6，兩者個數不相符，故答案不正確。

對於選項 (A) 和 選項 (C) 來說，即使不重新排列所有詞語，也可以透過檢查第一個詞語，很快找出答案是選項 (C)。

- 選項 A　2, (4, 3, 8, 7, 5, 6, 1, 2)

  "Y _", "EV", " _ I", " MI", "M _", "UO", "OL", " _ I"

  ⇒ " _ Y", "VE", "I _", "IM", " _ M", "OU", "LO", "I _"

  ⇒ "LO", "I _", "VE", " _ Y", " _ M", "OU", "IM", "I _"

  ⇒ LOI VE Y MOUIMI

- 選項 B　3, (3, 2, 5, 4, 1)

  "Y _ E", "V _ I", "MIM", " _ UO", "OL _", "I _ _"

  ⇒ "E _ Y", "I _ V", "MIM", "OU _", " _ LO", " _ _ I"

  ⇒ " _ LO", "I _ V", "E _ Y", "OU _", "MIM"

  ⇒ LOI VE YOU MIM

- 選項 C　4, (2, 4, 3, 1)

  "Y _ EV", " _ IMI", "M _ UO", "OL _ I"

  ⇒ "VE _ Y", "IMI _", "OU _ M", "I _ LO"

  ⇒ "I _ LO", "VE _ Y", "OU _ M", "IMI _"

  ⇒ I LOVE YOU MIMI

- 選項 D　5, (2, 3, 1)

  "Y _ EV _", "IMIM _", "UOOL _", "I _ _ _ _"

  ⇒ " _ VE _ Y", " _ MIMI", " _ LOOU", " _ _ _ _ I"

  ⇒ " _ LOOU", " _ VE _ Y", " _ MIMI"

  ⇒ LOOU VE Y MIMI

資訊科學上的意義

密碼學 cryptography 是研究如何使訊息安全地被傳遞與接收的學問，通常利用數學方法來對資料加密和解密。**加密 encryption** 則是將原始訊息重編為不同形式的密文。加密方法有很多種，使用金鑰進行加密就是其中一種。

金鑰 key 是指某組用以進行加密、解密、驗證完整性的資訊。在此任務中「AMOR ECUPID」是原始訊息，「PUCOMA DIE R」是加密後的資料，「3 和 (3, 1, 4, 2)」則是金鑰。由於加密、解密都是使用 3 和 (3, 1, 4, 2) 這一組金鑰，這種方式又稱為對稱式加密 symmetric encryption。另外還有非對稱式加密 Asymmetric encrypt　ion。

網路上大部份的加密方式都是使用對稱式加密，它就像我們替大門上鎖、解鎖都用同一把鑰匙一樣的意思，若是金鑰被別人取得了，危險性就會增加；非對稱式加密則是將金鑰分為公開金鑰與私密金鑰，應用在數位簽章、區塊鏈等必須兼顧公開性與安全性的技術上。

關鍵字

加密金鑰、解密

37. 電子鎖

海狸小巴在門上安裝了一個電子密碼鎖。

要打開它，必須按照正確的順序按下幾個不同的數字按鈕。

原始密碼是：

0	2	4	3	1

為了增加密碼安全性，他依照以下規則重新編碼

" Y>>X " 表示在數字 X 的左邊，恰好有 Y 個比 X 大的數字。

0>>0, 0>>0, 3>>1, 0>>2, 1>>3, 0>>4

" 0>>0, 0>>2 和 0>>4 " 表示密碼中的數字 0、2、4 左邊都沒有更大的數字。

" 3>>1 " 表示在數字 1 的左邊有三個更大的數字，它們是 2、3、4。

而 " 1>>3 " 表示在數字 3 的左邊有一個更大的數字，它是 4。

小巴將密碼改為更複雜的 8 位數，編碼為：

3>>0，2>>1，4>>2，4>>3，1>>4，1>>5，1>>6，0>>7。

小巴的新密碼是什麼？

A. 70145623

B. 74105623

C. 74165320

D. 74105326

正確答案是：B

7	4	1	0	5	6	2	3

將所有可能的 n! 選項列舉出來，而得到正確密碼，是不可行的（數字有可能很大）。

本任務快速的解法是，從最小的數字（數字 0）開始，填到最大的數字（數字 7），依序填在它們應該放置的空格內。

1. 從 3>>0 可以得出，數字 0 左邊應該有三個空格。也就是說，數字 0 一定是放在左邊數來的第四個格子。

			0				

2. 從 2>>1 可以得出，數字 1 左邊應該有兩個空格。也就是說，數字 1 必須放在第三個格子中。

		1	0				

3. 從 4>>2 可以得出，數字 2 左邊應該有四個空格（不包含 0 和 1 的位置）。也就是說，數字 2 需放在第七個格子中：

		1	0			2	

4. 從 4>>3 可以得出，數字 3 左邊應該有四個空格（不包含 0、1 和 2 的位置）。也就是說，數字 3 需放在第八個格子中：

		1	0			2	3

其實解到這裡就可知道答案為選項 B，以此類推，依序放置剩餘數字，就能得到唯一解。

資訊科學上的意義

本任務包含了「密碼學加密」與「組合學解密」兩個主題。**密碼學 cryptography** 是研究如何將訊息安全地被傳遞與接收；**組合學 combinatorics** 也稱為組合數學，則是研究計數、排列、機率和運算等問題。在本任務中，電子鎖密碼被重新編寫為不同形式的密文，應用了密碼學中加密的技術；而從加密後的線索列出所有可能的組合，並且從中找出符合條件的正確解答，這是組合學領域研究的課題。組合學與密碼學都是數學重要理論，也是近代推動資訊科學、電機通訊、資通安全領域發展的重要力量。

關鍵字

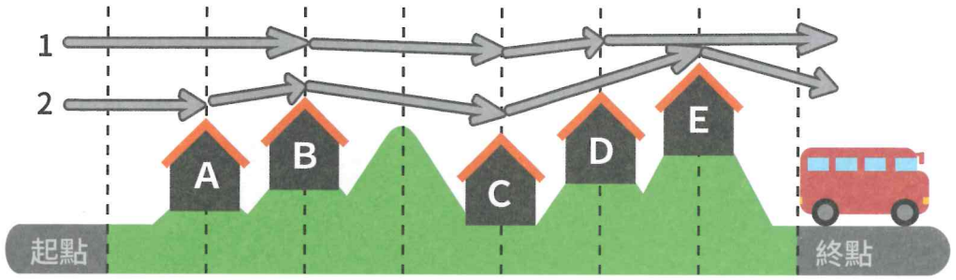
加密、解密

38. 徒步旅行

米亞喜歡規劃假日的徒步旅行，並將行程畫成圖。如下圖標示了行程的起點、終點，以及她可以在晚上住宿的地方，並在圖上用虛線分隔步行路段。

米亞每天只能步行一個或兩個路段，並且每晚要住在不同的地方。

圖中顯示了兩條可能的路線： 路線 1 有 3 個晚上住宿點。 路線 2 有 4 個晚上住宿點。



請問米亞在徒步旅行中有幾種不同的路線可選擇？（包含圖中顯示的兩條路線）

A. 4 種

B. 5 種

C. 6 種

D. 7 種

E. 8 種

正確答案是：C

首先，我們可以透過「分解」來解決問題：我們以米亞每天可步行的最長距離（兩個步行路段）來分解。我們可以將完整行程拆解成三個較小的行程：起點至 B、B 至 C、C 至終點。

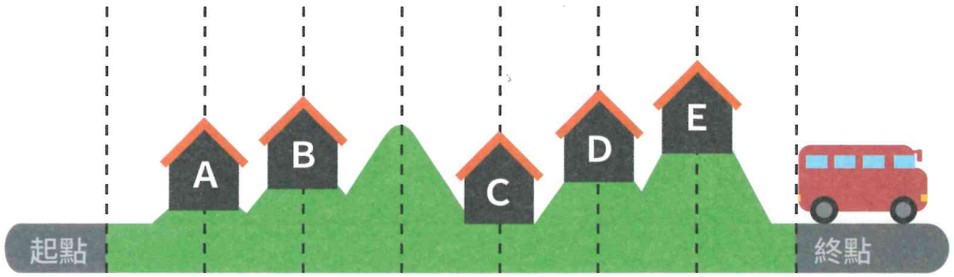
分別來看這三個行程的可能路線：

(1) 起點至 B：有兩種路線（即任務顯示的路線 1 和路線 2）

(2)B 至 C：只有一種路線

(3)C 至終點：有三種路線（C → D → E → 終點、C → E → 終點、C → D → 終點）

因此總共有 2 X 1 X 3 = 6 種路線。



資訊科學上的意義

組合學 combinatorics 也稱為組合數學，內容包含研究計數、排列、機率和運算等問題。組合學在資訊科學中特別有用，可用於演算法分析。

本任務要找出米亞在徒步旅行中有多少種不同的路線，可以透過分解為較小的問題來解決任務，最後將可能的路線排列組合，完成解題，這是屬於組合學的概念。

組合學在資訊科學、物理、化學、生物等學科均有重要的應用，資訊方面主要應用於圖論、編碼、密碼學與網路安全等領域。

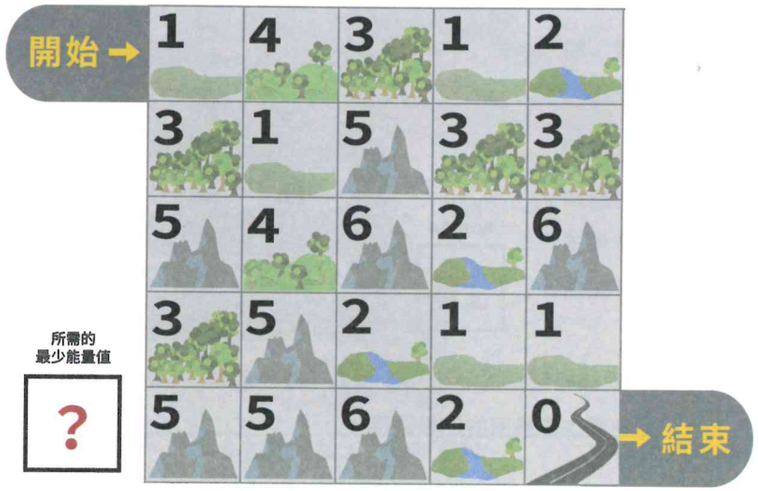
關鍵字

組合學

39. 棋盤遊戲

小狸在玩棋盤遊戲 (如下圖)，遊戲需要從棋盤左上角的「開始」移動至右下角的「結束」。棋盤上每個方格內的數字表示小狸經過該方格時需使用的能量值。

移動過程中小狸只能「往右」或「往下」移動。

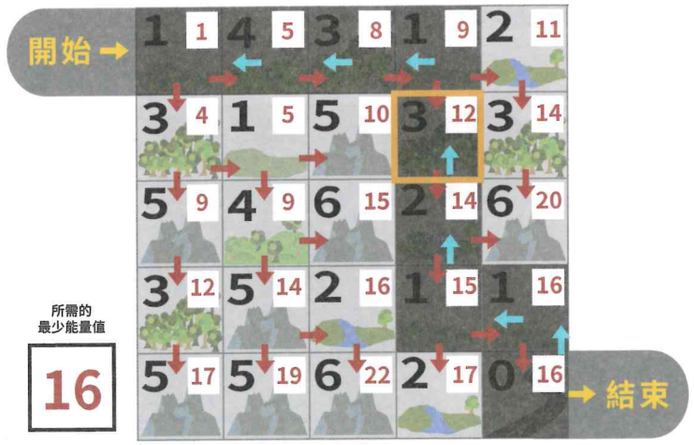


小狸想要用最少的能量值完成遊戲 (如上圖)，請問能量值最少為何？(範圍為 [0~30] 的整數。)

正確答案是：16

由於小狸只能「往右」或「往下」移動，因此對於棋盤上的任何一個方格而言，必定是從上方或左側的方格而來。所以我們只要檢查某方格的上方或左側，哪一方所需的能量較少，就可以確定到達該方格的最佳路徑。我們可以在每一格區塊，計算小狸從「開始」一路走到該格，所使用的能量值。每一格的能量最小值的計算原則如下：

- 如果該格只能由「某一格」走到，則該格的能量最小值，只可能是由走到「某一格」的能量最小值，加上該格所需能量值的和。
- 如果該格可以從上方或左側其中一格走到，則選擇能走到這兩格的能量值的最小值並加上該格所需能量值的和。



上圖中的紅色數字代表走到該格所需的能量和最小值；紅色箭頭代表該格的最佳路徑來自哪個方向。藍色箭頭代表可以得到最少能量值的回推路徑。

資訊科學上的意義

本任務使用 **動態規劃 dynamic programming, DP** 的概念來解決最佳化問題。 動態規劃是將複雜問題分解為較簡單的子問題，若子問題非常相似，則只需解決每個子問題一次。並運用記憶法儲存這些子問題的答案，避免重複求解，從而減少計算量，再根據各子問題的解來找出原問題的解。

在此任務中，問題指的是從棋盤左上角的「開始」移動至右下角的「結束」，要用最少的能量值完成遊戲。訣竅是在每一格區塊，計算小狸從「開始」一路走到該格，所使用的能量最小值。 當推導到右下角「結束」時，我們即得到了最後的結果，並能回推最少能量值的路徑。

近年來，資訊科學家設計各種演算法，協助有效率尋找圖中的最短路徑。其中一種演算法由荷蘭電腦科學家 Edsger Dijkstra 所發佈，現在被稱為 Dijkstra 演算法。這種演算法可以在點與點間找出最短距離，用以解決生活中常見的問題，例如導航系統，或是盡可能使用較少燃油消耗起降的客機等，都屬於路徑最佳化應用。

關鍵字

動態規劃

40. 字詞變換

小換、小刪和小加每天起床就一起到河邊排字母牌，排完後離開，如下圖。

小刪總是刪除一個字母牌。

小加總是加入一個字母牌。

小換總是用一個新的字母牌替換其中一個字母牌。

今天早上排好的單字是 "LEAVES"，到了晚上變成了 "BLUEBERRIES"。

請問今天動物們合計最少回到河邊幾次？

A. 11 次

B. 8 次

C. 7 次

D. 6 次

正確答案是：C

利用下表比較早上單字（α）與晚上單字（β）的變化，可以發現表格中有 4 欄的字母相同（藍色底表示），另有 7 欄的字母不同（黃色底表示）。

α		L				E	A	V		E	S
執行的動作	+		+	+	+		↔	↔	+		
β	B	L	U	E	B	E	R	R	I	E	S

由此推論動物們應該至少到河邊 7 次 刪除 (-)、加入 (+)、或替換（↔）字母牌。只是若要確定 **7 次是最少的次數，就要進一步驗證。**

若要驗證動物們回到河邊的**最少**次數，可以透過下方表（d），從單字最前面的字母開始，逐一計算並紀錄將 α 的前 i 個字母變成 的前 j 個字母時，動物回到河邊的最少次數。

	""	B	L	U	E	B	E	R	R	I	E	S
""												
L												
E												
A												
V												
E												
S												

詳細的步驟如下：

1. 如下表，在第一行的黃色格子中填入「把 α [1] 到 α [i] 刪除」的最少次數。例如：早上河邊排著 L 時，動物最少要回到河邊 1 次，才能刪除河邊所有字母牌；早上河邊排著 LE 時，動物最少要回到河邊 2 次，以此類推。同樣的，在第一列的紅色格子中填入「在沒有字母牌的河邊加入 [1] 到 [j] 」的最少次數。例如：若早上河邊沒有字母牌，動物最少要回到河邊 3 次，才能加入 BLU 字牌；動物最少要回到河邊 4 次，才能加入 BLUE，以此類推。

	""	B	L	U	E	B	E	R	R	I	E	S
""	0	1	2	3	4	5	6	7	8	9	10	11
L	1											
E	2											
A	3											
V	4											
E	5											
S	6											

而藍色格子中的 0 則代表河邊的早上和晚上都沒有字母的話，動物最少回到河邊 0 次。

2. 接下來便分別由每一格（$d[i][j]$）的左上方、上方及左方推算 的前 i 個字母變成 的前 j 個字母時，動物回到河邊的次數，並把三者中的最少次數紀錄下來，推算方式如下：

- **左方格（$d[i][j-1]$）的值加 1**

  左方格中的值是把 的前 i 個字母變成 的前 j-1 個字母時，動物回到河邊的最少次數。在這之後小加再回到河邊加入 的第 j 個字母，因此從左方格推算就是左方格的值加 1。

- **上方格（$d[i-1][j]$）的值加 1**

  上方格中的值是把 的前 i-1 個字母變成 的前 j 個字母時，動物回到河邊的最少次數。在這之後小刪再回到河邊刪除 的第 i 個字母，因此從上方格推算就是上方格的值加 1。

- **左上方格**

  左上格中的值是把 的前 i-1 個字母變成 的前 j-1 個字母時，動物回到河邊的最少次數。在這之後，若是 的第 i 個字母（[i]）和 的第 j 個字母（[j]）不同時，小替會再回到河邊（次數 +1）將 [i] 替換成 [j] ；若是 [i] 和 [j] 相同時，動物就不會再回到河邊。因此從左上方格推算會有兩種可能：

  ＊ $d[i-1][j-1]+1$　　**（當 α[i] ≠ β[j] 時，表示小替須回到河邊）**

  ＊ $d[i-1][j-1]$　　**（當 α[i] = β[j] 時）**

以下表為例，因為 L 和 B 不相同，因此綠色格子中會填入 1 + 1 、 1 + 1 、或 0 + 1 的最小值，也就是 1 。

	""	B	L	U	E	B	E	R	R	I	E	S
""	0	1	2	3	4	5	6	7	8	9	10	11
L	1	*min*(1+1,1+1,0+1)=1										
E	2											
A	3											
V	4											
E	5											
S	6											

再以下表為例，因為 L 和 L 相同，因此綠色格子中會填入 1 + 1 、 2 + 1 、或 1 的最小值，也就是 1 。

	""	B	L	U	E	B	E	R	R	I	E	S
""	0	1	2	3	4	5	6	7	8	9	10	11
L	1	*min*(1+1,2+1,1)=1										
E	2											
A	3											
V	4											
E	5											
S	6											

3. 逐一將表格填滿之後，表格右下角格子的數值就是今天動物回到河邊的最少次數，也就是 7 次。

	""	B	L	U	E	B	E	R	R	I	E	S
""	**0**	**1**	2	3	4	5	6	7	8	9	10	11
L	1	1	**1**	**2**	**3**	**4**	5	6	7	8	9	10
E	2	2	2	2	2	3	**4**	**5**	6	7	8	9
A	3	3	3	3	3	3	4	5	**6**	7	8	9
V	4	4	4	4	4	4	4	5	6	**7**	8	9
E	5	5	5	5	4	5	4	5	6	7	**7**	8
S	6	6	6	6	5	5	5	5	6	7	8	**7**

資訊科學上的意義

編輯距離 Edit distance 是一種量化兩字串間差異的演算法，代表從一個字串轉換為另一個字串，至少需要經過多少次的處理。

萊文斯坦距離 Levenshtein distance 是以俄羅斯數學家 Vladimir Levenshtein 命名（1935 - 2017 年），在字串距離量測時，可以刪除、加入、取代字串中的任何一個字元。在本任務中每隻動物（小刪、小加和小換），都代表了一個允許的編輯操作。萊文斯坦距離概念容易理解，實作簡單，為最具代表性的編輯距離演算法。

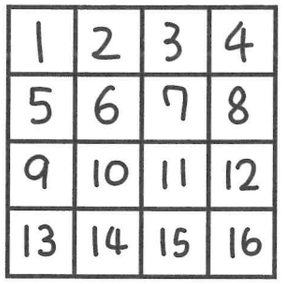
而編輯距離演算法，常用在簡易拼字修正與建議、自動更正或光學字元辨識等應用。

關鍵字

編輯距離、萊文斯坦距離、電腦語言學

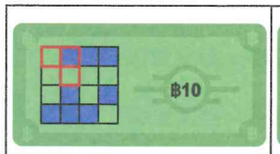
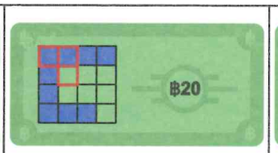
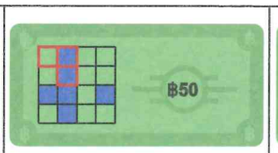
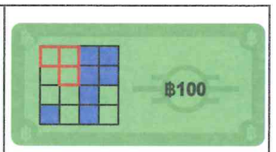
41. 紙鈔辨識

海狸國的紙鈔上，都會印有一個方塊條碼，方塊條碼包括 16 個方塊，方塊標號如下所示。



每個方塊可以塗色或是空白。

藉由紙鈔方塊條碼中標號 1、2、6 位置的方塊是否塗色，自動販賣機可以區分四種面值的紙鈔，如下所示：

			
1：空白 2：塗色 6：空白 ⇨ ฿10	1：塗色 2：塗色 6：空白 ⇨ ฿20	1：空白 2：塗色 6：塗色 ⇨ ฿50	1：空白 2：空白 6：空白 ⇨ ฿100

小狸發現其實只需觀察方塊條碼中特定兩個方塊是否塗色，就可以區分出 4 種面值。請問是哪兩個標號的方塊？

A. 1, 2

B. 9, 10

C. 10, 12

D. 14, 16

正確答案是：B

	฿10	฿20	฿50	฿100
選項 A　1, 2	1 2	1 2	1 2	1 2
選項 B　9, 10	9 10	9 10	9 10	9 10
選項 C　10, 12	10 12	10 12	10 12	10 12
選項 D　14, 16	14 16	14 16	14 16	14 16

- 選項 A：觀察四種面值的方塊條碼中方塊 1 與 2 的塗色方式，฿10 和 ฿50 的塗色方式相同，因此無法區分 ฿10 和 ฿50。
- 選項 B：四種面值的方塊條碼在方塊 9 和 10 的塗色方式皆不同，可以區分四種面值。
- 選項 C：觀察方塊 10 和 12 的塗色方式，฿10 和 ฿50 有相同的塗色方式，因此無法區分。฿20 和 100 有相同的塗色方式，因此也無法區分。
- 選項 D：觀察方塊 14 和 16 的塗色方式，฿10、฿20 和 ฿50 皆有相同的塗色方式，因此無法區分這三種面值。

因此只有選項 B 是正確答案。

資訊科學上的意義

在機器學習和模式識別中，**特徵**是指被觀測資料的可測量數值或特性。舉例來說，水果的顏色跟形狀都可以是描述它們的特徵。**模式辨識 pattern recognition** 是一項在電腦中用於處理資料的一種技術，其主要功能是辨識和理解資料中的規律和相似 / 相異處，使電腦能夠對未知數據進行自動識別。舉例來說，當提供多個蘋果跟橘子，用來分辨蘋果和橘子，顏色跟形狀都可以是特徵，不過可以發現以顏色當作辨識的特徵，會比用形狀容易區分兩者。

挑選能有效識別資料特徵來進行辨識，就是本任務需用到的概念：找出辨別紙鈔要觀察哪幾個方塊 (特徵) 的塗色方式。模式辨識技術的應用在我們日常生活中十分常見，例如文字、語音和指紋指紋辨識等。

關鍵字

模式辨識

42. 可愛生物

小狸想要分析「可愛生物」的長相，牠研究了以下蒐集的 5 張照片，並寫下觀察到的共同特徵。

但是當小狸蒐集到第 6 張照片時，發現之前寫下的特徵描述中，有一條不成立了。

請問是哪一條特徵描述變為不成立了？

A.　「可愛生物」都會有牙齒。

B.　「可愛生物」有些會長翅膀。

C.　「可愛生物」會有角或有 3 隻眼睛，但不會同時有這 2 個特徵。

D.　有 2 隻手臂的「可愛生物」，一定剛好有 2 條腿。

正確答案是：D

第 6 張照片顯示 1 隻有 2 隻手臂和 5 條腿的「可愛生物」，因此有 2 隻手臂的「可愛生物」並不代表牠剛好有 2 條腿。選項 D 的特徵敘述變為不成立。

選項 A 的特徵描述仍然成立：所有 6 張照片都顯示「可愛生物」有牙齒。

選項 B 的特徵描述是成立的：第 2 張照片顯示 1 隻有翅膀的「可愛生物」。

選項 C 的特徵描述仍然成立：第 1、第 2、第 5 和第 6 張照片顯示「可愛生物」有角，但「只有」2 隻眼睛。第 3 張和第 4 張照片顯示「可愛生物」有 3 隻眼睛但沒有角。

資訊科學上的意義

在電腦中資料是以 0 或 1 來表示，而邏輯運算是資訊科學中的重要運算基礎。在邏輯的世界中用 1 表示「**真** (true)」，0 表示**假**。一個能辨別真假的語句稱為一個**敘述**，敘述成立稱為「真 (true)」，不成立就稱為「假 (false)」。例如：「狗有 4 隻腳」這個敘述根據為真，「貓會生蛋」為假。運用邏輯運算子「且 (and)」、「或 (or)」、「否定 (not)」，可以形成複雜的邏輯敘述；例如「狗是胎生且貓有 4 隻腳」。

若敘述用在多筆資料上，判斷真假就要加上範圍限制是「對所有」都要成立，或是「存在一個」成立就可以。此任務中提供了一些可愛生物的圖案，每個選項的特徵描述就是邏輯敘述，可由給定的圖案判斷敘述是「真」或「假」(不成立)。

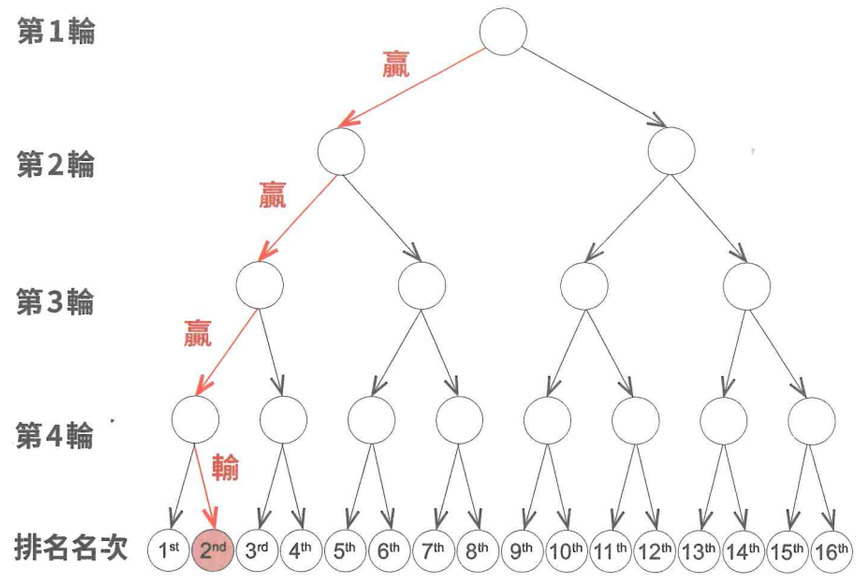
邏輯推論是進一步從一些已知的事實進行條件判斷，並推論出結果。例如：知道「如果我要出門時正在下雨，就會帶傘。」，又知道「我昨天要上學時正在下雨。」就可以推論出：「我昨天有帶傘。」

關鍵字

邏輯敘述、邏輯推理

43. 海狸球賽

如下圖，16 隻海狸將進行 4 輪比賽來決定排名。每場比賽都是一群海狸一同比賽；依比賽 贏 或輸 的結果分成兩群，這兩群球員再分別進行下一輪的比賽，直到分出所有名次為止。



例如，小海贏了第 1 和 2 輪比賽，但輸了第 3 和 4 輪比賽，最後得到第 4 名。

小狸在 4 輪比賽中，只輸了其中 1 輪，請問他不可能得到哪個名次？

A. 第 2 名

B. 第 3 名

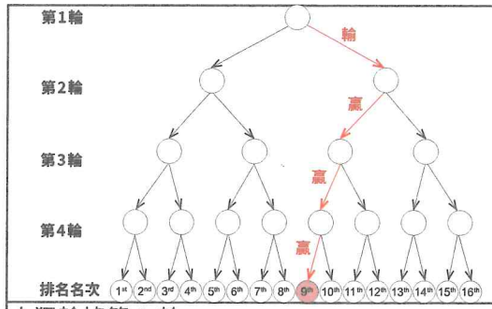
C. 第 5 名

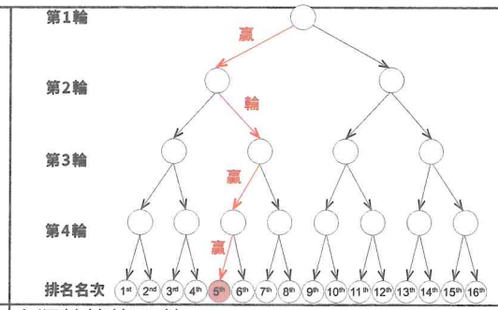
D. 第 7 名

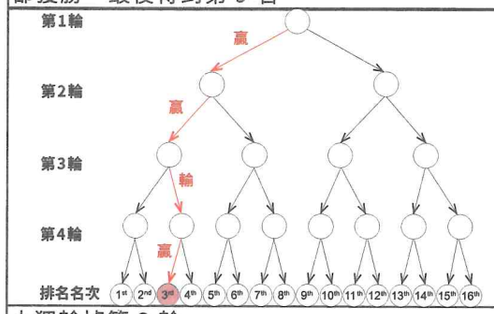
E. 第 9 名

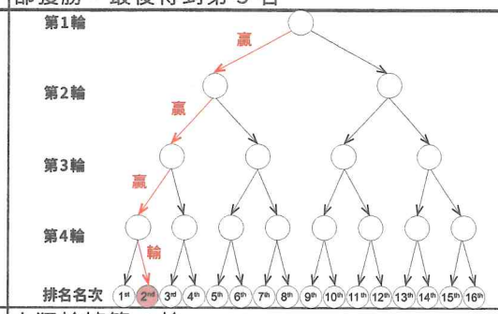
正確答案是：D

比賽共有 4 輪，而小狸只在其中 1 輪是輸家，因此可以分成以下 4 種情況：

<table>
<tr>
<td> 

小狸輸掉第 1 輪
如下圖所示，若小狸只輸了第 1 輪，其餘比賽都獲勝，最後得到第 9 名。</td>
<td> 

小狸輸掉第 2 輪
如下圖所示，若小狸只輸了第 2 輪，其餘比賽都獲勝，最後得到第 5 名</td>
</tr>
<tr>
<td> 

小狸輸掉第 3 輪
如下圖所示，若小狸只輸了第 3 輪，其餘比賽都獲勝，最後得到第 3 名。</td>
<td> 

小狸輸掉第 4 輪
如下圖所示，若小狸只輸了第 4 輪，其餘比賽都獲勝，最後得到第 2 名。</td>
</tr>
</table>

綜上所述，小狸可能得到的名次為：第 9 名、第 5 名、第 3 名、第 2 名。故小狸不可能得到第 7 名。

資訊科學上的意義

在電腦的世界中，數字是採用二進位表示。我們一般使用的十進位制表示法，有 0 至 9 共 10 種數字，每數到 10 就會進位。二進位則只使用 0 和 1 兩個數字，每數到 2 就會進一位，因此二進位制表示法由一串 0 與 1 表示，其中每一位稱為一個位元 (bit)。在此任務中，每一輪比賽結果會依「贏」或「輸」而將海狸分成兩堆，這跟二進制依位元由高而低進行大小比較是類似的。若 4 個位元的二進位數第一位元是 1，就比第一位元是 0 的數大。「贏」往左邊分支，而「輸」則往右邊分支，再進行下一輪比賽，也就是下一位元比大小。經過 4 個回合，就可以對所有 4 個位元的二進位數排出大小順序。因此 4 個位元最大的二進位數為 1111，比較後為第一名 (最大)。

關鍵字

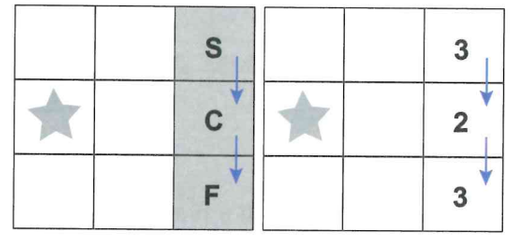
二進位表示

44. 更近或更遠

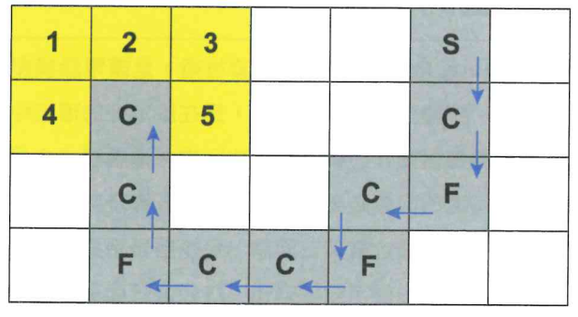
小明在玩尋寶遊戲，要找出寶藏在方格藏寶圖中的確切位置。

小明從方格 S 出發，每步只能水平或垂直移動到相鄰方格。每次移動一步後，小明將收到更接近（C）或更遠離（F）寶藏的提示訊息，以下所說的「距離」表示小明所在方格位置到寶藏所須移動的最短步數。

例如，在下圖 3x3 方格藏寶圖中，寶藏在標記 ★ 的方格處。小明依照左圖箭頭方向前進兩步，右圖顯示他所在位置到寶藏的距離。小明在每次移動一步後收到“C”（更接近）和“F”（更遠離）的提示訊息。



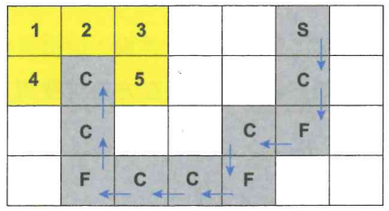
現在，小明得到另一個 4x7 的方格藏寶圖，寶藏就埋在編號 1~5 其中之一的位置下，若按照箭頭路徑前進，將同時顯示距離提示訊息如下圖。

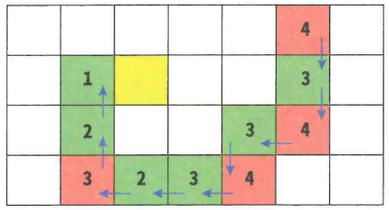


請問寶藏在哪個編號方格之下？(範圍為 [1~5] 之間的整數)

正確答案是：5

透過在方格藏寶圖上寫下每一個方格到方格 5 的距離，來檢查提示訊息 "C" 和 "F" 的正確性。從下圖中可以看到依照箭頭方向移動，到方格 5 的距離增加則標示為 "F"；到方格 5 的距離減少則標示為 "C"。我們得出的提示訊息順序與方格藏寶圖提供的結果一致。因此，寶藏被放在方格 5 處。

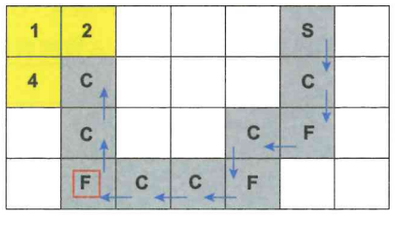




我們可以透過檢查方格藏寶圖上的線索，發現寶藏放在其他位置可能性的錯誤提示：

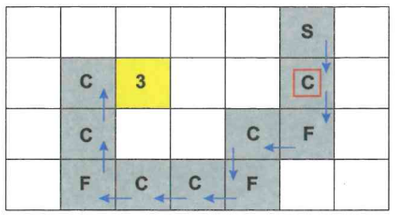
1. 對於寶藏放在方格 1、2 或 4 的可能性評估：

   如果寶藏被埋在方格 1、2 或 4 其中之一處，那麼第 4 列第 2 行的提示訊息 "F" 必須是 "C"，因為該方格比它前一步所在位置（即第 4 列第 3 行）更接近寶藏（位於方格 1、2 或 4 其中之一）。



2. 對於寶藏放在方格 3 的可能性評估：

   如果寶藏被埋在方格 3 下，那麼第 2 列第 6 行的提示訊息必須是 "F"，因為該位置比小明前一步所在位置（即第 1 列第 6 行）離寶藏更遠。



資訊科學上的意義

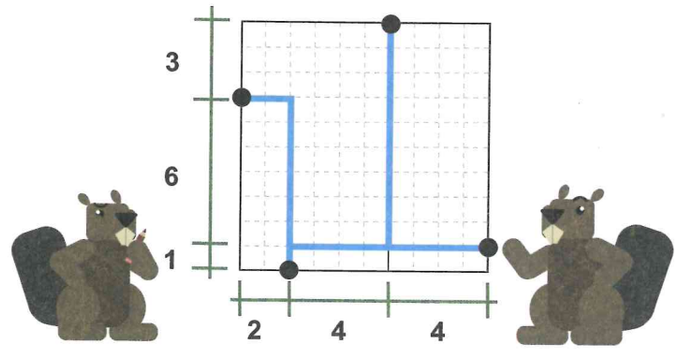
強化學習 Reinforcement learning, RL 是一種機器學習技術，是讓電腦模擬人類由得到獎勵或懲罰的過程，建立行為習慣的判斷。這種技術的學習目標，是在設定好的環境中採取行動，學習如何讓累積的利益最大化。訓練過程的每個動作必須設有正負回饋，電腦程式會根據不同的環境狀況嘗試各種決定，再根據此決定行動得到的回饋結果，調整內部的計算及判斷策略。本任務中，每次移動到新位置會收到提示距離的訊號，更接近表示正回饋，更遠離表示負回饋。為了替下一步移動做出最佳決策，必須考慮先前及目前方格的提示訊號變化，這類似於強化學習中的價值函數。例如貪食蛇遊戲，遊戲中的角色如果碰到障礙物會減少生命值，這就是一種負回饋。根據負回饋，玩家會嘗試避開障礙物，最後就可能學習到破關方式及取得高分的操作策略。此外本任務中，方格之間的距離是透過**曼哈頓距離 Manhattan distance**，玩家只能水平或垂直移動，與一般經常採用的歐氏距離 (又稱直線距離) 的計算方式不同。

關鍵字

強化學習、曼哈頓距離

45. 下水道規劃

海狸國規定：挖掘下水道必須沿垂直或平行於房屋的方向。莫莫須挖掘連結四間房屋的下水道；他規劃的下水道如下圖所示，黑色圓點表示房屋所在地，藍色線條表示他預計挖掘的下水道。



為了節省體力，挖掘下水道的總長度愈短愈好；以莫莫目前的規劃，下水道的總長度為 26 單位。

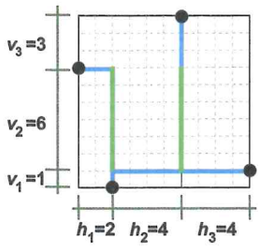
請問連結這四間房屋的下水道總長度最短是幾個單位？(範圍為 [20~25] 的整數)

正確答案是：24

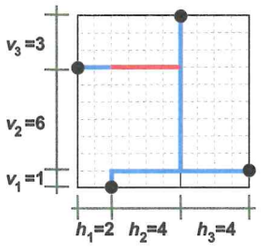
根據房子的相對位置，下水道至少應該包含水平距離 $h_1 + h_2 + h_3 = 2+4+4 = 10$ 和垂直距離 $v_1 + v_2 + v_3 = 1+6+3 = 10$。因此，所有下水道的總長度最短一定大於或等於 20。而最短長度若要等於 20，那麼房屋必須分別排列在同一條水平線和垂直線上。

由圖一莫莫的設計圖可以發現，v_2 距離的下水道被建造了兩段，因此下水道總長度為 20 + 6 = 26。若要找到長度最短的下水道規劃，可以將其中一段 v_2 距離的下水道，換成圖二中 h_2 距離的下水道；如此一來，下水道總長即可縮減為 20 + 4 = 24。

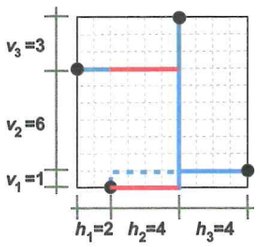
如圖三及圖四所示，若考慮以 v_1、v_3、h_1 或 h_3 來取代 v_2，都沒辦法只建造一條下水道就將四棟房屋連接起來，反而可能會讓路徑總和比 26 個單位長還長。因此，下水道的最短總長度就如圖二所示，是 24 個單位。



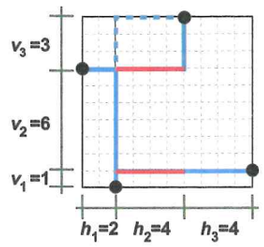
圖一



圖二



圖三



圖四

資訊科學上的意義

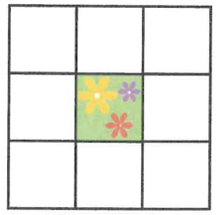
計算幾何 computational geometry 是資訊科學的一個分支，專門研究解決幾何問題的演算法。此任務中使用到的正是計算幾何問題之一的直線史戴那樹問題。**直線史戴那樹問題 rectilinear Steiner tree problem** 是在直角座標系中找出一棵樹 tree，這棵樹由若干指定點 node（此任務中的房屋）和連接點的水平或垂直方向邊 edge（下水道）組成，並希望樹的總邊長越短越好。直線史戴那樹問題可運用於超大型積體電路 (VLSI) 設計，由於線路需連接上千個元件，且線路通常被限制只能是水平或垂直方向，如何規劃才讓線路總長度最短是個相當困難的問題。

關鍵字

計算幾何、直線史戴那樹

46. 生命遊戲

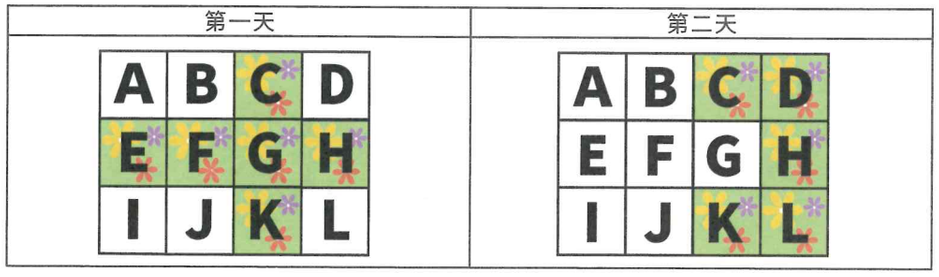
在海狸國裡有一個方格花園，有顏色的方格種了花朵，而白色方格則沒有。每個方格周圍有 8 個鄰近的方格：



花朵生長的規則如下：

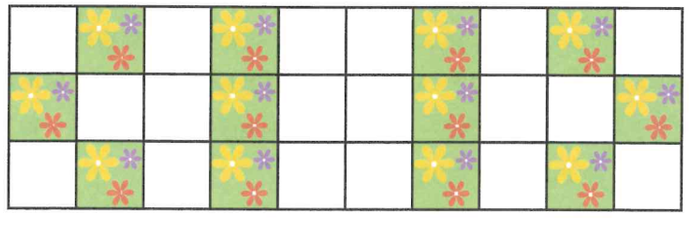
1. 如果有顏色的方格周圍少於 2 個有顏色的鄰近方格，它將在第二天變成白色。
2. 如果有顏色的方格周圍有 2 個或 3 個有顏色的鄰近方格，它將在第二天保持有顏色。
3. 如果有顏色的方格周圍有多於 3 個有顏色的鄰近方格，它將在第二天變成白色。
4. 如果白色的方格周圍恰好有 3 個有顏色的鄰近方格，它將在第二天變成有顏色。

範例：



- E 格的顏色，因為規則 1 而在第二天變成白色。
- F、G 兩格的顏色，因為規則 3 而在第二天變成白色。
- C、H、K 三格的顏色，因為規則 2 而保持有顏色到第二天。
- D、L 兩個白色方格，因為規則 4 而在第二天變成有顏色。

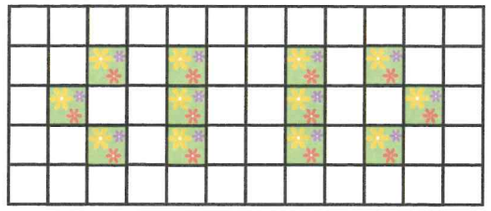
以下是第一天的花園情況（這個花園的外圍並無有顏色方格）：



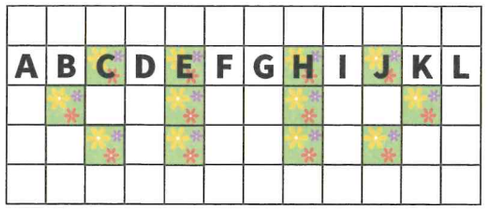
第二天會有多少個有顏色的方格？（範圍為 [0~30] 的整數）

正確答案是：12

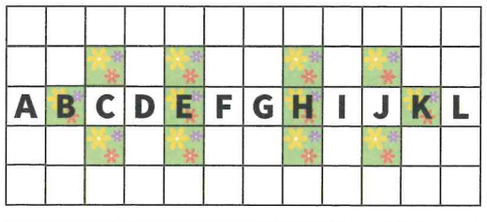
讓我們模擬這個遊戲並找到第二天的情況。第一天的情況如右圖。我們依序由第二列開始一列一列套用規則，觀看變化：



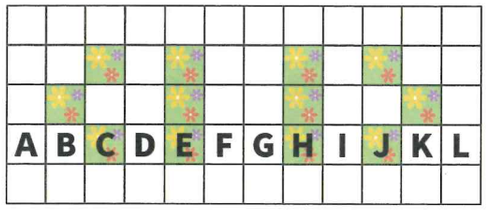
第二列的有顏色方格（CEHJ）符合規則 1，第二天全部變為白色；第二列的白色方格（DI）符合規則 4，第二天變成有顏色方格。



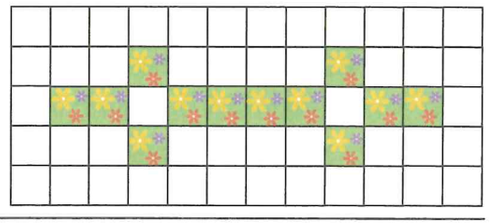
以此類推，第三列的有顏色方格（BEHK）符合規則 2，第二天保持有顏色；第三列的白色方格（CFGJ）符合規則 4，第二天變成有顏色方格。



第四列的有顏色方格（CEHJ）符合規則 1，第二天全部變為白色；第四列的白色方格（DI）符合規則 4，第二天變成有顏色方格。



所以第二天會變成如右圖所示，共有 12 個有顏色的方格。



資訊科學上的意義

本任務的生命遊戲可以用 **有限狀態機 Finite-State Machine** 表示，有限狀態機是一種數學計算模型，由狀態列表（白色及有顏色）、初始狀態（此任務中第一天花園的情況）、以及狀態轉移規則（花朵生長規則） 三部份構成。每個時間點會位於當下狀態，之後根據輸入的訊號或動作，轉換到下個狀態，這個過程被稱為狀態轉移。日常生活中有許多設備都具有限狀態機的行為，根據動作決定轉換到哪個狀態，例如：自動販賣機投入硬幣後，即自動更新顯示金額，按下飲品按鈕後，若投入金額大於飲品金額，則供應飲品並找錢。

關鍵字

康威生命遊戲、有限狀態機