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。

2024 運算思維挑戰賽題組

Benjamin 組

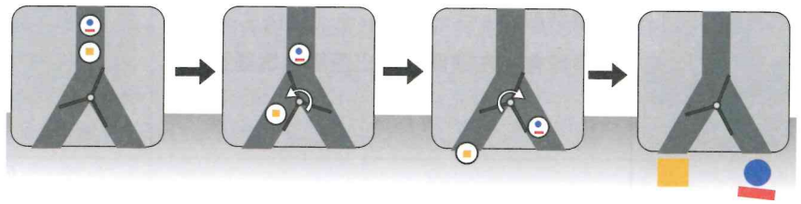
Cadet 組

Junior 組

Senior 組

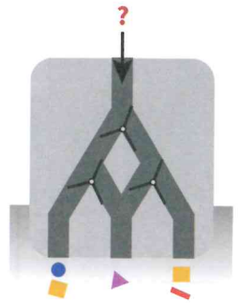
1. 圖案設計師

用下圖機器，能在地板上創作出具有不同色塊的圖案。設計師將內含不同色塊的球從機器上方投入，球會按照閘門的方向滾動，當一顆球通過後，閘門會自動切換方向，讓下一顆球朝另一個方向滾下；最後，球中的色塊會散落在地板上，在對應區域印出有那些色塊的圖案。

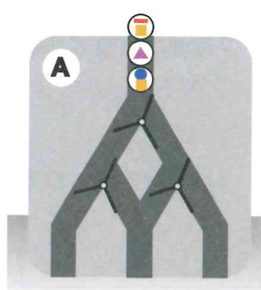


如上圖，在一開始，閘門開向左邊，因此第一顆球會滾向左側，並切換閘門方向；接著第二顆球滾向右側，並切換閘門的方向。

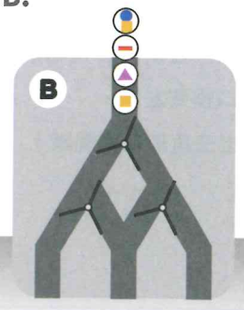
請問下列哪個選項的球，會讓機器印出如右圖的色塊圖案？



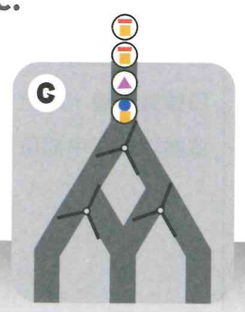
A.



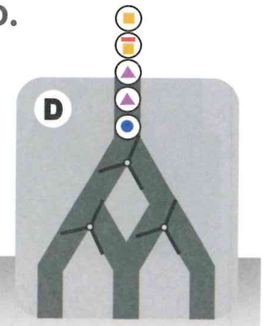
B.



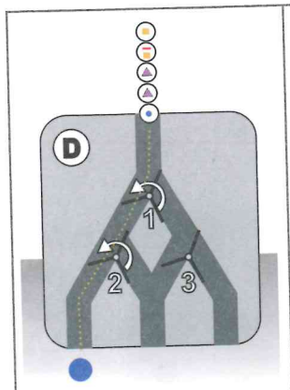
C.



D.

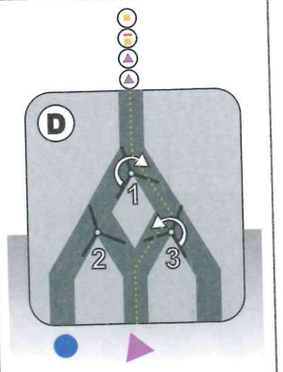


正確答案是：D



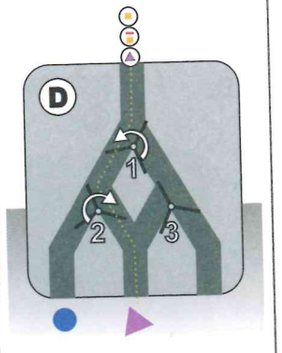
第一顆球會沿著最左邊的路徑往下滾，滾過之後使閘門 1 和閘門 2 開口換到右邊。

這顆球會在左邊區域印出圈圈色塊圖案。



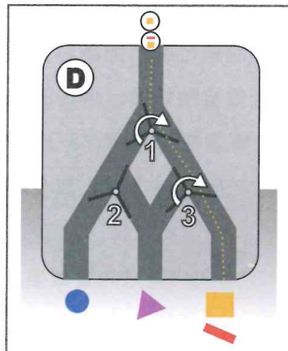
第二顆球在閘門 1 往右滾，在閘門 3 向左滾，滾過之後使閘門 1 開口換到左邊，閘門 3 開口換到右邊。

這顆球會在中間區域印出三角形色塊圖案。



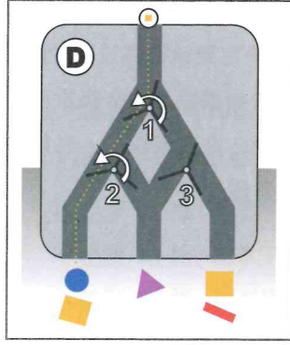
第三顆球在閘門 1 向左滾，在閘門 2 向右滾，滾過之後使得閘門 1 開口換到右邊，閘門 2 開口換到左邊。

這顆球會在中間區域印出三角形色塊圖案。



第四顆球在閘門 1 向右滾，在閘門 3 向右滾，滾過之後使得閘門 1 和閘門 3 的開口都換到左邊。

這顆球會在右邊區域印出正方形跟長方形色塊圖案。



最後，第五顆球會沿著最左邊的路徑往下滾；

在左邊區域印出正方形色塊圖案。

依照上方示範的閘門切換方式，我們也可以推導出，ABC 選項印出的圖案都不是任務想要的色塊圖案。

選項 A 創造的圖案：

選項 B 創造的圖案：

選項 C 創造的圖案：

資訊科學上的意義

邏輯閘 logic gates 是由電晶體組成的基本元件，用來控制電流是否通過，進而進行邏輯判斷與資訊處理。當我們操作這台機器時，會發現每當一顆球通過閘門，閘門的方向就會自動切換。這樣的機制，其實在資訊科學中是一種「狀態變化（state change）」的概念，也與邏輯閘的運作原理很相似。特別是這種「每次進來就反轉方向」的行為，就像邏輯運算中的 NOT 閘，每次輸入都會讓輸出產生一次反轉效果。

此外，在解這個任務時，需要一步一步模擬每顆球的滾動路徑，並記住每次經過閘門後方向會改變。這種過程稱為「**程式碼追蹤 program tracing**」：我們不是真的去執行機器，而是依照規則，模擬每一個步驟會發生什麼狀況，進而預測輸出結果。

在程式設計中，邏輯判斷是控制流程的基本工具，能根據變數目前的狀態，決定要執行哪段程式碼。例如，智慧家電能根據設定的溫度，自動判斷是否開啟冷氣；自動販賣機會根據投入的金額是否足夠來決定是否出貨。這種「根據狀態判斷 → 執行動作 → 更新狀態」的流程，在遊戲設計、家電控制、機器人決策等應用中都很常見，也是電腦程式能夠做出判斷與反應的基礎。

關鍵字

邏輯閘、程式碼追蹤

2. 卡片數值

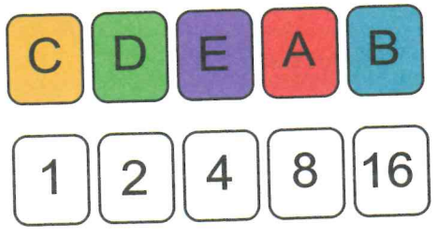
以下有 5 種不同數字的卡片，正面分別印 A、B、C、D、E，每種卡片都有很多張：

我們不確定每種卡片背面的數字，只知道這五個數字為 1、2、4、8 和 16。但以下三位同學知道卡片的數字，每位同學拿了最少數量的卡片，使卡片背面的數字總和等於他們的年齡。

- 小天 17 歲，他拿了兩張卡片，其中一張是 C。
- 小雅 18 歲，她拿了兩張卡片，其中一張是 B。另一張跟 哈利 的卡片一樣，而且不是 E。
- 哈利 15 歲，他是唯一拿 A 卡片的人，且 A 卡片是他數字最大的卡片。他還拿了一張 C 卡片。

請依照上面的線索，推理出 A、B、C、D、E 這 5 種卡片背面的數字，並依各卡片對應的數字，將 A、B、C、D、E 由小到大排列。

正確答案是：CDEAB



小天 17 歲，需要數字 1 和 16 的卡片。

小雅 18 歲，需要數字 2 和 16 的卡片。

哈利 15 歲，需要數字 1、2、4 和 8 的卡片。

因為 小天 和 哈利 都拿了 C 卡片，且 小天 和 哈利 都需要數字 1，所以 C 卡片是 1。

小雅 拿了 B 卡片，但 哈利 並沒有拿 B 卡片，表示 B 卡片 是 小雅 需要的數字，但 哈利 並不需要的數字，所以 B 卡片是 16。

哈利 是唯一擁有 A 卡片的人，也是唯一需要 4 和 8 的人。因此，A 卡片是這兩個數字中的一個，且 A 卡片是 哈利 的最大數字，所以 A 卡片是 8。

剩下卡片 D、E 以及數字 2、4 還沒對應。由於 小雅 和 哈利 有一張相同的卡片，但不是 E 卡片，且 小雅 和 哈利 唯一共同需要的數字是 2，因此 E 卡片不是 2，一定是 4，而 D 卡片是 2。

各種卡片對應的數字是：A=8，B=16，C=1，D=2，E=4，

由小到大排列的順序是 CDEAB。

資訊科學上的意義

二進位法 binary system 是電腦儲存與處理資料的基礎。這是因為電腦內部由大量電晶體組成，而每個電晶體都像是一個微小的開關，只能處於開（1）或關（0）兩種狀態。因此，電腦系統普遍採用僅由 0 與 1 組成的二進位制，來表示各種資訊，例如文字、圖片、聲音或數字等。

在本任務中，每張卡片可以視為一個特定位數的「位元」，是否被選取就相當於該位元是 1（選取）或 0（未選取）。舉例來說，小天的年齡是 17，轉換成二進位就是 10001，代表他所選的卡片數值總和為 16 + 1。如果我們已經知道其中一張卡片是 C，那就能利用 17 的二進位組合來反推出 C 及另一張卡片的數值。透過這樣的方式，我們可以逐步推理出三位同學選擇的卡片與年齡間的對應關係，最後找出五張卡片（A、B、C、D、E）所代表的實際數字

卡片數值	挑選 (1: 是 , 0: 否)
1	1
2	0
4	0
8	0
16	1

在日常生活中，許多數位設備其實都是建立在二進位系統的基礎上運作的。像是電腦或手機，裡面儲存的所有資料：無論是文字、圖片還是音樂，最後都會轉換成由 0 和 1 組成的訊號，讓機器能夠理解與處理。除了這些複雜的裝置之外，一些家電產品也會用簡單的二進位方式來呈現設定狀態。例如有些風扇，雖然我們通常只看到一顆按鈕切換多種模式，但機器裡面會用一個像「計數器」的方式記錄目前按了幾次，這個數值是用二進位的方式儲存，對應到不同的功能。

關鍵字

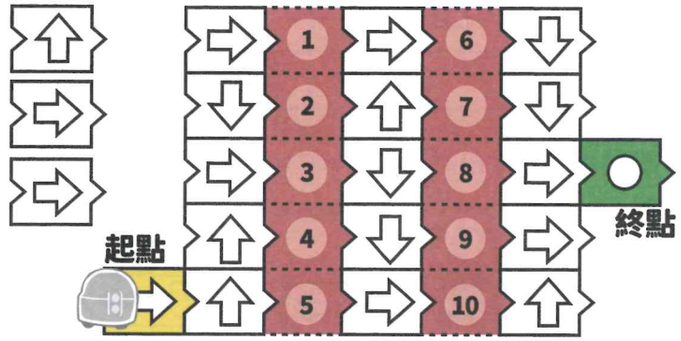
二進位法

3. 機器人的路徑

海狸機器人從黃色地磚 為起點啟動後會依照下列規則在地磚上移動：

1. 依照所在地磚上的箭頭方向移動到下個地磚。
2. 如果下方沒有地磚，或是已到達 終點，機器人就會停下來。

已鋪好的地磚如下圖所示，左方還有 3 塊地磚可以放到紅色標示編號 ① 到 ⑩ 的位置 (與地磚剛好大小)，讓機器人能夠從 起點啟動後抵達 終點才停下來。



請問這 3 塊地磚中的向上箭頭 地磚要放在哪個編號位置？

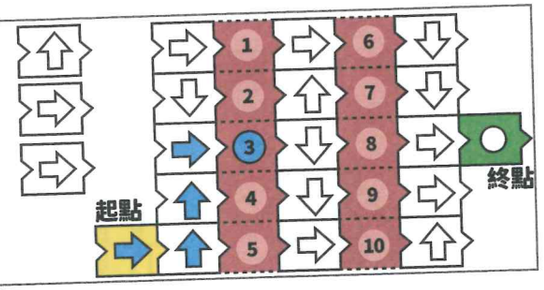
本任務解析由 2024 運算思維推動計畫製作，係採用創用 CC 姓名標示 - 非商業性 - 禁止改作 3.0 台灣 授權條款授權。

正確答案是：3

「向上」 地磚要放在編號 3 的位置。

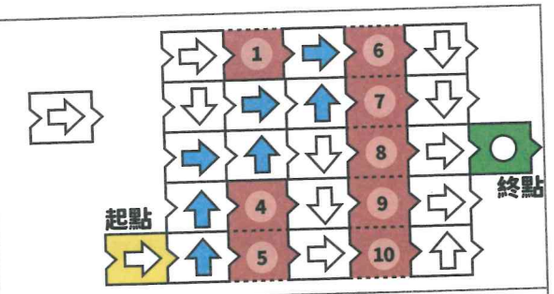
這個任務的目標是填入給定的三塊地磚，讓機器人可由黃色地磚出發並到達終點，安排地磚的思考過程如以下說明。

由起點出發，根據已排好的地磚，走到最左直排的第三塊地磚後會向右走到編號 3 的位置，因此這個位置必須選一塊地磚放。這時候我們有兩個選擇：放「向上」 或「往右」 。

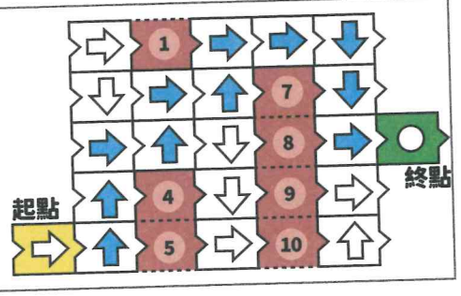


(1) 如果在編號 3 的位置選擇放「向上」 ：

在編號 3 的位置放「向上」 會走到編號 2 的位置，因此編號 2 的位置需要接著一個「往右」 跨到中間直排。



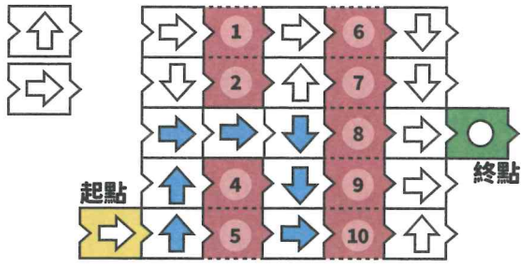
按照中間直排的箭頭，機器人會走到中間直排最頂端的地磚，並走到編號 6 的位置。在編號 6 位置擺放最後一塊「往右」 ，可讓機器人跨到最右直排頂端，再循著箭頭地磚到達終點。



我們分別找到 3 個地磚的擺放位置，且可讓機器人從起點順利走到終點停止。

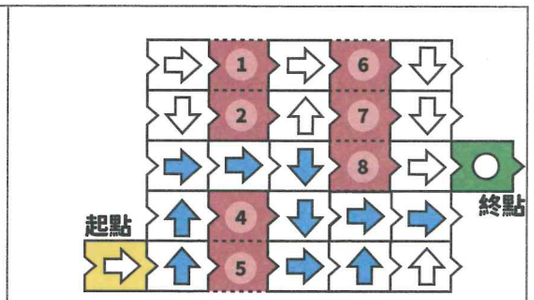
所以「向上」 地磚要放在 3 號位置。

(2) 如果在編號 3 的位置選擇放「往右」 ：機器人會走到中間直排底部的地磚後向右走到編號 10 的位置。



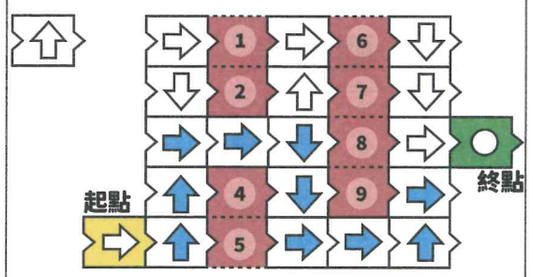
在這種情況，編號 10 的位置必須放第二塊地磚，接下來分成以下兩種狀況。

若在編號 10 的位置擺放「向上」 ，會走到編號 9 的位置，只剩下一個「往右」 一定要放在標記 9 的位置，機器人才不會停在紅色區域。但這樣的擺放，機器人無法停在終點。



若在編號 10 的位置擺放「往右」 ，使機器人直接跨到最右直排的底部，再根據最右直排地磚的箭頭走。

在這種情況，最後一塊「向上」 地磚沒有用到 (因為只允許放在編號 1 到 10 的位置，而放在其他編號的位置也無法從起點被走到)，機器人也無法停在終點。



因此「向上」 地磚一定要放在編號 3 的位置。

資訊科學上的意義

當我們寫程式時，經常會遇到程式沒有按照預期的方式執行。這時就需要用到一種叫做 **程式碼追蹤 program tracing** 的技巧，來一步步觀察程式在執行時到底發生了什麼。我們會從第一行開始，追蹤變數的變化、判斷條件流程的走向，就像當一位「程式偵探」，找出程式中的邏輯或指令錯誤的問題。

在本任務中，海狸機器人會依照地磚上的箭頭前進，目標是從起點順利走到終點。為了設計正確的地磚配置，我們需要事先「模擬」機器人每一步的動作，觀察每一塊地磚會讓它往哪裡移動，以及最後是否能走到正確的位置。這個過程就像在進行程式碼追蹤；雖然我們沒有真的讓機器人程式執行一遍，但透過步驟推演，能預測執行結果並找出可能出錯的地方。

其實，在日常生活中，我們也常用類似的方式來出問題所在。例如在烘焙時，如果蛋糕沒有膨脹，我們會回頭檢查材料比例、攪拌方式或烘烤時間；又或者解數學題時發現答案不對，我們會一步步倒回去檢查運算過程。這些做法和程式碼追蹤很像，都是靠「逐步驗證」與「邏輯推理」，幫助我們有效地找出錯誤並修正問題。

關鍵字

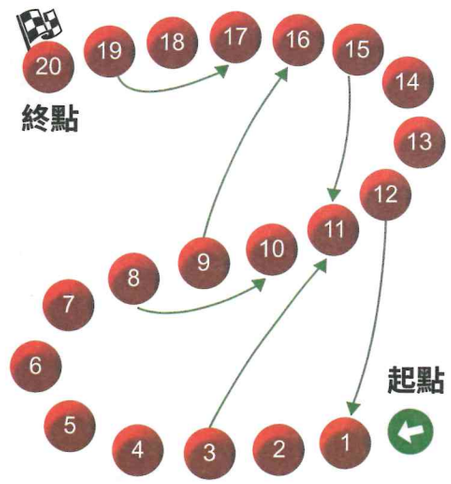
程式碼追蹤

4. 消失的箭頭

趣味競賽的賽道如下圖所示。有 4 位同學 A、B、C、D 依序從起點進入賽道。遊戲規則如下：

1. 每次輪一位同學，輪到的同學會從原先的位置先往前跳一個位置。
2. 如果跳到畫有箭頭尾端的位置，可以快速跳到該箭頭所指向的位置。不過箭頭為一次性的路徑，有人使用後的箭頭隨即失效，其他人不能再用同樣的箭頭快速往前跳或往後跳。

A、B、C、D 四位同學從起點開始，依序輪流往前跳一個位置，經過 2 輪都到 2 號位置。在第 3 輪時，A 會是第一個跳到 3 號位置的同學，他會立即跳到 11 號位置；但接下來的 B 就無法使用這個快速路徑。

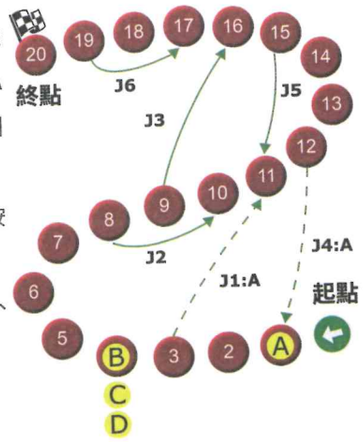


請問 A、B、C、D 四位同學最後到達終點的先後順序？

正確答案是：CDBA

A 是第一個移動的，他會最先到達 3 號位置並使用箭頭 J1 到達 11 號位置，這時候其他同學都在 3 號位置。接著下一個回合，A 會到 12 號位置，並使用箭頭 J4 回到 1 號位置。因此在第 4 回合後，A 會回到第 1 個位置，其他同學都在 4 號位置。

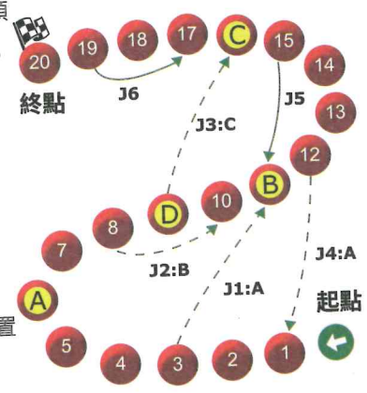
圖 1 顯示第 4 回合後，每個人的位置及箭頭使用狀態；箭頭按 J1 到 J6 編號，箭頭線為實心表示會由該處直接跳到另一位置，虛線則表示該箭頭已經有人用過而失效，虛線旁邊標示的 A、B、C 或 D 表示使用該箭頭的同學。



圖一

按照順序，第 8 回合後，B 會第一個到達 8 號位置，並使用箭頭 J2 到達 10 號位置，因此不會用到箭頭 J3。所以在第 9 回合後，C 會第一個到達 9 號位置，並使用箭頭 J3 到達 16 號位置。

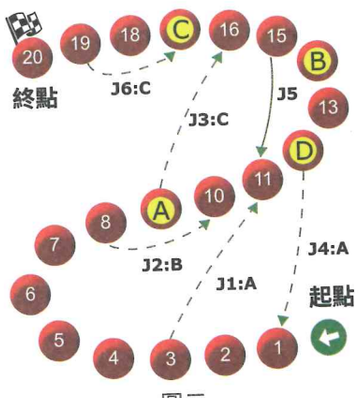
圖 2 顯示第 9 回合後，每個人的位置及箭頭使用狀態。



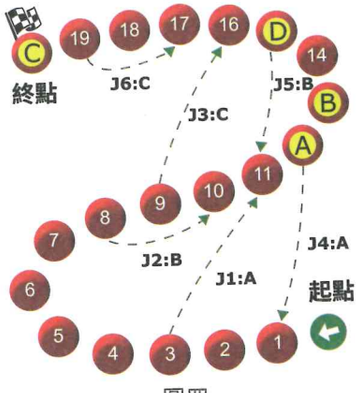
圖二

在第 12 回合， C 會到達 19 號位置，由箭頭 J6 並退回 2 個位置到 17 號位置，但他仍然保持領先。

圖 3 顯示第 12 回合後，每個人的位置及箭頭使用狀態。



圖三



圖四

在第 15 回合後，C 會最先到達終點，其他人的位置及箭頭使用狀態如圖 4。

因為到此時，所有箭頭都已被使用過，所以最後他們到達終點的順序是 CDBA。

資訊科學上的意義

程式碼追蹤 program tracing 是一種幫助我們理解程式執行流程的技巧。當我們寫下一段演算法，或讀到一組規則時，可以照著它的執行順序，一步步模擬每個指令動作，觀察變數或狀態的變化，進而預測最終的執行結果。

在本任務中，四位同學依序在賽道上移動。當有人第一個踩到綠色箭頭節點時，就會立刻跳躍到箭頭所指的位置，而且這條捷徑只能被使用一次。要解出答案，必須逐輪模擬每位同學的行動，觀察誰先踩到箭頭觸發捷徑，以及路線會如何因此改變，一步步推理出最終的抵達順序。這樣的過程就像是在「追蹤」一段程式的執行，分析每個條件在何時會被觸發，並理解每一步對整體結果的影響。

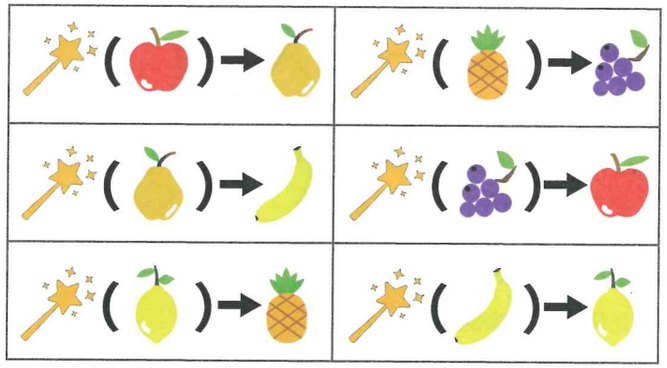
在生活中，桌遊「大富翁」就是一個類似的例子。玩家輪流擲骰子移動，根據停下的位置可能會購地、繳費或抽卡。當某位玩家買下某塊地，地圖上的狀態就改變了，也會影響其他玩家後續的行動與選擇。這種根據規則和事件逐步推演、追蹤每一步對結果所造成影響的過程，和程式碼追蹤的思維很相似。

關鍵字

程式碼追蹤

5. 魔術師

班班是一位著名的海狸魔術師，他能用魔法棒把一種水果變成另一種。
下圖為各種水果經過一次魔法棒後變成另一種水果的變換列表。



班班用魔法棒對水果進行多次變換，例如從蘋果 開始，點第一次把它變成梨 ，點第二次又把梨變成香蕉 。這樣經過兩次魔法棒變換的過程，可以用下列式子表示：

班班的忠實粉絲用下面的式子記錄班班使用魔法棒的過程，請問這個過程最後會變出哪一種水果呢？

A.  　**B.**  　**C.**  　**D.** 

本任務解析由 2024 運算思維推動計畫製作，係採用創用 CC 姓名標示 - 非商業性 - 禁止改作 3.0 台灣 授權條款授權。

正確答案是：A

依照任務顯示的式子，可以得知班班使用了 5 次魔法棒。

根據魔法棒的水果變換列表，我們可以一步一步得出水果每次的變換結果而得到答案。

第 1 次變換：

第 2 次變換：

第 3 次變換：

第 4 次變換：

第 5 次變換：

其他選項都不正確。

選項 B 是錯誤答案，是變換了 4 次的結果。

選項 C 是錯誤答案，是變換了 3 次的結果。

選項 D 是錯誤答案，是只變換了 1 次的結果。

資訊科學上的意義

在程式設計中，**函數 function** 是一種模組化的工具，用來接收輸入並產生對應的輸出結果。透過函數的設計，我們可以將重複的操作集中處理，使程式的架構更加清晰、易讀且容易維護。而**遞迴 recursion** 則是一種特殊的函數應用方式，指的是函數在自己的內部再次呼叫自己。遞迴的核心精神在於：透過相同的處理流程，一步步由規模較小的子問題先解決，最終再合成整體問題的解答。

在本任務中，魔術師使用魔法棒將一種水果變成另一種。這個轉換動作可以視為一個函數：輸入一種水果，輸出對應的轉換結果。當魔術師對轉換後的水果再次施展相同的魔法，並一層層重複這個轉換過程時，就形成了遞迴結構。每一次的變換都依賴上一次的結果，直到最後完成所有轉換為止。

這樣的遞迴邏輯其實在日常生活中也可運用。舉個例子：當你想知道自己在隊伍中排第幾個時，可以問你前面那個人他排第幾，對方則問他前面的人，依此類推，一直追溯到第一位。當第一位說「我是第一個」，這個資訊就像是遞迴的終止條件，然後每個人再把結果「往回（後）傳」，知道自己是前一個人告訴你的結果再加一位。這就是遞迴思維的具體應用：不直接處理整體問題，而是將其分解為規模更小、結構相同的子問題，逐層處理，最後得到完整的結果。只要正確設計好終止條件，遞迴就能幫助我們以簡潔而有條理的方式，處理許多具結構性的問題。

關鍵字

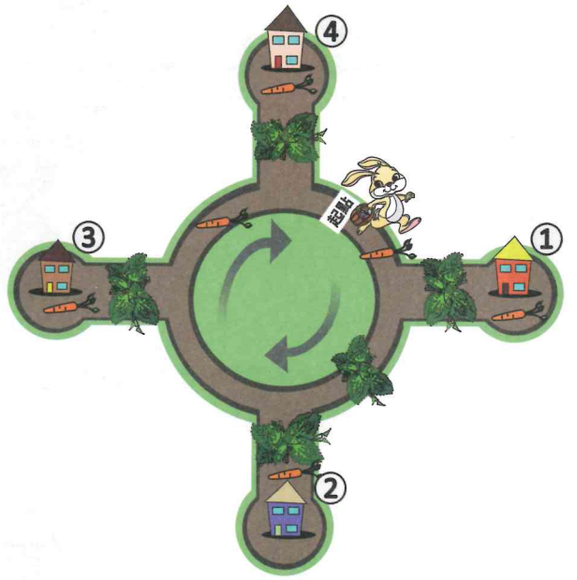
函數、遞迴

6. 送蛋

小兔要把彩蛋送到村中四戶人家裡，但途中有障礙樹叢擋住去路；

小兔需要吃下紅蘿蔔才有力氣清除障礙樹叢，且每吃一根紅蘿蔔就有力氣清除一棵障礙樹叢。

下圖為目前各戶人家及障礙樹叢位置圖，在路途中看到的紅蘿蔔，小兔都可以吃。



小兔打算從「起點」的位置出發，在圓形路徑上依順時針方向前進，經過每戶人家門前時，若能清除障礙樹叢，則進行送貨；若不能清除，則依圓形路徑繼續前往下一戶人家，依此規則直到四戶人家都收到彩蛋為止。

哪一戶人家最後收到復活節彩蛋呢？

A. ①

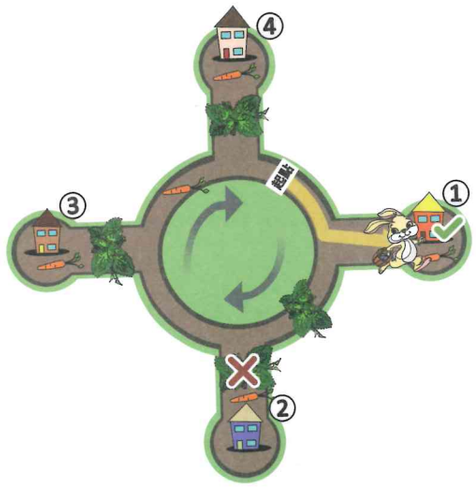
B. ②

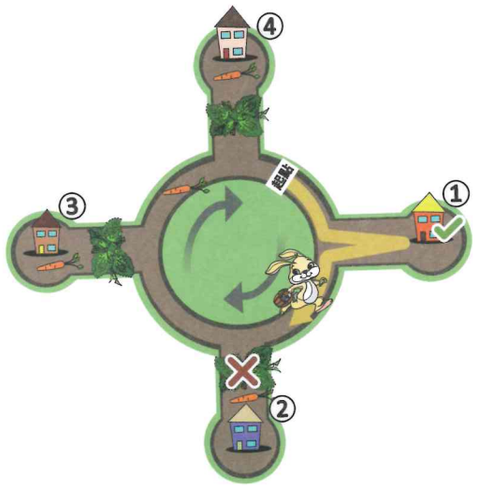
C. ③

D. ④

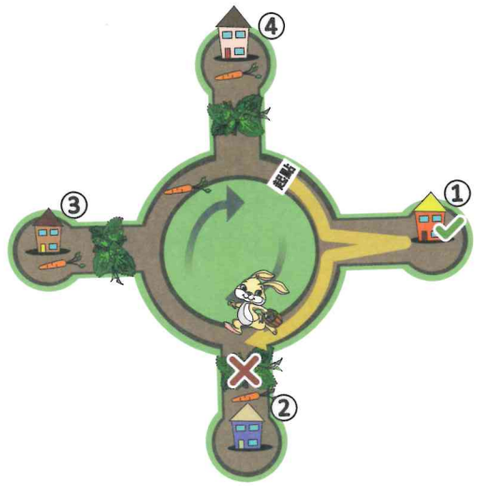
正確答案是：C

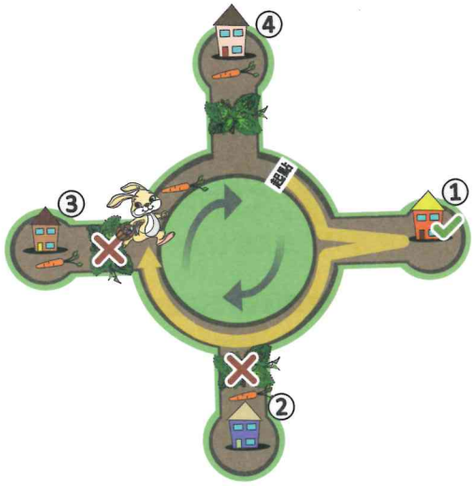
小兔從起點出發後，先吃了一根路上的紅蘿蔔，所以到①號住戶前可以順利清除障礙樹叢並送貨。接著吃下 ① 號住戶旁的紅蘿蔔，並在順著圓形路徑前進時，清除圓形路徑上的障礙樹叢。



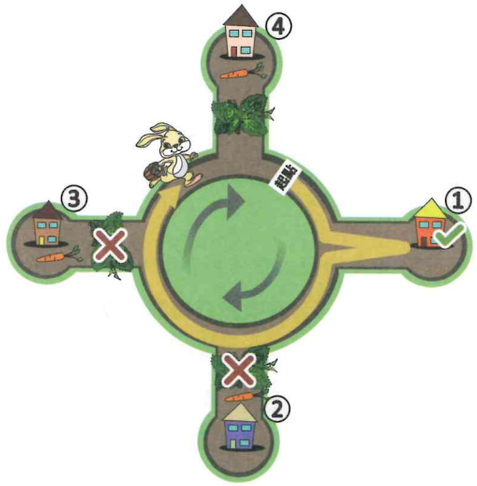


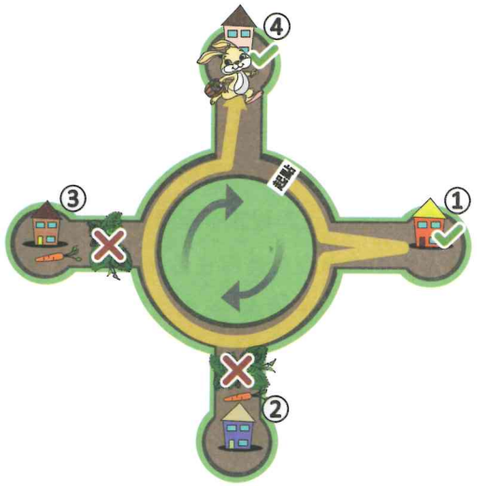
接下來到了 ② 號住戶及 ③ 號住戶前，因為沒有紅蘿蔔的能量可以清除障礙樹叢，所以只能繼續前進。





至 ④ 號住戶前吃了路上的紅蘿蔔，可以清除 ④ 號住戶的障礙樹叢並送貨，接著吃下 ④ 號住戶旁的紅蘿蔔。





再依圓形進入第二圈，到 ② 號住戶前清除障礙樹叢並送貨，接著吃下 ② 號住戶旁的紅蘿蔔，繼續前進至 ③ 號住戶，清除障礙樹叢並送貨。因此最後收到復活節彩蛋的為第 ③ 戶。

資訊科學上的意義

在程式設計中，**迴圈 loops** 是一種用來執行重複動作的程式結構。當某個條件尚未達成時，程式會不斷重複執行相同的操作，直到條件被滿足為止。

在本任務中，小兔從起點出發，沿著路徑順時針方向繞行村莊，並重複執行以下步驟：「吃紅蘿蔔補充體力 → 檢查是否可以清除障礙 → 若成功則送出彩蛋」。如果到某戶人家的路徑仍被障礙擋住，小兔就會繼續繞圈，重複上述步驟，直到所有人家都順利收到彩蛋為止。這種不斷重複直到任務完成的流程，正是程式中迴圈邏輯的典型應用。

迴圈邏輯的概念經常運用在日常生活中，例如洗衣機在清洗衣物時，會依照設定的洗程重複執行「注水 → 旋轉 → 排水 → 再注水」等動作，只要尚未達到預定的清洗次數或時間，就會重複這些步驟，直到整個洗衣程序完成為止。這樣的流程就像程式中的迴圈：只要條件未滿足，就持續執行相同的操作。另一個例子是背單字。當我們在學習新單字時，經常會反覆「看單字 → 嘗試背誦 → 測驗 → 檢查是否記住」，如果發現還沒記起來，就會再重頭開始。這樣的學習過程其實也是一種迴圈結構，持續進行相同的行動直到目標達成為止。因此「迴圈」不只是程式設計的技術概念，更是我們日常行為中常見且自然的思考與行動方式。

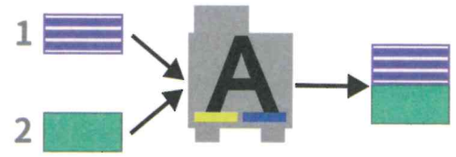
關鍵字

迴圈

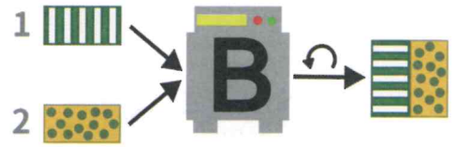
7. 拼布工廠

拼布機 A 和 B 能依照不同規則縫合兩種花色的布料：

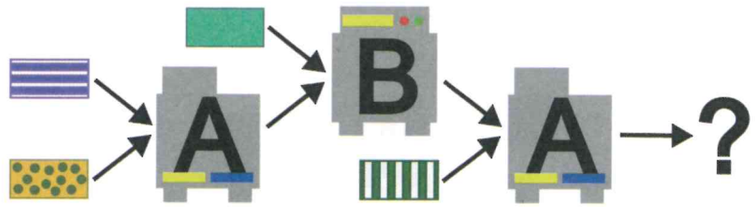
- 拼布機 A 將布料 1 放在布料 2 上方，縫合成布料 。



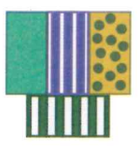
- 拼布機 B 先將布料 1 放在布料 2 上方，縫合後向左旋轉 90 度，形成布料 。



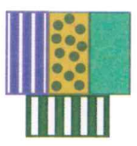
請問下圖的布料經過拼布機縫合後，最終形成的布料為何？



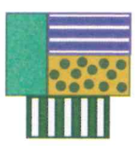
A.



B.



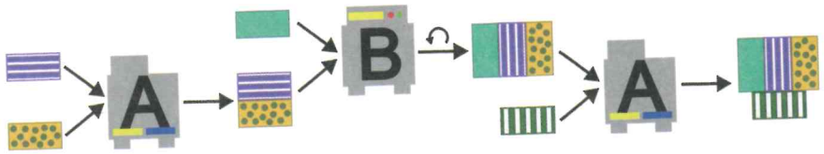
C.



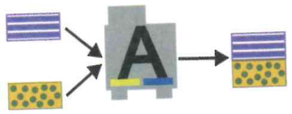
D.

正確答案是：A

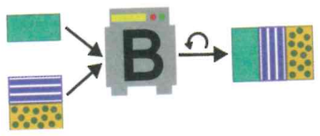
正確答案是 A。



首先，拼布機 A 將兩張布上下縫合成新布料 1。

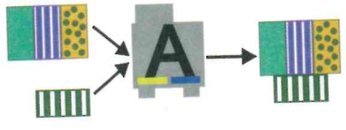


然後，另一張布料和新布料 1 一起放進拼布機 B，拼布機 B 將兩張布料縫合並旋轉，得到新布料 2。



最後，新布料 2 在上，和另一張布料在下，一起放進拼布機 A，縫合得到最終的布料。

因此，正確答案是 A。



資訊科學上的意義

在程式設計中，模組化是一種將特定功能封裝成可重複使用元件的做法，有助於提升程式的結構性與維護效率。**函數 function** 是實現模組化的常用工具之一，它透過接收一組輸入值，依照函數中所定義的處理步驟進行運算，最終產生對應的輸出結果。這種「輸入 → 處理 → 輸出」的結構，使得程式碼可以更清楚地分工與重新利用。

在本任務中，拼布機 A 與拼布機 B 分別可視為兩個具有不同處理目的函數。每台拼布機會接收兩塊布料作為輸入，依照指定規則進行縫合（例如：直疊，或是旋轉後再疊），並產出一塊新的布料圖樣作為輸出。進一步來看，這些產出的布料也可以作為另一台拼布機的輸入，顯示函數之間可以組合 (composition) 與嵌套 (nesting) 使用的特性，這是模組化設計中重要的概念。

在日常生活與數學中，我們也常接觸到類似的函數概念。例如在數學裡，給定一個函數 $f(x, y) = x^2 + 3xy + 4y^2$，我們可以將不同的 x、y 值帶入，計算出對應的輸出，這與程式中的函數操作如出一轍。其他像拍照後的影像處理，濾鏡效果 (如模糊、銳化、亮度調整) 都是典型的函數應用。以模糊濾鏡為例，可以被視為一個函數，輸入為一張圖片，內部透過對每個像素與其鄰近像素進行加權平均的運算，輸出則是一張經過模糊處理的新圖片。再如圖像合成時，可使用色彩合成函數將兩張圖片依據透明度參數進行像素層級的融合，產生一張合成後的結果。這些處理流程雖然複雜，但都可以用「輸入 → 處理 → 輸出」的函數概念來實現與重複使用。

關鍵字

函數

8. 烏龜圖形

海龜機器人收到指令後會在畫板上畫出對應的圖案。當收到「畫一個圓 (d)」指令時，它會根據 d 的數值大小畫出一個對應大小的圓。

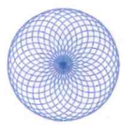
圓 (10)

圓 (7)

圓 (5)

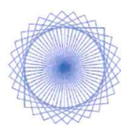
一開始 d 設為 1 ，每次執行指令畫完一個圓後，將 d 的值增加 1。

請問連續執行 30 次後，機器人會畫出什麼樣的圖案？

A. 

B. 

C. 

D. 

正確答案是：C

機器人第一次 d 設定為 1，畫了一個非常小的圓，然後回到起始位置。

第二次加大 d，畫了一個更大的圓，再回到起始位置。

這兩個圓會重疊在起始位置上。

經過 30 次畫圓後，最終圖案會有 30 個大小不同的圓，且都由一個共同的起始點畫出去，如選項 C 所示。

其他選項為何不正確：

A：圖中是多個相同大小的圓，以固定相隔角度疊在一起，但機器人應該畫了 30 個大小不同的圓。

B：這些圓是圓心相同，但不是由一個共同的起始點開始畫。

D：這張圖不是由圓所組成。

資訊科學上的意義

在程式設計中，模組化是一種將特定功能封裝成可重複使用元件的做法，有助於提升程式的結構性與維護效率。**函數 function** 正是實現模組化的常用工具之一，它透過接收一組輸入值，依照內部定義的運算邏輯進行處理，產生對應的輸出結果。這樣的設計方式能讓我們在需要時重複使用相同的邏輯，而無需重新撰寫程式碼。

在函數的設計，參數改變對其執行結果有關鍵影響。在本任務中，「畫一個圓」可以視為一個函數，其中的參數 d 決定了圓的大小。雖然我們呼叫的是同一個函數，但只要輸入的參數不同，輸出的結果也可能跟著改變。例如，「畫一個圓 (d)」 每次接收到不同的 d 值，就會畫出不同大小的圓，但開始畫圓的起點一樣。每當程式呼叫這個函數一次，便會根據當時的 d 值畫出對應大小的圓。接著，程式會將 d 加 1，再次呼叫同樣的函數畫出下一個圓。這種方式再結合**重複結構 repetition structure** ，便可重複執行相同的動作流程，但改變呼叫函數的參數 (d 值)，使下次執行結果有變化。搭配迴圈使用能讓程式自動畫出多個圓，而不需重複寫多行畫圓的指令。函數不變，但透過參數改變就產生不同執行，這正是函數執行可彈性調整的方式。

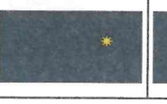
在日常生活中也有許多情境用到函數。例如智慧洗衣機中已設計好多種洗衣流程函數，使用者只需輸入衣物種類與重量等參數，系統就能依照條件執行對應的處理步驟。像是輸入「棉質、6 公斤」與「嬰兒衣物、2 公斤」，雖然執行的是同一個「洗衣」功能，但因為參數不同，執行的動作也會不同。在程式中，我們常在迴圈中呼叫函數，針對一組資料逐一處理。這種結構在生活中也很常見。例如：廚師為多位客人煎蛋，每份點單可能要求不同熟度（如半熟、全熟），就像在迴圈中針對每筆訂單呼叫一個煎蛋函數。又利如聊天機器人會依序對每則輸入訊息呼叫一個回應函數產生回應訊息。這些都是以迴圈重複呼叫函數處理多筆資料，函數則負責依每次輸入參數進行處理的例子。

關鍵字

函數、演算法 (重複結構)

9. 安娜的畫作

安娜正使用繪圖軟體創作一幅圖畫，她按照下表中的步驟進行：

步驟 1	步驟 2	步驟 3	步驟 4	步驟 5	步驟 6	步驟 7
						

她完成所有步驟後，得到的圖畫如下：

安娜想從她的圖畫中刪除一些圖案，所以她退回到 3 個步驟前的結果。請問她會得到以下哪個選項的圖畫？

A. 

B. 

C. 

D. 

E. 

正確答案是：E

逐步退回 3 個步驟的過程，這張圖畫的變化如下表所示：

退回步驟 7	再退回步驟 6	再退回步驟 5
		

因此選項 E 是答案。

資訊科學上的意義

堆疊 stack 是一種常見的資料結構，它的特性是 **後進先出 last in first out, LIFO**，也就是最後放入的資料會最先被取出。就像把盤子一個一個往上疊，新的只能放在最上面，要拿的時候也只能從最上面開始。堆疊有兩個基本操作：push 是將資料放進堆疊，pop 則是把最上面的資料取出。

以本任務為例，安娜完成了七個繪圖步驟，這就好比每一步執行前的狀態都依序被記錄到堆疊中。當她想退回到三步驟前的狀態時，就相當於從堆疊中執行了三次 pop 操作，依序移除最上層的三個動作，留下前四步的結果。這樣的過程正好展現了堆疊「後進先出」的特性。

這種結構常被用來記錄程式執行時的函數呼叫流程，以及使用者的操作歷程。當使用者修改、刪除或插入內容時，系統會依序將每個操作透過 push 記錄到堆疊中；而當使用者按下「復原」鍵時，就會透過 pop 將最近的操作一個個還原。在日常生活中，像是瀏覽器的返回功能，也運用了相同的原理；每次瀏覽的新頁面都被記錄起來，當使用者想回到上一頁時，就可由堆疊依序取出最近造訪的頁面。

關鍵字

堆疊、後進先出

10. 線上課程

海狸老師在家裡進行線上授課。有 9 位學生在學校的圖書館裡，使用同一排相鄰的 9 臺電腦上線上課。

在老師的電腦螢幕上，老師看到這 9 位學生上線的畫面跟名字，因為位子的間距不大，學生左右兩邊坐的同學也被拍到了一部分。

請問這 9 位學生中，是誰坐在正中間呢？

A. 小戴
B. 阿力
C. 小李
D. 小貝
E. 娜娜

正確答案是：E

答案是 E 娜娜。

根據電腦螢幕上的畫面，我們知道每位學生旁邊坐的是誰。

從左圖的畫面，我們知道小詹坐在最旁邊，而且小愛坐在他旁邊。

從左圖的畫面，我們知道小愛坐在小詹和小戴之間。

用同樣的方法查看其他畫面，我們可以依序排出這 9 位學生的位置：

總共 9 位學生，因此坐在正中間的是從左邊數來第 5 位，或者從右邊數來第 5 位的人。

那個人就是娜娜。

資訊科學上的意義

雙向鏈結串列 doubly linked list 是資訊科學中常見的一種資料結構，由一連串節點（nodes）串接而成。每個節點除了儲存資料，還會記錄自己前一個和下一個節點的位置，因此可以從任意位置開始，向前或向後移動。相比只能往一個方向前進的單向鏈結串列，雙向鏈結串列更具彈性，特別適合需要雙向搜尋或存取資料的情境。

在本任務中，學生們的畫面排列就像是一條雙向鏈結串列，每位學生相當於其中的一個節點。每個畫面會同時顯示自己與左右兩側同學的部分畫面，就像每個節點都知道自己的前一位和下一位是誰。我們可以從其中任一位已知的學生出發，根據左右的線索一路推理出整個排列順序，最後找出正中央的那位學生。

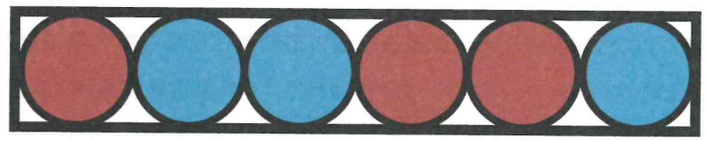
這種資料結構在日常生活中也有實際應用。以音樂播放程式的播放清單為例，使用者可以自由點選「上一首」或「下一首」切換歌曲，這樣的功能通常會採用雙向鏈結串列來記錄播放順序。因為每首歌都能快速找到前後的曲目，系統不需要重新從清單開頭搜尋，就能順暢完成切換。

關鍵字

雙鏈接串列

11. 球的排列

有一串紅色球和藍色球的排列如下：



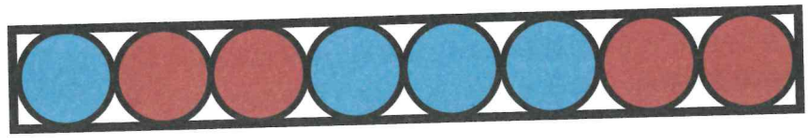
小狸想出一個記錄這兩種色球排列的方式：從左到右，從每個位置開始計數到最右邊，記下藍色球的累積數量。

首先，從左邊第一顆球開始，數到最右邊，看有幾顆藍色球，得到答案為 3；第 2 次，從左邊數來第二顆球開始算，數到最右邊看藍色球的數量，得到答案也是 3；第 3 次，則從左邊第三顆球開始計算藍色球的數量，數到最右邊得到答案是 2，反覆依照這個方式計數，最後得到的計數結果為：3, 3, 2, 1, 1, 1。

接下來將這些計數值轉換為 0 和 1，如果計數值是偶數就寫 0（計數值 0 是也是寫 0），計數值是奇數就寫 1，得到以下字串： 110111

小狸剛看到另一串由 8 個色球形成的排列，根據他的規則記下的字串是： 01110100 ，請問這串球中有幾個紅色球 ?(範圍 [1~8] 的整數)

正確答案是：4



已知記下的字串是： 01110100，其中每個數字表示「從該位置到最右邊的藍球數是偶數（0）或奇數（1）」。我們可以利用這個資訊，從最後一個位置開始往左推論出每個位置應該放什麼球。

一開始都沒有球，所以藍色球的數目為 0(為偶數)；然後從記下的字串由右往左逐一檢查：

- 如果看到 '1'，代表需要奇數個藍色球。這時如果累計的藍球數是偶數，這個位置就放藍色球；如果是奇數，就放紅色球。
- 如果看到 '0'，代表需要偶數個藍色球。這時如果累計的藍球數是偶數，這個位置就放紅色球；如果是奇數，就放藍色球。

我們只要依這個方法從右到左，就可以一步步決定每個位置該放什麼球，過程如下所示。

字串位置 8，字元 '0'(應有偶數個藍球)，累積藍色球數 0(偶數)，放紅球。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色								紅球

字串位置 7，字元 '0'(應有偶數個藍球)，累積藍色球數 0(偶數)， 放紅球。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色							紅球	紅球

字串位置 6，字元 '1'(應有奇數個藍球)，累積藍色球數 0(偶數)， 放藍球 (藍球數變 1)。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色						藍球	紅球	紅球

字串位置 5，字元 '0'(應有偶數個藍球)，累積藍色球數 1(奇數)， 放藍球 (藍球數變 2)。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色					藍球	藍球	紅球	紅球

字串位置 4，字元 '1'(應有奇數個藍球)，累積藍色球數 2(偶數)，放藍球 (藍球數變 3)。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色				藍球	藍球	藍球	紅球	紅球

字串位置 3，字元 '1'(應有奇數個藍球)，累積藍色球數 3(奇數)， 放紅球。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色			紅球	藍球	藍球	藍球	紅球	紅球

字串位置 2，字元 '1'(應有奇數個藍球)，累積藍色球數 3(奇數)，放紅球。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色		紅球	紅球	藍球	藍球	藍球	紅球	紅球

字串位置 1，字元 '0'(應有偶數個藍球)，累積藍色球數 3(奇數)，放藍球 (藍球數變 4)。

位置	1	2	3	4	5	6	7	8
字串	0	1	1	1	0	1	0	0
珠色	藍球	紅球	紅球	藍球	藍球	藍球	紅球	紅球

累積藍色球數為 4，另外有 4 個位置放紅球，分別是在位置 2、位置 3、位置 7、及位置 8。

資訊科學上的意義

在資訊科學中，常會透過 **資料編碼 data encoding** 的方式，將原始內容依照特定規則轉換為由 0 和 1 組成的位元（bit）序列，方便電腦系統進行儲存及運算。在本任務中，小狸不是記下每顆球的顏色，而是取出這串排列中某種可觀察、可計算的樣式特徵：透過「從每個位置往右的藍色球數量」，再進一步轉換為簡單的 0/1 表示，形成一串二進位序列。這樣的處理步驟，就是典型的資料編碼過程。

將數字轉為 0 和 1 的位元序列是資料壓縮的基本技巧之一。很多壓縮格式 (如 Huffman 編碼、Run-Length Encoding) 會根據資料出現頻率或連續性，把原始資料重新映射成較短的 bit 表示。小狸的轉換方式，雖然不是為了節省空間，但展示重新編碼的（re-encoding）概念，正是許多**資料壓縮 data compression** 技術的核心基礎。壓縮方法通常會透過模式發現、統計性分析與重新編碼，來簡化資料表示、降低冗餘；以減少資料儲存空間並加快傳輸速度。

在日常生活中，條碼與 QR Code 是常見的資料編碼方式。像超市掃描商品條碼，或用手機掃描 QR Code，都是在用特定的格式把資料轉換成機器能讀懂的樣子，方便快速處理，內容本身並沒有改變。而像 .zip 或 .rar 這類壓縮檔，則是資料壓縮的例子。它們會找出資料中的重複和規律，把檔案變小，好儲存也好傳送，還能在需要時還原成原本的內容。

關鍵字

資料編碼、資料壓縮

12. 冰淇淋店

在海狸自助冰店，客人只要按箭頭控制按鈕，將冰勺機器人移動到想要的口味，再按一下按鈕 ● ，機器人就會取一球冰淇淋，取完三球，會將冰淇淋送給客人。

每次服務一個新客人，冰杓機器人都會回到底部中間的位置。

例如有一位客人依序按下以下按鈕：

機器人就會取巧克力 、梅子 和咖啡 口味的冰淇淋各一球，送給這位客人。

小狸想買櫻桃 、香草 和草莓 口味的冰淇淋各一球，並希望按最少次的按鈕，以簡省時間。請問小狸最少要按幾次按鈕？(範圍 [1~15] 的整數)

正確答案是：9

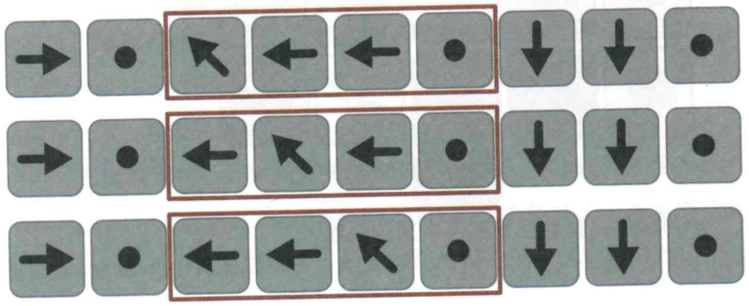
小狸最少需要按 9 次按鈕。

在這個任務中，因為冰杓機器人會從底部中間出發，所以最好先取離底部中間較近的口味。距離底部中間最近的是香草 （只需按 2 次 → ● 就可取得），如果先取香草，接下來不論取櫻桃 還是草莓 都要按 4 次，所以可以分成以下二種挖取順序：

1. 先取香草，再取櫻桃，最後取草莓 。

2. 先取香草，再取草莓，最後取櫻桃 。

取了香草 後，共有三種不同的方法可以透過按 4 次按鈕取得草莓 。



以上四種方式都是按 9 次按鈕的取法。

此外，如果一開始取櫻桃 或草莓 ：

- 一開始先取櫻桃 （按 4 次按鈕），接下來取跟櫻桃 較接近的草莓 （按 3 次按鈕），最後取香草 （按 4 次按鈕），共要按 11 次按鈕。
- 一開始先取草莓 （移動 3 次），接下來取跟草莓 較接近的櫻桃 （按 3 次按鈕），最後取香草 （按 4 次按鈕），共要按 11 次按鈕。

所以最少為按 9 次按鈕。

資訊科學上的意義

在資訊科學中，**路徑搜尋 path finding** 是一種常見問題，目標是在空間中尋找從起點到目標的最佳路徑。在本任務中，機器人要以最少的按鈕次數依序取得三種冰淇淋口味，這就像是在地圖上安排從一個點依序拜訪多個目的地的最短路徑。要取得一個口味，要先移動到該位置，再按下取用按鈕一次。若找出機器人從起點去取三種口味冰淇淋的順序之最少移動步數，則最少按鍵次數 = 機器人移動步數 +3。要走訪多個目的地的最短路經搜尋屬於一種 **最佳化問題 optimization problem**，必須從各種可能的組合中選出最符合目標的那一組結果。

此外，本任務中機器人移動的最少步數估算，使用了 **切比雪夫距離 Chebyshev distance** 這個概念。這種距離常用於棋盤型的網格中，當移動允許上下左右與斜向時，它能估算出兩點間最少移動步數，計算公式為：Chebyshev 距離 = $\max(|x_2 - x_1|, |y_2 - y_1|)$。舉例來說，當機器人從起點走到櫻桃冰淇淋的位置，其座標差為 x 軸相差 2 格、y 軸相差 3 格，那麼距離為 $\max(2, 3) = 3$。

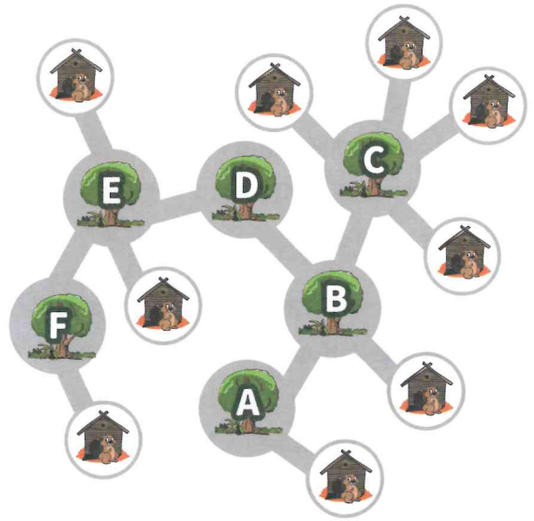
在日常生活中，多目的地最短路徑搜尋可以幫助我們安排拜訪多個地點時，找出總路程最短或所需時間最少的方法。舉例來說，物流配送 / 外送員在接到多筆訂單後，可根據每位顧客的位置，安排一條最有效率的送貨 / 送餐順序。

關鍵字

路徑搜尋、最佳化問題、切比雪夫距離

13. 集合點

下圖是一個海狸聚落地圖，聚落中有海狸各自的巢穴及種植的大樹，並鋪設了一些長度相同的小徑（　　）互相連接。



這群海狸想選一棵大樹作為開會地點，希望讓住最遠的海狸從巢穴到這棵樹的距離最短。請問應該選哪一棵大樹開會呢？

A. A

B. B

C. C

D. D

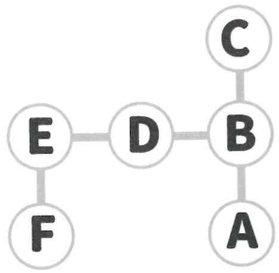
E. E

F. F

正確答案是：D

因為每段小徑長度都相同，我們可稱為 1 單位長距離。先讓每隻海狸都走 1 單位到最近的大樹，我們就可以忽略巢穴的位置，聚焦在這些大樹。

我們將大樹與小徑簡化表示如下：每棵大樹用一個有標記的圓圈表示。



上圖中像 A 這樣只連接到另一個圓圈 B，若選在 A 開會，走到 B 的海狸必須再走 1 單位，已到 C、D、E、F 的海狸也都得先走到 B 再多走 1 單位；反之，若選在 B 開會，換成走到 A 的海狸要再走 1 單位，但到 C、D、E、F 的海狸只要走到 B，比走到 A 短。所以若想讓其他海狸（例如在 D）減少移動距離，在 B 開會比在 A 開會好；因此讓走到 A 的海狸移動到 B。

C 跟 F 也是只連接到另一個圓圈，根據上述分析，想減少其他海狸的移動距離，在 B 開會比在 C 開會好，在 E 開會比在 F 開會好。因此讓走到 C 的海狸移動到 B，走到 F 的海狸移動到 E。

最多移動 2 單位後，海狸走到的大樹可簡化如下圖所示。

接下來，在簡化的圖中，應該選擇 D 作為會議地點，這樣已到達 B 跟 E 的海狸都只需要再多走 1 個單位。

選擇在 D 開會，最遠的海狸從巢穴出發，只需走 3 個單位，為最短距離。

資訊科學上的意義

圖 graph 是由點（nodes）和邊（edges）組成的資料結構，而 **樹 tree** 則是一種沒有環的連通圖，常用來表示層級結構。在樹中，上層的節點稱為父節點（parent node），下層為子節點（child node）；最上層是根節點（root node），最下層沒有子節點的稱為葉節點（leaf nodes）。

本任務中的海狸聚落地圖，就是一個沒有環的連通圖，也就是一棵樹。我們的目標是找出這棵樹的 **樹中心 center of a tree**，也就是讓所有海狸到達此點的最長路徑長度最短的節點。這樣可以讓每隻海狸從自己家出發，以最少步數到達開會地點。尋找樹的中心點，通常可以透過逐步從最外層往內層縮減的方式來進行：不斷移除葉節點，直到剩下一或兩個節點時，這些剩下的節點就是整棵樹的中心。這個方法能有效找出整體路徑最平均、距離最短的位置。

這種「找中心點」的想法，在日常生活中有許多應用，例如：城市交通規劃中可以用來決定大家通勤時間最短的交會點，物流配送中可作為最佳倉儲中心位置，無線基地台部署或網路伺服器設計中，也能以此提高整體覆蓋率與傳輸效率。

關鍵字

圖、樹中心

14. 題組一 - 朋友 A

新學期海狸班有 7 位同學（A ~ G）。有些人以前就認識，在網路上已互加好友，有些人彼此還不認識。

老師逐一訪問同學，在以下記錄表中，用打勾 ✓ 代表兩位同學已互加好友可以私訊。

	A	B	C	D	E	F	G
A		✓	✓				
B	✓						✓
C	✓				✓		
D						✓	
E			✓				
F				✓			
G		✓					

每隻海狸如果收到學校的通知，就會立刻傳私訊給認識的同學；而收到私訊的海狸們也會立刻轉發訊息給其認識的所有同學。

今天放學後，學校發了一則訊息給班長 G。請問經過轉發後，以下哪隻海狸<u>不會</u>收到這則訊息？

A. 海狸 A

B. 海狸 C

C. 海狸 D

D. 海狸 E

15. 題組二 - 朋友 B

新學期海狸班有 7 位同學（A ~ G）。有些人以前就認識，在網路上已互加好友，有些人彼此還不認識。

老師逐一訪問同學，在以下記錄表中，用打勾 ✓ 代表兩位同學已互加好友可以私訊。

	A	B	C	D	E	F	G
A		✓	✓				
B	✓						✓
C	✓				✓		
D						✓	
E			✓				
F				✓			
G		✓					

每隻海狸如果收到學校的通知，就會立刻傳私訊給認識的同學；而收到私訊的海狸們也會立刻轉發訊息給其認識的所有同學。

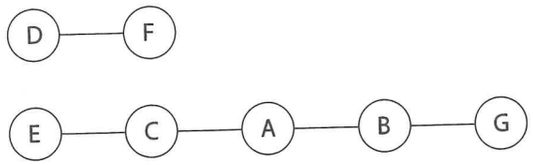
今天放學後，學校發了一則訊息給班長 G。請問經過轉發後，有幾隻海狸<u>不會</u>收到這則訊息？（範圍 [1~6] 的整數）

題組一 - 朋友 A 正確答案是：C
題組二 - 朋友 B 正確答案是：2 隻

我們可以將班長做的記錄表轉換成圖，顯示班上海狸間的好友關係。

把海狸用圓圈表示，表格中只要有打勾，表示彼此是好友，就將 2 個對應的圓圈用一條線連接起來。

例如第一橫列的 A 認識 B 和 C，就將 A 連接 B，A 也連接 C，以此類推最後會得到以下的圖：



從上圖可看出圓圈分為 2 組各自有線連接，但對另一組都沒有線連接。表示班上海狸分成 2 個群組，A、B、C、E、G 會收到彼此之間轉發的訊息，但 D、F 只會收到 2 人彼此傳送的訊息，沒有管道可轉收到由班長 G 發出的訊息。

所以 D 和 F 這 2 隻海狸不會收到班長的訊息，其他海狸都會收到。

資訊科學上的意義

圖 graph 是資訊科學中重要的抽象工具，常用來表示實體之間的關係。圖由節點（nodes）和邊（edges）組成，能模擬人際關係、網路連線、路徑規劃等各種結構。在本任務中，每位同學可視為一個節點，好友關係是互相的，因此可用 **無向圖 undirected graph** 來表示，也就是連接線是雙向的。

相鄰矩陣 adjacency matrix 是記錄圖結構的一種方式，使用一個方形表格來標示節點間是否有連結。若 A 和 B 是朋友，則矩陣中第 A 行第 B 列與第 B 行第 A 列都記為 1；若不是朋友，則記為 0。在本任務的表格中，「有打勾」就代表 1，「沒打勾」則是 0。因為是無向圖，這個矩陣會是對稱矩陣（symmetric matrix）。透過相鄰矩陣記錄的資訊，我們可以判斷哪些同學彼此有直接或間接的好友連結關係，哪些同學無法透過朋友接收到訊息。

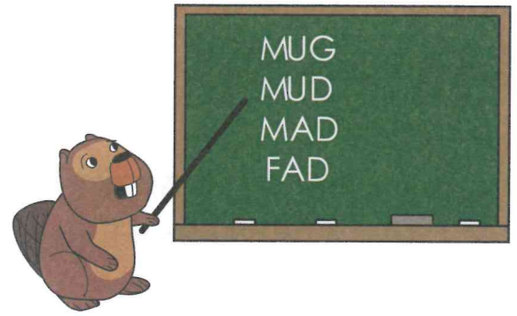
圖的概念在生活中有許多應用。例如在社群平台中，圖可以分析人際關係並推薦可能認識的朋友；在疫情模擬中，圖可模擬病毒透過接觸擴散的路徑；在網路與路由設計中，圖結構則能幫助檢查裝置之間是否有有效連線，確保資料能順利傳輸。

關鍵字

圖、無向圖、相鄰矩陣

16. 字詞鏈

大衛先生正在教學生閱讀。為了幫助他們學習，他創建了字詞鏈，也就是有一系列的單詞，其中每個單詞只改變其中一個字母就會形成下一個單詞。例如，MUG → MUD → MAD → FAD 就是一個字詞鏈。



大衛先生正要教以下九個單詞：BOT、SAD、BAT、CAB、COT、BAD、COB、CAT 和 SAT。他想將這些單詞分成三組，各組都包含三個單詞，每個單詞都只出現在某一組，且各組皆可形成長度 3 的字詞鏈。

以下每個選項中的 3 個單詞組合雖然都可形成一條字詞鏈，但若選了該組合，會導致剩下的 6 個單詞無法產生另外兩條包含三個單詞且不重複用字的字詞鏈。

請問大衛先生不應該先選哪個組合？

A. SAD → BAD → BAT

B. COT → COB → CAB

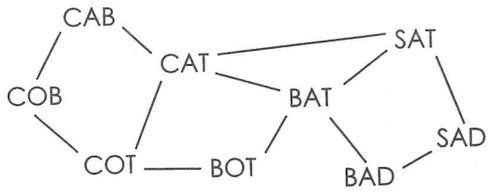
C. BAT → CAT → COT

D. CAT → CAB → COB

正確答案是：C

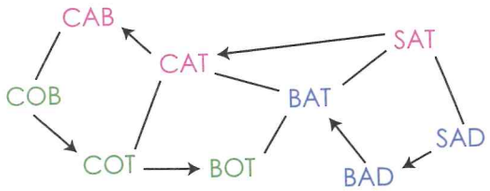
為了解決這個問題，可以繪製一張圖來表示單詞間的關係。

在這張圖中，兩個單詞若只有一個字母不同，就畫一條線把這兩個單詞連起來，表示它們可以在字詞鏈中相互接續。所以可以找到兩條線把三個單詞連起來，就形成一條字詞鏈。



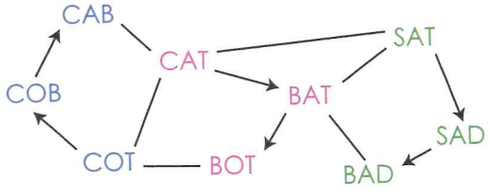
- 選項A

  如果以選項 A 的 SAD → BAD → BAT 形成一條字詞鏈，另外兩條字詞鏈可以是 SAT → CAT → CAB 和 COB → COT → BOT，如下圖所示。所以選項 A 是可選的組合。



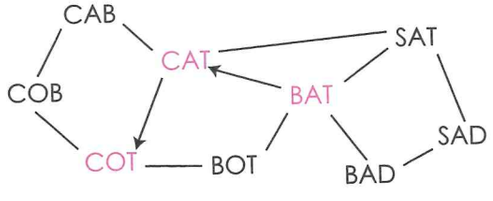
- 選項B

  如果以選項 B 的 COT → COB → CAB 形成一條字詞鏈，另外兩條字詞鏈可以是 CAT → BAT → BOT 和 SAT → SAD → BAD，如下圖所示。也可以是 CAT → SAT → SAD 和 BAD → BAT → BOT。 所以選項 B 是可選的組合。



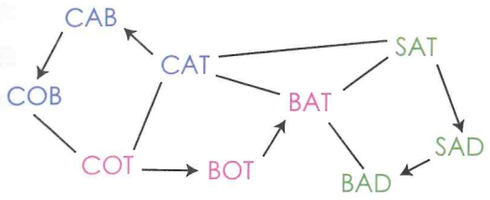
- 選項C

  如果以選項 C 的 BAT → CAT → COT 形成一條字詞鏈 ，根據圖中剩下的字詞，雖然 SAT → SAD → BAD 可形成另一條字詞鏈，但在不重複用字的情況下，BOT 則無法跟 CAB 及 COB 形成字詞鏈。因此不應該用選項 C 這個組合。



- 選項D

  如果以選項 C 的 BAT → CAT → COT 形成一條字詞鏈 ，根據圖中剩下的字詞，雖然 SAT → SAD → BAD 可形成另一條字詞鏈，但在不重複用字的情況下，BOT 則無法跟 CAB 及 COB 形成字詞鏈。因此不應該用選項 C 這個組合。



資訊科學上的意義

本任務運用了 **圖論 graph theory** 中尋找簡單路徑 (simple path) 的概念。首先，每個單字可視為圖中的節點（node），若兩個單字僅差一個字母，則在兩者之間建立一條邊（edge），代表它們可互相轉換並連結成字詞鏈。在此無向圖結構中，每條長度為三的字詞鏈對應到圖中的一條簡單路徑（simple path），而此任務是要在此圖中尋找三個互不重疊且長度為 2 的簡單路徑子圖，完成整體分組。

在比對字詞是否能相連時，任務中判斷兩個固定長度的字串是否僅差一個字母，應用到 **海明距離 Hamming distance** 的概念。這種距離比對方式常用於自然語言處理、錯誤校正與相似字建議系統，來處理拼字變化與相似字詞辨識問題。例如當你在手機或電腦輸入英文單字時，輸入系統會檢查你輸入的單字是否正確；若不正確，它會推薦「最接近的」正確拼法單字。

此外，本任務要求對九個單字中分組，且每組都須符合特定連結條件，屬於一種 **組合最佳化 combinatorial optimization** 問題。我們需在所有可能的三個詞組合中，找出能夠覆蓋全部單字且彼此不重複的三組合法字詞鏈。這類問題與集合分割、圖著色與子圖分解問題相關，常見於資源分配、網路分群與排程演算法設計等領域。

關鍵字

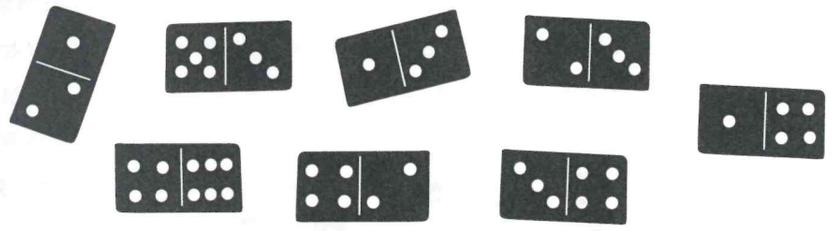
圖、海明距離、組合最佳化

17. 骨牌遊戲

多米諾骨牌是一種長方形的骨牌，上面的圖案為兩個正方形接在一起；且每個正方形內都有畫出點數，分別為 1、2、3、4、5 或 6 點。

骨牌遊戲的規則如下：只有當兩個骨牌連接端的正方形點數相同時，兩個骨牌才可以相鄰擺放。例如，下圖左邊的兩個骨牌可以相鄰擺放，因為連接端都是點數 3。而右邊的兩個骨牌，因為連接端的點數分別是 3 和 4，這樣擺放是違反規則的。

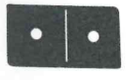
如果已經有以下 8 個多米諾骨牌：

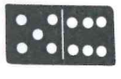


請問要再加上哪個骨牌，可以將 9 個骨牌依照遊戲規則接續擺放成一整列？

A. 

B. 

C. 

D. 

正確答案是：D

對已有的 8 個骨牌上出現的點數，分析他們各出現在幾個骨牌中：

具有點數	骨牌個數
	3
	3
	4
	4
	1
	1

若要讓骨牌連續排成一列，由於排列在中間的骨牌，它出現的點數一定要跟左側 / 右側骨牌相鄰的點數一樣，所以出現這些點數都要有偶數個骨牌。如果某種點數出現的骨牌數是奇數，一定要擺在最左邊或最右邊，所以這樣的點數只能有 2 種。

目前我們有 4 種點數：  、  、  及  出現的骨牌數是奇數。在四個選項中，只有添加  後能使出現的骨牌數是奇數的點數恰好為 2 種：  跟  。

因此添加骨牌後，這 9 個多米諾骨牌可排列成以下一整列：

資訊科學上的意義

在資訊科學中，**尤拉路徑 Eulerian path** 是**圖論 graph theory** 中的一個經典概念，它指的是一條經過圖中每一條邊恰好一次且不重複的路徑。要判斷一張無向圖是否存在尤拉路徑，有一個簡單的判斷方式：整張圖中，最多只能有兩個節點邊數（也就是連接數，degree）為奇數，其餘節點的連接數都必須是偶數。這項特性可以幫助我們快速判斷：是否可能找到一條連貫不重複的路徑，而不需要全部窮舉。

在本任務中，每個點數可看作是圖中的節點，而一張骨牌中出現的兩種點數，就可將這兩個對應節點間建立一條邊。題目要求選出一張適合的補充骨牌，使得九張骨牌能依照「相同點數相接」的方式排成一條長列，這其實就是要在圖中找出一條尤拉路徑的過程。

解題時，我們只要統計每種點數有出現的骨牌數 (相當於在圖中的連接數)，加上某一張骨牌後，只要剛好有兩種點數的出現次數為奇數，而且形成一個連通圖 (任兩個節點都有路徑可相連)，那就一定能找到一條尤拉路徑，以這兩個節點作為尤拉路徑的起點或終點 (排在最左 / 右邊)；而那些出現次數是偶數的點數，則只可能是路徑經過的中間節點。

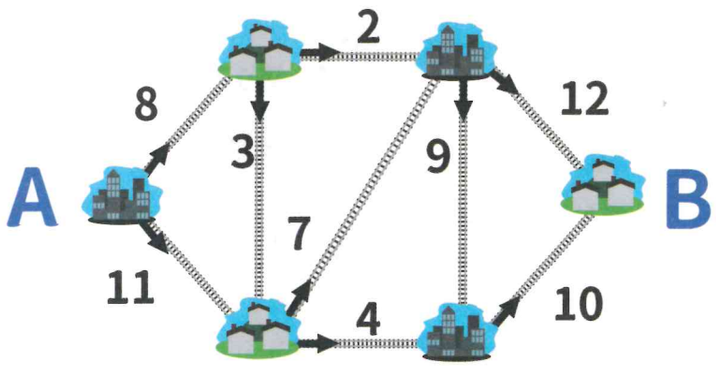
在日常生活中，當我們遇到「每條路都要走到，但不能重複走」的情況時，其實就是在找一條尤拉路徑。舉例來說：學校園遊會你想要走過每條通道上的每個攤位，但又不想走重複的路，這時可以設計一條尤拉路徑，確保每條路線只走一次就能逛完所有地方。在電路設計過程中，工程師會根據電路功能決定哪些元件要相連（可視為圖的節點與邊）；然後思考是否能以尤拉路徑的方式走完所有導線連結，也就是每條連線只走一次且不交錯；如果可行，就會依據這條路徑來決定元件的擺放位置，讓布線最短、交錯最少。

關鍵字

圖、尤拉路徑

18. 鐵路路線圖

下圖是海狸市的鐵路路線圖，圖中的數字代表每條鐵路每日最多可通過的列車數量，不同的列車可共用同一條鐵路，列車必須遵循鐵路箭頭方向前進。



請問每天可從 A 點出發到 B 點的列車數量最多為何？

A. 13

B. 15

C. 19

D. 22

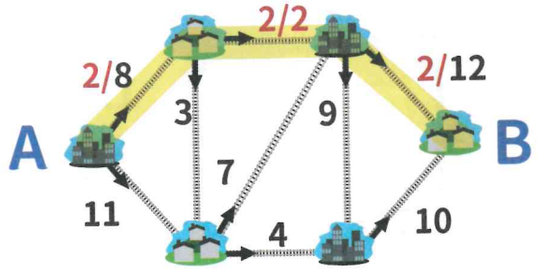
正確答案是：A

首先我們觀察 A 點每日最多只可能派出 8 + 11 = 19 台列車到 B 點，所以答案 (C) 跟 (D) 22 一定錯誤。

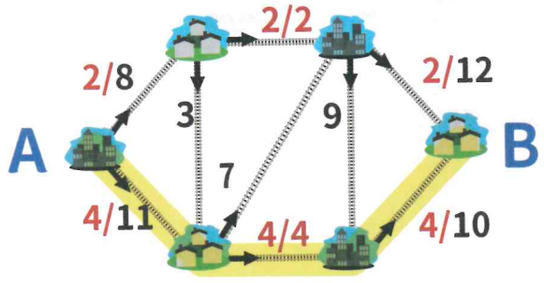
解決這個問題的方法之一，是漸次找出可從 A 點到 B 點的鐵路路線 (也稱路徑)，並累積能通過的最多列車數。

從 A 點到 B 點，有以下 3 條用黃色粗線顯示之鐵路路線：

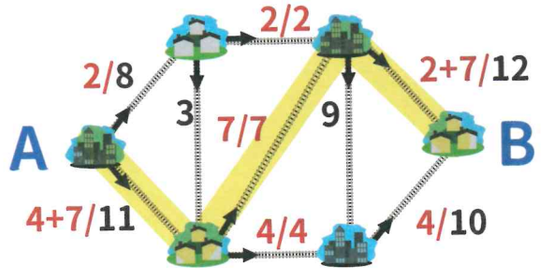
1. 路線 1：能通過的列車數量最多為 2；



2. 路線 2：能通過的列車數量最多為 4；



3. 路線 3: 能通過的列車數量最多為 7；



所以透過這 3 條路線，最多可通過 2 + 7 + 4 = 13 台列車。

我們要如何確認 13 台列車是最多的數量？

如果考慮經過四個路段或五個路段的路徑，會發現以目前的安排，從 A 點到 B 點中間要經過的路段都已達到通車上限，所以無法再派出更多列車從 A 點到 B 點。

資訊科學上的意義

在資訊科學中，**最大流問題 maximum flow** 是一個經典的圖論優化問題。它的目標是：在一個有向圖（directed graph）中，找出從來源點（source）到匯點（sink）所能通過的最大流量。圖中的每條邊代表一條通道，例如表示鐵路、公路、水管或電纜，並且有給定容量限制，也就是最多能通過多少單位的流量。我們必須在不超過容量的情況下，想辦法安排最好的流動方式，達到指定點間的最高輸送量。

在本任務中，城市的鐵路系統可以被看作一張有向圖：每條鐵路對應到一條邊，而該鐵路能通過的最大列車數量就是這條邊的容量。任務的目標是算出每天最多可以從 A 點運送多少列火車到 B 點，這其實就是要計算從來源到目的地的「最大流量」。

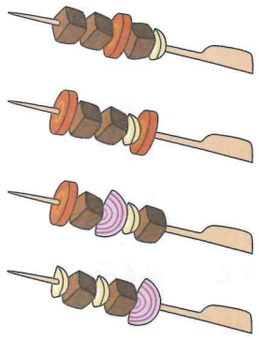
最大流問題在現實生活中有許多實際應用，因為很多系統都像圖一樣，由節點與有容量限制的通道組成。舉例來說：在電力網路中，發電廠與用電地區透過高壓電線輸送，每條電線能承受的電流量有限。最大流模型可用來找出在不超載的情況下，從電廠輸送到用戶端的最大供電能力，並協助在尖峰時段的電力調配，避免斷電。在網路傳輸中，資料從伺服器經過多條路由器與線路傳送到使用者端，每條線路的頻寬也就是其容量。最大流演算法可以幫助網路系統從規劃設置的線路及頻寬，估算最多能同時傳送多少資料，讓網路服務保持順暢減少延遲。

關鍵字

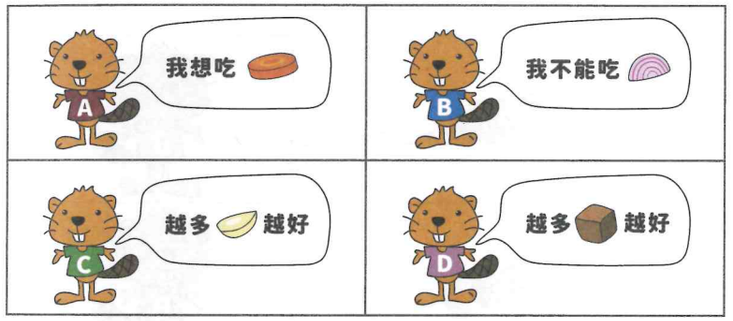
圖、最大流問題

19. 燒烤派對

海狸派對中烤好了以下四根烤肉串。



海狸們分別說出他們想吃的肉串要求：



經分配後四隻海狸後都有拿到想吃的肉串，請問海狸 A 拿到哪一根肉串？

A. 

B. 

C. 

D. 

正確答案是：C

依據 4 隻海狸的要求，烤肉串的食材限制如下：

- 海狸 A：至少有 1 塊 。
- 海狸 B：不能有 。
- 海狸 C： 越多愈好。
- 海狸 D： 越多愈好。

我們將每根烤肉串依序編號以方便說明。

1 號烤肉串有最多 ，所以分配給 D 。

4 號烤肉串有最多 ，所以分配給 C 。

剩下的 2 號和 3 號烤肉串都至少有 1 塊 ，但因為 3 號有 ，但 B 不想有 ，所以把 2 號烤肉串分配給 B 、3 號烤肉串分配給 A 。

分配結果如下表：

1		
2		
3		
4		

如果海狸 A 分配到其他編號的烤肉串，都會造成其他有海狸無法拿到想吃的肉串，因此海狸 A 拿到 3 號烤肉串是唯一的可能。

資訊科學上的意義

在資訊科學中，當所解任務需要在多個變數上同時滿足一系列限制條件時，這類問題被稱為 **條件約束問題 constraint satisfaction problem**。常見的求解策略包含先決定最具約束力的變數（例如和最多限制相關的變數），逐步縮小可能的解空間，並在必要時透過回溯法修正不合法的選擇，直到找出一組滿足所有限制條件的解答為止。

在本任務中，每隻海狸可以視為一個變數，烤肉串的種類構成其可能的取值範圍，而海狸們的口味偏好則形成限制條件。我們可以根據這些條件逐一為海狸分配符合要求的烤肉串；若在指派過程中發現某個限制無法被滿足，則需回退先前的選擇，重新嘗試其他可能的分配組合，直到所有海狸的偏好都能同時被滿足為止，即為一組正確配對。

條件約束問題在生活中的應用很常見，例如餐廳在接受客戶訂位時，需要根據多項限制來安排座位：包括顧客人數與桌位大小的匹配、避免時間重疊、顧客偏好位置（如靠窗、非吸菸區）等。餐廳必須根據這些限制，為每組顧客分配最適合的座位；若發現無法滿足所有條件，則需回溯並調整之前的安排，直到所有訂位要求能被滿足為止。這與本任務中的解題策略是類似的。

關鍵字

條件約束問題

20. 生日日期

小路、小愛、小菲、小內、小莎都在同一年出生；以下 5 個日期是他們 5 人的生日：

3 月 17 日、4 月 4 日、4 月 29 日、5 月 1 日、5 月 4 日

我們不知道誰在哪一天出生，只知道以下事實：

- 小愛 比 小莎 早 5 天出生
- 小菲 是在 4 月出生的
- 小路 比 小菲 早出生
- 小內 比 小菲 晚出生

請問 5 月 1 日是誰的生日？

A. 小路

B. 小愛

C. 小菲

D. 小內

E. 小莎

本任務解析由 2024 運算思維推動計畫製作，係採用創用 CC 姓名標示 - 非商業性 - 禁止改作 3.0 台灣 授權條款授權。

正確答案是：D

3月17日	4月4日	4月29日	5月1日	5月4日

按照第一個提示，5 個日期中相差 5 天的只有「4 月 29 日」和「5 月 4 日」，且小愛是先出生的，所以可判斷出小愛的出生日期為「4 月 29 日」、小莎的出生日期為「5 月 4 日」。

3月17日	4月4日	4月29日	5月1日	5月4日
		小愛		小莎

第二個提示：小菲是在 4 月出生的，剩下 4 月的日期只有「4 月 4 日」，所以小菲的出生日期是「4 月 4 日」。

3月17日	4月4日	4月29日	5月1日	5月4日
	小菲	小愛		小莎

接下來依照第三個提示：小路 比 小菲 早出生，在小菲生日前的日期只剩「3 月 17 日」，因此是小路的出生日期。

3月17日	4月4日	4月29日	5月1日	5月4日
小路	小菲	小愛		小莎

最後一個提示：小內 比 小菲 晚出生，剩下的「5 月 1 日」是小內的出生日期。

3月17日	4月4日	4月29日	5月1日	5月4日
小路	小菲	小愛	小內	小莎

資訊科學上的意義

條件約束問題 constraint satisfaction problem 是資訊科學中的一種問題類型，其核心目標是在一組變數中，為每個變數選出一個合適的值，使整體同時滿足所有給定的條件限制。解這類問題的基本方式是：根據已知條件，一步步排除不可能的選項，慢慢縮小可能的答案，直到每個變數都只剩下一個符合條件的值可選。在這個過程中，會用到邏輯推理，也會利用「條件傳播」：意思是當確定了一個選項，其他有關的選項也會跟著受到影響，可能性變得更少。如果中途發現選錯了，使條件之間發生衝突，可以退一步重新選擇，這種方式叫做「回溯法」。透過這樣持續推理、排除和調整，就能找出一組符合所有條件的正確答案。

在本任務中，學生需要根據事實提示（例如：「小愛比小莎早 5 天」）來推理出每個人對應的正確生日。這些提示其實就是一組條件限制，而解題的過程就是找出一組滿足所有條件的日期與人物對應組合。這種根據條件逐步排除不合邏輯的選項，最終找出唯一解的推理過程，正是條件限制問題的應用。

在電腦系統中，條件限制問題常見於排程設計、邏輯推理、資源分配等情況。在日常生活也經常出現，例如：班級排課表要同時滿足教師、教室與時間的限制、排班系統需同時考慮人力、工時與假期的分配，又例如數獨或邏輯謎題等遊戲，解題的邏輯推演過程也跟此類問題一樣。

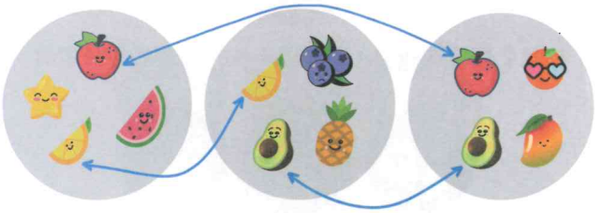
關鍵字

條件約束問題

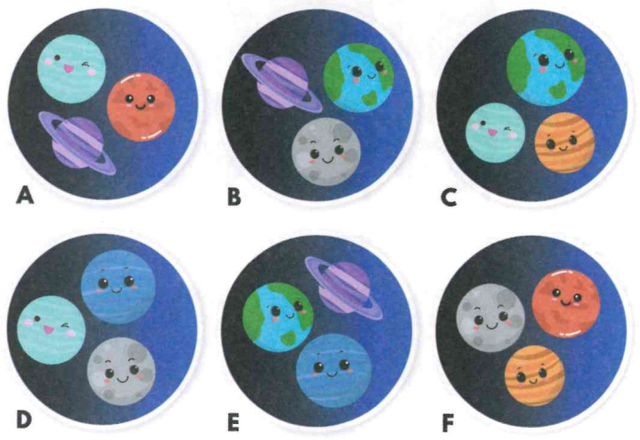
21. 重複圖樣

小狸想製作遊戲卡，遊戲卡上會有一些不同圖樣，任兩張卡片上的圖樣要有重複，但只能有一個重複。

例如，在這一組 3 張遊戲卡中，每張卡片上都有 4 個圖樣。每兩張卡片，都找得到恰好只有一個圖樣是相同的。



以下是小狸製作的 6 張卡片，要去掉哪張卡片才會符合遊戲卡的規定呢？

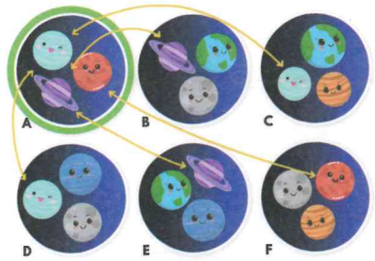


正確答案是：E

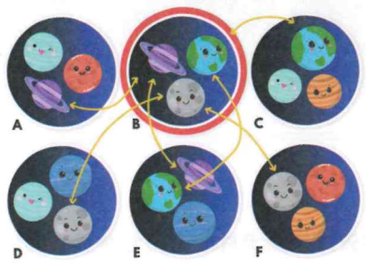
正確答案是移除卡片 E。

要回答這個問題，我們需要檢查每張卡片和其他 5 張卡片，看有多少圖樣是相同的。檢查結果如下：

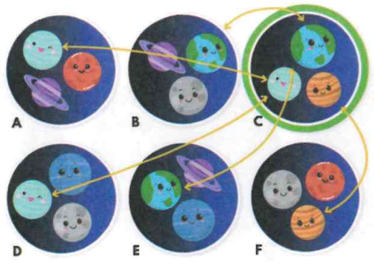
1. 卡片 A 和其他卡片都只有 1 個圖樣相同。



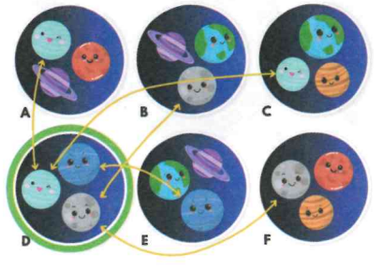
2. 卡片 B 和卡片 E 有 2 個圖樣相同，與其他卡片只有 1 個圖樣相同。



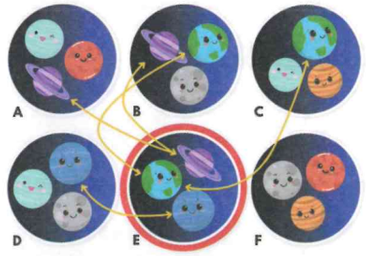
3. 卡片 C 與其他卡片都只有 1 個圖樣相同。



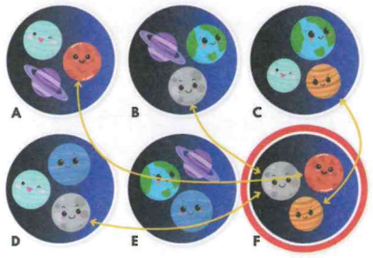
4. 卡片 D 與其他卡片都只有 1 個圖樣相同。



5. 卡片 E 與卡片 B 有 2 個圖樣相同，與卡片 F 沒有相同圖樣。

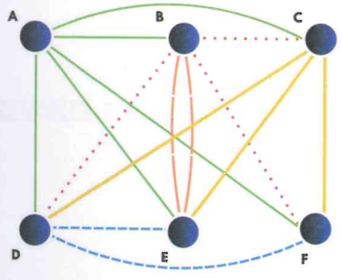


6. 卡片 F 與卡片 E 沒有相同圖樣， 與其他卡片都只有 1 個圖樣相同。

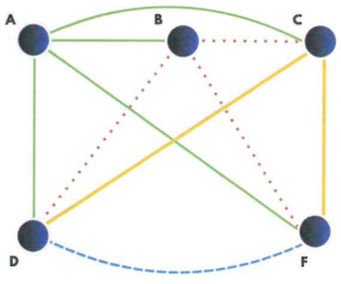


依照上述檢查結果，發現檢查狀況 (2)(5)(6) 有問題，發生問題的共同卡片是 E。將卡片 E 移除後，就符合遊戲卡規定了。

上述檢查狀況也可以用簡化的圖來表示，把六張卡片各用一個圓形節點表示（A、B、C、D、E 和 F），如果兩張卡片有相同的圖樣，對應節點就畫一條連線。在圖一中，以卡片 A 為例，用綠色線條表示卡片 A 與其他卡片有相同圖樣。顏色或線條的類型只是為了更清楚地看到節點之間的連線。重要的是每對節點之間必須只能有一條連線。



圖一



圖二

在圖一，可以更容易地看到卡片 B、E 和 F 存在的問題：

- 卡片 B 與卡片 E 有 2 個相同圖樣。
- 卡片 E 與卡片 F 沒有相同圖樣

因此，如果移除卡片 E，就會去掉圖表中的重複線條，且每個節點與其他節點間都正好只有一條連線。所以答案是移除卡片 E。

資訊科學上的意義

在資訊科學中，當解任務時需要同時滿足一系列條件或限制，這類型的問題被稱為 **條件約束問題 constraint satisfaction problem**。一項常見的解法是：在每次進行選擇前，先觀察有哪些選項不可能滿足條件，把這些選項提前排除掉。這種做法可以縮小需要嘗試的範圍，不必把所有可能都一一試過，節省求解的時間。

在本任務中，每兩張卡片都必須恰好只有一個相同圖樣，這即是一項明確的限制條件。在推導過程中，我們可以先比對所有卡片的配對情形，觀察是否有違反「只重複一個圖樣」的組合，例如某些卡片與多張卡片重複了兩個以上的圖樣（如卡片 B、卡片 E），或完全沒有重複圖樣（如卡片 F）。接著可以找出導致條件衝突的卡片（卡片 E），並將其移除，讓剩餘卡片能夠完全符合限制條件。

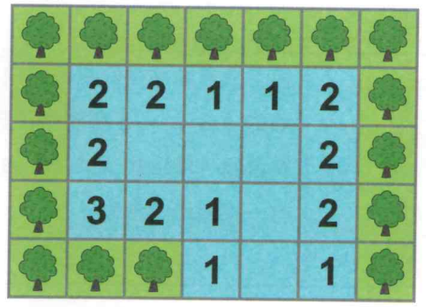
條件約束問題在生活中的應用非常廣泛。例如在學校排課時，每堂課可視為一個變數，授課時間與教室則為其可選擇的指定值，而教師、時間與場地不可重疊等則是限制條件。排課時需逐一檢查各課程安排是否有衝突，並在遇到限制衝突時進行調整，最終找出一組同時滿足所有條件的完整課表。

關鍵字

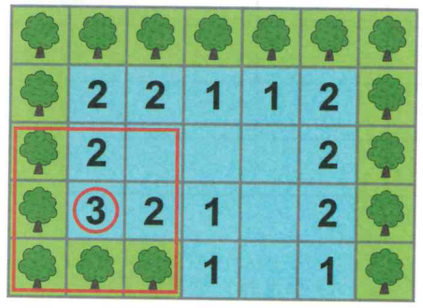
條件約束問題

22. 食物地圖

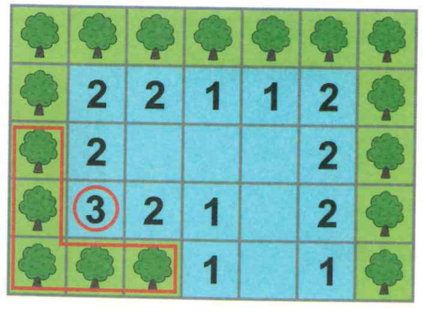
海狸尼克把食物藏在池塘岸邊 17 棵樹中的 9 棵樹下，他做了一張地圖，在池邊每個格子上寫了周圍的格子區域有藏食物的樹木數量。



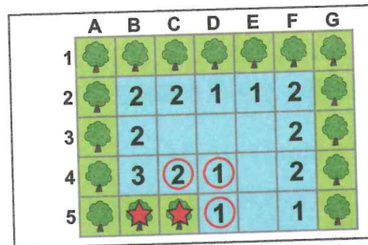
例如，下圖中標有「③」，表示這個格子周圍區域（紅色框起來）所接觸到的 5 棵樹 中，有 3 棵樹藏有食物。



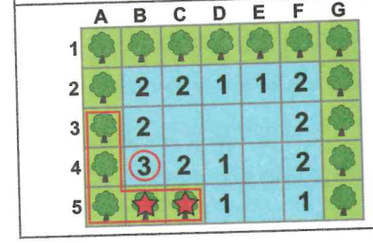
針對下圖框起的 5 棵樹中（分別以座標 A3,A4,A5,B5, 及 C5 表示），請問哪幾棵樹確定藏有食物？



正確答案是：A4B5C5



解題過程可以從左下角開始。首先，C5 樹下必須有食物，以滿足 D4 和 D5 位置上的「1」。B5 樹下也必須有食物，以滿足 C4 位置上的「2」。這兩棵樹如圖所示。



接下來考慮左圖紅框圈出來的樹，以滿足 B4 位置上記錄的「3」。5 棵樹中已確定有兩棵樹（B5 和 C5）有食物，所以在 A3、A4 和 A5 這三個格子的樹中，恰有一棵樹藏有食物。

將這些記錄的數字，依順時針方向推論其周圍格子，哪幾個格子的樹可能藏有食物：

線索	方格位置	數值	哪些方格中有幾棵樹可能藏食物
(1)	B4	3	A3、A4、A5 中有 1 棵
(2)	B3	2	A2、A3、A4 中有 2 棵
(3)	B2	2	A1、A2、A3、B1、C1 中有 2 棵
(4)	C2	2	B1、C1、D1 中有 2 棵

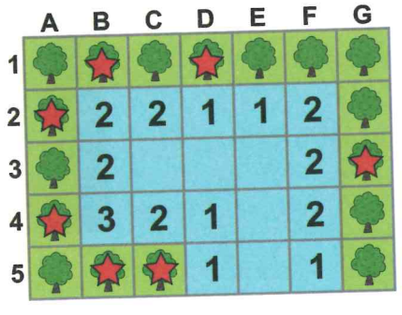
根據線索 (1)：

如果是 A5 中的樹有食物，表示 A3、A4 都沒有，這樣就不可能符合線索 (2) 在 B3 記錄數值 2。

如果是 A4 中的樹有食物，表示 A3 跟 A5 中都沒有，且根據線索 (2) 在 B3 記錄的數值 2，表示 A2 中的樹有食物。

如果是 A3 中的樹有食物，表示 A4 跟 A5 中都沒有，且根據線索 (2) 在 B3 記錄的數值 2，表示 A2 中的樹有食物。但這樣根據線索 (3) 會推論到 A1、B1、C1 都沒有，線索 (4) 就不可能滿足。

因此可以確定紅框圈出來的區域中，有食物的樹是在 {A4、B5、C5}。



如果繼續往下推論，

可以找出整張地圖中藏有食物的 7 棵樹

為右圖中標有  的樹。

資訊科學上的意義

在資訊科學中，當一項任務需要同時滿足多個條件或限制時，我們稱這類問題為 **條件約束問題 constraint satisfaction problem**。這類問題的解題核心是：根據已知條件，一步步推理並排除不可能的情況，最終找出能同時符合所有限制的解答。

在本任務中，每個格子上的數字代表該格子周圍相鄰區域中藏有食物的樹木數量，而這些數字就構成了解題時要遵守的限制條件。由於相鄰的格子可能會重疊計算到同一棵樹，因此我們可以先找出那些已確定藏有食物的樹木位置加以標記，然後根據剩下的條件，繼續推理出其他未知的食物位置。這種根據部分資訊、逐步縮小可能性、同時處理多組交錯限制的過程，就是條件約束問題最典型的解題方式。

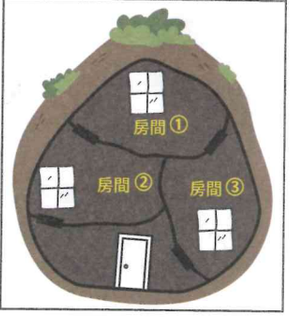
在生活中，有許多邏輯型益智遊戲，例如踩地雷、數獨 (Sudoku)、或填字遊戲，都是條件約束問題的實際應用。玩這些遊戲都需要根據部分已知的資訊，透過邏輯推理與條件分析，逐步推導出正確解答。

關鍵字

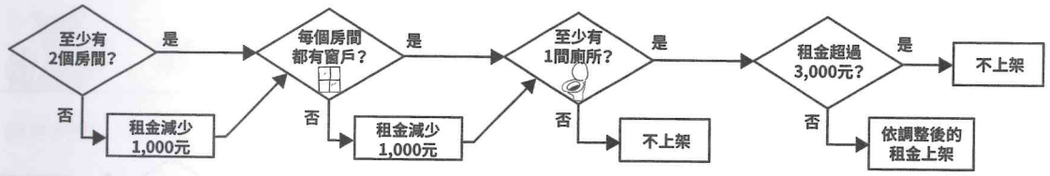
條件約束問題

23. 租賃網站

海狸奶奶擁有下面的三棟房子，她打算透過租屋網站出租：

	房子 1	房子 2	房子 3
奶奶提出的租金	$4,000	$4,000	$4,000
房屋平面圖			

租屋網站會根據以下的規則，調整屋主提出的租金後，再將房屋依調整後的租金上架：



根據租屋網站的規則，以下何者敘述正確？

A. 房子 1 和 3 各以 $3,000 的租金上架。

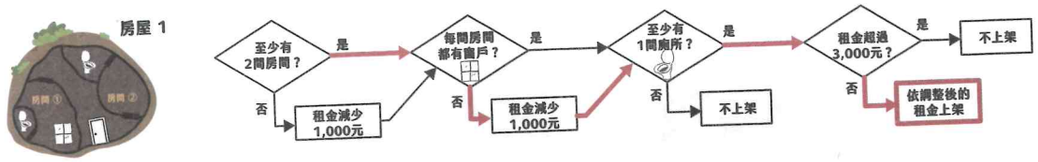
B. 房子 1 以 $3,000 的租金上架，其他房子不能上架。

C. 房子 1 以 $3,000 的租金，房子 2 以 $2,000 的租金上架。

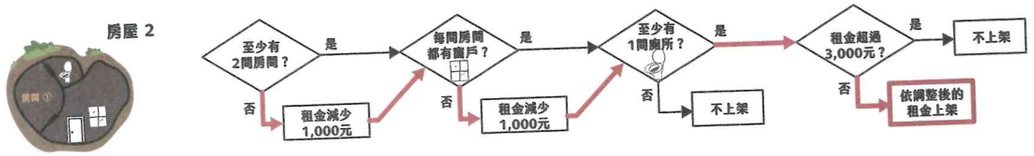
D. 所有的房子都不能上架出租。

正確答案是：C

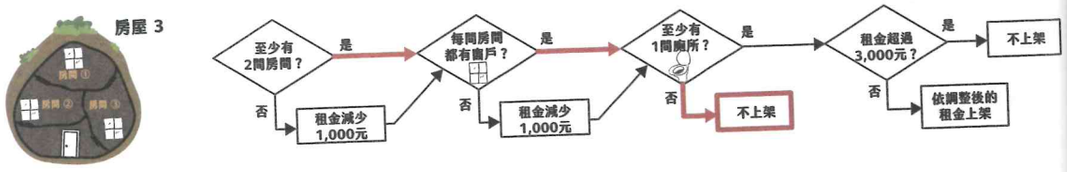
按照租屋網站的規則，房子 1 因為有一個房間沒有窗戶，租金從 $4,000 減少到 $3,000，但其他條件都符合，且租金未超過 $3,000，所以依調整後的租金 $3,000 上架出租。



房子 2 因為房間數量不足且有一個房間沒有窗戶，租金從 $4,000 降低到 $2,000，租金未超過 $3,000，也可以上架出租。



房子 3 因為沒有廁所，所以不能上架出租。



選項中只有 C 的敘述完全正確。

資訊科學上的意義

流程圖 flowchart 是一種常用來表示演算法、工作流程或程序處理邏輯的圖形工具，經常應用於程式設計與系統規劃中。其中，條件判斷 conditional statements 通常以菱形符號表示，用來判斷某項條件是否成立，並依據結果決定後續的執行路徑。在程式語言中，條件判斷的語法形式通常為：「如果…那麼…否則…」（英文為 `if ... then ... else ...`），讓程式能根據不同條件是否成立的情況執行不同內容。

在本任務中，使用流程圖來判斷一間房屋是否需要調整租金，並最終決定該房屋是否符合條件，可以上架至租賃網站。網站的審核條件包含四項限制：「至少有兩個房間」、「每個房間都有窗戶」、「至少有一間廁所」以及「租金不得超過 3000 元」。我們必須根據這些條件逐一檢查，並依照流程圖判斷是否需要修改房屋資訊，最後決定是否符合上架標準。

在生活中，我們常常也會遇到「如果…那就…，否則…」的情境。像是如果今天下雨，就要帶傘；如果作業寫完了，就可以玩電腦；如果訂了午餐，就能去領便當。這些就是條件判斷的邏輯：根據不同的條件，做出不同的行動選擇。電腦程式也是一樣，只是用程式語言把這些條件寫下來，讓電腦自動判斷要做什麼；例如智慧家電會根據環境的溫度或濕度，自動開啟或關閉特定功能。

關鍵字

條件判斷、流程圖

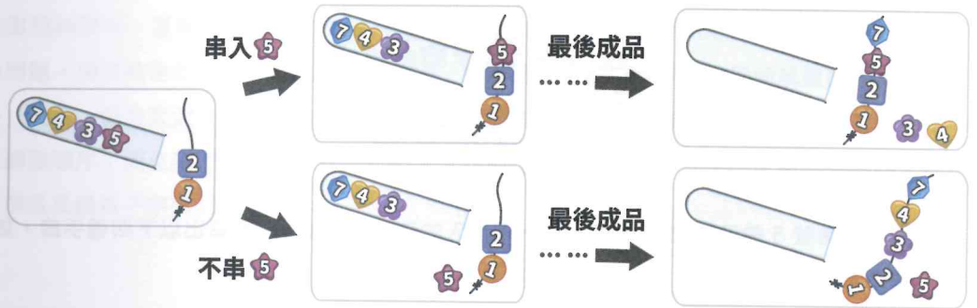
24. 串珠手鍊

海狸正在製作手鍊。他會從管子裡一次取出一顆有編號的珠子。取出的珠子如果要串到手鍊上，必須符合以下要求：

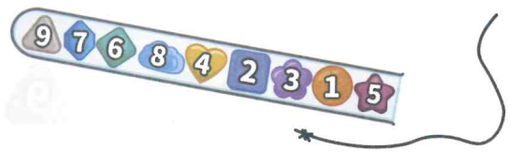
1. 手鍊上還沒有珠子。
2. 編號要比手鍊上最後一顆珠子的編號更大。

不過即使珠子符合要求，海狸仍可以選擇要串入手鍊，或者放到旁邊不串。

舉例如下圖：當手鍊上最後一顆珠子的編號是 2，取出的 5 號珠子符合可串入手鍊的條件，但海狸可選擇串入手鍊（上方圖示），或者放到旁邊不串（下方圖示），最後串出的手鍊就會不一樣。



海狸正準備用以下這個管子中的珠子製作一條手鍊，請問最多可以把幾顆珠子串入手鍊？



A. 3 顆珠子

B. 4 顆珠子

C. 5 顆珠子

D. 6 顆珠子

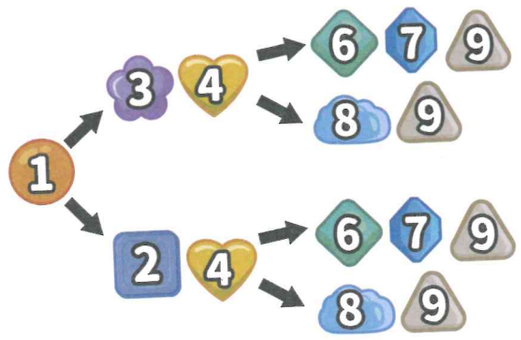
E. 7 顆珠子

正確答案是：D

根據規則，可能串出的手鍊樣式如以下幾種，再看看哪個手鍊的珠子數量最多。

1. 第一顆珠子是編號 5，如果把它串上手鍊，之後只有編號 6、7、8、9 的珠子能串入，串出的手鍊中珠子編號可能依序是 5679 (不把 8 串入) 或 589，如下圖所示。

2. 如果把編號 5 的珠子 放到一旁，第一顆串入的珠子是編號 1，會串出以下四種手鍊，如下圖所示：



(1) 接下來串入 3 號珠子，會串出的手鍊中珠子編號為 134679 或 13489。

(2) 如果不把 3 號串入，接下來串入 2 號珠子，
會串出的手鍊中珠子編號是 124679 或 12489。

可發現最多能串出有 6 顆珠子的手鍊，編號依序是 134679 或 124679，所以答案是 (D)。

資訊科學上的意義

本任務的目標是找出一個最長的手鍊串法，而且串入的珠子數字必須由小到大排列。在資訊科學中，這種從一串給定數字序列中挑出部分元素使它們依序遞增，並找出其中最長可能的子序列，稱為**最長遞增子序列 longest increasing subsequence** 問題。解這類問題時，除了列舉出所有答案，也可以採用**動態規劃 dynamic programming, DP**。動態規劃的想法是：把一個大問題拆成一些較小的子問題，並把各子問題找出的答案記下來，避免重複計算。這樣就可以用比較少的計算，把原本複雜的問題解決掉。

在本任務，子問題是找拿到每顆珠子為止能形成的最長遞增子序列。每當新拿出一顆珠子，都有兩種選擇：要不要把這顆珠子串入手鍊裡。這時候便可根據前面已找出的最長遞增子序列，比較哪一種選法配上目前的珠子可以讓手鍊串得更長，然後一步步推算下去，直到最後一顆珠子。記錄前面每個位置的最佳結果，可避免重複計算，這樣的策略是動態規劃處理最長遞增子序列問題的典型做法。

在資訊科學中，最長遞減子序列 longest decreasing subsequence 與 最長遞增子序列是類似的問題，兩者的差別只在於所選出的子序列元素為遞減或遞增，可運用相同解題概念。以「打包盤子」為例，假設我們依序收到不同尺寸的盤子，每次只能決定是否要將這項物品放進箱子中，而不能調整順序；而且我們希望後放進去的物品尺寸要比前一件更小，以免壓壞脆弱的物品，就是在解一個最長遞減子序列問題。

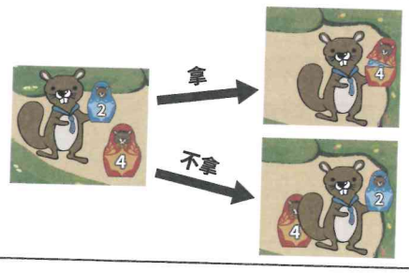
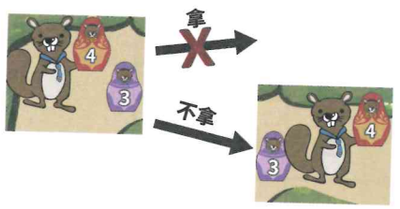
關鍵字

最長遞增子序列、動態規劃

25. 娃娃迷宮

娃娃迷宮地圖如下，裡面散落著不同號碼的娃娃。號碼越大，娃娃就越大隻。

小狸想蒐集散落在迷宮各處的娃娃並穿過迷宮到出口，但是在迷宮中，她必須按照箭頭方向移動，並依照下列規則決定是否帶走地上的娃娃：

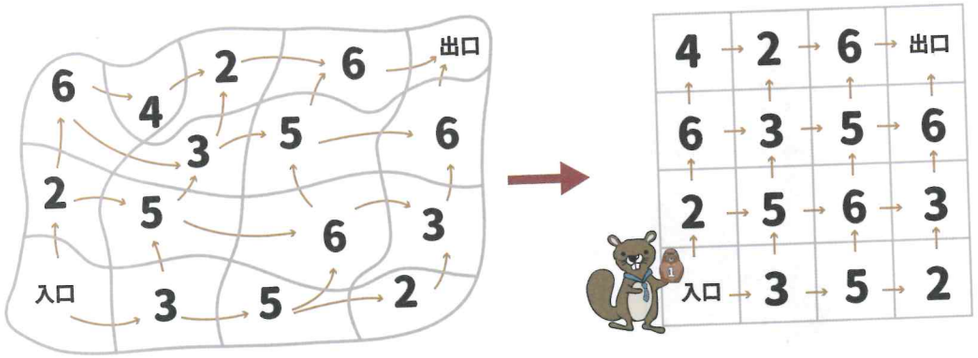
	如果地上的娃娃號碼比她已攜帶娃娃中的最大號碼大，她可以選擇帶走或不帶走。
	否則，她就不能帶走這個娃娃。

小狸在進入迷宮時攜帶著一個 1 號娃娃，請問她在離開迷宮時，最多能蒐集到多少個娃娃（包括她最初帶的 1 號娃娃）？（範圍 [1~6] 的整數）

本任務解析由 2024 運算思維推動計畫製作，係採用創用 CC 姓名標示 - 非商業性 - 禁止改作 3.0 台灣 授權條款授權。

正確答案是：5

依照任務畫出迷宮的簡化地圖如下（所有箭頭都指向右或向上走）。

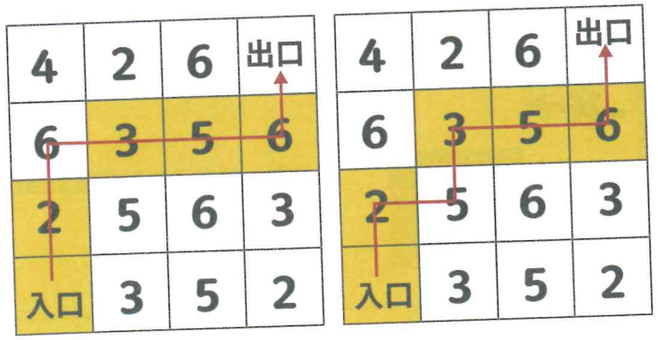


從入口到出口的所有路徑都是 5 步長，因此，若小狸在每一步都可帶走娃娃，包含一開始帶的娃娃，最多能蒐集到 6 個娃娃。

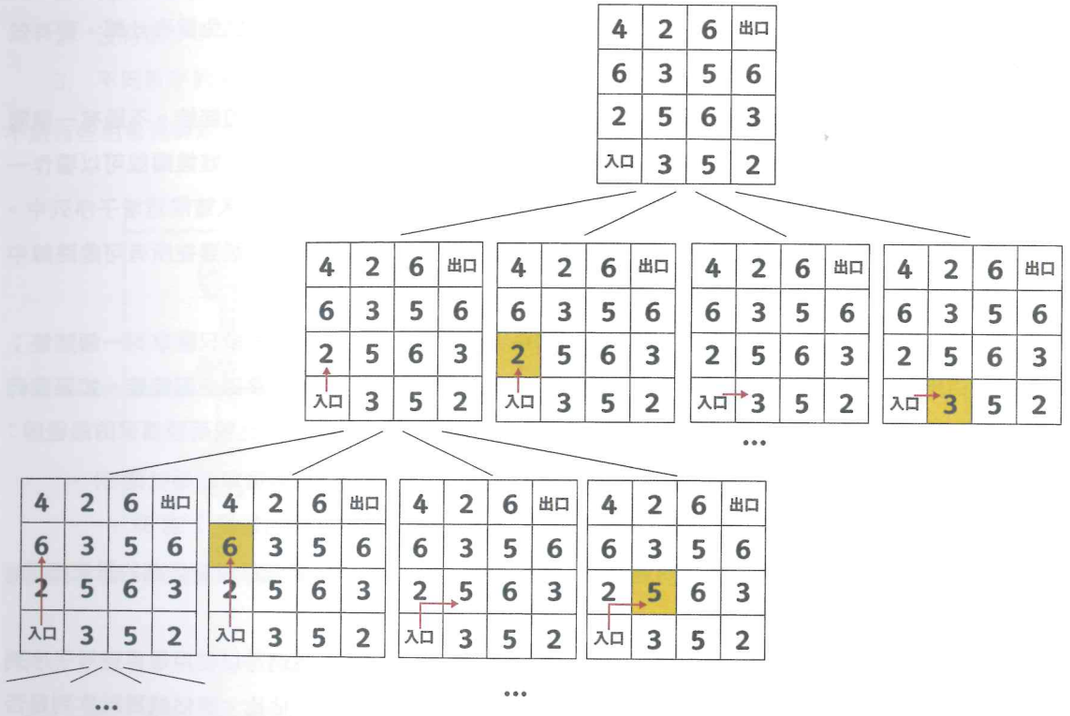
但是要根據規則，娃娃要帶走就必須按 1、2、3、4、5、6 的號碼，由小而大的順序蒐集。

在地圖上，若是要帶 4 號的娃娃，之後就沒有路徑可以再拿到 5 號的娃娃了，所以在這個娃娃迷宮中，小狸不可能同時帶走 4 號和 5 號的娃娃。這表示她最多能蒐集到 5 個娃娃。

以下顯示兩個能蒐集到 5 個娃娃的路線（路線上的白色格表示不帶走那個娃娃）。



用電腦解題時，會將所有從入口到出口能行進的路線及帶或不帶娃娃的狀況列出來，類似下面樹狀圖展開的方式，每一層是多走一步的可能情況，來完整推導小狸在迷宮中可能的行進路線及是否帶走娃娃的選擇。



資訊科學上的意義

在資訊科學中，從一串資料序列中挑選出部分元素，使這些資料依序遞增，並形成一段最長的子序列，這樣的任務稱為 **最長遞增子序列 longest increasing subsequence** 問題。這類問題可以透過一種稱為 **動態規劃 dynamic programming, DP** 的技巧來加速解題。動態規劃的核心概念是：將原本複雜的問題拆解為較小的子問題，並記住這些子問題已經算過的結果，從而避免重複計算，更有效率地找出整體的最佳解答。

在本任務中，小狸要在娃娃迷宮中選擇一條行走路線，並盡可能地拿到最多的娃娃。不過有一個限制：每次拿到的娃娃編號必須依照從小到大的順序。因此，每條路線上出現的娃娃編號可以看作一個數字序列，而每到一格，是否拿起該娃娃，就等同於在選擇是否將該編號納入這段遞增子序列中。由於從起點到終點有多條可能路線，我們不只要計算一條路線的最佳解，還需要在所有可能路線中找出最好的那一條。

舉例來說，從入口走到入口上方的位置，唯一的路徑是「向上走一步」，最多只能拿到一個娃娃；而從入口走到右方的位置，也只有「向右走一步」的路徑，同樣最多只能拿到一個娃娃。如果我們要到第二排的第二格，可能是「從下方來」或「從左方來」，這時我們就要比較兩種情況的最佳解：

- 若拿起該位置的娃娃，可以延續前面的遞增序列，得到長度為 2（最後拿的編號是 5）。
- 若不拿，則維持前一步的最長遞增子序列長度為 1（最後拿的編號可能是 2 或 3）。

這樣一步一步地計算每個位置從不同路線來，以及「拿或不拿」的兩種情況下所能得到的最佳結果，我們就能逐步構建整體的最佳解。

這種分析方法也經常運用在不同領域。例如在觀察資料變化趨勢時，我們可以使用最長遞增子序列演算法來找出資料中最長的上升趨勢，再進一步計算其在整體資料中的佔比，評估該資料序列是否呈現長期持續成長。

關鍵字

最長遞增子序列、動態規劃

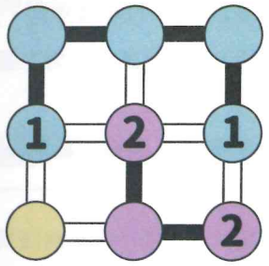
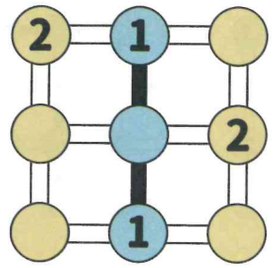
26. 伯魯克

伯魯克謎題由節點 ○ 和節點 ○ 間的連接線 ▭ 所組成。要解開謎題，就需要將一些連接線塗黑，使兩個數字相同的節點能透過塗黑線段 ▬ 連接起來。

線段塗黑的限制如下：

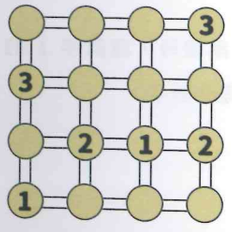
1. 有數字的節點只能有一條連接線塗黑。
2. 沒有數字的節點最多只能有兩條連接線塗黑。
3. 不同數字的節點不能有塗黑線段讓它們可連起來。

不過有些伯魯克謎題是無解的，例如下表左邊的謎題有解，但右邊的謎題則無解：

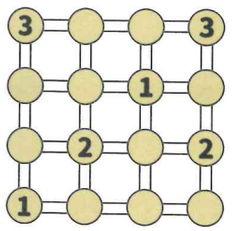
有解	無解
	

請問以下哪一個伯魯克謎題<u>無解</u>？

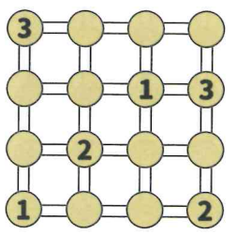
A.



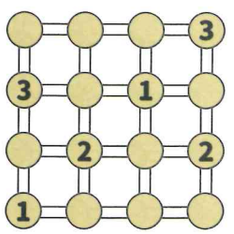
B.



C.

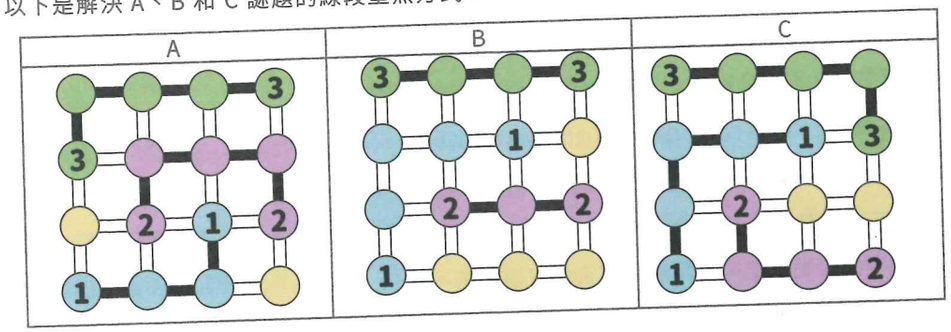


D.

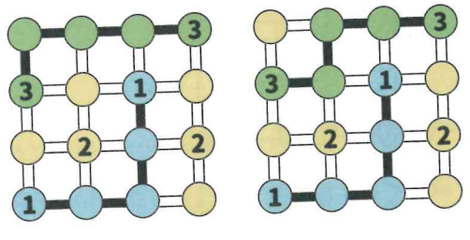


正確答案是：D

以下是解決 A、B 和 C 謎題的線段塗黑方式：



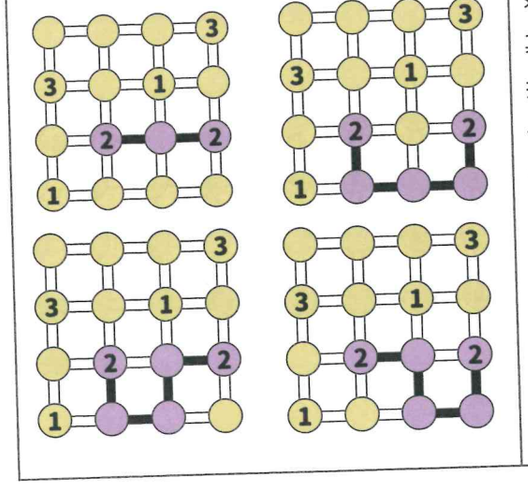
下圖顯示嘗試解決 D 謎題的線段塗黑方式：



將可連接數字 1 的兩個節點，且不連到其他數字的線段，可塗黑的畫法如左圖所示，是唯一的可能。

接下來對數字 3 的兩個節點，進行塗黑連接線段的選擇有 2 種。

但這兩種狀況，接下來皆無法找到可塗黑線段讓數字 2 的兩個節點連接，但不連到其他數字。



如果先將可連接數字 2 的兩個節點，且不連到其他數字的線段圖像左圖塗黑，有左圖 4 種選擇。

但接下來皆無法找到可塗黑線段，讓數字 1 的兩個節點連接，且不連到其他數字。

皆無法找出符合條件的線段塗黑方式。因此 D 謎題無解。

資訊科學上的意義

在資訊科學中，**回溯演算法 backtracking algorithm** 是一種用來找出所有可能解的策略。它的核心精神是：一步一步嘗試建立解答，如果在某一步發現當前的選擇會導致後續無法完成任務，就會「退回」前一步，重新嘗試其他可能的選擇。這樣的過程會不斷重複，直到找到一個同時符合所有條件的正確解。

在本任務中，伯魯克謎題的解法就是運用回溯法的一個典型例子。解題的過程中，玩家需要不斷嘗試各種線段塗黑的方式，並在每一步即時檢查目前的配置是否符合規則。如果某個數字的塗黑選擇導致其他數字後續無法找到正確的塗法，那就代表這條路走不通了，必須退回上一步，換一種可能的塗法重新嘗試。

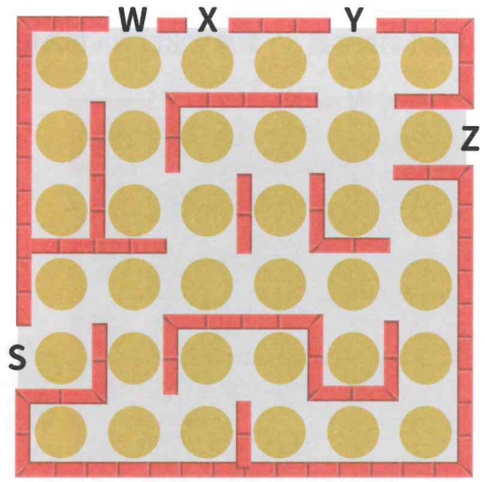
在日常生活中，其實我們也常不自覺地使用回溯法來解決問題。像是在走迷宮時，如果某條路走到底卻發現是死巷，就會回頭改走另一條路。這種「先嘗試 → 發現不通 → 回頭 → 改走別的方向」的方式，就是回溯的思考模式。

關鍵字

回溯演算法

27. 迷宮

下圖是一個迷宮，從入口 S 進入迷宮後，只能走在黃點 上移動，無法穿過牆壁 ；只要走到 W，X，Y，Z 任何一個點，即可離開迷宮。



要離開這個迷宮，最少會經過幾個點 ？

A. 5

B. 6

C. 7

D. 8

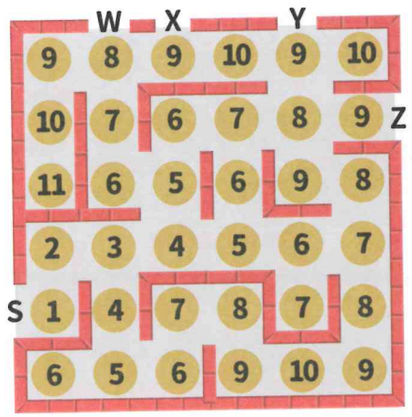
E. 9

正確答案是：D

要解這個任務，我們可以先將跟 S 點緊鄰的點標記為 1，表示經過 1 個點會到。接著跟標記 1 緊鄰可走到且尚未標記的下個點標記為 2，表示經過 2 個點會到。也就是根據點的標記由小到大，對於已標記為「n」的點，將跟「n」一步可到達但尚未標記的點標為「n + 1」。繼續這樣進行編號，直到所有點都被標記為止。

全部標記完成之後，每個出口緊鄰點的標記數字，就知道從那個出口離開迷宮最少需要經過幾個點。

這些出口緊鄰點的標記數字中最小的數字是 8，對應到出口 W，即為解答。



資訊科學上的意義

廣度優先搜尋 breadth-first search, BFS 是圖論中一種常見且實用的搜尋方法。它的核心概念是：從起點出發，依照距離由近到遠，以逐層擴展的方式探索所有節點，就像水波一圈一圈向外擴散一樣。這種方法特別適合用來找出「步數最少」或「最短時間到達目標」的解法。

在本任務中，我們從起點 S 出發，利用廣度優先搜尋的策略，一層一層地找出所有可以到達的黃點。當我們走到出口 W、X、Y 或 Z 時，就代表成功逃出迷宮。由於 BFS 會優先探索距離最近的選擇，因此特別適合用來找出逃離迷宮的最短路徑。

在生活中，也常有類似的應用情境。例如社群平台推薦朋友時，系統會先找出你最直接的朋友（第一層），再擴展到朋友的朋友（第二層），這種「由近到遠」的搜尋方式，就是廣度優先搜尋的應用。此外，BFS 也被用來解決許多實際問題：像是在查找轉機次數最少的航班組合、網頁爬蟲依照層級擴展來蒐集資料、又或是在社群網路中分析訊息如何一層層地傳播出去。

關鍵字

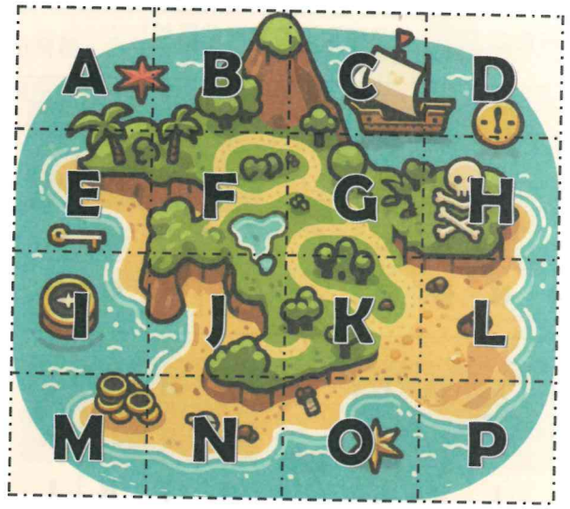
廣度優先搜尋

28. 尋找寶藏

船長比比在下圖中的島嶼上尋找一個寶藏，他把島嶼平均劃分為 16 塊區域（A 到 P）。

比比有個特殊探測機器，每次使用時，只要輸入區域代碼（不限個數）詢問機器，機器會告訴他寶藏是否在這些區域中。

例如：輸入「A、C、D」，若機器回答「是」，就表示寶藏在區域 A、C 或 D 之中。回答「否」，就表示寶藏不在這 3 個區域。



比比最少詢問機器幾次，就一定可以找到寶藏所在的區域呢？（範圍 [0~99] 的整數）

正確答案是：4

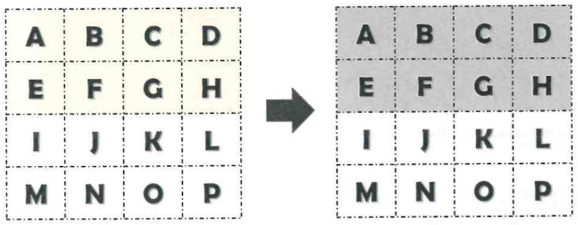
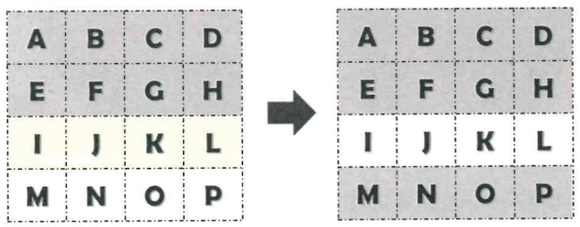
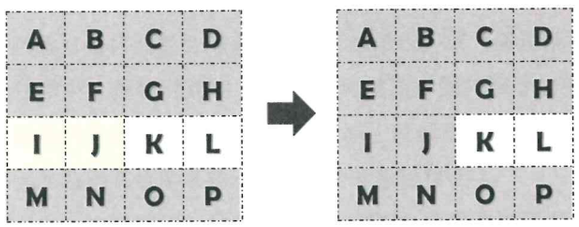
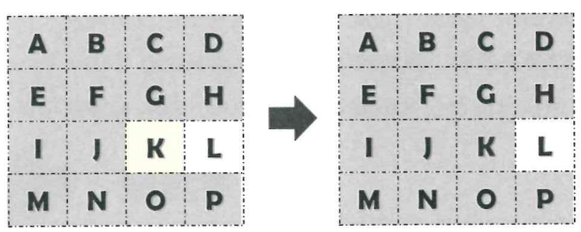
只要比比將「不確定（是否有寶藏的）區域」取一半輸入機器中，機器的回答一定是以下兩種情況：

1. 「是」：表示寶藏在比比輸入的其中一個區域；另外一半的區域確定沒有寶藏。
2. 「否」：表示比比輸入的這些區域確定沒有寶藏；寶藏藏在另一半的某一個區域。

無論是哪一種情況，在獲得機器回答後，都可以確定其中一半的區域沒有寶藏。

以這種方式，「不確定區域」的數目每次都可減少一半；島嶼有 16 個區域，因此只要使用機器 4 次，一定可以確定寶藏所在的區域。

舉例來說，假設寶藏在 L 區域，可採用下列尋找步驟（下圖以黃色底 ▢ 表示該次輸入機器的區域，以灰色底 ▢ 表示確定沒有寶藏的區域）：

	「輸入：「ABCDEFGH」，機器回答：「否」。比比得知寶藏在區域 I 至 P 中。
	「輸入：「IJKL」，機器回答：「是」。比比得知寶藏在這四個其中一個區域。
	輸入：「IJ」，機器回答：「否」。比比得知寶藏在區域 K 或 L 中。
	輸入：「K」，機器回答：「否」。比比得知寶藏在區域 L 中。

資訊科學上的意義

本任務運用了資訊科學中的 **二元搜尋 binary search** 概念。二元搜尋是一種可在有序資料中快速搜尋出目標資料的演算法。其核心策略是：每次都選擇目前資料範圍的中間項目進行比較，若尚未找到，可根據比較結果判斷目標在大的那邊或小的那邊，排除其中一半，讓搜尋範圍快速縮小，從而確保在固定次數內找到目標資料。

在本任務中，寶藏藏在 16 個區域中的某一處。雖然這些區域本身沒有並沒有排序，但每次可以用探測裝置選擇一半的區域進行詢問。如果機器回答「是」，就表示目標在選定的一半，保留這一半繼續查找；如果回答「否」，就改查另一半。這樣每次都能將範圍減半，搜尋範圍會依序從 16 個 → 8 個 → 4 個 → 2 個 → 1 個，最多只需 4 次詢問，就能找到寶藏的正確位置。

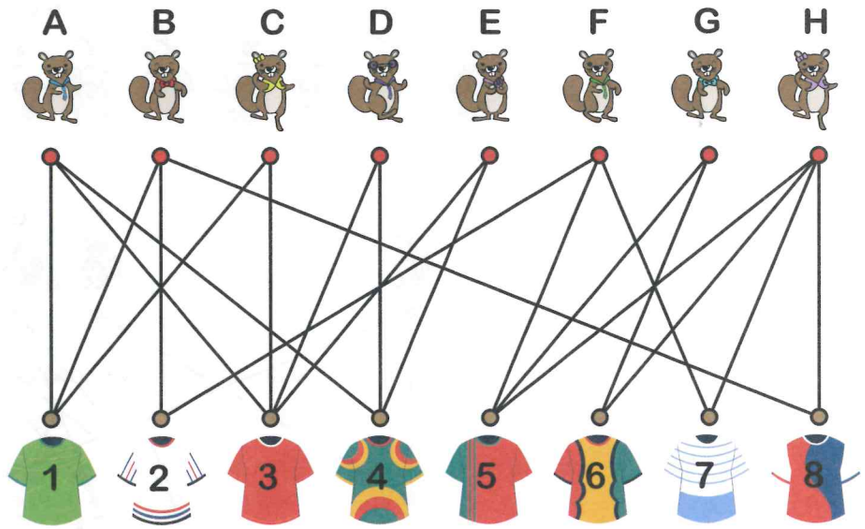
我們在生活中也常採用類似的方式來加快搜尋，例如查字典時，會先翻到中間看看要找的字是比較前面還是後面，再逐步縮小範圍。另外跟朋友玩「猜猜你心中的數字」遊戲，如果數字範圍是 1 到 100，你會根據朋友的提示「太大」或「太小」，每次將猜的數字範圍排除一半，快速縮小目標範圍。這些都用到了二元搜尋的思考方式。

關鍵字

二元搜尋

29. 海狸的球衣

海狸球隊有八位隊員，經理今天送來 8 件球衣，供海狸挑選。下圖的連線表示每隻海狸想要的球衣，例如海狸 A 想要編號  、  或  的球衣，以此類推。



每隻海狸只能分配一件球衣，最多能讓幾隻海狸拿到自己想要的球衣呢？

A. 8

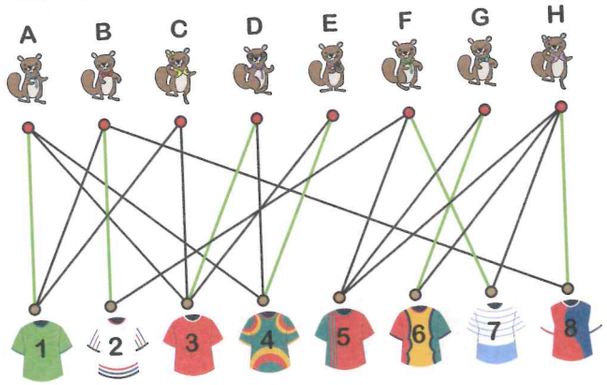
B. 7

C. 6

D. 5

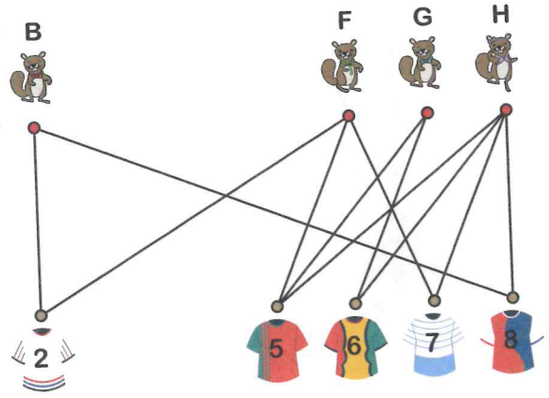
正確答案是：B

綠色的線條顯示一種分配方法，讓 7 隻海狸可挑到他們想要的球衣。



但是讓 7 隻海狸挑到球衣是不是最多，需要進一步分析。

觀察編號 2、5、6、7 和 8 的球衣，只有海狸 B、F、G 及 H 想選。



換句話說，這 5 件球衣只能分配給這 4 隻海狸，即使這 4 隻海狸都能分配到一件球衣之後，一定會有一件球衣剩下來，但其他海狸都不想要。

因此不可能讓所有的海狸都分配到想要的球衣，而上面已經找到了一種方法可以讓 7 隻海狸拿到喜歡的球衣，因此最多分配到球衣的海狸數量就是 7。

在分配的時候，可以將題目提供的關係圖，以下表的形式呈現：

	A	B	C	D	E	F	G	H	被挑選數
1	■	■	■						3
2		■				■			2
3	■		■	■	■				4
4	■			■	■				3
5						■	■	■	3
6							■	■	2
7						■		■	2
8		■						■	2
被選數	3	3	2	2	2	3	2	4	

如果從球衣開始，決定要分配給誰。"當某件球衣想要的海狸數越多，可能的分配可能就會越多種"。因此，我們可以試著從"想要的海狸數比較少的球衣開始"分配，比方 6 號球衣只有兩隻海狸挑選它，所以可以先從 6 號球衣開始分配。

由於挑選 6 號球衣有兩隻海狸 G 跟 H，依照上述簡化可能性的方式，可以接著從這些海狸中，選擇球衣數偏好數較少的海狸，先分配牠的球衣。如果把 6 號球衣分配給海狸 G，因此表格內容也進行更新，去掉海狸 G 對其他球衣的偏好。

	A	B	C	D	E	F	G	H	被挑選數
1	■	■	■						3
2		■				■			2
3	■		■	■	■				4
4	■			■	■				3
5						■		■	2
6							■		1
7						■		■	2
8		■						■	2
被選數	3	3	2	2	2	3		3	

然後再依照上述處理原則，發現接下來有 4 件球衣可考慮優先分配

1. 2 號球衣，還有海狸 B, F 想選
2. 5 號球衣，還有海狸 F, H 想選
3. 7 號球衣，還有海狸 F, H 想選
4. 8 號球衣，還有海狸 B, H 想選

這 4 件球衣，只有三隻海狸想選，表示最後一定會有一件球衣無法分給想要的海狸。

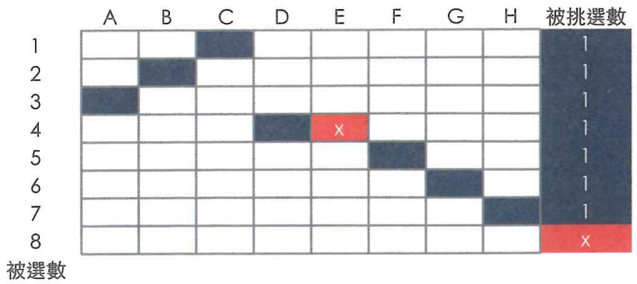
先對這三隻海狸的偏好找出一種球衣分配方式：例如給海狸 B 分配 2 號球衣，表示海狸 F 就只能分配 5 號或 7 號。若把 5 號球衣分配給海狸 F，海狸 H 可以從 7 號或 8 號中選一件。所以 7 號或 8 號球衣會有一件會被留下，分配後的結果如右表。

	A	B	C	D	E	F	G	H	被挑選數
1	■		■						2
2		■							1
3	■		■	■	■				4
4	■			■	■				3
5						■			1
6							■		1
7								■	1
8									X
被選數	3		2	2	2				

接著我們依照同樣的方式，先分配 1 號球衣，因為海狸 C 的偏好球衣數較少，所以分配給牠。

	A	B	C	D	E	F	G	H	被挑選數
1			■						1
2		■							1
3	■			■	■				3
4	■			■	■				3
5						■			1
6							■		1
7								■	1
8									X
被選數	2			2	2				

接下來，我們會發現出現類似的狀況：剩下的兩件球衣有三隻海狸想要，因此一定會有一隻海狸分配不到。球衣只能分配給其中兩隻海狸，例如把 3 號球衣給海狸 A，剩下的 4 號球衣只能分配給海狸 D 或海狸 E 其中一隻。



請注意這個分配結果只是其中一種可能，但無論如何都不可能讓所有的海狸分配到想要的球衣。因此這八件球衣，最多只有七件球衣可被分配給想要的海狸，表示最多也只有七隻海狸可拿到想要的球衣。

資訊科學上的意義

本任務運用了 **二分圖 bipartite graph** 和 **最大雙族匹配 maximum bipartite matching** 這兩個資訊科學中的重要概念。所謂二分圖，是一種可以把節點分成兩組的圖，圖中的連線（邊）只會連接兩組之間的節點，而不會連到同一組裡的節點。

在這個任務裡，我們可以把海狸們看成左邊這一組，把球衣看成右邊那一組。只要某隻海狸喜歡某件球衣，就在他們之間畫一條線，表示這是可能的配對。任務的目標是：讓最多的海狸拿到自己喜歡的球衣，而且每隻海狸只能拿一件、每件球衣也只能分給一隻海狸。這就相當於在二分圖中找出最多條「不互相衝突」的配對線，這個問題就叫做最大雙族匹配問題。

在日常生活中，常有需要在「兩組對象之間」進行配對的情況，而我們也希望能讓配對的數量盡可能多，這就對應到圖論中的「最大雙族匹配」問題。例如在志願服務活動中，主辦單位有多個攤位需要人力支援，而每位志工只能參與自己感興趣或擅長的項目。此時，需要根據志工的志願與攤位需求，安排出最多志工參與的組合，讓更多工作能有人負責。又像租屋媒合平台：房東有房屋出租，租客有喜歡的房型與地點。平台會根據雙方需求做配對，希望讓最多房屋成功出租。

關鍵字

二分圖、最大雙族匹配

本任務解析由 2024 運算思維推動計畫製作，係採用創用 CC 姓名標示 - 非商業性 - 禁止改作 3.0 台灣 授權條款授權。

30. 題組一 - 遊樂園 A

阿福去海狸文化村參觀。下圖是文化村遊樂器材的地圖和每項遊樂設施玩一次的時間。文化村規定：

1. 每項設施每個人最多只能玩一次，別人才有機會玩。
2. 更換設施必須順著黑色箭頭路徑走，不能穿越廣場公園。
3. 到下班時間，所有遊樂設施都要結束，不能有人玩到一半。

阿福發現 20 分鐘後文化村的工作人員就要下班。

請問阿福在 20 分鐘內最多可以玩幾項器材設施？（範圍 [1~5] 的整數）

31. 題組二 - 遊樂園 B

阿福去海狸文化村參觀。下圖是文化村遊樂器材的地圖和每項遊樂設施玩一次的時間。文化村規定：

1. 每項設施每個人最多只能玩一次，別人才有機會玩。
2. 更換設施必須順著黑色箭頭路徑走，不能穿越廣場公園。
3. 到下班時間，所有遊樂設施都要結束，不能有人玩到一半。

阿福發現 20 分鐘後文化村的工作人員就要下班。

阿福希望這 20 分鐘盡量花在玩遊樂設施上。請問扣掉移動的時間，他玩到遊樂設施的時間總計最多是幾分鐘？（範圍 [1~20] 的整數）

題組一正確答案是：4

因為玩全部的 5 項遊樂器材，不計移動時間就需要 21 分鐘，所以不可能在 20 分鐘內玩 5 項遊樂器材。

可將花最多時間的摩天輪刪掉，剩下的 4 個遊樂器材，玩一次的時間及移動到另一器材的時間如下：

遊樂設施							遊玩 + 移動時間
5 分鐘	2 分鐘	3 分鐘	2 分鐘	3 分鐘	2 分鐘	2 分鐘	19 分鐘
溜滑梯 	移動	射箭 	移動	盪鞦韆 	移動	飛鏢 	

如果將這些器材遊玩的順序反過來也是只需要 19 分鐘，因此在 20 分鐘內最多可以玩完 4 個項目。

題組二正確答案是：16

正確答案是 16 分鐘。遊玩組合是摩天輪 + 溜滑梯 + 射箭。

因為玩全部的 5 項遊樂器材，不計移動時間就需要 21 分鐘，所以不可能在 20 分鐘內玩 5 項遊樂器材。

要在這 20 分鐘內盡量花在玩遊樂設施上，可能做以下安排：

1. 刪除「遊玩時間最少」的飛鏢。
2. 必選「遊玩時間最多」的摩天輪。

滿足以上 2 個條件的路線就是「摩天輪 + 溜滑梯 + 射箭」。這樣花費的時間是：

 8 分鐘 + 移動 2 分鐘 +  5 分鐘 + 移動 2 分鐘 +  3 分鐘 = 20 分鐘

待在遊樂器材上的時間是 16 分鐘。

為了證明 16 分鐘是最佳答案，以下是其他可在 20 分鐘內玩完的組合：

包含摩天輪的組合			遊玩時間	遊玩 + 移動時間
摩天輪 8 分鐘 	飛鏢 2 分鐘 	盪鞦韆 3 分鐘 	13 分鐘	17 分鐘
溜滑梯 5 分鐘 	摩天輪 8 分鐘 	飛鏢 2 分鐘 	15 分鐘	19 分鐘

不包含摩天輪的組合				遊玩時間	遊玩 + 移動時間
飛鏢 2 分鐘 	盪鞦韆 3 分鐘 	射箭 3 分鐘 	溜滑梯 5 分鐘 	13 分鐘	19 分鐘
不包含摩天輪的其他器材，任選 3 種也可以玩完，但待在遊樂器材上的時間一定比 13 分鐘還少。					

所以摩天輪 + 溜滑梯 + 射箭是在 20 分鐘內，能花在玩遊樂設施上最久的玩法。

資訊科學上的意義

本任務結合了圖論 graph theory 與 **貪心演算法 greedy algorithm** 的概念。整張地圖可以視為一個圖（graph）：每個遊樂設施是一個節點（node），而設施之間的道路則是邊（edge）。圖的結構在資訊科學中應用非常廣泛，例如路徑規劃、網路架構設計、甚至遊戲地圖建構等情境。

貪心演算法 greedy algorithm 的核心精神是：在每一個決策點，只考慮當下最有利的選擇，不回頭也不做全局比較，而是希望透過一連串局部最佳的選擇，累積出一個整體還不錯甚至接近最好的結果。舉例來說，若阿福希望在 20 分鐘內玩到最多項設施，那麼每次選擇時應該優先避開那些花費時間最多的項目（如摩天輪），改選擇時間較短的設施，讓他能多玩幾個設施。反過來，如果阿福希望盡可能用滿這 20 分鐘、增加總遊玩時間，那麼他可以先選擇耗時長的設施，再依剩餘時間刪掉短時間項目（如飛鏢），讓時間分配更集中。這樣的策略雖然無法保證絕對是最佳解（不像窮舉法會試過所有可能），但它的優點是：計算速度快，可以在時間與資源有限的情況下，迅速做出「夠好」的決策，因此在實務中經常運用。

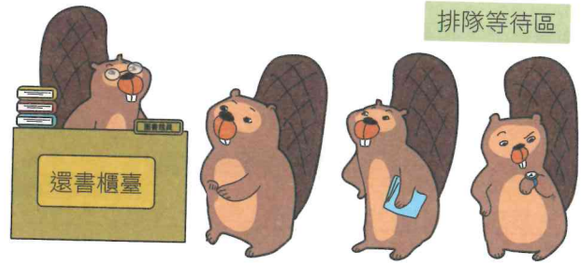
舉例來說：導航軟體會根據即時路況推薦目前最快路線；行程安排時，會選出最多能參加的活動組合；廣告系統則會優先展示目前效益預測最高的廣告。這些情境背後，都可採用了貪心式的策略選擇，來幫助系統快速作出反應並提升整體效能。

關鍵字

貪心演算法

32. 還書

圖書館通常會有海狸排隊等待還書。因此圖書館員引入一條處理還書的規則：「書本最少的海狸先還書。」



無論海狸在什麼時間到圖書館，圖書館員會從排隊隊伍中找出書本最少的海狸，優先處理他要歸還的書。

圖書館員每一分鐘處理一本書，當她處理完一個海狸歸還的所有書籍後，就會從隊伍中找出歸還書本最少的海狸。

某天早上有 5 隻海狸來還書。牠們到達的時間和歸還的書本數如下表：

海狸	到達時間	書本數
小貫	9:00	4
小以	9:02	6
小柄	9:03	2
小釘	9:05	4
小勿	9:11	1

小貫在圖書館開門時到達，因此圖書館員能立即處理小貫要還的書。

圖書館員處理這些海狸的還書的先後順序為何？

A. 小貫，小以，小柄，小釘，小勿

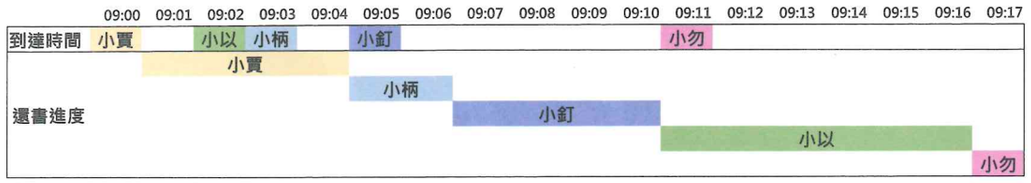
B. 小貫，小柄，小以，小釘，小勿

C. 小貫，小柄，小釘，小以，小勿

D. 小勿，小柄，小貫，小釘，小以

正確答案是：C

讓我們觀察以下時間對應表：



小賈是第一個到達圖書館的。因為他是當時唯一的海狸，所以可以立即處理他的還書。他歸還了 4 本書，因此圖書館員會在 9:04 時完成。

在 9:04 時，等待隊伍中有 2 隻海狸：小以在 9:02 到達，帶著 6 本書，小柄在 9:03 到達，帶著 2 本書。由於小柄是其中還書量最少的，因此優先處理他的還書，在 9:06 完成。

在 9:06 時，等待隊伍中有 2 隻海狸：包括原先的小以，以及帶著 4 本書在 9:05 到達的小釘。由於小釘的書較少，因此優先處理他的還書，並且在 9:10 時完成。

在 9:10 時只有小以在等待中，所以處理他的 6 本還書，處理完時為 9:16。

這時候，等待隊伍中只剩下在 9:11 到達的小勿，他要還 1 本書。所以圖書館員在 9:17 時完成這些海狸的還書。

還書的處理順序是 小賈，小柄，小釘，小以，小勿。

答案 A 是依照先排隊先處理的方式；顯然因為處理的優先規則不同，會影響處理順序。

資訊科學上的意義

在電腦系統中，中央處理器 (CPU) 一次只能處理一項工作，稱為程序 (process)。當有多個程序同時等待執行時，系統會依據**排程演算法 scheduling algorithm** 來決定各程序的執行順序。其中一種常見的策略是 **最短工作優先 shortest job first, SJF**，也就是優先執行所需時間最短的程序。這樣可以有效減少整體平均等待時間，進而提升系統整體效率。

在本任務中，我們可以將圖書館員比喻為 CPU，而每隻等待還書的海狸就像是一個等待執行的程序。圖書館員一次只能幫助一隻海狸還書，且每次都會優先處理手上書本數最少的海狸，這正好對應到 SJF 排程的邏輯：先完成「花最少時間」的任務，以加快整體流程。

在日常生活中也有許多應用需要有排程策略決定哪個工作優先處理，例如：

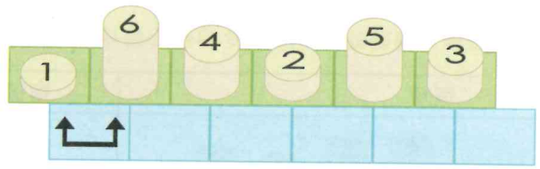
- 印表機管理系統：可能會優先列印頁數較少的文件，讓更多使用者可以快速完成列印，縮短總等待時間。
- 醫院急診室的診療順序：醫師會根據病患狀況的緊急程度決定先後順序，而不僅按照到達時間，目的在提高急診醫療資源的有效使用。

關鍵字

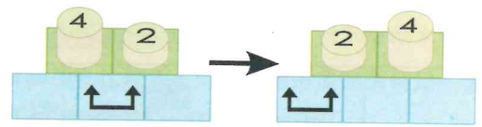
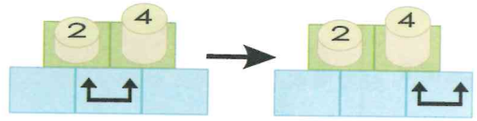
排程演算法

33. 排列遊戲

如下圖，遊戲桌上有 2 排格子。上排綠色格子分別放了 6 個高度不同的積木，積木上的數字代表它的高度；下排藍色格子中有 1 個指標符號 ，表示上排那兩格的積木正在進行高度比較。

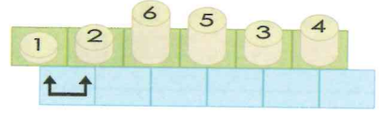


遊戲規則如下：

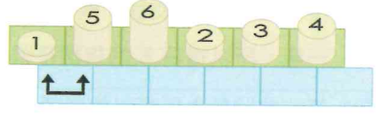
1. 積木高度比較以後，如果左邊的積木比右邊高，就交換兩個積木位置，並將指標符號向左移動一格，如右圖。	
2. 如果左邊的積木比右邊矮，兩個積木都不動，並將指標符號向右移動一格，如右圖。	
3. 當指標符號移到最右邊的格子時遊戲就結束。	

請問下列各個遊戲的初始積木排列，哪一個在遊戲結束時，指標符號累積的移動次數最少？

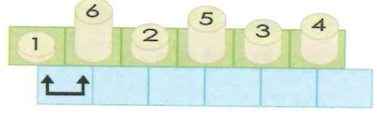
A.



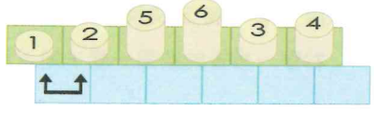
C.



B.

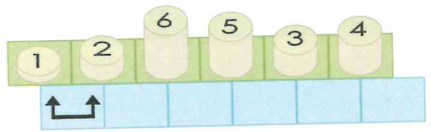
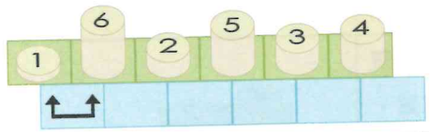
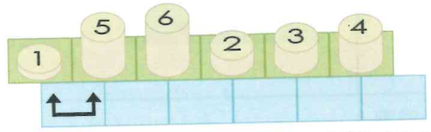
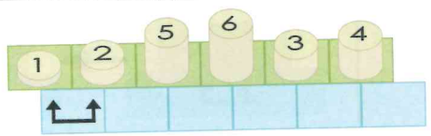


D.



正確答案是：D

依照遊戲規則，如果小的在前大的在後，可以讓指標符號順利右移，減少來回移動的次數。這 4 個選項只有前面 4 個積木的前後順序不同，選擇前 4 個已經由小到大排好的 D 選項，指標符號移動次數會最少。以下是每一個選項的指標符號移動次數：

選項		指標符號移動次數
選項 A		15 次
選項 B		17 次
選項 C		17 次
選項 D		13 次

除了逐一計算外，我們也可以用觀察比較的方式找答案：

1. 首先觀察狀態類似的選項 A 和 B：

   選項 B：指標符號先右移一格，1 6 2 5 3 4 → 1 6 2 5 3 4。

   接下來 6 和 2 交換，同時指標符號左移一格，變成 1 2 6 5 3 4。這跟選項 A 的初始狀態一樣。所以選項 B 的指標符號移動次數 > 選項 A。

2. 再比較狀態也很類似的選項 A 和 D：

   選項 A 前 2 步是：1 2 6 5 3 4 → 1 2 6 5 3 4 → 1 2 6 5 3 4。

   接下來 6 和 5 交換，同時指標符號左移一格，變成 1 2 5 6 3 4。這跟選項 D 在指標符號右移一格後的狀態一樣，所以選項 A 的指標符號移動次數 > 選項 D。

3. 最後檢查狀態比較不同的選項 C 和 D：

   選項 C 前 2 步是：1 5 6 2 3 4 → 1 5 6 2 3 4 → 1 5 6 2 3 4。

   接下來 6 和 2 交換，同時指標符號左移一格，變成 1 5 2 6 3 4，

   再接著 5 和 2 交換，同時指標符號左移一格，變成 1 2 5 6 3 4。這是選項 D 的初始狀態，所以選項 C 的指標符號移動次數 C> 選項 D。

從上面幾點比較得到的結論，可以得出選項 D 的指標符號移動次數最少。

資訊科學上的意義

本任務運用了 **排序演算法 sorting algorithm** 的核心概念，透過比較與交換相鄰積木的高度，讓數列逐漸接近按大小排列。整體的操作方式乍看之下與經典的氣泡排序（bubble sort）相似：從左到右，依序比較每一對相鄰的積木，如果左邊比較高，就將它們交換位置；反之則不動，接著繼續比較下一對。但仔細觀察會發現，本任務的操作規則與氣泡排序略有不同：當兩個積木交換後，指標會往左退一步重新比較上一組積木，而不是繼續往右比較下一對。這樣的處理邏輯是一種叫做侏儒排序（gnome sort）的演算法。

侏儒排序的處理是：如果順序正確，就往前走；如果順序錯誤，就交換並退回一步再比一次。這樣前進與退後交錯的流程，會持續進行，直到整個數列由小到大排好。這正好對應到本任務中交換與移動指標的方式。

在日常生活中，排序演算法的應用非常普遍，尤其在各種數位工具中更是不可或缺。以網路商店為例，當我們在購物網站上瀏覽商品時，可以依照「價格高低」、「熱銷程度」、「評價分數」等條件進行排序，讓使用者容易找到符合需求的商品。在手機或雲端的相簿管理，照片依拍攝時間、地點或檔案大小進行排序後，可方便查找或清理。而日常使用的電子郵件系統中，收件匣也常提供按「最新 / 最舊時間」、「寄件人姓名」、「主旨」等方式排序，幫助有效率地瀏覽與搜尋郵件。

關鍵字

排序演算法

34. 動物排序

有 7 隻不同的動物排成一列，每隻動物都能跳到和自己身高一樣的高度。如果你呼叫其中一隻動物的名字，牠會連著跳過排在前面比自己矮的動物，直到遇到比自己高的動物為止。

例如，如果你呼叫狐狸，牠會先跳過兔子，再跳過海狸。因為狐狸比狼矮，所以狐狸會停在海狸和狼之間。

動物的高度及初始排列順序如下圖所示：

以下哪個選項的呼叫順序，能將這 7 隻動物從左至右由矮排到高？

A.

B.

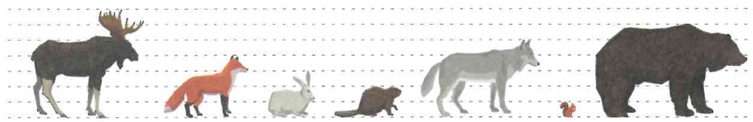
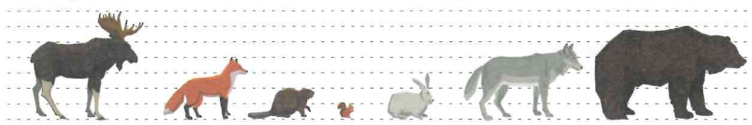
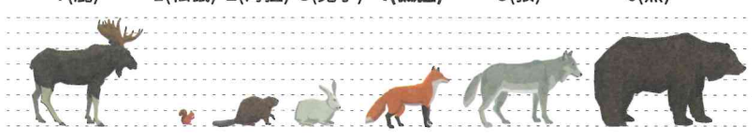
C.

D.

正確答案是：B

因為每隻動物的身高不同，為簡化表示，下面呼叫順序直接以動物高度表示。

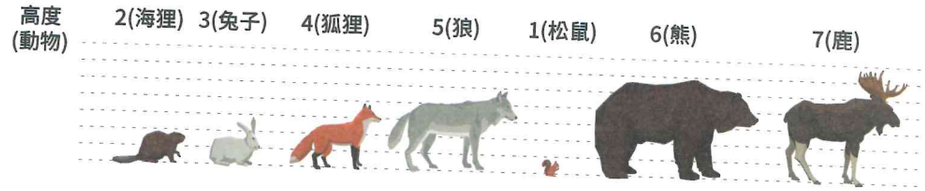
選項 (B) 呼叫順序為： 5 – 3 – 2 – 4 – 7，每次呼叫後的排列變化過程如下。

原排列	高度 (動物)　7(鹿)　4(狐狸)　3(兔子)　2(海狸)　5(狼)　1(松鼠)　6(熊) 
呼叫 5（狼）	高度 (動物)　7(鹿)　4(狐狸)　3(兔子)　2(海狸)　1(松鼠)　**5(狼)**　6(熊) 
呼叫 3（兔）	高度 (動物)　7(鹿)　4(狐狸)　2(海狸)　1(松鼠)　**3(兔子)**　5(狼)　6(熊) 
呼叫 2（海狸）	高度 (動物)　7(鹿)　4(狐狸)　1(松鼠)　**2(海狸)**　3(兔子)　5(狼)　6(熊) 
呼叫 4（狐狸）	高度 (動物)　7(鹿)　1(松鼠)　2(海狸)　3(兔子)　**4(狐狸)**　5(狼)　6(熊) 
呼叫 7（鹿）	高度 (動物)　1(松鼠)　2(海狸)　3(兔子)　4(狐狸)　5(狼)　6(熊)　**7(鹿)** 

完成由矮排到高，因此，選項 (B) 為正確答案 。

其他選項錯誤的原因：

選項 (A): 呼叫 7 – 4 – 3 – 1 後 ，排列變成：



動物 1（松鼠）會在 5（狼）和 6（熊）之間。

選項 (C): 呼叫 5 – 4 – 3 – 2 後，排列變成：

動物 7（鹿）會排在最左邊，順序不對。

選項 (D): 呼叫 2 – 5 – 7 – 4 – 3 後，排列變成：

動物 2（海狸）會排在動物 1（松鼠）左邊，順序不對。

資訊科學上的意義

本任務運用了 **排序演算法 sorting algorithm** 的概念，並且其邏輯運作方式與 **插入排序 insertion sort** 非常相似。插入排序的核心概念是：逐一處理資料項目，並將每個新處理項目與前面已排序的資料進行比較，將其插入到正確的位置。這樣一來，整個序列會隨著每次插入逐步趨於有序。

在本任務中，每呼叫一隻動物時，牠會跳過前面所有比自己矮的動物，直到遇到第一個比自己高的動物後停下來。這個過程就像插入排序中的「插入」步驟：新加入處理的元素會被插入到適當位置，讓序列前方維持排序狀態。

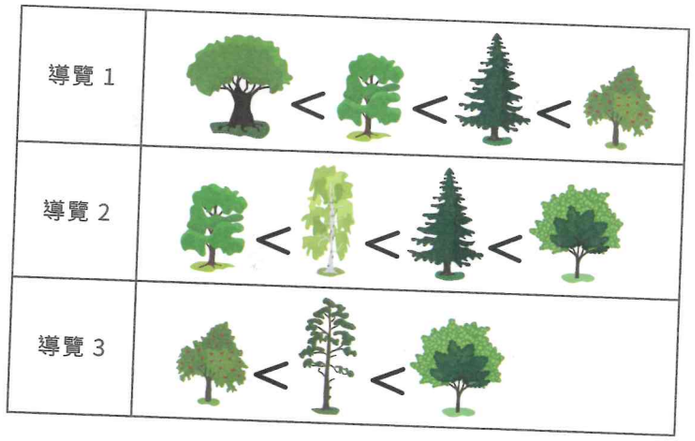
這樣的排序方式在日常生活中也很常見。例如：撲克牌遊戲時玩家會把新抽到的牌插入手牌中正確的位置，讓整手牌的花色從小到大排列。在通訊錄管理，系統會將新輸入的聯絡人依字母順序自動插入到依字母順序的位置。

關鍵字

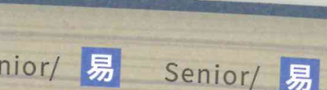
排序演算法、插入排序

35. 樹木導覽

小葵是海狸森林的樹木導覽員。每次導覽她都會介紹幾棵特別的樹。在前三次的導覽，她記得有些樹比其他樹更受遊客喜愛。下圖中，以樹 1 < 樹 2 表示在那一次的導覽，樹 2 比樹 1 更受喜愛。



小葵希望在下次導覽時，安排把較受喜愛的樹較晚再介紹。舉例來說，在導覽 1 中  比  更受喜愛，所以在下次導覽時  要比  晚介紹。

根據前三次導覽觀察到中樹木受喜愛的比較結果，小葵想安排這 5 顆樹：  和  在下次導覽的介紹順序。

請問以下哪個介紹順序符合要求？

A. 

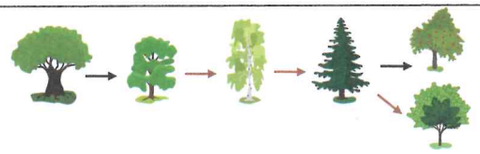
B. 

C. 

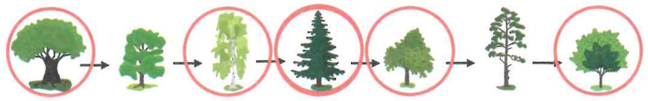
D. 

正確答案是：A

依序從之前導覽受喜愛的資訊，決定這些樹必須符合的先後介紹順序。

先考慮導覽 1 的喜愛順序，排出這 4 顆樹必須符合的介紹順序。	**只考慮導覽1** 
再考慮也要滿足導覽 2 的喜愛順序，可得出這 6 顆樹必須符合的介紹順序。	**同時考慮導覽2** 
最後要同時滿足導覽 3 的喜愛順序，可以得出這 7 顆樹必須符合的介紹順序。	**再考慮導覽3** 

因此對於下次導覽的 5 顆樹 ，選項 A  是唯一符合要求的介紹順序。



資訊科學上的意義

拓撲排序 topological sort 是一種用來安排有先後順序限制的項目排列方法。當我們知道某些事情必須在其他事情之前完成，但又無法直接看出所有項目的完整順序時，就可以透過拓撲排序來找出一個符合所有條件的合理順序。

在本任務中，導覽記錄會顯示哪棵樹比哪棵更受歡迎，例如：如果 A 樹比 B 樹受歡迎，而 B 又比 C 樹受歡迎，那就能推論 A 也比 C 更受歡迎。因此在安排介紹順序時，就應讓 A 樹出現在 C 樹的後面。我們只要整理這些「誰比誰更受歡迎」的條件，就能排出一組符合規則的介紹順序，這就是拓撲排序在實際問題中的應用方式。

在生活中，我們也常常遇到類似的情境。像是準備晚餐時，「要先洗菜才能炒菜」、「要先煮飯才能盛飯」，但煮湯或擺餐具的順序則比較自由。只要遵守那些有明確前後順序的任務，整體的執行順序就有多種可能，而這正是拓撲排序要解決的問題。這個概念也常用在電腦中，例如流程控制、任務排程，甚至遊戲中「完成前一關才能解鎖下一關」的設計，都與拓撲排序有關。

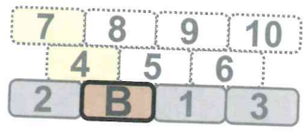
關鍵字

拓撲排序

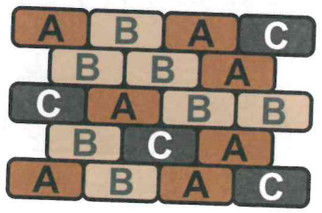
36. 磚牆

小狸的院子裡有一面牆（甲牆），上面的磚塊有三種 (A B C)，他想把這面牆移到別處，並堆砌成形狀一樣的牆（乙牆）。

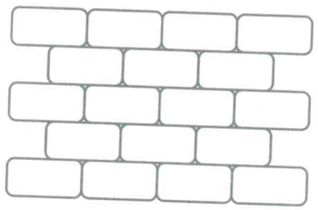
小狸拆除甲牆跟堆砌乙牆的規則如下：

1. 每次拆除甲牆的一塊磚，就馬上堆砌到乙牆。	
2. 拆除甲牆時，一次選一塊上方已沒有其他磚的磚塊拆除；例如：右圖中只能選 A 磚 或 B 磚拆除。	 可拆除範例
3. 磚塊堆砌到乙牆時，可堆砌的位置是下方為地基，或是該位置有接觸的下方位置皆已堆砌磚塊。舉右圖為例，當已堆砌 B 磚，接下來可堆砌的位置為編號 1、2、3 這 3 個地基位置。位置 4 必須等位置 2 也已堆砌磚塊後才可被選擇堆砌；而位置 7 則需要等位置 4 已被堆砌後就可被選擇。	 可堆砌範例

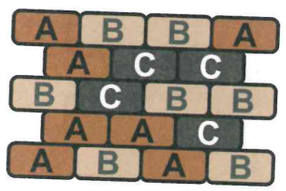
下圖為小狸要拆除的甲牆牆面上三種磚塊的圖案，以及乙牆要堆砌磚塊的設想位置。

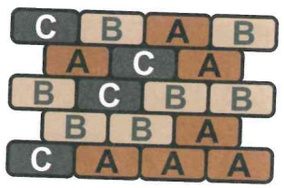


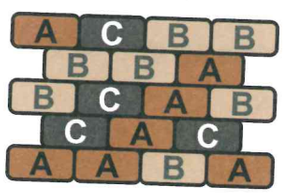
甲牆

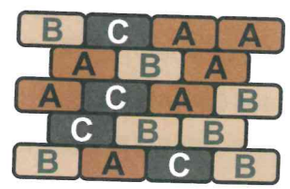

乙牆

請問下列哪些牆面花樣有可能是小狸所砌出的乙牆呢？

A.


C.


B.


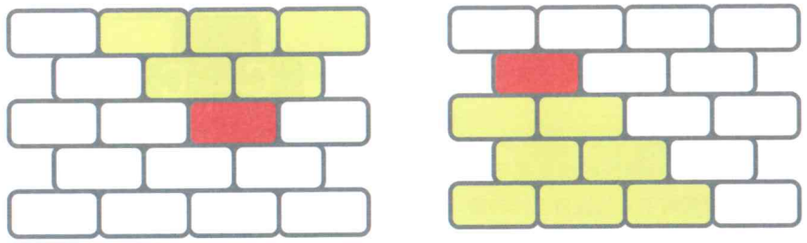
D.


正確答案是：AC

只有 A 和 C 選項的牆面可能建造成功。

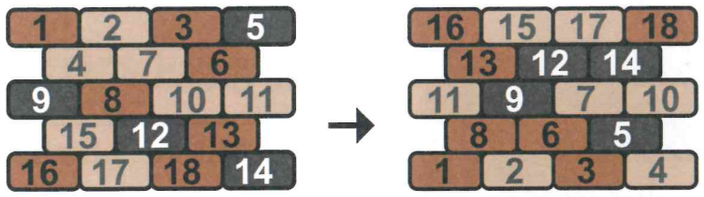
拆除原牆磚塊時，要符合「選取上方沒有被其他磚塊壓住的磚塊拆除」。以左下圖為例，要拆除紅色的磚塊，必須先將其上方所有的黃色的磚塊都先拆除。

此外，堆砌新牆時，要符合「只能堆在新牆地基上，或是該位置接觸的下方位置皆已堆砌磚塊」。以右下圖為例，要堆砌紅色磚塊在該位置前，必須等將下方黃色磚塊位置都擺完後，才能堆砌紅色磚塊。



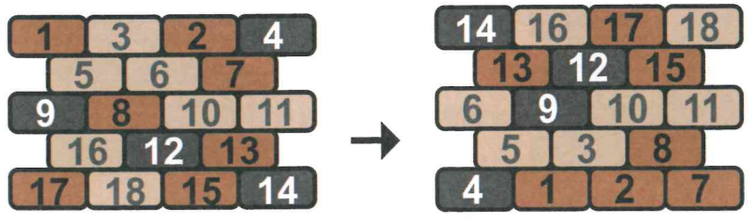
要拆除的牆面　　重新建造的牆面

對於選項 A，我們將原牆拆除順序編號顯示如下左側圖，並依下右側圖順序編號建造新牆，可得到選項 A 的圖面。因此選項 A 的牆面可能建造出來。



原牆（要拆除的牆面）　　新牆（重新建造的牆面）

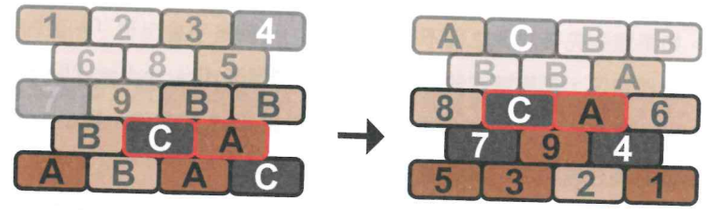
選項 C，我們將原牆拆除順序編號顯示如下左側圖，並依下右側圖順序編號建造新牆，可得到選項 C 的圖面。因此選項 C 的牆面亦可建造出來。



要能符合拆除及堆砌規則，經觀察上述選項 A 及 C 磚塊拆除及堆砌的順序編號，可發現以下特性：

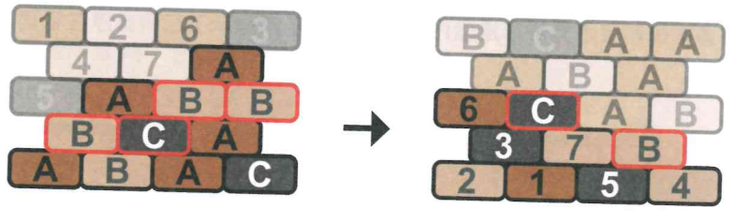
1. 原牆中每個磚塊拆除編號都比其相連的下方磚塊編號小。
2. 新牆中每個磚塊堆砌編號都比其相連的下方磚塊編號大。

要建造選項 B 的新牆，可將原牆依下左側圖示編號，依序拆除 9 個磚塊，並依下右側圖順序編建造；接下來新牆的第 3 層需要堆砌 A 或 C 磚塊後才能繼續往上堆，但此時原牆中的 A 及 C 磚塊上方都仍有磚塊壓住無法拆下，所以選項 B 的牆不可能被造出來。



原牆（要拆除的牆面）　　新牆（重新建造的牆面）

要建造選項 D 的新牆，可將原牆依下左側圖示編號，依序拆除 7 個磚塊，並依下右側圖順序一一堆砌。接下來新牆需要堆砌 B 或 C 磚塊才能繼續堆，但此時原牆中的 B 及 C 磚塊上方都仍有磚塊壓住無法拆下，所以選項 D 的牆不可能被建造出來。



原牆（要拆除的牆面）　　新牆（重新建造的牆面）

資訊科學上的意義

有向圖 directed graph 是一種圖的結構，每條邊都有方向，從某個節點指向另一個節點。這類圖常用來表示任務之間的先後關係：每個節點代表一項任務，而箭頭表示「某件事必須在另一件事完成之後才能進行」。如果這些先後關係之間**不會形成循環**，也就是不會出現「A 依賴 B、B 依賴 C、C 又反過來依賴 A」這種情況，那麼這樣的圖就叫做 **有向無環圖 directed acyclic graph, DAG**。當任務之間形成 DAG 時，我們就可以用 **拓撲排序 topological sort** 的方法，找出一個符合所有條件的執行順序，也就是讓每項任務都在它的前置任務之後開始。

在本任務中，小狸要拆除甲牆與建造乙牆。拆牆時要遵守「下方有支撐的磚才能先拆」，建牆時則是「每塊磚必須堆在有支撐的位置上」。這兩種情境其實都符合拓撲排序的概念：一個是決定拆除的順序，一個是決定堆砌的順序，必須同時滿足這兩套條件，才是一組正確的答案。

拓撲排序的概念在生活中也經常出現。舉例來說，蓋房子時一定要先完成地基，才能搭建牆面，再蓋屋頂，這些工序彼此之間就有清楚的先後依賴關係；又例如在準備一道餐點時，有些食材要先切、先煮，才能進行下一步的調味或裝盤，整個料理過程也可以視為一個有向無環圖。而在學校或工作中，安排專案進度或選修課程也會遇到類似情形：某些作業要等資料整理完才能開始寫報告，有些課程也需要先修過基礎科目才允許修進階內容。這些例子都顯示了拓撲排序在管理「有順序限制的任務」是很實用的。

關鍵字

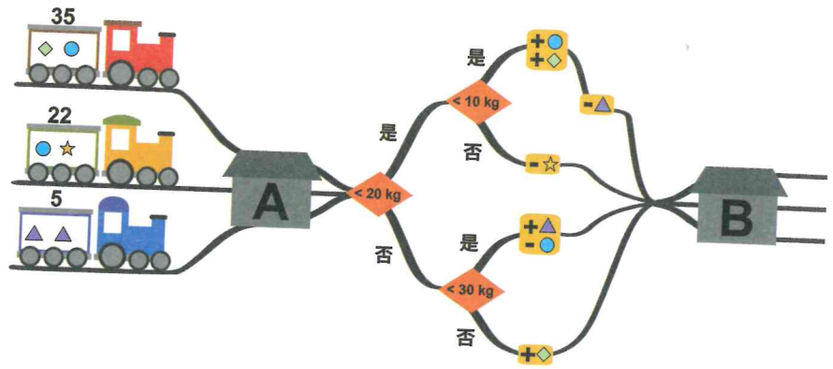
有向圖、拓撲排序

37. 鐵路

下圖是城市 A 跟城市 B 之間的火車路線圖。

海狸會根據區分載重的橘色標誌 ◆ 來選擇路線，並根據鐵路上的黃色標誌 ▬ 來裝載 (+) 或卸下 (-) 貨物。

現在有三輛火車要從城市 A 開往城市 B，火車裝載的貨物重量分別是 5 公斤、22 公斤和 35 公斤。



當這三輛火車到達城市 B 時，它們分別裝載了哪些貨物？

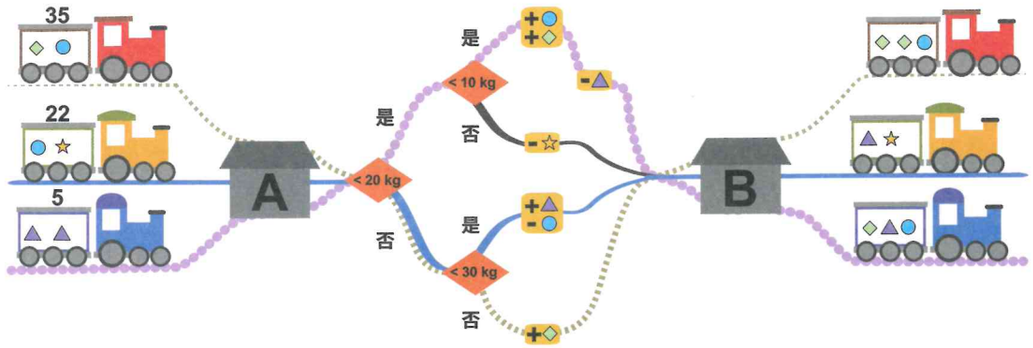
A.

B.

C.

D.

正確答案是：D



- 載運 35 公斤貨物的火車會沿著上圖黃色的路徑行駛，沿途它多裝載了 1 個 ◇，因此到城市 B 時，車上載了 2 個 ◇ 和 1 個 ○。
- 載運 22 公斤貨物的火車會沿著上圖藍色的路徑行駛，沿途它多裝載了 1 個 △，並卸下了 1 個 ○。最後到達城市 B 時，車上載了 1 個 ☆ 和 1 個 △。
- 載運 5 公斤貨物的火車會沿著上圖紫色的路徑行駛，沿途它多裝載了 1 個 ◇ 和 1 個 ○，卸下了 1 個 △。最後到達城市 B 時，車上載了 1 個 ◇、1 個 △ 和 1 個 ○。

資訊科學上的意義

當我們在寫程式時，經常會用到「條件判斷」來控制程式接下來要做什麼。這種做法在資訊科學中稱為 **控制結構 control structures** 的一種，讓程式可以根據不同情況做出不同反應。常見的語法是「如果……那麼……否則……」（英文是 `if...then...else...`），意思是：如果某個條件成立，就做某件事；如果不成立，就做另一件事。在某些任務中，還會出現「巢狀條件判斷」，也就是在一個判斷裡又包含另一個判斷，讓選擇變得更細緻。

在本任務，火車會根據軌道上橘色的載重標示來決定路線，如果重量符合要求就走指定那條路，不符合就走另一條；之後還經過另一個橘色的載重標示再細分路線，經過後續的黃色標誌則直接指示要裝哪些貨或卸哪些貨。這就是一種巢狀的條件判斷結構：每個條件的結果會影響下一步行動。

以日常生活中的選午餐為例，就像是巢狀條件判斷的邏輯。你可能會先想：「如果今天下雨，那就不出門吃飯。」接下來再想：「如果家裡冰箱有做好的食物，那就熱來吃；否則就叫外送。」但如果今天沒下雨，你會選擇出門吃飯，然後又根據不同情況判斷：「如果附近餐廳人太多，那就走遠一點；否則就在附近吃。」這就是在一個條件判斷下，又根據不同情況再做進一步選擇的邏輯流程，正如程式中的巢狀條件一樣，幫助我們一步步做出決策。

關鍵字

演算法（控制結構）

38. 字母繪圖機

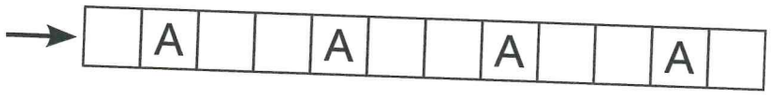
有一台字母繪圖機可以在一張字卡的固定間隔位置印出指定字母所形成的字串，這台機器接受的一組指令必須包含三項資訊，對應的意義如下：

第一個符號：想要印的字母。

第二個數值：字母在字卡上第一次印出的位置。

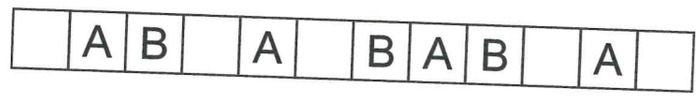
第三個數值：要隔幾個位置再印一次這個字母。機器會繼續對指定字母計算印出位置，直到位置超出字卡長度就結束。

例如，指令「A 2 3」會在字卡的第 2、5、8、11… 個位置印出字母「A」。



這台機器可以接收多組指令，每組指令都會從字卡的開頭開始算位置。如果某個位置已經有印過字母，不會印出覆蓋原有字母，但會繼續計算下個要印出的位置。

例如，執行「A 2 3」和「B 3 2」會得到：



各選項給定的 4 組指令依序執行，何者會印出下面字卡的字串？

A. A 2 2, L 5 4, D 9 5, B 1 2

B. D 9 5, L 5 4, B 1 2, A 2 2

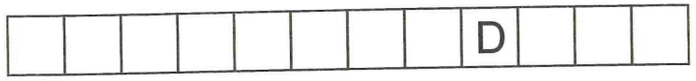
C. A 2 2, B 3 4, D 9 5, L 5 4

D. B 1 2, A 2 2, L 5 4, D 9 5

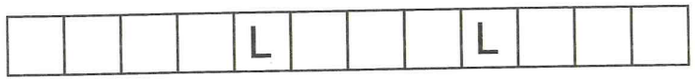
E. L 5 10, D 9 3, A 2 2, B 1 2

正確答案是：B

在空白字卡上，D 9 5 指令執行的結果如下：



在空白字卡上，L 5 4 指令執行的結果如下：



在空白字卡上，B 1 2 指令執行的結果如下：

在空白字卡上，A 2 2 指令執行的結果如下：

因為某個位置若在前面的指令已印過字母，不會覆蓋原有字母。因此循序執行這 4 組指令後的印出結果如下：

其他選項的印出結果如下：

A.

C.

D.

E.

資訊科學上的意義

在資訊科學中，**演算法 algorithm** 是一組用來解決問題的一連串步驟與規則。我們可以把它想成是一套「怎麼做事情」的指令流程，而要讓這些指令執行起來有條不紊，演算法中可運用三種基本結構：**循序結構**（照順序執行）、**重複結構**（重複做一樣的事），以及 **選擇結構**（根據條件決定要怎麼做）。

在本任務中，字母繪圖機接收到一組指令後，會從指定的開始位置，根據輸入的間隔，一步步往右反覆印出字母，直到超出整張卡片的長度，這就是「重複結構」的應用。而如果印的位置上已經有其他字母，機器就會跳過不印，這就是「選擇結構」的例子：根據條件決定動作。

在生活中，舉例來說，做料理的食譜本身就是一種「演算法」：你會按照順序一步步操作（循序結構），像「攪拌三次」就是重複動作（重複結構），而「如果太鹹就加水」這種根據狀況處理的方式，就是條件判斷（選擇結構）。另一個很貼近日常的例子是搭車通勤：你會根據「今天塞車嗎」來決定是搭公車還是改搭捷運（選擇結構）；搭車時，車子會一站一站開（循序結構）；而且這整個流程幾乎是每星期五天的早上都要重複的（重複結構）。

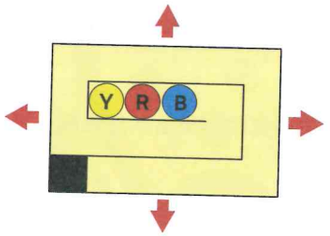
關鍵字

演算法（重複結構、選擇結構）

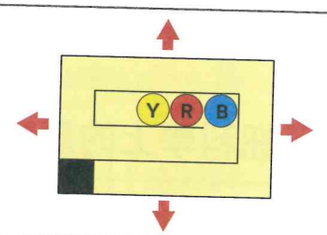
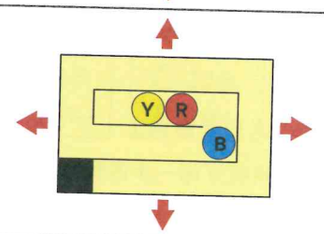
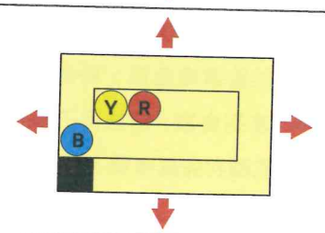
39. 糖果盒

透明盒子裡有三顆糖果球，小羅將盒子水平放置，然後將盒子每次往 ↑、↓、← 或 → 其中一個方向擺動來移動糖果球，這樣重複幾次，直到把想要的糖果球從盒子的左下角黑色開口處取出來。

從盒子的上方可看到裡面有 3 顆糖果球（黃色、紅色、藍色，分別用 Y、R、B 三個字母代表）。



將糖果盒擺動後，糖果球的移動範例如以下說明：

步驟一：執行 → 方向擺動，3 顆糖果如圖所示方向移動。因為右邊有阻隔，所以 3 顆糖接續往右移動，碰到阻隔後停住。且因為糖果盒是水平擺放，所以 B 糖果不會往 ↓ 方向移動。	
步驟二：執行 ↓ 方向擺動，只有 B 糖果往 ↓ 方向移動，另外 2 顆糖果被阻隔而未移動。	
步驟三：執行 ← 方向擺動，3 顆糖往 ← 方向移動，碰到阻隔後停住，如圖所示。	

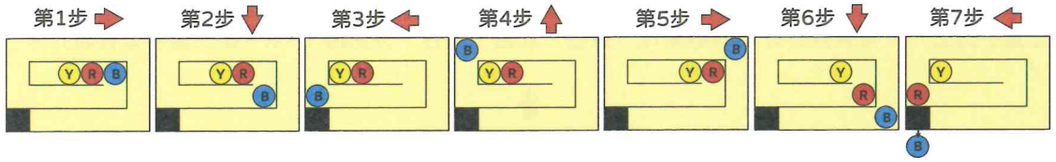
請問，照這樣的方式擺動糖果盒，最少需要幾個步驟可將紅色糖果球 (R) 取出來？（範圍 [1~20] 的整數）

正確答案是：11 步

最佳解法是 11 步。

要取出紅色糖果前，必須先取出藍色糖果。

用以下 7 個步驟標示的方向擺動糖果盒，第 7 步後，藍色糖果 B 會從洞裡掉出來。每一步的操做，其他糖果的位置也會跟著改變。



接下來，再按以下 4 個步驟標示的方向擺動糖果盒，紅色糖果 R 就會從洞裡掉出來：

第 8 步：將盒子向上傾斜

第 9 步：將盒子向右傾斜

第 10 步：將盒子向下傾斜

第 11 步：將盒子向左傾斜

總共需要 11 步，沒有更好的解法，因為紅色糖果只能用這個方式跟步驟取出來。

資訊科學上的意義

循序結構 sequential structure 是演算法中最基本的控制流程之一，意思是程式指令要一行接一行、照順序執行，每個步驟的結果都會影響下一步。當一個任務必須按照正確的流程進行，循序結構就變得非常重要。

在本任務中，機器人只能透過上、下、左、右四個方向來移動球，還必須先拿出藍球、才能拿紅球。需要我們事先規劃好每一步的動作順序，才能讓機器人用最少的步驟順利完成任務。如果順序錯了，就要花更多步才能解題成功。

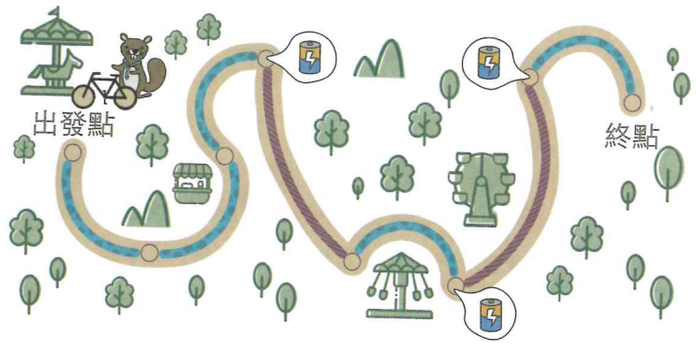
在日常生活中，我們也常會遇到這種「不能跳步」的情況。像是在 ATM 提款時，我們一定要先插卡、再輸入密碼、然後選擇金額，最後才能領錢。到學校餐廳吃飯必須先排隊、點餐、付款、 等待取餐 ，才能順利拿到食物。這些例子都說明了循序結構的特性：每個步驟要照順序來，不能跳、不能亂調順序，才能順利完成事情。

關鍵字

演算法（循序結構）

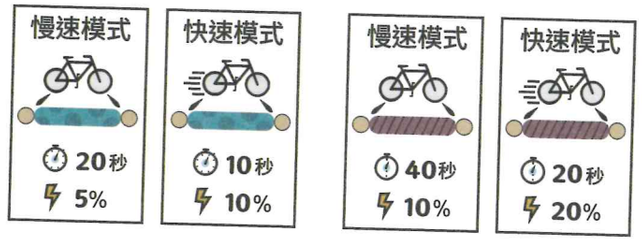
40. 電動自行車

小狸發現遊樂園快要關門了，他跳上電動腳踏車，往出口前進。下面的地圖顯示了小狸從出發點到終點的路線圖。路上有 、 兩種路段及充電站 ，充電站可以瞬間讓腳踏車的電量 回升 20%。



小狸的電動腳踏車有兩種速度模式：慢速和快速。每個路段他可以選擇一種速度模式。

下表是不同路段的快速模式與慢速模式花費的時間 和消耗電量百分比 ：



小狸的電動腳踏車在出發時有 20% 的電量 ，他必須在電量耗盡之前到達出口。

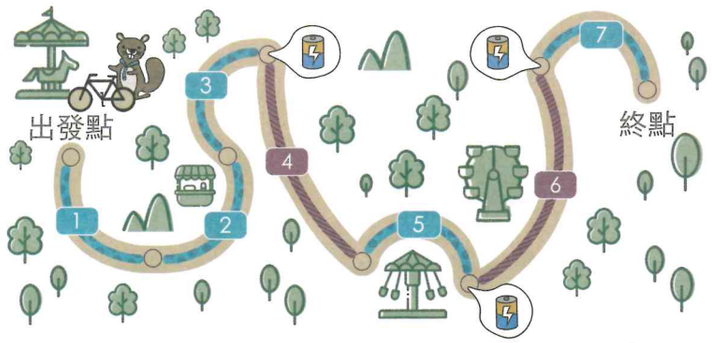
小狸最快花幾秒可以到達出口？（範圍 [1~200] 的整數）

正確答案是：130

答案是 130 秒。

為了儘快完成任務，小狸需要在電量足夠的情形下使用最快的速度模式。

小狸可以根據充電站的位置將路段分為四個區段 (路段 1-3、路段 4-5、路段 6、路段 7) 來規劃電量使用，如下表所示：



剩餘電量花費時間

區段	自行車速度模式	使用電量	剩餘電量	花費時間
從入口到第一個充電站間 (路段 1-3)	三個路段可用總電量至多 20%，因此規劃 -1 段 用快速模式 -2 段 用慢速模式	10%+5%x2 = 20%	0%	10 秒 +20 秒 x2 = 50 秒
到第一個充電站 電量回升20%			20%	
從第一個充電站到第二個充電站間 (路段 4-5)	兩個路段可用總電量至多 20%，因此規劃路段 4 - 用慢速模式 路段 5 - 用快速模式	10%+10% = 20%	0%	40 秒 +10 秒 = 50 秒
到第二個充電站 電量回升20%			20%	
從第二個充電站到第三個充電站間 (路段 6)	路段 6 - 用快速模式	20%	0%	20 秒
到第三個充電站 電量回升20%			20%	
從第三個充電站到出口間 (路段 7)	路段 7 - 用快速模式	10%	10%	10 秒

因此，小狸至少需要 50 秒 + 50 秒 + 20 秒 + 10 秒 = 130 秒才能到達出口。

資訊科學上的意義

在資訊科學中，當我們希望在多個可行方案中找出在特定條件下表現最好的選擇，這類問題被稱為**最佳化問題 optimization problem**。為了解決這類問題，通常需要設計合適的策略，從各種可能的組合中選出最能達成目標的那一組結果。

在本任務中，小狸要從出發點前往終點，每段路程都可以選擇快速模式或慢速模式前進。不同模式下的耗時與電量消耗不同，加上小狸的起始電量有限，而且充電站在固定位置，因此必須在有限條件下找出耗時最少的移動策略。這正是一個典型的最佳化問題：在多種可行方案中，找出花費時間最短的路徑組合。

在解決最佳化問題時，條件限制不僅僅是約束，也是加速計算的重要依據。透過分析條件與目標之間的關係，可以在搜尋過程中提前排除不可能成為最佳解的選項，節省運算。這種方式稱為剪枝 (pruning)，例如當某個方案已違反限制 (如電量不足)，就可略過而不需考慮。

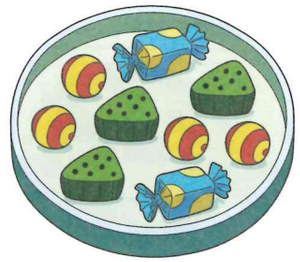
類似的情境在生活中也經常遇到。例如當我們要從甲地前往乙地，可能有多種交通方式可選，例如搭捷運、公車、騎車或走路等。若目標是花費時間最短，我們會選擇最快的方案；但若希望節省費用，可能會改選價格最低的交通組合。這些情況屬於不同目標下的最佳化決策問題。其實人工智慧 (AI) 系統在訓練 (學習) 的過程中，也是運用最佳化的概念。我們可以把訓練 AI 想成是「調整參數，讓它的表現越來越好」。一開始，AI 模型的表現可能很差，但系統會不斷嘗試不同的設定，看看哪一種設定讓結果更接近正確答案。這個過程就像是在從成千上萬種可能中，慢慢找到最好的那一組設定方式，讓 AI 的預測或判斷越來越準確。無論是讓電腦學會辨識人臉、理解語句，還是推薦影片，其實背後都是透過不斷比較與調整來找出表現最好的做法。這些調整步驟，就是一種最佳化的過程。

關鍵字

最佳化問題

41. 糖果

小蓉有 9 顆糖果，想要分給她的朋友們：



- 阿俊會拿走盤子中所有  的一半（如果是小數，就無條件捨去，例如 2.5 會變成 2）。
- 只要該形狀的糖果至少有 2 顆，小明會拿走這些形狀的糖果各 1 顆。
- 小美會拿走 2 顆 

小蓉想留下最多的糖果，應該按照什麼順序讓朋友們拿？

A. 阿俊、小明、小美

B. 小美、小明、阿俊

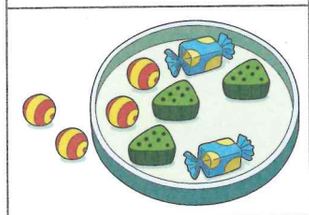
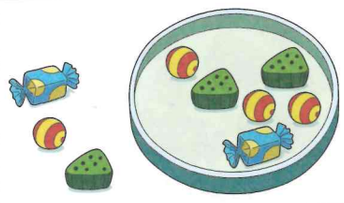
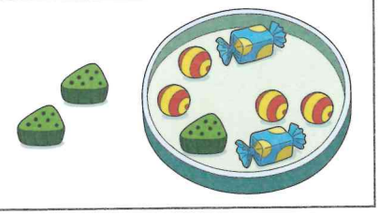
C. 小明、阿俊、小美

D. 小美、阿俊、小明

正確答案是：B

正確答案是 B，先給小美，再給小明，最後再給阿俊拿。

分別考慮三位朋友如果是第一個拿糖的人，拿完後的結果如下：

阿俊第一個拿	小明第一個拿	小美第一個拿
		

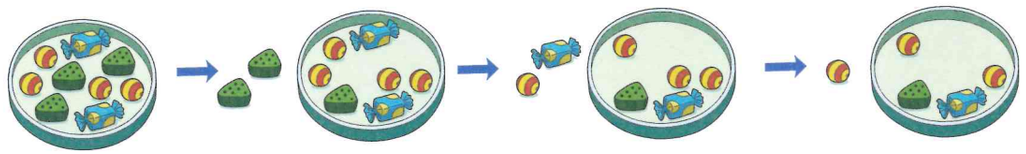
根據接下來第 2 個拿糖果的朋友是誰，可列舉出 6 種可能的拿糖順序。

但我們也可以不用列舉出每一種拿糖順序的結果，而是分析每個人拿糖果的數量是否會受其他人的先後順序影響。

1. 小美無論第幾個拿，都會拿走 2 顆  。阿俊不會拿  ，所以小美拿糖的順序只會影響小明拿走的糖果數量。
2. 小明如果在小美之後拿，會拿走 3 顆（每種形狀各一顆）；如果在小美之前拿，小明只會拿走 2 顆（沒有  ）。因此，小明應該在小美之後拿糖果，才能留下較多的糖果。
3. 阿俊拿走的糖果數量，則受到小明是否比他先拿而影響。如果阿俊比小明先拿，會拿走 2 顆  ；如果在小明之後，則只會拿走 1 顆  。所以，要留下較多的糖果，阿俊要在小明之後拿。

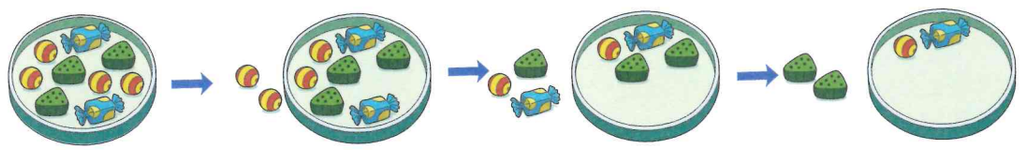
綜合上述分析，讓小美第一個拿，接著是小明，最後是阿俊，可以留下最多糖果。

過程如下圖，小蓉會留下 4 顆糖果。

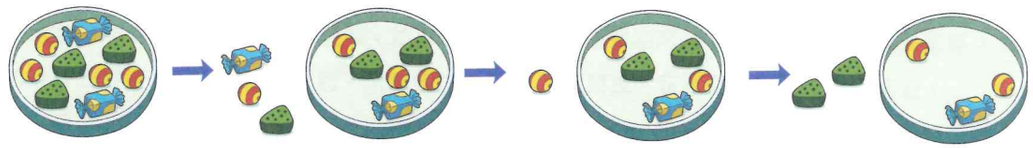


在其他選項 A、C 和 D 的情況下，小蓉留下的糖果都不到 4 顆。

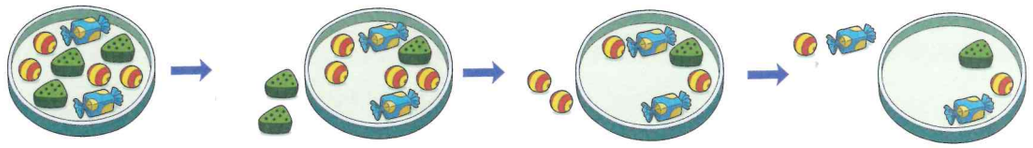
選項 A 的拿糖過程如下圖，只會留下 2 顆糖果；



選項 C 的拿糖過程如下圖，將留下 3 顆糖果，



選項 D 的拿糖過程如下圖，也是留下 3 顆糖果。



根據 B 的順序，小蓉會留下最多的糖果。

資訊科學上的意義

在資訊科學中，當我們希望在多個可行方案中找出符合特定條件的最佳選擇，這類問題被稱為**最佳化問題 optimization problem**。要解決最佳化問題，其中一種常見且直觀的方法是**窮舉法 exhaustive search**。窮舉法的核心概念是：將所有可能的方案一一列出，再進行模擬與比較，從中挑選出達到目標值最高的那個方案。

在本任務中，小蓉希望能留下最多的糖果，但三位朋友在拿糖果時各有不同的規則，且拿取的順序會影響最終的結果。為了找出小蓉留下糖果最多的順序，我們可以使用窮舉法：先列出三位朋友所有可能的拿糖順序組合（共有 3!=6 種），然後逐一模擬每種情況，根據規則計算每種順序下小蓉剩下多少糖果。最後，我們從中挑選出讓小蓉糖果數量最多的組合，這就是這個任務中的最佳解。

這類最佳化與窮舉策略的思維方式在生活中也很常見，舉例來說：當我們在有限預算下購物時，若想從購物清單中選出最多需要的物品組合，就可能需要列出各種採購組合進行比較，最後挑出既不超出預算又能達到效益最大化的選項。

關鍵字

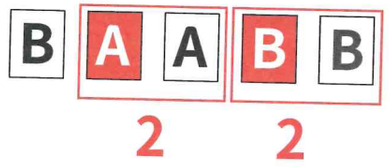
最佳化問題、窮舉法

42. 字母排列最高分

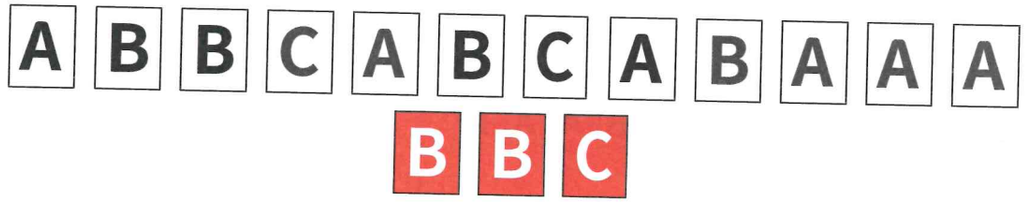
這是字母卡 A、B 和 C 排列的遊戲，如果排列順序中有兩個相同的字母連在一起，玩家獲得 2 分，如果有三個相同的字母連在一起，玩家獲得 3 分，以此類推。玩家要盡可能獲得最高分。

如下圖，上排五張的字母卡排列，因為沒有相同的字母連在一起，這樣獲得 0 分。但下排會提供兩張紅色卡片，必須覆蓋到上排的字母卡上，讓覆蓋後的字母卡排列順序得到最高分。

例如將紅色 A 卡片覆蓋在第一個 C 上面，紅色 B 卡片覆蓋在第二個 C 上面，因為有兩組連續 2 個相同字母，得分是 2 + 2 = 4 分；第一個 B 旁邊沒有連著相同的字母，所以不會得分；總得分是 4 分。

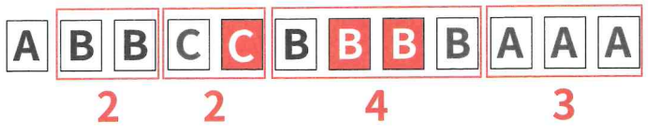


如果提高難度，一開始拿到的 12 張字母卡排列如下，要玩家將三張紅色卡片覆蓋到原字母排列的任 3 張卡上面。



請問玩家能拿到最高的分數為多少？（範圍 [1~30] 的整數）

正確答案是：11 分

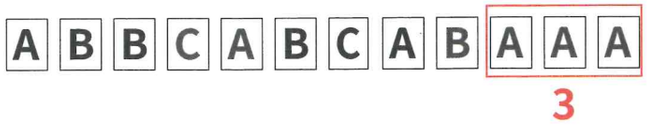


覆蓋紅色卡片後排列最高分的解如上圖所示，可得到 2 + 2 + 4 + 3 = 11 分。

找出最高分的思考方式如下。

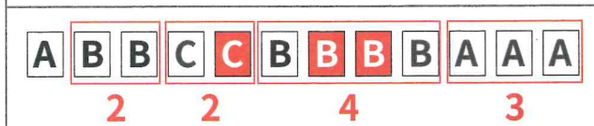
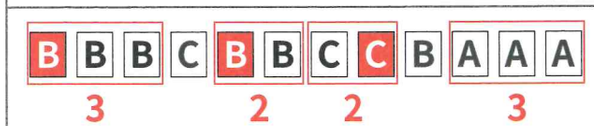
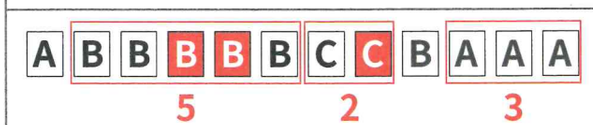
觀察紅色卡片及原字母排列。

1. 紅色卡片中沒有 A ，沒辦法讓原來單獨不連續的 A 卡片連成更長。A 卡片最多形成最右邊連續 3 張牌，得到 3 分。



2. 原來單獨不連續的 C 卡片有 2 張，中間相隔 2 張卡片；1 張紅色 C 卡片覆蓋位置要在原 C 卡片的旁邊 (增加 C 卡片連續長度)，最多能形成連續 2 張牌；不要覆蓋到 B 卡片上，才能讓 B 卡片連續更多張。
3. 2 張紅色 B 卡片覆蓋位置要在原 B 卡片的旁邊，以增加 B 的連續卡片數。

因此只需考慮以下幾種覆蓋方式：

紅色 C 卡片覆蓋在第 5 張卡片 A 上	
	得分 3 + 2 + 2 + 3 = 10。
	得分 2 + 2 + 2 + 2 + 3 = 11。
紅色 C 卡片覆蓋在第 8 張牌卡片 A 上	
	得分 3 + 2 + 2 + 3 = 10。
	得分 2 + 3 + 2 + 3 = 10。

因此找出能拿到的最高分是 11 分。

資訊科學上的意義

在資訊科學中，當解任務時希望在多個可行方案中找出符合特定條件，找出表現最好的選擇，這類問題被稱為**最佳化問題 optimization problem**。為了解決這類問題，其中一種常見且直觀的方法是**窮舉法 exhaustive search**。窮舉法的核心概念是：將所有可能的方案一一列出，逐一模擬執行與比較，最後從中挑選出能達到目標值最高的那一組作為最佳解。

在本任務中，若要獲得最高分數，我們可以使用窮舉法來列出紅色字母卡所有可能的放置位置組合，然後根據任務規則計算每一組放法所能得到的總得分。最後從這些結果中，挑出分數最高的那一組組合，即為此任務中的最佳解。但在實際操作中，並不一定要執行每一個可能。我們可以在過程中加入基本的評估機制，提前排除某些明顯不會產生更好結果的選項。這種做法類似於演算法中的「剪枝 (pruning)」技巧，能有效減少不必要的計算，提高效率。例如，如果某個字母卡的放置方式明顯遮住高分區域，或重疊後產生的得分不可能高於目前的最佳結果，就可以不需要考慮。

這類透過列舉與比較來找到最佳解的策略，以點餐為例，當我們有預算限制時，若希望從菜單中選出最多喜歡的餐點，就可能先排除價格太高、或包含不喜歡食材的選項，再比較剩下的套餐與單點的各種搭配方式。這種在窮舉過程中加入初步判斷與過濾的方式，就是最佳化問題實務上的常見做法。

關鍵字

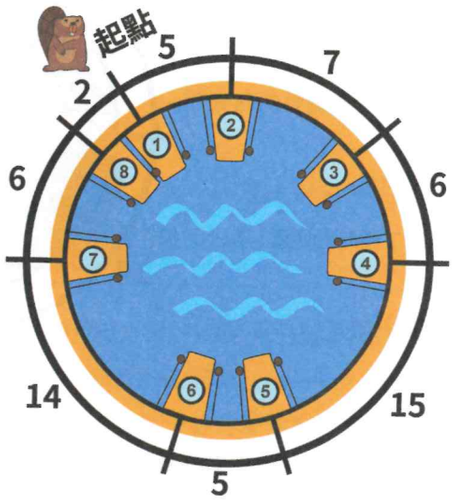
最佳化問題、窮舉法

43. 繞圈圈

海狸小波要負責到湖泊周圍的每個碼頭檢查一遍。

他從起點碼頭開始，沿著湖邊走，每到一個碼頭檢查完，可以往順時針或逆時針方向走到下一個碼頭。當檢查完所有碼頭後，他最後檢查的碼頭會有一艘船送他回起點，所以不必擔心最後這段返回的路程。

下圖顯示碼頭間的距離。



例如：小波如果用以下順序檢查碼頭：

①→②→③→④→⑤→⑥→⑦→⑧，這樣要走的距離是 58 單位；

如果檢查碼頭的順序是：

①→②→③→④→⑧ →（逆時針）⑦→⑥→⑤，這樣要走的距離是 63 單位。

小波想找出可檢查完所有碼頭需要走的最短距離，請問他最短要走多少單位？

（範圍 [1~60] 的整數）

正確答案是：54

小波必須檢查所有碼頭。沿著湖邊走一圈的距離是 60，一定能對每個碼頭都檢查一遍。

如果想省下走相鄰碼頭 A 到 B 之間的路 (碼頭 A 順時針方向的下個是碼頭 B)，可以先從起點 (1 號碼頭) 順時針沿著湖邊檢查到 A 碼頭 ，然後回到起點，再逆時針沿著湖邊檢查到 B 碼頭，省下碼頭 A 和 B 之間的路，但會多走從 A 逆時針方向回到起點的距離。

另一個方式是從起點逆時針沿著湖邊檢查到 B 碼頭，然後回到起點，再順時針沿著湖邊檢查到 A 碼頭，省下碼頭 A 和 B 之間的路，但會多走從 B 順時針方向回到起點的距離。

例如：要省下走 ④ 到 ⑤ 這一段，有兩種走法：

第一種：沿湖邊順時針檢查到 ④，逆時針方向回到起點，再逆時針沿湖邊檢查到⑤。

①→②→③→④→③→②→①→⑧→⑦→⑥→⑤

「省下」**距離**(④,⑤) = 15，多走了**逆時針距離**(④,①) = (5+7+6) = 18。

第二種：沿湖邊逆時針檢查到 ⑤，順時針方向回到起點，再順時針沿湖邊檢查到④。

①→⑧→⑦→⑥→⑤→⑥→⑦→⑧→①→②→③→④

同樣「省下」**距離** (④ , ⑤) = 15，但需要再多走 **順時針距離** (⑤ , ①) = (2+6+14+5) = 27。

兩種方法中，選擇第一種走法，相對於走一圈的距離，最多可以「省下」的距離是 -3。(負的意思代表沒有省下，其實是更長的走法)。

距離 (④ , ⑤) – 較小值 (逆時針距離 (④ , ①), 順時針距離 (⑤ , ①)) = 15 - 較小值 (18,27)= -3。

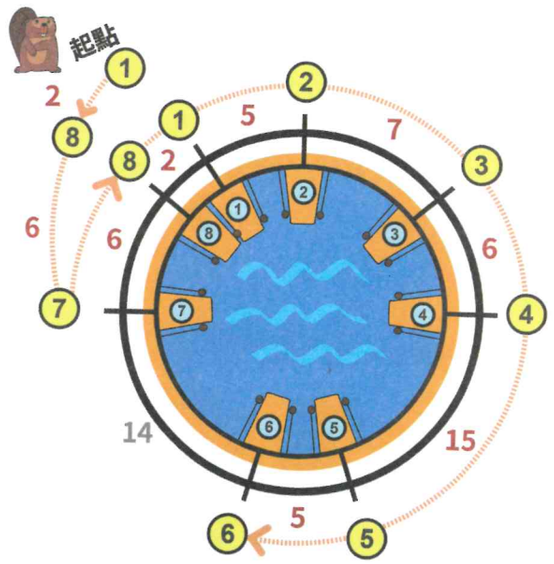
因此，要找到最短距離，就要找到可省下最多距離的兩個順時針相鄰碼頭配對。

對所有順時針相鄰碼頭配對，其可省下最多距離的計算如下：

①, ②	②, ③	③, ④	④, ⑤	⑤, ⑥	⑥, ⑦	⑦, ⑧	⑧, ①
5 - 較小值 (0, 55) = 5	7 - 較小值 (5,48) = 5	6 - 較小值 (12,42) = -6	15 - 較小值 (18 ,27) = -3	5 - 較小值 (33 22) = -17	14 - 較小值 (38, 8) = 6	6 - 較小值 (52, 2) = 4	2 - 較小值 (58, ,0) = 2

小波從起點 ① 出發，逆時針方向先檢查碼頭 ⑧ 和 ⑦，回到起點①，再順時針方向檢查碼頭 ① 到 ⑥ 的所有碼頭。

①→⑧→⑦→⑧→①→②→③→④→⑤→⑥



可比繞一圈節省最多距離 (6)。

所以，最短距離為：

2+6+6+2+5+7+6+15+5=54。

資訊科學上的意義

在資訊科學中，當我們希望在多個可行方案中找出在特定條件下表現最好的選擇，這類問題被稱為**最佳化問題 optimization problem**。**旅行推銷員問題 traveling salesman problem, TSP** 是一個經典的最佳化問題，這個問題的目標是：在給定一系列城市及它們之間的距離，找出一條能走遍所有城市一次，且回到起點的最短路線。這類問題在地點數量一多時，可能的走法會爆炸性成長，因此被認為是計算上非常困難的問題。

在本任務中，小狸需要走訪所有的碼頭，這是旅行推銷員任務的簡化情境，因為所有碼頭位置剛好圍成一圈，且最後一趟可搭船回到起點。若單純按照順時針或逆時針的順序繞行，可能會因為某兩個碼頭之間的距離過長，導致總路程大幅增加。我們需要考量避開那些距離特別遠的路段，同時也要注意有些區段會被來回經過，可以利用節省路段跟重覆走增加的長度來相較，計算可節省多少總距離，就能找到整體最短的行走方案。

這類路徑最佳化問題在生活中也非常常見。例如郵差在投遞信件時，要規劃出一條最有效率的送信路線；物流配送業則經常需要安排貨車收 / 送貨順序，讓整體行程最短、時間最省。這些情境類似於一個旅行推銷員問題。

關鍵字

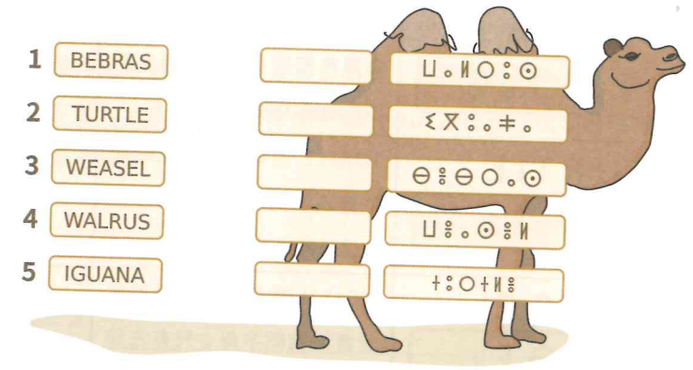
最佳化問題、旅行推銷員問題

44. 提非納字母

提非納字母是北非貝爾族使用的文字符號，提非納字母表會將每個提非納字母對應到一個英文字母。小丁查了提非納字母表，將下列五種動物的英文名稱轉成用提非納字母編碼：

1.BEBRAS　2.TURTLE　3.WEASEL　4.WALRUS　5.IGUANA

編碼後的結果如下圖：

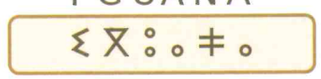
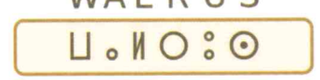


依照由上到下的順序寫出這五個編碼單字分別為哪一種動物的編號？
（請寫出 5 位數字，代表由上到下的順序，如：12345）

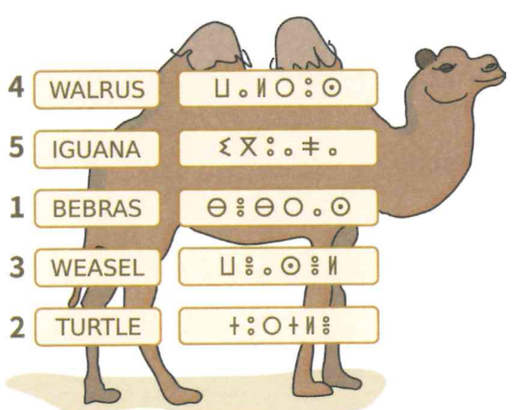
正確答案是：45132

我們可以由英文單字跟提非納字母編碼結果，觀察其中字母跟符號的出現數量及位置，找出提非納字母跟英文字母的對應關係。

1. 首先檢查每個動物單字的字母數量，所有單字都有 6 個字母，無法依照字母數量來找出對應。
2. 因此，我們接著查看每個單字中字母出現的模式，例如：一個單字中有相同字母出現的次數及位置、不同單字中有相同字母出現的位置等。

動物編號	英文單字對應提非納字母編碼	說明	編碼單字
1	BEBRAS 	BEBRAS 的第一個和第三個位置為相同字母；只有左圖這個編碼出現這種模式。	第 3 張編碼
2	TURTLE 	TURTLE 的第一個和第四個位置為相同字母；只有左圖這個編碼出現這種模式。	第 5 張編碼
3	WEASEL 	WEASEL 的第二個和第五個位置為相同字母；只有左圖這個編碼出現這種模式。	第 4 張編碼
5	IGUANA 	IGUANA 的第四個和第六個位置為相同字母，只有左圖這個編碼出現這種模式。	第 2 張編碼
4	WALRUS 	所以剩下動物單字 WALRUS 對應到左圖。	第 1 張編碼

因此這五個編碼單字分別對應的動物編號，依序為 45132。



資訊科學上的意義

本任務結合了 **密碼學 cryptography** 與模式識別 (pattern recognition) 的核心概念，重點在於不同字母系統之間的轉換與辨識能力。在資訊科學中，資料經常需要透過各種字母系統進行編碼與處理。例如電腦系統使用二進位 (binary) 由 0 和 1 組成資料，來進行儲存與運算；凱撒密碼 (Caesar cipher) 等密碼系統則利用符號的替換與重新編排，來達成資訊加密與保護隱私的目的。

在本任務中，英文單字被編碼為提非納字母，這是一種典型的字母映射問題 (alphabet mapping)。要完成解碼，參與者必須透過模式匹配 (pattern matching) 技術，觀察符號之間的對應關係，推敲出每個符號代表的英文字母，進而還原出原始英文詞彙。

在日常生活中，字母映射的概念有許多實際應用；其核心概念在於：將一套符號 (如字母、代碼) 對應到另一套符號或意義上，以達成溝通、轉譯或保密的目的。舉例來說，輸入法系統就是一種常見的字母映射應用：使用者在鍵盤上輸入注音或拼音，系統會將這些音碼「映射」到正確的中文字上，讓使用者完成文字輸入。此外，國際摩斯密碼 (morse code) 也是將每個英文字母對應為一串點與劃的符號，用來傳遞訊息。

關鍵字

密碼學

45. 時間會告訴你

鑽石藏在一個需要輸入密碼才能打開的盒子裡，且每輸入一個字母就會檢查是否與密碼匹配，每檢查一個字母需要 1 秒鐘。

如果輸入的字母與密碼不匹配，警報會立即響起。例如，如果前兩個字母匹配，但第三個字母不匹配，警報會在 3 秒後響起。

海狸嘗試了許多不同的密碼，並用筆記記下了警報響起前經過的秒數。但海狸不小心把筆記掉進田裡，導致部分字母被泥巴覆蓋了，被泥巴覆蓋的字母在下表以星號（＊）表示。

猜測	警報響起前經過的秒數
＊CORN＊DOG＊＊＊	1
＊＊＊＊＊＊DOG＊＊＊	8
＊E＊VER＊Y＊＊＊＊	7
＊＊＊VER＊Y＊＊＊＊	2
BE＊B＊R＊AS＊＊＊	4
＊＊＊＊＊R＊AS＊＊＊	9
＊＊A＊＊＊DAM＊＊＊	5
＊＊＊＊＊＊＊＊M＊＊＊	10

儘管如此，海狸說他仍然可以分析出盒子密碼前 9 碼的字母。

請問盒子密碼前 9 碼的字母是什麼？

正確答案是：BEAVERDAM

從任務中我們知道，如果警報在 n 秒後響起，那麼海狸可以確定他猜測的前 n-1 個字母是正確的，而第 n 個字母是錯的。

我們用一個例子來說明，如果警報在 3 秒後響起：

- 1 秒時，猜測的第一個字母與實際密碼的第一個字母進行比對。警報沒有響起，表示猜測的第一個字母是正確的。
- 2 秒時，將猜測的第二個字母進行比對。警報沒有響起，表示猜測的第二個字母也是正確的。
- 3 秒時，將猜測的第三個字母進行比對。然後警報響起，表示猜測第三個字母是錯的。
- 警報已經響起後，猜測密碼包含第四個字母之後的正確性就無從得知。

根據這樣的推理方式，可以對題目表格中提供的資訊逐一分析，辨認出實際密碼的某幾個字母。以下表格中的「猜測」列中的字母中若有陰影，表示是猜測正確的字母，**確認的密碼字母中的粗體字母，表示透過該次猜測正確且沒被蓋住的字母**。

猜測	警報響起前經過的秒數	確認的密碼字母
*CORN*DOG***	1	*********
******DOG***	8	******D**
*E*VER*Y****	7	*E*VERD**
VER*Y*	2	*E*VERD**
BE*B*R*AS***	4	BE*VERD**
*****R*AS***	9	BE*VERDA*
A*DAM***	5	BEAVERDA*
********M***	10	BEAVERDAM

因此最後可分析出正確密碼。

資訊科學上的意義

其中，「時間分析攻擊 timing attack」是常見的一種方式，透過測量比對密碼所花費的時間，逐步還原出正確的密碼組合。

在 **資訊安全 information security** 領域中，有一類特別的攻擊方式稱為 **旁路攻擊 side-channel attack**。這類攻擊並不是直接破解密碼，而是利用電腦或系統在運作過程中所洩漏的間接資訊，例如運行時間、電力消耗、處理頻率等，來間接推測出密碼或機密資料。其中一種常見的旁路攻擊方式是「時間分析攻擊 timing attack」。攻擊者會透過精確測量系統處理每個步驟所需的時間，來逐步還原密碼。例如：如果某一組輸入花費的處理時間比其他組合稍長，就可能代表該組輸入有部分是正確的。

本任務就模擬了這種攻擊方式。每當輸入的密碼字母與正確密碼不一致時，系統會立刻響起警報，並且每比對一個字母需耗時一秒。因此，攻擊者可以根據警報響起的秒數，推測密碼的前幾個字母是否正確。舉例來說：若輸入 "A" 或 "C"，警報在 1 秒後響起，表示第一個字母不正確；若輸入 "B"，警報在 2 秒後響起，則代表第一個字母正確，但第二個錯誤；依此類推，攻擊者可逐步推敲出完整密碼。即使密碼總共有高達 5.4 兆種組合 (9 位英文字母的排列)，但透過這種方法，最多只需嘗試 234 次（26 × 9），就能將密碼完全破解。

在日常生活中，旁路攻擊的概念其實也經常用於行為分析與數據推測。舉例來說，在智慧家庭中，可以從電力消耗的變化判斷哪台電器正在運作，例如冷氣開啟時電表跳動會特別快；在辦公室裡，如果某位同事的電子郵件發信時間在特定時段，可能反應他工作作息的固定模式；在醫療監測中，也可藉由心跳、血壓等生理數據的微小變化，間接發現身體潛在的異常狀況。這些例子雖然不是電腦系統上的資安攻擊，但都運用了旁路攻擊的推理原理：觀察間接線索，推論出背後的事實。

關鍵字

資訊安全、旁路攻擊