中原大學資訊工程系 演算法分析第一次機測

Deadline: 4 / 12 / 2024 (星期五)
(限期中考前測完，逾期不得補繳)

【程式設計說明】

1. 每組限 2～3 人，組員須固定，本學期不得任意變更。原則上以專題組員為主。
2. 組員應合作共同解題，但嚴禁跨組合作。
3. 程式設計必須使用 Python 程式語言，版本請採用目前最新版本 (原則上，請直接下載與安裝 Anaconda)。
4. 可參考課本、參考書籍或網站資料等進行解題，解題方法及演算法不限，但嚴禁抄襲他組程式，組員均有責任保護程式不被他組抄襲。**<u>若發現抄襲屬實，兩組均以零分計。</u>**
5. 輸入與輸出採用標準格式或讀寫檔案方式進行。
6. 每一支程式均須附上組員姓名及學號，例如:

```
#    演算法分析機測
#    學號: 11027XXX / 11027XXX
#    姓名: 陳○○ / 林○○
#    中原大學資訊工程系
```

程式命名依該組學號在前之同學 [學號+題號] 為原則。例如:
11027001_1.py
11027001_2.py
…
11027001_5.py

【機測須知】

1. 評分以解題成功之題數多寡與執行時間決定。
2. 程式必須能處理不同的輸入資料 (但輸入格式與範例相同)，並輸出正確結果 (輸出格式必須與範例相同)，組員應能說明程式設計內容，方可視為成功。程式的輸出結果錯誤、輸出格式與範例不符、或在執行後超過 5 秒 (以每筆測資為基準) 仍未結束，均視為失敗。若程式測試失敗給予基本分數，未繳交程式則以零分計。
3. 本機測於規定之期限前，各組應攜帶程式原始碼至電學大樓 603 室找助教測試 (電話：265-4726)，每組限繳交一次，不可分題或多版本繳交，逾期不得補繳。
4. 助教將使用不同之輸入資料作為測試與評分依據，同學應在繳交前充分測試程式。
5. 機測成績納入學期平時成績計算，請同學把握。

指導教授: 張元翔

【執行時間測試】

機測預計採用個人電腦 CPU Intel i7、8G RAM、作業系統以 Windows 10 為主。建議同學在繳交程式前先使用下列 Python 程式進行初步的執行時間測試：

```
import time
start_time = time.time()
……
total_time = time.time() – start_time
print(total_time)
```

I. 二元樹密碼 (Binary Tree Secret Code)

國防部擬發展一套簡易的編碼方式來傳遞訊息，其編碼方法如下：首先將傳遞的訊息放在一棵二元樹，每一個節點放一個字元，且**後序搜尋** (Postorder Traversal) 所產生的字串即是解碼後的訊息。當傳遞此訊息時，乃是將這個二元樹的**前序**及**中序搜尋** (Preorder and Inorder Traversals) 所得的字串傳送出去，當對方接收到這兩個字串，則可將其轉換成原先存在二元樹中的訊息，請編寫程式來完成這件事。

我們要傳送的訊息只包含不同的大小寫英文字母 A～Z、a～z 或數字 0～9，同時該訊息長度最多為 60 個字元，大小寫英文字元視為不同。

輸入說明:
輸入為兩兩一組之傳送訊息，分別為前序及中序搜尋，以換行隔開。0 代表結束。

輸出說明:
顯示解碼後的訊息(即後序搜尋結果)。

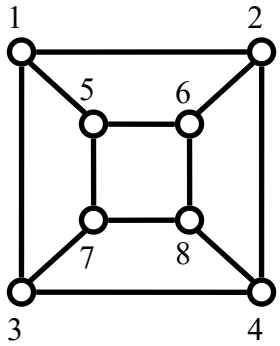
輸入範例:
abcde
cbdae
ACEIFJBLGKDH
IECJFAGLKBHD
0

輸出範例:
cdbea
IEJFCGKLHDBA

II. 漢密爾頓迴圈 (Hamiltonian Cycle)

給定一圖形 $G = (V, E)$，其中 V 代表**頂點** (Vertex) 集合，依阿拉伯數字 1, 2, … 等安排；E 代表**邊** (Edge) 的集合，依其連接的頂點定義。以下圖為例，則頂點的個數為$|V| = 8$，邊的個數為$|E| = 12$。

漢密爾頓迴圈 (Hamiltonian Cycle) 的定義如下：從某一頂點出發（如頂點 1），陸續對其他頂點走訪，但每一頂點僅能走訪一次，且回到原出發點形成迴圈。本題中，將給定圖形，且漢密爾頓迴圈一定存在，試寫一程式列出其中一個漢密爾頓迴圈，且其出發點為頂點 1。



以上圖為例，則其中一個漢密頓迴圈為：1 2 4 3 7 8 6 5 1 (起點及終點均為頂點 1)。注意：本題中，答案不一定是唯一，但僅須列出其中一個漢密頓迴圈即可。

輸入說明:

第一行為頂點及邊的數目（以空格隔開），緊接依序為連接邊的兩個頂點，0 0 代表結束。注意：測試時可能包含多個圖形，但頂點的個數不會超過 10。

輸出說明:

列出其中一個漢密頓迴圈（分別以空格隔開），但起點及終點均為頂點 1。

輸入範例：(同上圖形)

```
8 12
1 2
1 3
1 5
2 4
2 6
3 4
3 7
```

4 8
5 6
5 7
6 8
7 8
0 0

輸出範例

1 2 4 3 7 8 6 5 1

III. 水桶謎題 (Water Jug Puzzle)

假設有兩個水桶及一個水池 (無限供應水)，兩個水桶的容量均為已知，但是都沒有刻度，所以你只能進行下列三種動作：

(1) Fill　　將水桶的水裝滿
(2) Empty　　將水桶的水倒光
(3) Pour　　將其中一個水桶的水倒到另一個水桶

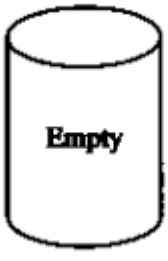
其中，第三種動作僅有兩種可能，即第一個水桶的水須全部倒光、或是第二個水桶已裝滿便算結束。舉例說明，假設水桶 A 及水桶 B 都可容納 8 公升，若此時水桶 A 有 5 公升，水桶 B 有 6 公升，第一種動作可將水桶 A 裝滿，第二種動作可將水桶 A 倒光，第三種動作可將水桶 A 的水倒入水桶 B，但僅可將水桶 B 裝滿到 8 公升，使得水桶 A 剩下 3 公升。

水桶謎題的目的在使水桶 B 達到某給定的水量 (公升)，如圖所示為範例，若水桶 A 的容量為 3 公升，水桶 B 的容量為 5 公升，目標水量為 4 公升，則可達到目標的順序如下 ：

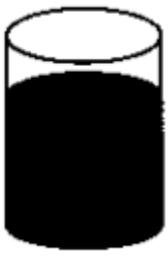
Fill A
Pour A B
Fill A
Pour A B
Empty B
Pour A B
Fill A
Pour A B
Success

其中，Pour A B 表示將水桶 A 倒水倒水桶 B 中。

水桶 A



水桶 B



目標 (水桶 B)

注意：

1. 本題中你可以假設給定的謎題一定有解。
2. 水桶 A 與水桶 B 在剛開始時皆是空的。

輸入說明:

每組有三個數字，第一個數字為水桶 A 的容量，第二個數字為水桶 B 的容量，第三個數字為目標容量，單位均為公升。輸入為 0 0 0 時則結束。

輸出說明:

列出達到目標的順序。

輸入範例:

3 5 4
5 7 3
0 0 0

輸出範例:

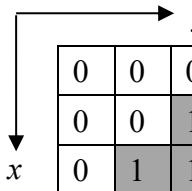
Fill A
Pour A B
Fill A
Pour A B
Empty B
Pour A B
Fill A
Pour A B
Success
Fill A
Pour A B
Fill A
Pour A B
Empty B
Pour A B
Success

※ 本問題曾經出現在電影「終極警探 3」中。

IV. 連通元標記 (Connected Component Labeling)

連通元標記 (Connected Component Labeling) 常用於數位影像處理中，主要目的在標記影像中的某一特定連通元，說明如下：

假設給定一數位影像 (如圖)，其基本構成元素稱為**像素** (Pixel)，本例為 10 × 10 之影像，其中像素值僅含 0 或 1 值，因此這樣的影像又常稱為**二值影像** (Binary Image)。數位影像常以二維矩陣表示，其 (x, y) 座標如圖表示，因此左上角座標為 (0, 0)、左下角座標為 (9, 0)、右上角座標為 (0, 9)、右下角座標為 (9, 9)，其他座標以此類推。



0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	0	0
0	1	1	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0
0	0	1	0	0	1	0	0	0	0
0	1	1	1	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

以下圖為例，考慮某像素 p，則像素 x 均視為與像素 p「相鄰」(Adjacent) 之像素。若兩像素間含有互相相鄰之像素路徑，則視為「相連」 (Connected) 。以下圖為例 (未標記之像素值為 0)，像素 p 與像素 q 相鄰且相連，像素 p 與像素 r 相連但不相鄰，像素 p 與像素 s 不相鄰也不相連。連通元 (Connected Component) 則可定義為相連通之像素集合。

x	x	x
x	p	x
x	x	x

	p				
	q		1	s	
	1			1	
	1				
		r			

連通元標記的目的即是在給定某一二值影像，標記影像中的連通元，上圖結果範例如下：

	0	0	0	0	0	0	0	0	0	0
	0	0	1	0	0	0	2	2	0	0
x	0	1	1	0	0	2	0	0	2	0
	0	0	1	0	0	0	0	0	2	0
	0	0	1	0	0	0	0	2	0	0
	0	0	1	0	0	0	2	0	0	0
	0	0	1	0	0	2	0	0	0	0
	0	1	1	1	0	2	2	2	2	0
	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0

由圖可見連通元標記的結果即是對影像中每一連通元像素給予特定**標籤** (Label)，且依 1、2、3…等順序安排。連通元標記後可決定影像中連通元的個數。標記之標籤原則上依由左而右、由上而下依序排列，且沒有跳號現象。

此外，可以依每個連通元計算其面積 (即總像素個數)，上例中，連通元 #1 之面積為 10、連通元 #2 之面積為 12。

輸入說明:

輸入依下列次序安排，每組輸入資料代表一張二值影像，首先為影像的大小，依高 × 寬安排 (最大為 100 × 100)，影像大小為 0 × 0 代表結束，接著為二值影像，像素值僅含 0 或 1，但影像中可能含有多個連通元。

輸出說明:

輸出應包含下列資訊：(1) 輸入影像編號；(2) 連通元個數；及 (3) 各連通元面積。

輸入範例:

10 10
0000000000
0010001100
0110010010
0010000010
0010000100
0010001000
0010010000
0111011110
0000000000

```
0000000000
8 5
00001
00011
00111
00000
11001
11001
10000
00000
0 0
```

輸出範例:

```
Image #1
Number of Connected Components = 2
Connected Component #1 Area = 10
Connected Component #2 Area = 12

Image #2
Number of Connected Components = 3
Connected Component #1 Area = 6
Connected Component #2 Area = 5
Connected Component #3 Area = 2
```

※ 提示：本問題可參考 Disjoint Set 資料結構。

V. 15-拼圖 (15-Puzzle)

15-拼圖 (15-Puzzle) 是個經典的益智遊戲 (如下圖)，包含 : 4 × 4 個方格，共有 15 個可以自由移動的數字方塊，編號分別為 1 ~ 15。剛開始時，15-拼圖的數字方塊可能是任意排列，遊戲者可以將空格旁邊的數字方塊移入空格內，例如 : 左(L)、右(R)、上(U)、下(D)等，最終目標是排成如圖的整齊排列，且移動的次數必須最少。

輸入說明

輸入含有幾個 15-拼圖，緊接為 4×4 的數字，每個數字均以 2 位數表示，且數字中間以空格隔開。0 代表 15-拼圖的空格，旁邊的數字方塊可以自由移入。拼圖與拼圖間則分行。

輸出說明

首先，列出 15-拼圖的編號。接著，印出最少**移動次數** (Number of moves)。接著，印出所有的移動方式，包含：移動的數字方塊 (以 2 位數表示) 與移動方向。若移動的次數超過 5 次，則以 5 次為單位分行列印。

輸入範例

```
2
 1  2  3  4
 5  6  7  8
 9 10 15 11
 0 13 14 12

 0  1  2  3
 5  6  7  4
 9 10 11  8
```

13 14 15 12

輸出範例

15-Puzzle #1
Number of moves = 5
13L 14L 15D 11L 12U

15-Puzzle #2
Number of moves = 6
 1L 2L 3L 4U 8U
12U

每行最多列印 5 個移動次數，
數字與方向必須對齊