LCS Computing

(10 pts)

設計一函式 LCS_compute()，其功能為計算輸入的兩個序列其最大共同子序列的長度，並輸出。

Input format

輸入的兩個序列，數個數字以空格隔開。

Output format

輸出如 Sample output 所示，輸出最大共同子序列的長度。

Sample input

3 1 4 1 5 9 2 6 5 3

1 4 1 4 2 1 3 5 6 2 3

Sample output

6

說明:
範例中的最大共同子序列可以為
[1, 4, 1, 5, 2, 3] (其中之一例)
所以長度為6。
注意: 若兩序列完全沒有交集，則長度輸出為0。

LCS Checking

(10 pts)

設計一函式 LCS_check()，其功能為檢查函式的第三個輸入的序列是否為前兩個輸入序列其最大共同子序列的其中一例。是則輸出 True，反之輸出 False。

Input format

輸入的三行序列，數個數字以空格隔開。

Output format

第三個輸入的序列為前兩個輸入序列其最大共同子序列的其中一例，輸出 True，反之輸出 False。

Sample input

3 1 4 1 5 9 2 6 5 3

1 4 1 4 2 1 3 5 6 2 3

1 4 1 5 2 3

Sample output

True

說明:
範例中的最大共同子序列可以為
[1, 4, 1, 5, 2, 3]
[1, 4, 1, 2, 5, 3]
[1, 4, 1, 2, 6, 3]
[1, 4, 1, 5, 6, 3]
所以輸出皆為 True
但 [1, 4, 1, 5, 2] 不為最大共同子序列，輸出為 False

LIS Computing

(10 pts)

設計一函式 LIS_compute()，其功能為計算輸入序列其最長遞增子序列的長度，並輸出。

Input format

輸入數個數字，以空格隔開。

Output format

輸出最長遞增子序列的長度。

Sample input

3 1 4 1 5 9 2 6 5 3

Sample output

4

說明:
範例中的最大遞增序列可為
[3, 4, 5, 9] 、 [1, 4, 5, 9]
也可以為 [1, 1, 5, 9]、[1, 1, 2, 6] 等，
所以輸出長度為 4 。
注意: 題目要求是遞增(不是嚴格遞增)，
所以相同數字也符合要求。

Fractional Knapsack Problem

(10 pts)

在可取部份物件(fractional)的條件下，輸入依次為物件的價值與物件的重量兩序列，再加上背包的最大承載上限值，設計一函式KP_fractional()，其輸出為計算背包在承載限制下的最高取物總價。

Input format

輸入的三行序列，數個數字以空格隔開。

第一行輸入物件的價值。

第二行輸入物件的重量。

第三行輸入背包的最大承載上限值。

Output format

輸出如 Sample output 所示，輸出計算背包在承載限制下的最高取物總價。(四捨五入取至整數)

Sample input

6 10 12

1 2 3

5

Sample output

24

說明:
此範例中在可Fractional下的取法為: 單位重量價值最高和第二高的兩物件全拿，第三高的物件只拿三分之二。所以總價值為 6+10+12*2/3 = 24。
注意: 別忘了四捨五入取到整數。

0/1 Knapsack Problem

(10 pts)

在不能部份選取的情況下(0/1)，輸入依次為物件的價值與物件的重量兩序列，再加上背包的最大承載上限值，設計一函式 KP_01()，其輸出為計算背包在承載限制下的最高取物總價。

Input format

輸入的三行序列，數個數字以空格隔開。

第一行輸入物件的價值。

第二行輸入物件的重量。

第三行輸入背包的最大承載上限值。

Output format

輸出如 Sample output 所示，輸出計算背包在承載限制下的最高取物總價。

Sample input

6 10 12

1 2 3

5

Sample output

22

說明:
此範例中在可0/1取法的限制下， 背包可承重的範圍內，只取排序第二和第三的物件能有最高的總價為 10+12＝22。

Solution Checking

(10 pts)

給定一方程式 $W0 + W1 * x1 + W2 * x2 + \ldots + Wn * xn = 0$

之參數向量 [W0, W1,..., Wn]與一組解[x1, x2,..., xn] 且其中 xi 非 0

即 1，也就是 $xi \in \{0, 1\}$

請設計一函式 solution_check()，其功能為判定所輸入的參數向量與一組 0/1 序列是否符合方程式(該方程式的一組解)。若符合方程式，則輸出 True，否則輸出 False。 (請注意:參數 Wi 的值可正可負)

Input format

第一行輸入參數向量，第二行輸入一組解，每行數字以空格隔開。

Output format

符合方程式，輸出 True，否則輸出 False。

Sample input

-5 1 2 3

0 1 1

Sample output

True

說明:
請注意參數項會比solution解中多了一項 bias (W_0)，代進公式的時候可能需要增加一個常數項的參數。

Solution exists

(10 pts)

給定一方程式 $W0 + W1 * x1 + W2 * x2 + \ldots + Wn * xn = 0$

之參數向量 [W0, W1,..., Wn]與一組解[x1, x2,..., xn] 且其中 xi 非 0 即 1，也就是 $xi \in \{0, 1\}$

請設計一函式 solution_exists()，其功能為判定所輸入的參數向量是否有解(有一組解[x1, x2,..., xn]符合方程式 且 xi 非 0 即 1)。若存在一組解符合方程式，則輸出 True，無解則輸出 False。(請注意:參數 Wi 的值可正可負)

Input format

第一行輸入參數向量，每行數字以空格隔開。

Output format

符合方程式，輸出 True，否則輸出 False。

Sample input

-5 1 2 3

Sample output

True

說明:
也可想成 W1*x1+W2*x2+…+Wn*xn = -W0
是否有一組 subset sum的問題。

Hamilton Cycle

(10 pts)

給定一有向圖之相鄰矩陣 A，若節點 i 至節點 j 可到達，則 A[i][j]為 1，若無路可達則 A[i][j]為無窮大(inf.)。設計一函式 hamilton_cycle() 可針對輸入所給的相鄰矩陣，判斷其所對應的有向圖是否存在經過所有節點且不重複拜訪下而回到原點的迴路。若存在則函式輸出 True，否則輸出 False。

Input format

第一行輸入 N，表示 NxN 大小之矩陣，後續輸入矩陣，每行數字以空格隔開，輸入矩陣，每行數字以空格隔開。

注意:以 x 表示為無窮大。

Output format

符合條件所述，輸出 True，否則輸出 False。

Sample input

4

x 1 1 1

x x 1 1

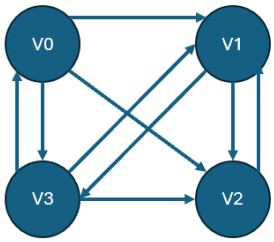
x 1 x x

1 1 1 x

Sample output

True

說明:
此案例中存在V0→V2→V1→V3→V0 之迴路，符合經過所有節點且不重複拜訪下而回到原點的要求，所以函式輸出為True。



TSP Computing

(10 pts)

給定一有向圖之相鄰矩陣 A，若節點 i 至節點 j 可到達，則 A[i][j]的值為節點 i 至節點 j 的旅行成本，若無路可達則 A[i][j]為無窮大(inf.)。設計一函式 TSP_compute() 可針對輸入所給的相鄰矩陣，計算其所對應的有向圖在經過所有節點且不重複拜訪下回到原點的迴路中可能的最小成本。若迴路存在，則輸出最小成本值，否則輸出-1。

Input format

第一行輸入 N，表示 NxN 大小之矩陣，後續輸入矩陣，每行數字以空格隔開，輸入矩陣，每行數字以空格隔開。

注意:以 x 表示為無窮大。

Output format

符合條件所述，輸出迴路中可能的最小成本，否則輸出-1。

Sample input

4
x 2 9 x
1 x 6 4
x 7 x 8
6 3 x x

Sample output

21

說明:
承上題所有可能的迴路中輸出最小成本的值，若不存在符合條件的迴路，則輸出-1。

Maze Solver

(10 pts)

給定一迷宮，_ 代表有路可走而 # 代表阻擋道路的牆壁。

設計一函式 maze_solver()，輸入依序為迷宮矩陣、(起點列 row,起點行 col)、(終點列 row,終點行 col)。若起點至終點存在可到達的路徑則函式輸出 True，否則輸出 False。(起點最左上角，終點最右下角)

注意: 起點與終點的位置不一定有路，也可能會是牆壁，則回傳 False。

座標 index 由 0 開始。

Input format

第一行輸入 row 與 col 數量，後續輸入矩陣，每行數字以空格隔開，輸入矩陣，每行數字以空格隔開。

Output format

起點至終點存在可到達的路徑，則輸出 True；否則輸出 False。

Sample input

```
5 5
_ _ # _ #
_ # _ _ _
_ # _ # _
_ _ _ _ #
# # # _ _
0 0
2 4
```

說明:
給定迷宮矩陣、(row,col)格式的起始座標和終點座標。
請注意迷宮的起始與終點座標可能為牆壁，則函式輸出False表示無解。

Sample output

```
True
```

Optimal Maze Solver

(10 pts)

承上題，設計一函式 maze_solver_optimal()，輸入依序為迷宮矩陣、(起點列 row,起點行 col)、(終點列 row,終點行 col)。若起點至終點存在可到達的路徑則函式輸出其最短路徑的步數(包含起點與終點的節點個數)，否則輸出 0。注意:若起點和終點同一位置，則根據定義步數為 1，若起點走不到終點，步數為 0。座標 index 由 0 開始。

Input format

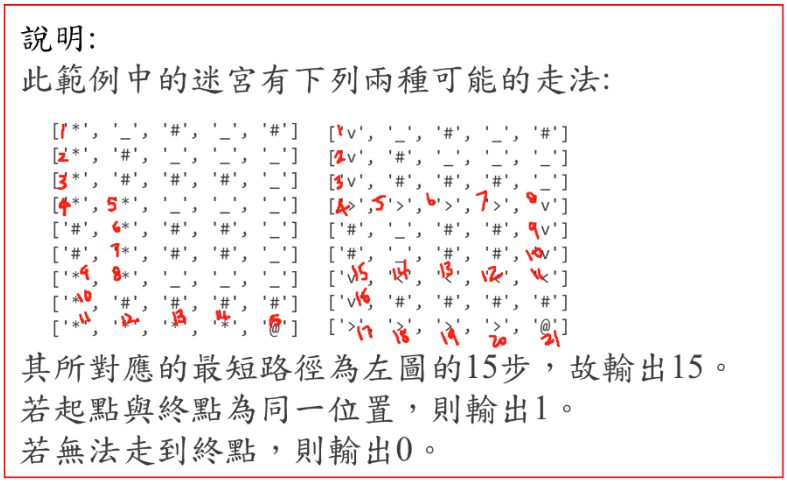
第一行輸入 row 與 col 數量，後續輸入矩陣，每行數字以空格隔開，輸入矩陣完，輸入起始節點索引、終點節點索引，每行數字以空格隔開。

Output format

起點至終點存在可到達的路徑，則輸出最短距離；否則輸出 0。

Sample input

```
9 5
_ _ # _ #
_ # _ _ _
_ # # # _
_ _ _ _ _
# _ # # _
# _ # # _
_ _ _ _ _
_ # # # #
_ _ _ _ _
0 0
8 4
```



Sample output

```
15
```

Shortest Path
(10 pts)

給定一有向圖的成本矩陣(cost matrix, CM)，成本矩陣中的 CM[i,j]為 Vi 到 Vj 的距離(沒有負值)，若 Vi 至 Vj (Vi≠Vj)無法直接到達，則成本為無窮大(或設置一個大數值常數代表無窮大)。

設計一函式 shortest_path()，其輸入依序為出發節點編號與終點節點編號，若可到達終點，則函式的輸出為起始點至終點的最短距離；若無法到達終點，函式輸出-1。

Input format

第一行輸入 N，表示 NxN 大小之矩陣，後續輸入矩陣，每行數字以空格隔開，輸入矩陣完，輸入起始節點與終點節點，每行數字以空格隔開。注意:以 x 表示為無窮大。

Output format

可到達終點，則輸出最短距離；否則輸出-1。

Sample input

```
8
0 x x x x x x x
300 0 x x x x x x
1000 800 0 x x x x x
x x 1200 0 x x x x
x x x 1500 0 250 x x
x x x 1000 x 0 900 1400
x x x x x x 0 1000
1700 x x x x x x 0
4 0
```

Sample output

```
3350
```

說明:
請考慮到起點不一定到每一點皆可到達的情況。例如下圖案例中V4出發無法走到V1(V4到V1的成本無窮大)，函式須輸出-1。