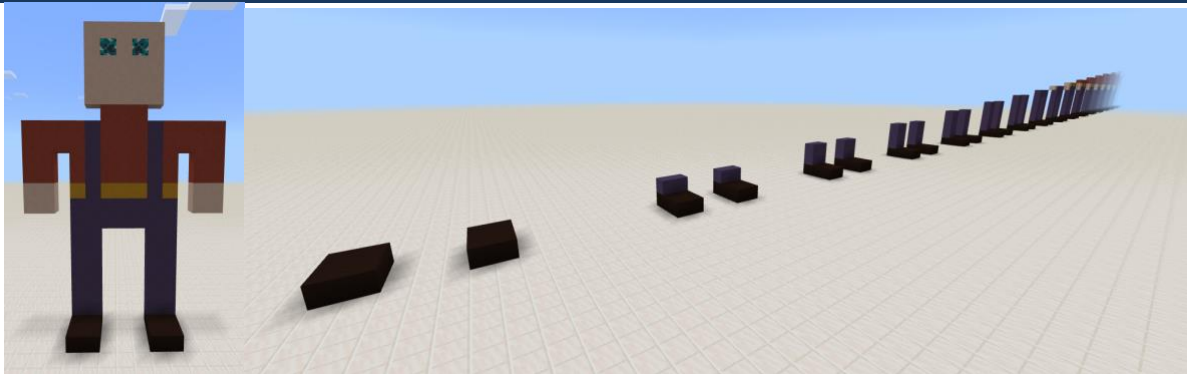
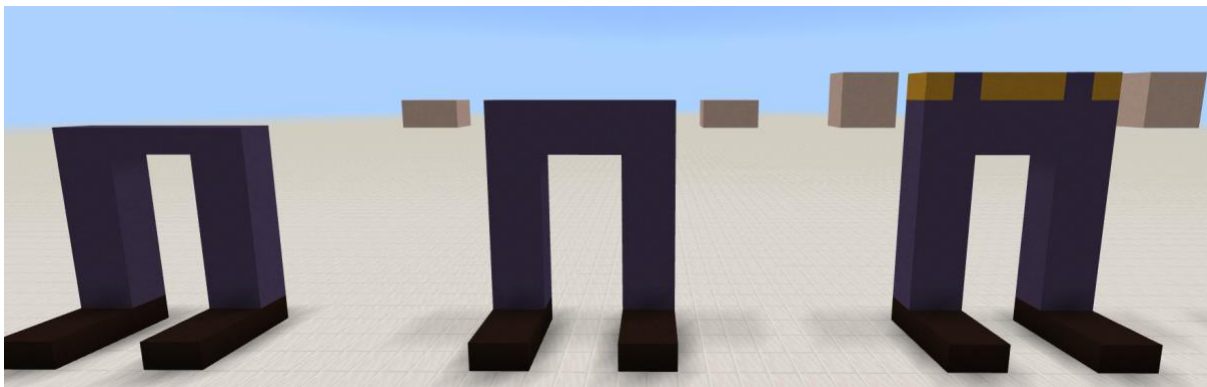


Problem3 Island Finder



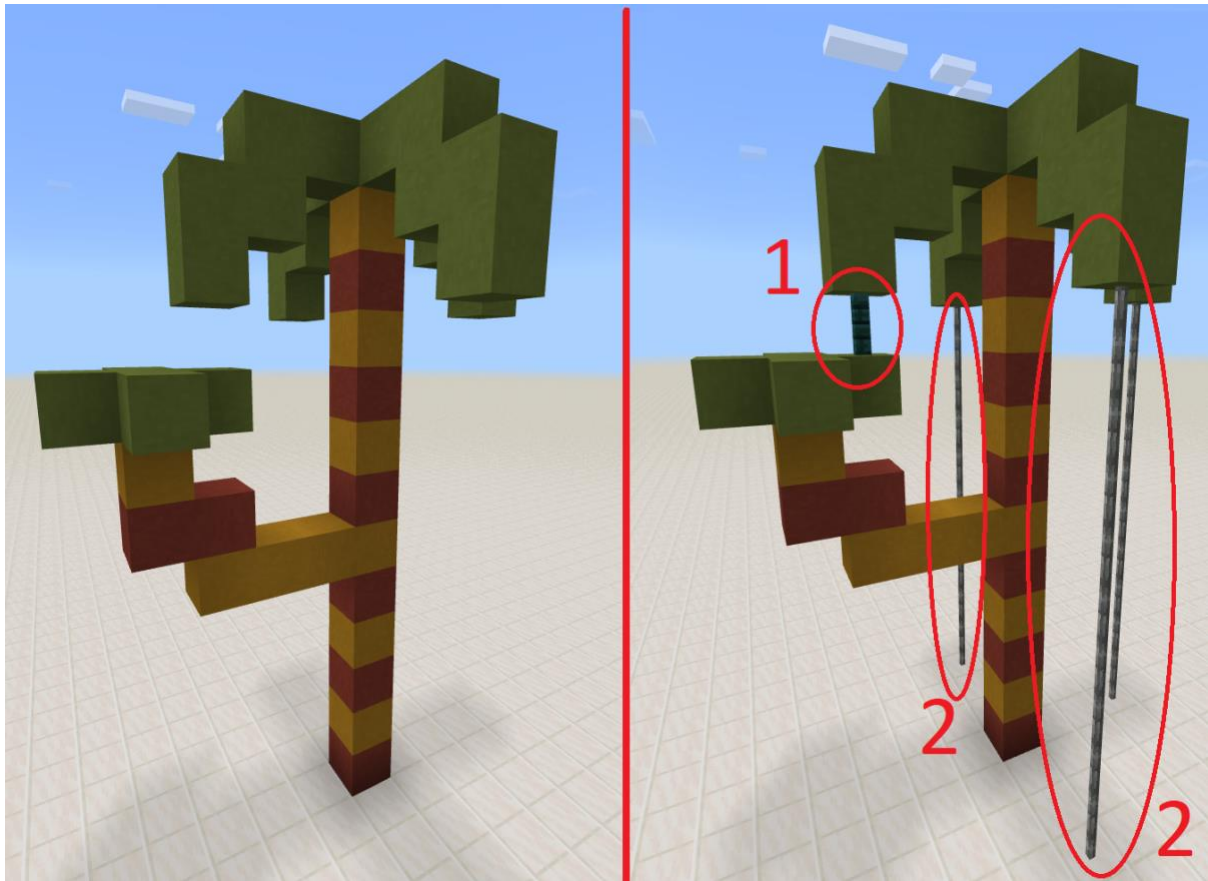
รูปภาพที่ 1 (ซ้าย) แสดงชิ้นงานที่สมบูรณ์แล้ว, (ขวา) ขั้นตอนการพิมพ์ทีละชั้นจากชั้นล่างสุด

กระบวนการพิมพ์ชิ้นงานสามมิติ จะสามารถทำได้โดยการหั่นชิ้นงานเป็นชั้น ๆ บาง ๆ ตามแนวตั้งแล้วเครื่องพิมพ์ก็จะค่อยๆพิมพ์ครั้งละ 1 ชั้นจากชั้นล่างสุด วางซ้อนทับเป็นชั้นใหม่ ซ้อนไปถึงชั้นบนสุด ได้จะชิ้นงานสามมิติออกมา



รูปภาพที่ 2 ปัญหา island ที่เกิดขึ้นในส่วนมือ

แต่ในกระบวนการซ้อนชั้นต่าง ๆ นั้น ก็มีข้อจำกัดคือ หากชั้นปัจจุบันมีชั้นส่วนที่ชั้นล่างไม่มีจะไม่สามารถพิมพ์ได้ ตัวอย่างเช่น การพิมพ์ชิ้นงานดังแสดงในรูปที่ 1 ซ้าย จะพิมพ์ทีละชั้นดังแสดงในรูปที่ 1 ขวา และจะมีปัญหาเกิดขึ้นในส่วนของมือแสดงในรูปภาพที่ 2 กล่าวคือ **มือนั้นถูกพิมพ์โดยไม่มีโครงสร้างมารองรับ** ซึ่งหากเป็นโลกจริงพลาสติกส่วนมือจะถูกฉีกลงไปติดกับพื้นระดับเดียวกับเท้า โดยมือที่ลอยอยู่นั้นมันถูกเรียกว่า **Island**



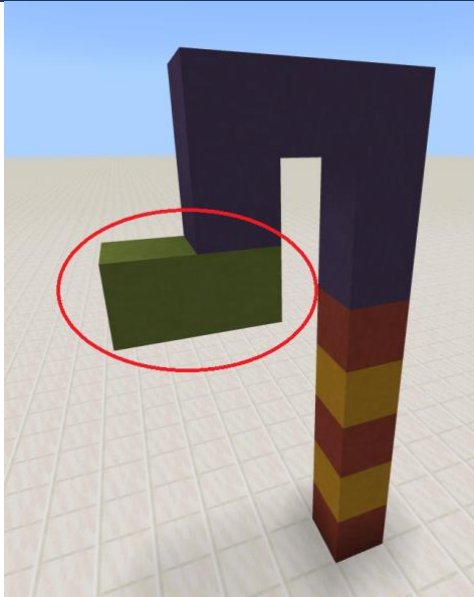
รูปที่ 3 (ซ้าย) ก่อนมี support, (ขวา) หลังจากมี support แล้ว 1 คือ Inner support (I) และ 2 คือ Main support (M)

ในการพิมพ์นั้นจึงต้องมีโครงสร้างที่สร้างขึ้นเพื่อ support รองรับการพิมพ์ไม่ให้พิมพ์ใส่อากาศ แต่พิมพ์ใส่โครงสร้างของเราแทน โดย Support จะมี 2 ประเภทคือ Main support และ Inner support ดังแสดงในรูปที่ 3 โดย Main support จะเป็น support ที่ฐานรับน้ำหนักมาจากที่พื้นเดียวกับชั้นแรกสุด แต่ Inner support จะเป็น support ที่มีฐานรับน้ำหนักเป็นเนื้อชิ้นงานชั้นต่ำกว่าที่อยู่ใกล้สุด

สิ่งที่ต้องทำ

1. ระบุว่า Island เกิดขึ้นที่ใดบ้าง
2. ระบุว่า Island ที่เกิดขึ้นจะสามารถถูกซ่อมแซมได้ด้วย Support แบบใด

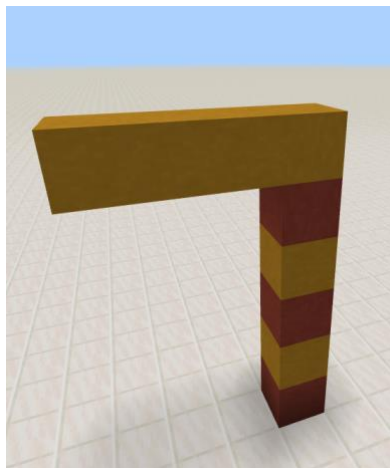
คำถามที่พบบ่อย



รูปที่ 4 ส่วนที่เป็น Island คือสี่เหลี่ยมในวงกลมทั้งสองอัน

Q: ในรูปที่ 4 หากมีหลายบล็อกที่ติดกัน ลอยอยู่ จะต้องตอบทุกบล็อกที่ติดกันหรือไม่

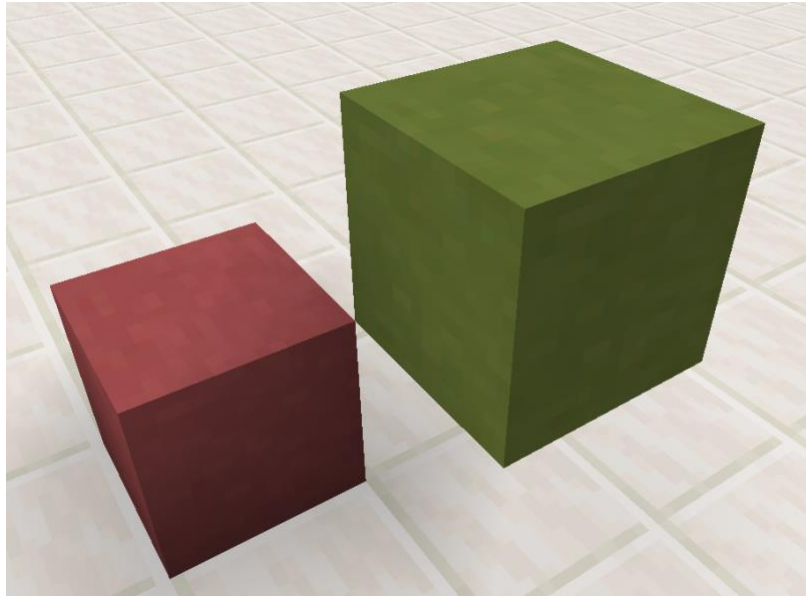
A: ใช่ครับ จะต้องตอบทุกบล็อกที่ติดกัน ตัวอย่างในรูป บล็อกสี่เหลี่ยมทั้งสองอันจะลอยอยู่ ดังนั้นจะต้องตอบทั้ง 2 บล็อกครับ



รูปที่ 5

Q: ในรูปที่ 5 แบบนี้นับว่าเป็นเกาะหรือไม่ครับ

A: ตัวอย่างนี้ไม่เป็นเกาะครับ เพราะว่าโครงสร้างด้านข้างของมันได้รับน้ำหนักไว้แล้ว



รูปที่ 6 สีเขียว(ขึ้นบนขวา)เป็นเกาะ

Q: ทิศทางใดบ้างที่ถือว่าโครงสร้างได้รับน้ำหนักแล้ว

A: เฉพาะ บน ล่าง ซ้าย และ ขวา เท่านั้น ในส่วนของทิศตามเส้นทะแยงมุมตัวอย่างในรูปที่ 6 ไม่จัดว่ารับน้ำหนักแล้วครับ

ข้อมูลนำเข้า

บรรทัดแรกตัวอักษรบอกประเภทของคำถาม Y หรือ N ถ้ารับอินพุตเป็น N ให้ตอบเพียงตำแหน่งของ island, ถ้ารับอินพุตเป็น Y ให้ตอบทั้ง ตำแหน่ง และ ประเภทของ support เป็นตัวอักษร I หรือ M แปลว่า Inner support และ Main support ตามลำดับ

บรรทัดที่สองบอกความกว้าง ความยาว และความสูง ถูกค้นโดยเครื่องหมายจุลภาค W,L,H

H*L บรรทัดถัดมาเป็นตัวอักษร x หรือ - ติดกันจำนวน W ตัว ทุก ๆ L บรรทัดติดกันจะรวมกันเป็น 1 ชั้นโดยมีทั้งหมด H ชั้น โดย x แปลว่ามีเนื้อชิ้นงาน - แปลว่าไม่มีชิ้นงาน

ข้อมูลส่งออก

จำนวนบรรทัดเท่ากับจำนวน island ที่มี โดยการตอบต้องเรียงลำดับจาก island ใน layer ล่างสุดก่อน และตามด้วย **ประเภทของ support (I ก่อน M) ตามด้วยแถว และสุดท้ายคือหลัก**

หากเป็นโหมด n แต่ละบรรทัดจะต้องตอบเป็นตัวเลข 3 ชุดค้นด้วยจุลภาค ชั้นที่,แถวที่,คอลัมน์ที่ (L,X,Y) โดยชั้นจะเริ่มที่ 0

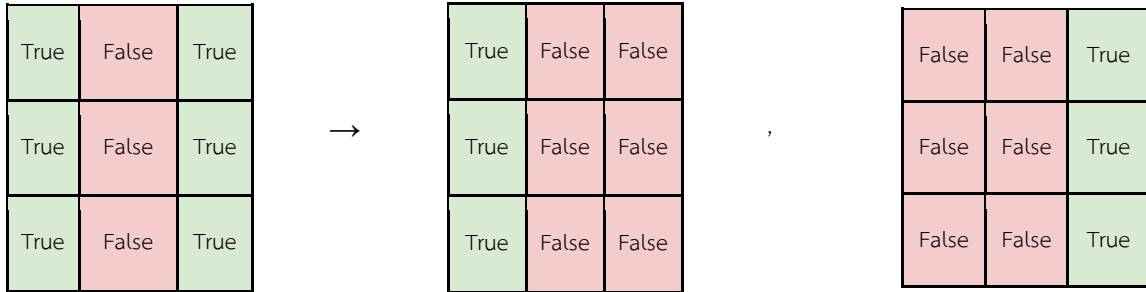
หากเป็นโหมด y แต่ละบรรทัดจะต้องตอบเป็นตัวเลข 4 ชุดค้นด้วยจุลภาค

ชั้นที่,ประเภทของโครงสร้างรับน้ำหนัก,แถวที่,คอลัมน์ที่ (L,T,X,Y) โดยชั้นจะเริ่มที่ 0 และ ประเภทของโครงสร้างมีเพียง I และ M เท่านั้น

หากไม่มี island ให้ตอบว่า “There is no island” โดยไม่มีเครื่องหมายอัฒภาคคู่

โค้ดตั้งต้น

สำหรับข้อนี้จะมีการให้ฟังก์ชัน `to_cluster` ซึ่งรับข้อมูลเป็น ข้อมูลของชั้นเพียงหนึ่งชั้นในรูปแบบของ numpy array 2 มิติ ซึ่งมีค่าเป็น `True` สำหรับเนื้อชิ้นงาน และ `False` หากไม่มีอะไร โดยฟังก์ชันนี้จะทำการแยกเนื้อของชิ้นงานที่นับว่าเป็นชิ้นเดียวกันสำหรับชั้นนี้ให้ โดยคืนค่ากลับมาเป็นลิสของ numpy array ที่มี dimension เหมือนข้อมูลนำเข้า



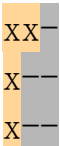

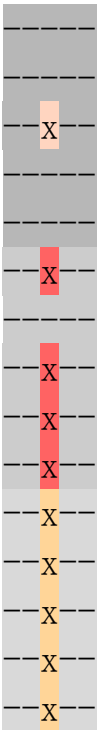
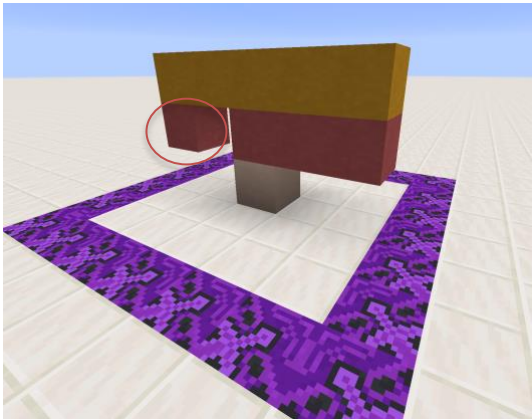
รูปที่ 7 รูปด้านบน มีกล่องอยู่สองเซต เมื่อเอาไปใส่ในฟังก์ชัน `to_cluster` ก็จะถูกแยกออกเป็นลิสต์ที่มีสมาชิกเป็นสอง

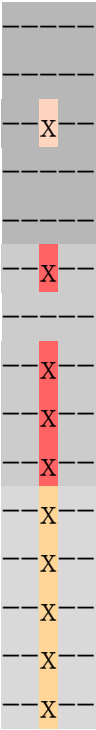
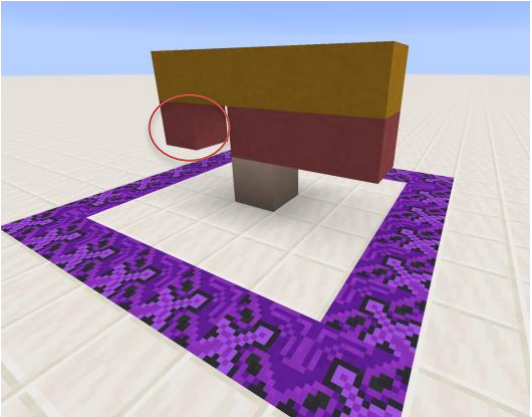

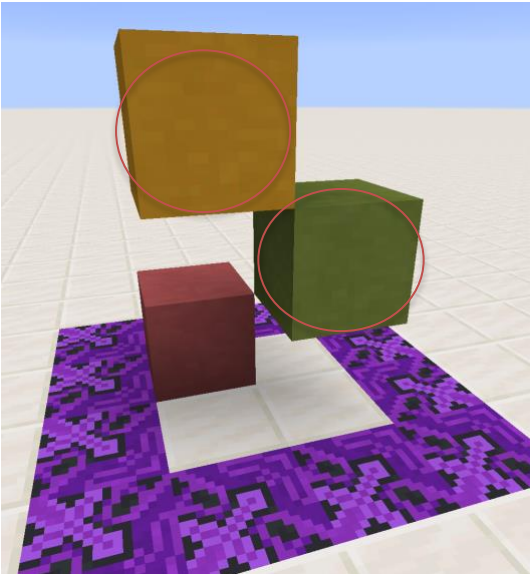
สามารถดาวน์โหลดโค้ดตั้งต้นได้ที่ [link](#)

ตัวอย่างการเรียกใช้

```
a = np.array([
    [1, 0, 1],
    [1, 0, 1],
    [1, 0, 1],
    ], np.bool)
to_cluster(a)
```

ตัวอย่าง

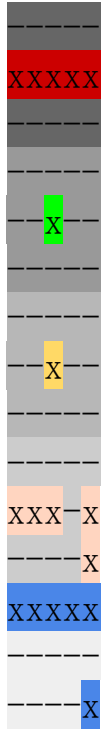
ชุดที่	Input (จากแป้นพิมพ์)	Output (ทางจอภาพ)
1	<div>N</div> <div>3,3,1</div> <div></div> <div></div>	<div>There is no island</div>
2	<div>N</div> <div>5,5,3</div> <div></div>	<div>1,0,2</div> <div></div>

<p>3</p>	<p>Y</p> <p>5,5,3</p> 	<p>1,M,0,2</p> 
<p>4</p>	<p>Y</p> <p>2,2,3</p> 	<p>1,M,1,1</p> <p>2,M,1,0</p> 

5

Y

5,3,5



3,l,1,4

3,M,2,4

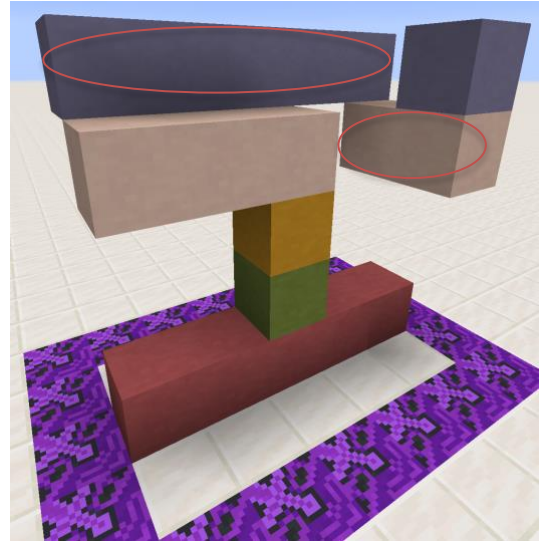
4,M,0,0

4,M,0,1

4,M,0,2

4,M,0,3

4,M,0,4



Code เริ่มต้น

```
import numpy as np

def to_cluster(a):
    """
    Seperate cluster from 2d array.

    # Parameters
    a: array_like - 2 dimension array of `True` or `False` with dtype np.bool.

    # Returns
    cluster_list: List - a list of 2d array each with only one cluster.

    # Notes
    This function doesn't guarantee order of output.

    # Examples
    >>> a = np.array([
        [1, 0, 1],
        [1, 0, 1],
        [1, 0, 1],
    ], np.bool)
    >>> np.array(to_cluster(a))
    array([[ True, False, False],
          [ True, False, False],
          [ True, False, False]],
          [[False, False,  True],
          [False, False,  True],
          [False, False,  True]])
    """
    height, width = a.shape
    output = []
    last_cross_section = []
    start_row = np.argmax(np.pad(np.max(a, axis=1), ((0, 1),), mode='constant', constant_values=np.inf))
    for row_index in range(start_row, height):
        cross_section = []
        seg_start = np.argmax(np.pad(a[row_index], ((0, 1),), mode='constant', constant_values=np.inf))
        for col_index in range(seg_start, width + 1):
            if (col_index == width and a[row_index][col_index-1]) or (col_index < width and a[row_index][col_index-1] and
not a[row_index][col_index]):
                seg = np.full_like(a, False)
                seg[row_index, seg_start:col_index] = True
                cross_section.append(seg)
            if col_index < width and not a[row_index][col_index-1] and a[row_index][col_index]:
                seg_start = col_index
        marked_remove = set()
        for lcs in last_cross_section:
            merged_to = None
            for ic, ccs in enumerate(cross_section):
                if np.any(np.logical_and(lcs[row_index - 1], ccs[row_index])):
                    if merged_to is None:
                        ccs[:, :] = np.logical_or(lcs, ccs) # inplace merge
                        merged_to = ccs
                    else: # merge into the same lcs, so need to remove from last_cross_section
                        merged_to[:, :] = np.logical_or(merged_to, ccs) # inplace merge
                        marked_remove.add(ic)
            if merged_to is None:
                output.append(lcs)
        last_cross_section = [cross_section[ic] for ic in range(len(cross_section)) if ic not in marked_remove]
        output.extend(last_cross_section)
    return output
```

Download here: http://pioneer.netserv.chula.ac.th/~psomchai/101/2563_1/grader_starter_kit.py

Good luck

