

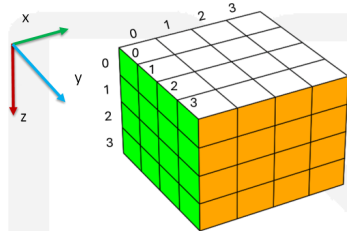
ข้อกำหนด

การเรียกใช้ฟังก์ชันเพื่อการทดสอบ ต้องอยู่ภายใต้เงื่อนไข `if __name__ == '__main__':` เพื่อให้สามารถ import ไปเรียกใช้งานจาก Script อื่น ๆ ได้อย่างเป็นมาตรฐาน

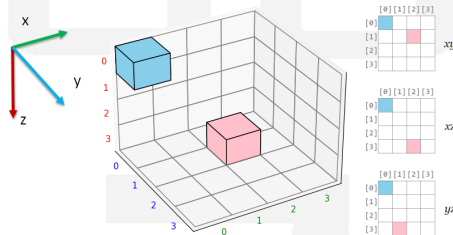
File Header (กรณีไม่เขียน Header จะเสียคะแนน 5%)

```
#!/usr/bin/env python3
# ชื่อ (ไม่ต้องใส่นามสกุล)
# รหัสศ
# Sec00x
```

- 1) **100 คะแนน (Q4P1_6XXXXXX.py)** คุณได้รับลูกบาศก์โปร่งใสขนาด $n \times n \times n$ ($0 < n \leq 300$) แต่ละเซลล์ในลูกบาศก์นี้มีการกำหนดพิกัดในระบบแกน x, y และ z ด้วยค่า (x_i, y_i, z_i) ตามลำดับ เมื่อ $0 \leq x_i, y_i, z_i < n$ รูปด้านล่างแสดงลูกบาศก์ที่ $n = 4$ และแนวแกน x, y และ z ที่อ้างอิง

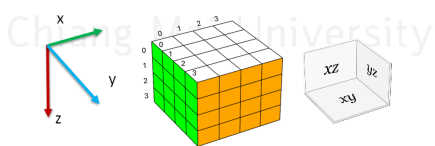


ลูกบาศก์ที่ได้รับจะมีบางเซลล์ที่ทึบแสงระบุด้วย List ของ Tuple ของพิกัดเซลล์ในรูปแบบ (x, y, z) เมื่อ $0 \leq x, y, z_i < n$ เช่นในรูปด้านล่าง เซลล์ที่ทึบแสงคือ $(0,0,0)$ ที่แสดงด้วยสีฟ้า และ $(2,1,3)$ ที่แสดงด้วยสีชมพู ซึ่งจะแทนด้วย List $[(0, 0, 0), (2, 1, 3)]$



หน้าที่ของคุณคือ ให้เขียนฟังก์ชัน

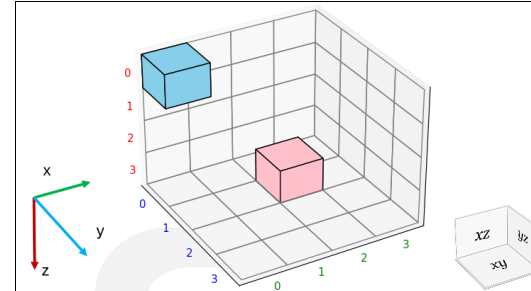
`projections(n: int, opaque_cells: list[tuple[int]]) -> dict[str, list[list[int]]]`
เพื่อคืนค่า Dictionary แทนผลลัพธ์การคำนวณหาภาพฉายสองมิติ (Projection) ของลูกบาศก์โปร่งใสขนาด $n \times n \times n$ ตามที่อธิบายด้านบน ลงบนระนาบทั้งสามเมื่อ `opaque_cells` เป็น List ของ Tuple แทนพิกัดของเซลล์ที่ทึบแสงที่สามารถเป็น List ว่างได้ ทั้งนี้จะไม่มี Coordinate ที่ซ้ำใน `opaque_cells`



ระนาบทั้ง 3 ได้แก่ xy เมื่อมองจากด้านสีขาวในรูป, xz เมื่อมองจากด้านสีส้มในรูป และ yz เมื่อมองจากด้านสีเขียวในรูป โดย Dictionary ที่คืนค่าจะมี key 3 ค่าเท่านั้น ได้แก่ String 'xy', 'xz' และ 'yz' โดยที่แต่ละ key จะมี value เป็น List 2 มิติขนาด $n \times n$ ซึ่งเป็นภาพฉายบนระนาบที่ระบุ กำหนดให้แต่ละเซลล์ใน List 2 มิตินี้ค่าเป็น 1 หากทึบแสง และ 0 หากโปร่งแสง โดยกำหนดให้ column ที่ 0 และ row ที่ 0 แทนพิกัด $(0,0)$ ในระนาบนั้นๆ

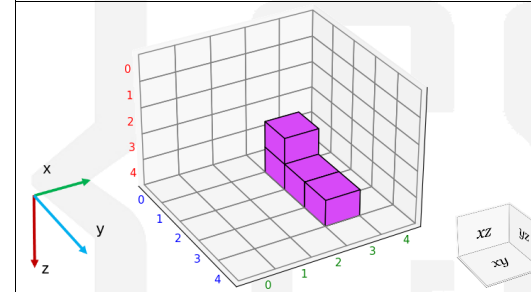
Function Call

Returned Value



`projections(4, [(0, 0, 0), (2, 1, 3)])`

```
{
  'xy': [[1, 0, 0, 0],
         [0, 0, 1, 0],
         [0, 0, 0, 0],
         [0, 0, 0, 0]],
  'xz': [[1, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 0, 1, 0]],
  'yz': [[1, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 0, 0, 0],
         [0, 1, 0, 0]]
}
```



`projections(5, [(3, 1, 3), (3, 1, 4), (3, 2, 4), (3, 3, 4)])`

```
{
  'xy': [[0, 0, 0, 0, 0],
         [0, 0, 0, 1, 0],
         [0, 0, 0, 1, 0],
         [0, 0, 0, 1, 0],
         [0, 0, 0, 0, 0]],
  'xz': [[0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0],
         [0, 0, 0, 1, 0],
         [0, 0, 0, 1, 0]],
  'yz': [[0, 0, 0, 0, 0],
         [0, 0, 0, 0, 0],
         [0, 1, 0, 0, 0],
         [0, 1, 1, 1, 0],
         [0, 1, 1, 1, 0]]
}
```

คำอธิบาย Test Case

มีทั้งหมด 10 Test Case คิดคะแนนเป็น 10 คะแนนต่อ Test Case

- Test Case ที่ 1: เหมือนกรณีตัวอย่าง
 - ต้องมีการคำนวณผลลัพธ์ตามข้อกำหนดของโจทย์ ไม่อนุญาต ให้ Hard Code หรือ ลอก Output จากตัวอย่างมาคืนค่าหรือแสดงค่า (ถือเป็นการทุจริต)
- Test Case ที่ 2 – 5: n จะมีค่าเป็น 4 เท่านั้น
 - Test Case ที่ 2: มีเซลล์ที่ทึบแสงเพียงเซลล์เดียว
 - Test Case ที่ 3: ตรวจสอบความถูกต้องแค่ระนาบ 'xy'
 - Test Case ที่ 4: ตรวจสอบความถูกต้องแค่ระนาบ 'xz'
 - Test Case ที่ 5: ตรวจสอบความถูกต้องแค่ระนาบ 'yz'
- Test Case ที่ 6 – 10: ทดสอบตามทุกข้อกำหนดของโจทย์

Python Tutor Visualizer: <http://10.4.28.251/tutor/visualize.html>

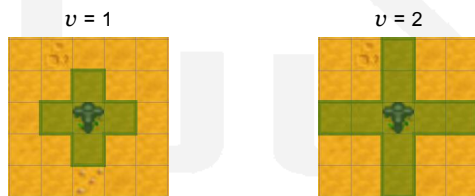
Grader: <http://10.4.28.251>

File Header

```
#!/usr/bin/env python3
# ชื่อ (ไม่ต้องใส่นามสกุล)
# รหัสสนศ
# Sec00x
```

- 2) 100 คะแนน (Q4P2_6XXXXXX.py) ในเกม Stardew Valley ซึ่งเป็นเกม RPG (Role Player Game) ที่ผู้เล่นสามารถเรียนรู้ที่จะใช้ชีวิตชนบทในการเปลี่ยนผืนดินและทุ่งนาที่รกร้าง ให้เป็นพื้นที่สำหรับการทำเกษตรกรรมและเลี้ยงปศุสัตว์ เพื่อให้พื้นที่เหล่านั้นมีความอุดมสมบูรณ์และมีผลผลิตที่ก่อให้เกิดรายได้ขึ้นมา การวางแผนเกษตรกรรมผู้เล่นสามารถใช้ Sprinkler (เครื่องพ่นน้ำ) เป็น item ในการรดน้ำให้ผืนดินมีความชุ่มชื้นเพื่อการเพาะปลูก โดยพื้นที่การทำเกษตรจะแทนด้วย board ขนาด $m \times n$ ช่อง

โดยการวาง Sprinkler ณ ช่องใดๆ จะส่งผลให้ช่องที่อยู่ติด Sprinkler ทั้ง 4 ด้าน (บน ล่าง ซ้าย ขวา) ได้รับปริมาณน้ำไหลออกไปด้านละ v ช่อง ทั้งนี้ในแต่ละรอบของการติดตั้ง Sprinkler ปริมาณน้ำที่ปล่อยจาก Sprinkler ทุกตัวจะให้ปริมาณน้ำเท่ากันเสมอ จากรูปที่กำหนดให้ด้านล่างนี้มีค่า $v = 1$ และ $v = 2$ ตามลำดับ



คุณได้รับคำร้องขอจากผู้ทดสอบเกมให้ทำการพัฒนาโปรแกรมที่สามารถกำหนดรูปแบบของพื้นที่เกษตรกรรมด้วยตนเองโดยให้เป็น board ที่สร้างจาก List ของ Tuple ที่ระบุขนาดของ board, ปริมาณของน้ำที่ปล่อยออกจาก Sprinkler และพิกัดของ Sprinkler จากนั้นทำการคำนวณปริมาณน้ำที่เกิดขึ้นในแต่ละช่องจาก Sprinkler ทุกตัวที่ติดตั้ง และคืนค่าเป็น Python Dictionary ที่มี key เป็นพิกัดช่องที่ไม่มี Sprinkler และ value เป็นจำนวนของ Sprinkler ที่ส่งน้ำมาถึงพิกัดช่องนั้นๆ เช่น ในตาราง 3x3 Sprinkler ให้ปริมาณน้ำไหลออกไป 1 ช่องและพิกัด Sprinkler เป็น (1, 1) ถ้า input คือ [(1, 1)] เมื่อ (i, j) หมายถึง row ที่ i และ column ที่ j Board ที่ได้จะมีรูปร่างดังนี้

	[0]	[1]	[2]
[0]			
[1]			
[2]			

โดยพิกัดของช่องที่วาง Sprinkler จะถือว่าไม่เปียกน้ำ

ดังนั้นพิกัดของช่องที่เปียกและตัวเลขแสดงจำนวนของ Sprinkler ที่ส่งน้ำมาถึง จะแสดงดัง

ภาพ

	[0]	[1]	[2]
[0]		1	
[1]	1		1
[2]		1	

หรือในรูป Dictionary $\{(0, 1): 1, (1, 0): 1, (1, 2): 1, (2, 1): 1\}$ โดยจะต้องไม่รายงานช่องที่ sprinkler ตั้งอยู่

รูปภาพตัวอย่าง กรณีสวน Sprinkler มากกว่า 1 ตัว

Function Call

Returned Value

wet_area(4, 5, 1, [(1, 0), (2, 0), (3, 2)])

$\{(0, 0): 1, (1, 1): 1, (2, 1): 1, (2, 2): 1, (3, 0): 1, (3, 1): 1, (3, 3): 1\}$

	[0]	[1]	[2]	[3]	[4]
[0]					
[1]					
[2]					
[3]					

	[0]	[1]	[2]	[3]	[4]
[0]	1				
[1]		1			
[2]		1	1		
[3]	1	1		1	

คำอธิบาย





- จำนวน Sprinkler มี 3 ตัว
พิกัดคือ (1, 0), (2, 0) และ (3, 2)

ให้ปริมาณน้ำไหลออกไป 1 ช่องทั้ง 4 ด้านของ Sprinkler แต่ละตัว

- ช่องที่ได้รับน้ำจาก Sprinkler (1, 0) คือ (0, 0) และ (1, 1)
- ไม่พิจารณา (2, 0) เนื่องจากมี Sprinkler วางอยู่
- ช่องที่ได้รับน้ำจาก Sprinkler (2, 0) คือ (2, 1) และ (3, 0)
- ไม่พิจารณา (1, 0) เนื่องจากมี Sprinkler วางอยู่
- ช่องที่ได้รับน้ำจาก Sprinkler (3, 2) คือ (2, 2), (3, 1) และ (3, 3)

Function Call





wet_area(4, 5, 1, [(0, 0), (1, 2), (1, 3), (3, 2)])

	[0]	[1]	[2]	[3]	[4]
[0]					
[1]					
[2]					
[3]					

- จำนวน Sprinkler มี 4 ตัว
พิกัดคือ (0, 0), (1, 2), (1, 3) และ (3, 2)
- ให้ปริมาณน้ำไหลออกไป 1 ช่องทั้ง 4 ด้าน
ของ Sprinkler แต่ละตัว

Returned Value

{(0, 1): 1, (0, 2): 1, (0, 3): 1,
(1, 0): 1, (1, 1): 1, (1, 4): 1, **(2, 2): 2**, (2, 3): 1, (3, 1): 1, (3, 3): 1}

	[0]	[1]	[2]	[3]	[4]
[0]		1	1	1	
[1]	1	1			1
[2]			2	1	
[3]		1		1	

คำอธิบาย

- ช่องที่ได้รับน้ำจาก Sprinkler(0, 0) คือ
(0, 1) และ (1, 0)
- ช่องที่ได้รับน้ำจาก Sprinkler(1, 2) คือ
(0, 2), (1, 1), และ **(2, 2)**
ไม่พิจารณา (1, 3) เนื่องจากมี Sprinkler วาง
อยู่
- ช่องที่ได้รับน้ำจาก Sprinkler(1, 3) คือ
(0, 3), (1, 4) และ (2, 3)
ไม่พิจารณา (1, 2) เนื่องจากมี Sprinkler วาง
อยู่
- ช่องที่ได้รับน้ำจาก Sprinkler(3, 2) คือ
(2, 2), (3, 1) และ (3, 3)
พิกัดช่องที่ได้รับน้ำจาก Sprinkler 2 ตัว
พร้อมกันคือ **(2, 2)** ส่วนพิกัดที่เหลือได้รับ
น้ำจาก Sprinkler เพียงตัวเดียว

หน้าที่ของคุณคือให้เขียนฟังก์ชัน `wet_area(m: int, n: int, v: int, sprinkler_list: list[tuple[int, int]]) -> dict[tuple[int, int], int]` เพื่อคืนค่า Dictionary แสดงพิกัดของช่องที่เปียกน้ำและจำนวน Sprinkler ที่ส่งเข้ามาในรูปแบบดังอธิบายด้านบน โดยให้ m ($3 \leq m \leq 500$) และ n ($3 \leq n \leq 500$) คือจำนวน row และจำนวน column ของ Board และ `sprinkler_list` คือ List ของ Tuple แสดงตำแหน่งของ Sprinkler โดยให้คืนค่าพิกัดของช่องที่เปียกน้ำและจำนวน Sprinkler ที่ส่งเข้ามาช่องนั้นเท่านั้น โดยจะต้องไม่รายงานช่องที่มี Sprinkler ตั้งอยู่

Test Cases มีทั้งหมด 10 Test Case คิดคะแนนเป็น 10 คะแนนต่อ Test Case

- Test Case ที่ 1: เหมือนกรณีตัวอย่าง
ต้องมีการคำนวณผลลัพธ์ตามข้อกำหนดของโจทย์ ไม่อนุญาต ให้ Hard Copy หรือ ลอก Output จากตัวอย่างมาคืนค่าหรือแสดงค่า (ถือเป็นการทุจริต)
- Test Case ที่ 2: ถึง 3 ค่าของปริมาณน้ำเป็น 1 หน่วยและจำนวน Sprinkler ไม่เกิน 10 ตัว
- Test Case ที่ 4 ถึง 5 ค่าของปริมาณน้ำเป็น 1 หน่วยและจำนวน Sprinkler 10 ตัวขึ้นไป
- Test Case ที่ 6 ถึง 7 ค่าของปริมาณน้ำเป็น 2 หน่วยและจำนวน Sprinkler 10 ตัวขึ้นไป
- Test Case ที่ 8 และ 9 ค่าของปริมาณน้ำเป็น 3 หน่วยและจำนวน Sprinkler 10 ตัวขึ้นไป
- Test Case ที่ 10 ค่าของปริมาณน้ำมีค่าเป็น 1-3 หน่วยและจำนวน Sprinkler 10 ตัวขึ้นไป
- Test Case ที่ 3-10 สามารถมีช่องที่ได้รับน้ำจากจำนวน Sprinkler มากกว่า 1 ตัวได้ (Overlapped)

Function Call

wet_area(3, 3, 1, [(1, 1)])

Returned Value

```
{
  (0, 1): 1,
  (1, 0): 1,
  (1, 2): 1,
  (2, 1): 1
}
```

wet_area(4, 5, 1, [(1, 0), (2, 0), (3, 2)])

```
{
  (0, 0): 1,
  (1, 1): 1,
  (2, 1): 1,
  (2, 2): 1,
  (3, 0): 1,
  (3, 1): 1,
  (3, 3): 1
}
```

wet_area(4, 5, 1, [(0, 0), (1, 2), (1, 3), (3, 2)])

```
{
  (0, 1): 1,
  (0, 2): 1,
  (0, 3): 1,
  (1, 0): 1,
  (1, 1): 1,
  (1, 4): 1,
  (2, 2): 2,
  (2, 3): 1,
  (3, 1): 1,
  (3, 3): 1
}
```

Python Tutor Visualizer: <http://10.4.28.251/visualize.html>

Grader: <http://10.4.28.251>