Homework7

Interval and Set

งานที่ 1 สร้างช่วง

def create_interval(x, y):

รับ x, y เก็บค่าจำนวนเต็ม

โดยช่วงที่คืนจะต้องไม่มีช่องว่างกั้นระหว่างเครื่องหมาย "," กับตัวเลข เช่น [1, 2] หรือ (0, 0)

ตัวอย่าง

create_interval(-2, 10)) # คืน [-2,10]

create_interval(7, 5)) # คืน (0,0)

แนวคิด

1. หาก x มีค่าน้อยกว่าหรือเท่ากับ y คืน สตริงแทนช่วงปิดตั้งแต่ x ถึง y “[x,y]”
2. หาก x มีค่ามากกว่า y คืน '(0,0)'

ตัวอย่างโค้ดเทียม

```
def create_interval(x, y):
    if x <= y: return “[x,y]”
    if x > y:   return “(0,0)”
```

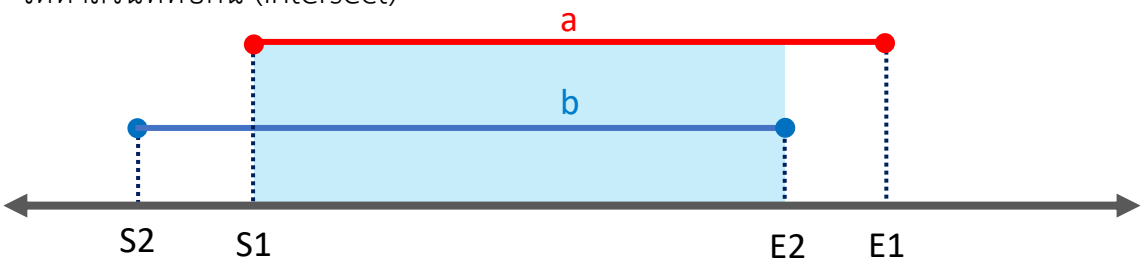
งานที่ 2 หาส่วนที่ทับกัน (Intersect)

def intersection(a, b):

รับ a เก็บสตริงแทนช่วงของจำนวนแรก สมมติว่าเป็น [S1, E1]

รับ b เก็บสตริงแทนช่วงของจำนวนที่สอง สมมติว่าเป็น [S2, E2]

*** ให้หาส่วนที่ทับกัน (Intersect)



แนวคิด

1. ถ้าเจอช่วงเปล่า “(0,0)” คืน ช่วงเปล่า
2. ปรับสตริงที่รับ (ใช้วิธีคล้ายกับข้อ 02_StrList_06 Add_vector)
3. หาว่าขอบซ้าย ให้ดูว่าใครมากกว่ากัน
4. หาว่าขอบขวา ให้ดูว่าใครน้อยกว่ากัน
5. คืนช่วงโดยที่ [ขอบซ้าย, ขอบขวา]

ตัวอย่างโค้ดเทียม

```
def intersection(a, b):
    if a, b is empty interval:
        return empty interval
    modify a, b
    l = max S1, S2; r = min E1, E2
    return [l,r]
```

งานที่ 3 เช็คซับเซต

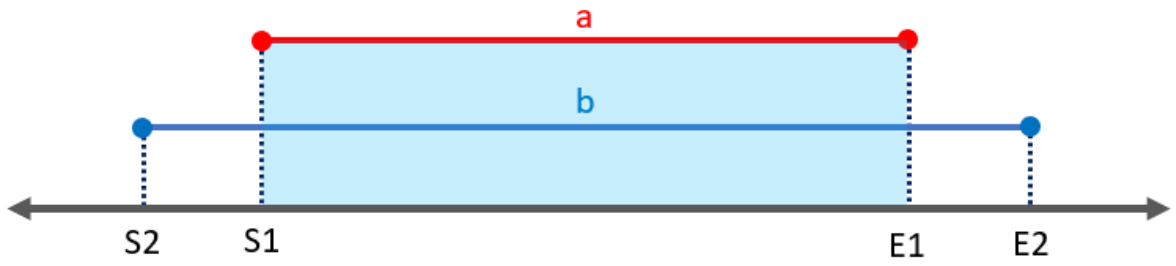
def is_subset(a, b):

รับ a เก็บสตริงแทนช่วงของจำนวนแรก สมมติว่าเป็น [S1, E1]

รับ b เก็บสตริงแทนช่วงของจำนวนที่สอง สมมติว่าเป็น [S2, E2]

- ช่วงจะประกอบด้วยเลขจำนวนเต็มสองจำนวน
- ตัวซ้าย <= ตัวขวา เสมอ

** ให้ตรวจสอบว่า a เป็น subset ของ b หรือไม่



แนวคิด

1. ถ้า a คือ ช่วงเปล่า “(0,0)” คืน จริง
2. ถ้า b คือ ช่วงเปล่า “(0,0)” คืน เท็จ
3. ปรับสตริงที่รับ (ใช้วิธีคล้ายกับข้อ 02_StrList_06 Add_vector)
4. จากภาพ S2 <= S1 <= E1 และ S1 <= E1 <= E2 แต่ S1 <= E1, S2 <= E2 อยู่แล้ว
5. ตรวจสอบว่า ข้อ 4 จริงหรือเท็จ แล้วส่งออก

หรือใช้ intersection มาช่วย (ลองคิดดูว่า subset ∩ set ได้อะไร) Credit @tew_isadawong

ตัวอย่างโค้ดเทียม

```
def is_subset(a, b):
    if a is empty interval: return True
    if b is empty interval: return False
    modify a, b
    B = (S2 <= S1 <= E1) and (S1 <= E1 <= E2)
    return B
```

งานที่ 4 หาซับเซต (วิธีที่ไม่ได้ Bonus)

def get_subsets(list_a, list_b):

รับ list_a, list_b เป็นลิสต์ที่มีสมาชิกแต่ละตัวเป็นสตริงที่เก็บค่าช่วง

- ซึ่งสมาชิกในลิสต์เรียงมาให้เรียบร้อย
- ไม่มีสมาชิกที่เป็นช่วงว่าง

คืน ลิสต์ ที่มีสมาชิกช่วงเป็นสมาชิก list_a ที่เป็น subset ของสมาชิกหนึ่งใน list_b

ตัวอย่าง

get_subsets(['[-2,3]','[4,6]', '[8,9]', '[10,10]', '[15,18]'], ['[0,10]', '[12,20]'])

คืน ['[4,6]', '[8,9]', '[10,10]', '[15,18]']

[-2,3]	[4,6]	[8,9]	[10,10]	[15,18]

[0,10]	[12,20]

OUTPUT

ตัวอย่างโค้ดเทียม

```
def get_subsets(list_a, list_b):
    create empty list
    for a in list_a:
        for b in list_b:
            if a is subset b: add a
    return list # ห้ามใช้คำสั่ง sort
```

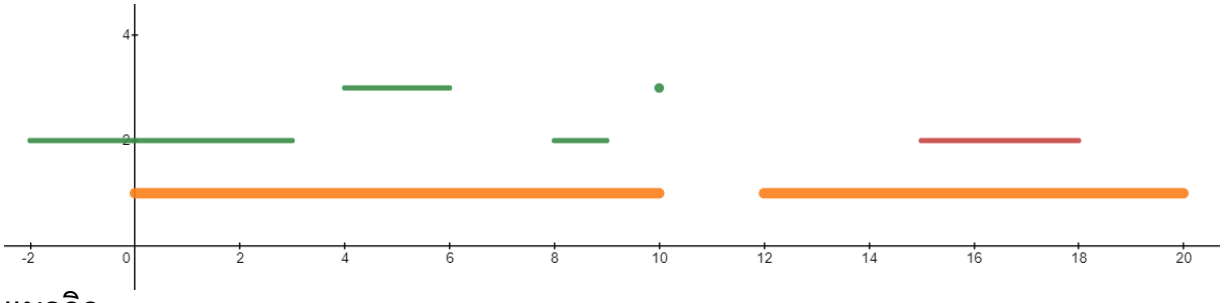
```
No Bonus
True
373.4165 second
```

หมายเหตุ list_size = 20000

ใช้โปรแกรม Visual Studio Code

```
>>> print('2' > '10')
True
```

งานที่ 4 หาซับเซต (วิธีที่ได้ Bonus วิธีที่ 1)



แนวคิด

1. สร้างตัวนับโดยเริ่มจาก 0 และตัวหยุด(ความยาวลิสต์) ของทั้ง 2
2. สร้างลิสต์เปล่า
3. ในขณะที่ list_a ยังไม่ถึงตัวสุดท้าย
    - ถ้า list_b เลยตัวสุดท้าย ให้จบการทำงาน
    - ถ้าเป็น subset ให้เพิ่มลิสต์ย่อย a นั้นไป และตั้งสถานะว่าตัวก่อนหน้าใช้ได้
    - ถ้าตัวก่อนหน้าใช้ได้ และไม่เป็น subset เลื่อน B ไป 1, ถอย A มา 1 และตั้งสถานะว่าตัวก่อนหน้าใช้ไม่ได้
    - เลื่อน A ไป 1
4. ส่งกลับลิสต์นั้น

<u>ตัวอย่างโค้ดเทียม</u>

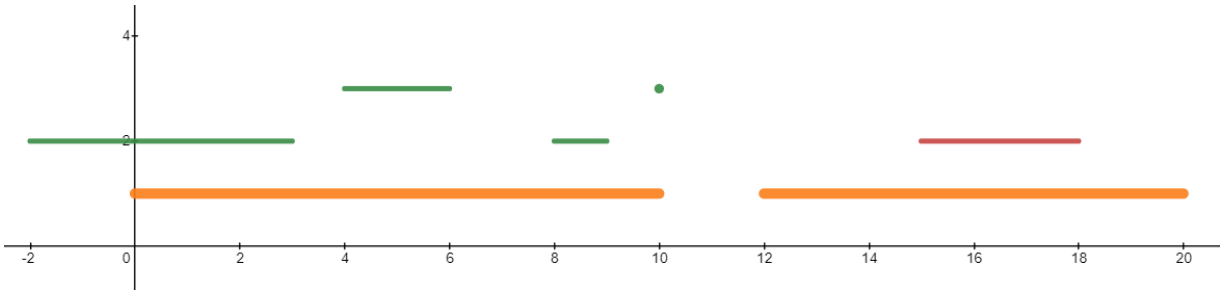
```
def get_subsets(list_a, list_b):
    prev = False; idxA = idxB = 0
    create empty list
    while idxA < size of A:
        if idxB >= size of B: break;
        if subset: add list_a[idxA]; prev = True
        if prev = True and not subset:
            idxA-- ; idxB++ ; prev = False
        idxA++
    return list
```

```
Solution 1
True
0.0479 second
```

<u>หมายเหตุ</u> list_size = 20000

ใช้โปรแกรม Visual Studio Code

งานที่ 4 หาซับเซต (วิธีที่ได้ Bonus วิธีที่ 2) Credit @Sali5a



แนวคิด

จากวิธีก่อนหน้าที่ไม่ได้โบนัส สังเกตว่าเราจะเริ่มรันที่ตัวแรกทุกครั้ง สรุปคอมพิวเตอร์จะต้องทำงานแปรผันกับ len(list_a) x len(list_b) ซึ่งเยอะมาก ถ้าลองมองดีๆ ข้อมูลที่ให้มาถูกเรียงลำดับแล้ว ดังนั้น จากภาพเราจะค่อยๆ เช็คทีละตัวจากซ้ายไปขวา ถ้าลิสต์ย่อย a ใช้ได้ แล้วจะจด index ในขณะนั้นของ b ในรอบต่อไปที่เข้ามาจะเริ่มที่ index นั้น แทนที่จะเริ่มจาก 0 ซึ่งทำให้คอมพิวเตอร์ทำงานแปรผันกับ len(list_a) + len(list_b) เท่านั้น แต่ต้องเช็คด้วยเสมอว่าขอบซ้ายของ sub_a น้อยกว่าขอบซ้ายของ sub_b ด้วย เนื่องจากจะรันใน b จนจบโดยไม่ได้อะไร (ตัวซ้ายสุดยังไม่ได้ ตัวไปซึ่งมากกว่าจะได้หรือ) โดยที่ให้หยุดแล้วเลื่อน a ไป 1 index

ตัวอย่างโค้ดเทียม

```
def get_subsets(list_a, list_b):
    create empty list; begin = 0
    --> i in 0 to len(list_a):
        --> j in begin to len(list_b):
                if subset:
                    add a[i]; begin = j
                if a[i][0] < b[j][0]: break
    return list
```

```
Solution 2
True
0.0708 second
```

หมายเหตุ list_size = 20000

ใช้โปรแกรม Visual Studio Code