Programming Fundamentals I Lab

8. Searching and Sorting

ชื่อ _____ รหัสนิสิต _____

งานชิ้นนี้จะช่วยให้คุณได้ฝึกฝนการออกแบบและวิเคราะห์อัลกอริทึมให้มีประสิทธิภาพ โดยใช้เทคนิคการค้นหาและการเรียงข้อมูล

8.1 Sorted List Intersection

ในปัญหานี้ เราต้องการให้คุณออกแบบฟังก์ชันเพื่อรับ list ของจำนวนเต็มจำนวนสองสายมาเป็น input ซึ่งมีคุณสมบัติสำคัญอย่างหนึ่งคือ ตัวเลขในแต่ละ list นั้นจะมีค่าเรียงจากน้อยไปหามาก และคำนวณหา list ที่เป็น intersection ของ list ทั้งสอง โดย intersection ของ list A และ B คือ list ที่สมาชิกแต่ละตัวเป็นสมาชิกของทั้ง A และ B ด้วย โดยให้ list ผลลัพธ์นั้นเรียงค่าจากน้อยไปหามากเช่นกัน

แนวคิดที่จะแก้ปัญหานี้ก็คือ เมื่อเรารับ list ทั้งคู่มาแล้ว (สมมติว่าเป็น list a และ b) ให้วนรอบตรวจสอบสมาชิกแต่ละตัวของ a ว่าเป็นสมาชิกของ b ด้วยหรือไม่ (ทำได้โดยการวนรอบใน b ว่ามีตัวเลขนี้หรือไม่) ถ้าเป็นสมาชิกของ b ด้วยก็ให้นำตัวเลขนั้นเพิ่มเข้าไปใน list ผลลัพธ์ที่จัดเตรียมไว้ เมื่อตรวจสอบสมาชิกของ a ครบทุกตัวแล้ว เราจะได้ list ผลลัพธ์เป็นคำตอบ

จงเขียนฟังก์ชันชื่อ intersect_1(a, b) ที่ทำงานตามแนวคิดข้างต้นและ return intersection ของ a และ b เป็นผลลัพธ์ และให้ฟังก์ชันนี้แสดงจำนวนครั้งที่ทำการอ่านค่าของสมาชิกของ list a และ b ตลอดการทำงานด้วย (การวนรอบใน a แต่ละรอบถือเป็นการอ่านค่าของสมาชิกของ a หนึ่งครั้ง การอ้างอิงโดยตรง เช่น b[i] ก็นับเป็นหนึ่งครั้งเช่นกัน ให้แสดงจำนวนครั้งทั้งหมดเป็นค่า Count โดยไม่จำเป็นต้องตรงกับตัวอย่าง แต่ไม่ควรจะมากไปกว่าความยาวของ a และ b คูณกัน)

ตัวอย่างการทำงาน

```
In [1]: a = [1, 3, 5, 6, 10]
In [2]: b = [2, 3, 5, 7, 9, 10]
In [3]: intersect_1(a, b)
```

```
Count = 21
Out [3]: [3, 5, 10]
```

8.2 เรียงอายุ

จงเขียนฟังก์ชัน ageSort ที่รับชื่อ และวันเกิดของคน n คน และทำการคืนค่าเป็นลิสต์ของชื่อคนทั้งหมด เรียงตามอายุจากมากไปน้อย ในกรณีที่อายุเท่ากัน ให้เรียงตามลำดับตัวอักษรในชื่อ

ประเด็นที่น่าสนใจของโจทย์ข้อนี้คือการเปรียบเทียบข้อมูลชนิด tuple เพื่อนำไปเรียงลำดับ

ตัวอย่างการทำงาน

```
In [1]: p1 = ('Mark', 20, 7, 1977)
In [2]: p2 = ('Andy', 11, 2, 1990)
In [3]: p3 = ('Tony', 5, 10, 1990)
In [4]: p4 = ('Pete', 30, 11, 1983)
In [5]: ageSort([p1, p2, p3, p4])
Out[5]: ['Mark', 'Pete', 'Andy', 'Tony']
```

8.3 List Differences

กำหนด L1, L2 เป็นลิสต์ของจำนวนเต็มไม่เรียงกัน เราต้องการหาลิสต์ S ที่ประกอบด้วยสมาชิกของ L1 ที่ไม่อยู่ใน L2 ในโจทย์ข้อนี้ เราจะให้คุณสร้างฟังก์ชัน diff(l1, l2) สำหรับคำนวนลิสต์ S ดังกล่าวโดยใช้แนวคิดดังนี้

- ทำการเรียงข้อมูลใน L2 ด้วยเทคนิคใดก็ได้
- สำหรับข้อมูลแต่ละตัวใน L1 ให้ทำการค้นดูว่าเป็นสมาชิกใน L2 หรือไม่ด้วย Binary search หากข้อมูลตัวไหนไม่เป็นสมาชิกใน L2 ให้ append เข้าสู่ลิสต์ S ที่เตรียมรอรับไว้
- เรียงข้อมูลใน S ด้วยเทคนิคใดก็ได้ ก่อนคืนค่ากลับสู่ผู้เรียกฟังก์ชัน

ตัวอย่างการทำงาน

```
In [1]: lst1 = [2, 5, 10, 6, 75, 3]
In [2]: lst2 = [4, 10, 74, 75, 92, 6, 18]
In [3]: diff(lst1, lst2)
Out[3]: [2, 3, 5]
```