Lab 02: Time Complexity

13 สิงหาคม 2567

รับช็อป (Deep Ocean Blue)

ในวันที่ [redacted] 2567 ภาควิชาวิศวกรรมคอมพิวเตอร์ได้จัดพิธีมอบเสื้อช็อปให้กับ CPE37 ทั้งหลักสูตรปกติ, นานาชาติ, HDS และ RC

ปรากฏว่าในวันนั้นมีนักศึกษาบางคนไม่สามารถเข้าร่วมงานได้ ภาควิชาจึงหาวันในการมอบเสื้อหากับนักศึกษาที่ไม่ได้เข้าร่วมงานจำนวน n คน โชคดีที่พิธีในวันนี้มีอาจารย์เข้าร่วมพิธีมอบเสื้อช็อปจำนวน n คน เท่ากันพอดี ในกรณีนี้สมมติว่ามีนักศึกษาและอาจารย์อย่างละ 4 คน ดังนี้: อ.ปิยนิตย์, อ.จุมพล, อ.สนั่น และ อ.ทวีชัย เรียงตามลำดับจากซ้ายไปขวา และนักศึกษาแทนด้วยรหัสนักศึกษา 1010, 1012, 1066, 1064 เรียงจากซ้ายไปขวาเช่นกัน

ปรากฏว่ามีนักศึกษาอยู่หนึ่งคนที่ไม่อยากรับเสื้อช็อปกับ อ.ทวีชัย (ซึ่งคือรหัส 1010) เพราะว่าอาจารย์ไม่ช่วยปัดเกรดให้ ทำให้นอยด์และไม่อยากรับกับอาจารย์ ทำให้การจัดลำดับการเข้ารับช็อปมีปัญหาเล็กน้อย

ในสถานการณ์ปกติ เราสามารถนับจำนวนวิธีในการพานักศึกษาเข้ารับช็อปด้วย $n!$ ในกรณีนี้คือ $4!$ ซึ่งเท่ากับ 24 วิธี แต่เมื่อมีนักศึกษา 1 คนไม่อยากรับกับอ.วี (ลำดับที่ 4 จากซ้าย) ทำให้นักศึกษารหัส 1010 ไม่สามารถยืนลำดับที่ 4 ได้ ทำให้จำนวนวิธีในการจัดเรียงลำดับเหลือเพียง **18** วิธีเท่านั้น

งานของนักศึกษา

จงหาว่า หากวันนั้นมีอาจารย์และนักศึกษาเข้าร่วมพิธี n คนเท่ากัน โดยที่อ.ทวีชัยยืนอยู่ที่ลำดับ v นักศึกษาแต่ละคนมีรหัสดังนี้ $\{S_1, S_2, \ldots, S_n\}$ มีนักศึกษาจำนวน p คนที่ไม่อยากเข้ารับกับอ.วี แต่ละคนมีรหัสนักศึกษาเป็น $\{s_1, s_2, \ldots, s_p\}$ จงหาว่ามีวิธีจัดเรียงลำดับนักศึกษาเข้ารับเสื้อช็อปเท่าไหร่ เมื่อคำนึงถึงกรณีนักศึกษาไม่อยากรับกับอ.วีแล้ว โดยมีข้อกำหนดว่านักศึกษาต้องใช้ Recursive Function อย่างน้อย 1 ฟังก์ชัน จากนั้นให้นักศึกษาหาด้วยว่า Algorithm ที่นักศึกษาเขียนเพื่อหาคำตอบนั้น มี Time Complexity เท่าใด เขียนเป็น O-Notation เช่น $O(n \log n)$ เมื่อ n คือจำนวนนักศึกษา (หรืออะไรก็ว่าไปในทำนองนี้)

โดยในขั้นแรกให้นักศึกษา**ออกแบบ**ขั้นตอนการแก้ปัญหาด้วย **Pseudocode** จากนั้นให้นักศึกษาคำนวณ Time Complexity โดยอยู่ในรูป O-Notation แล้วนำ Pseudocode ที่ออกแบบมาไปเขียนโค้ดภาษา C หรือ C++ จากนั้นในโค้ด ให้เขียนการนับ Basic Operation โดย Basic Operation ที่จะนับมีกลุ่มดังต่อไปนี้

- กลุ่ม $+, -$
- กลุ่ม $\times, \div$
- กลุ่มการเปรียบเทียบ (Comparison)

ข้อมูลนำเข้า (Input)

บรรทัด 1	จำนวนเต็ม n และ v แสดงจำนวนนักศึกษาและอาจารย์ และตำแหน่งที่อ.วียืนอยู่ โดยที่ $1 \leq n \leq 10$ และ $1 \leq v \leq n$
บรรทัด 2	จำนวนเต็ม n จำนวน ดังนี้ $S_1, S_2, \ldots, S_n$ (แต่ละตัวคั่นด้วยช่องว่างหนึ่งตัว) แสดงรหัสนักศึกษา ซึ่งเป็นจำนวนเต็ม 4 หลัก
บรรทัด 3	จำนวนเต็ม p คือจำนวนนักศึกษาที่ไม่อยากรับกับอ.วี โดยที่ $0 \leq p < n$
บรรทัด 4	จำนวนเต็ม p จำนวน ดังนี้ $s_1, s_2, \ldots, s_p$ (แต่ละตัวคั่นด้วยช่องว่างหนึ่งตัว) แสดงรหัสนักศึกษาที่ไม่อยากรับเสื้อช็อปกับอ.วี ซึ่งเป็นจำนวนเต็ม 4 หลัก

ข้อมูลส่งออก (Output)

บรรทัด 1	จำนวนวิธีที่สามารถจัดนักศึกษาเข้ารับช็อปได้

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
4 4 1010 1012 1066 1064 1 1010	18
5 3 1001 1002 1003 1004 1005 2 1001 1003	72
10 1 1003 1038 1046 1050 1059 1062 1063 1064 1079 1086 2 1003 1038	2903040
4 2 1002 1004 1099 1075 0	24

หมายเหตุ: รับประกันว่าจะต้องมีอย่างน้อย 1 คนที่อยากรับกับอ.วี (ไม่มีทางที่ $p = n$)

หมูปิ้ง (Grilled Pork)

ทุก ๆ ช่วงเดือนกุมภาพันธ์ถึงมีนาคม ภาควิชาวิศวกรรมคอมพิวเตอร์จะมีกิจกรรม CPE Games ซึ่งมีการแข่งขันทั้งกีฬา การละเล่นพื้นบ้าน รวมไปถึง E-Sports ต่าง ๆ และในค่ำคืนสุดท้ายของกิจกรรม ก็จะมีงานเลี้ยงปาร์ตี้ ซึ่งครั้งที่จะถึงนี้ (CPE Games 2025) จะมีซุ้มขายหมูปิ้งที่ขายโดย อ.ทวีชัย เพื่อหารายได้เข้าสู่ภาควิชา

อ.ทวีชัยวางแผนที่จะขายหมูปิ้งเป็นกล่อง ๆ ซึ่งมีทั้งหมดสี่กล่อง โดยในสถานการณ์สมมติแต่ละกล่องมีจำนวนไม้ดังนี้

- กล่องแบบแรก มีหมูปิ้ง 8 ไม้
- กล่องแบบที่สอง มีหมูปิ้ง 12 ไม้
- กล่องแบบที่สาม มีหมูปิ้ง 15 ไม้
- กล่องแบบที่สี่ มีหมูปิ้ง 20 ไม้

สมมติในการซื้อแต่ละครั้ง อ.วี จะกำหนดไม่ให้นักศึกษาซื้อเกิน 100 ไม้ภายในการซื้อครั้งเดียว หากนักศึกษาอยากซื้อมากกว่า 100 ไม้ นักศึกษาต้องไปต่อแถวใหม่เพื่อซื้อใหม่อีกครั้งหนึ่ง

สมมติว่าหากนักศึกษาอยากซื้อหมูปิ้งทั้งหมด 23 ไม้ นักศึกษาสามารถซื้อหมูปิ้งโดยซื้อกล่อง 8 ไม้ 1 กล่อง และกล่อง 15 ไม้ 1 กล่อง หรือหากอยากซื้อทั้งหมด 31 ไม้ นักศึกษาสามารถซื้อได้จากกล่อง 8 ไม้ 2 กล่อง กล่อง 15 ไม้ 1 กล่อง

สังเกตได้ว่า จะมีหมูปิ้ง**บางจำนวน**ที่ไม่สามารถซื้อได้ เช่น 34, 41, 49 ไม้ เพราะไม่สามารถรวมกันจากกล่องทั้งสี่รูปแบบได้เลย ซึ่งจำนวนไม้สูงสุดที่ไม่สามารถซื้อได้คือ **49** ไม้

งานของนักศึกษา

จงหาว่าถ้าอาจารย์วีทำหมูปิ้งขายแบบกล่องโดยมีทั้งหมด 4 กล่อง แต่ละกล่องมีหมูปิ้ง $\{p_1, p_2, p_3, p_4\}$ ไม้ หากเราสั่งได้มากที่สุด n ไม้ จำนวนไม้ที่มากที่สุดที่**ไม่สามารถสั่งได้**เป็นเท่าใด จากนั้นให้นักศึกษาหาด้วยว่า Algorithm ที่นักศึกษาเขียนเพื่อหาคำตอบนั้น มี Time Complexity เท่าใด เขียนเป็น O-Notation เช่น $O(2^n)$ เมื่อ n คือจำนวนไม้ (หรืออะไรก็ว่าไปในทำนองนี้)

โดยในขั้นแรกให้นักศึกษา**ออกแบบ**ขั้นตอนการแก้ปัญหาด้วย **Pseudocode** จากนั้นให้นักศึกษาคำนวณ Time Complexity โดยอยู่ในรูป O-Notation แล้วนำ Pseudocode ที่ออกแบบมาไปเขียนโค้ดภาษา C หรือ C++ จากนั้นในโค้ด ให้เขียนการนับ Basic Operation โดย Basic Operation ที่จะนับมีกลุ่มดังต่อไปนี้

- กลุ่ม $+, -$
- กลุ่ม $\times, \div$
- กลุ่มการเปรียบเทียบ (Comparison)

ข้อมูลนำเข้า (Input)

บรรทัด 1	จำนวนเต็ม 4 จำนวน แสดงข้อมูลจำนวนไม้ในแต่ละกล่อง p_1, p_2, p_3, p_4 (แต่ละตัวคั่นด้วยช่องว่าง 1 ตัว)
บรรทัด 2	จำนวนเต็ม n แสดงจำนวนหมูปิ้งที่มากที่สุดที่สามารถสั่งได้

ข้อมูลส่งออก (Output)

บรรทัด 1	จำนวนหมูปิ้งที่มากที่สุดที่ไม่สามารถซื้อได้ภายใน n ไม้

ตัวอย่างข้อมูลนำเข้า ส่งออก (Examples of Input & Output)

Input	Output
8 12 15 20 100	49
2 5 6 10 250	3
6 10 14 19 300	27
11 21 31 41 200	133

คำใบ้ (Hint)

หนึ่งในวิธีการคำนวณที่แนะนำ (ที่สามารถคำนวณ Time Complexity ได้ง่าย) คือการนำจำนวนหมูปิ้งแต่ละกล่องเทียบสัมประสิทธิ์ที่ต่าง ๆ กัน โดยในเคสแรก เราสามารถสร้างสมการได้ว่า

$$8a + 12b + 15c + 20d \leq 100$$

จากนั้นเมื่อเราเทียบความสัมพันธ์ของสัมประสทธิ์แต่ละตัว เราจะได้ว่า

$$0 \leq a \leq 12$$
$$0 \leq b \leq 8$$
$$0 \leq c \leq 6$$
$$0 \leq d \leq 5$$

(ให้เราลองคิดดูว่าเลข $12, 8, 6, 5$ มาจากไหน) จากนั้น เราสามารถทำการ Loop เพื่อหาผลรวมของจำนวนไม้ที่สามารถสั่งได้ ส่วจำนวนที่เหลือก็คือจำนวนไม้ที่ไม่สามารถสั่งได้นั่นเอง เราก็หาว่าจำนวนไม้ที่ไม่สามารถสั่งได้มากที่สุดเป็นเท่าใด

การหา Time Complexity

ให้นักศึกษาลองเขียนโค้ดแล้วให้คำนวณแต่ละขั้นด้วยการ $+1$ ไว้ทุก**Basic Operation** (กลุ่มการบวก, คูณ, การเปรียบเทียบ) เช่น

```
#include<iostream>
#include<some_other_libraries>
using namespace std;
int main(){
    int = some_complexity_variable = 0;
    int a;
    cin >> a;
    int n = a * 100;
    some_complexity_variable++; //for multiplication
    for (int i=0; i<=100;i++){
        some_complexity_variable++; //for comparison
        if (i <= something){
            some_other_thing = i + 4;
            some_complexity_variable++; //for addition
    }
}
```

จากนั้นเอามาคำนวณแยกว่า แต่ละอันมีจำนวน Operation เท่าใด เช่นเกิดกลุ่มบวกลบเท่าไหร่, คูณหารเท่าไหร่, มีการเปรียบเทียบค่าเท่าไหร่ แล้วลองคิดดูว่าควรจะเอาอันไหนเป็น Time Complexity