05506232 Mathematics for Computer Science

Lab 14

Objectives

1. เข้าใจการเข้ารหัสด้วย Shamir's Secret Sharing Algorithm ด้วยการเขียนโปรแกรมภาษาไพธอน
2. เข้าใจการบีบอัดข้อมูลด้วย Huffman Coding ด้วยการเขียนโปรแกรมภาษาไพธอน

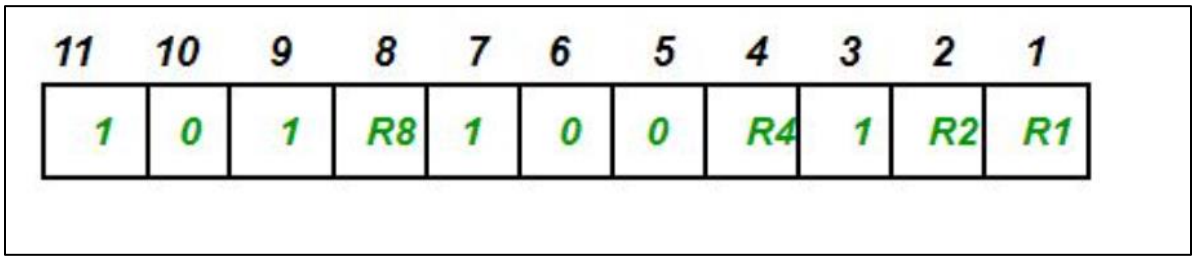
Hamming Code

Hamming Code เป็นอัลกอริทึมหนึ่งที่นิยมใช้ในการส่งข้อมูลผ่านระบบเครือข่ายที่อาจมีการเปลี่ยนแปลงหรือสูญหายของข้อมูลได้ตลอดเวลา ทำให้ผู้รับสามารถรับรู้ได้ว่ามีการผิดพลาดของการส่งข้อมูล (Error Detection) และสามารถแก้ไขข้อมูลให้ถูกต้อง (Error Correction) ได้

ในแล็บนี้ เพื่อความง่ายในการเข้าใจ เราจะใช้ข้อมูลตัวเลขจำนวนเต็มตั้งแต่ 0 – 127 (7 บิต)

การจำลองการเข้ารหัสด้วย Hamming Code ที่มีข้อมูล 7 บิต สามารถทำได้ดังนี้

- เพิ่มจำนวนบิตอีก 4 บิต เพื่อตรวจสอบข้อมูล (Redundant Bits) รวมข้อมูลทั้งหมดเป็น 11 บิต
- Redundant Bits จะเพิ่มที่ตำแหน่งที่ยกกำลัง 2 ในที่นี้จะเพิ่มที่ตำแหน่ง 1, 2, 4, และ 8 เรียกว่า R1, R2, R4, R8 นับจากด้านหลัง



ที่มา - https://www.geeksforgeeks.org/hamming-code-in-computer-network/

- R1 จะตรวจสอบบิตที่ 3, 5, 7, 9, 11 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ ให้ R1 เป็น 0 หากเป็นจำนวนคี่ ให้ R1 เป็น 1

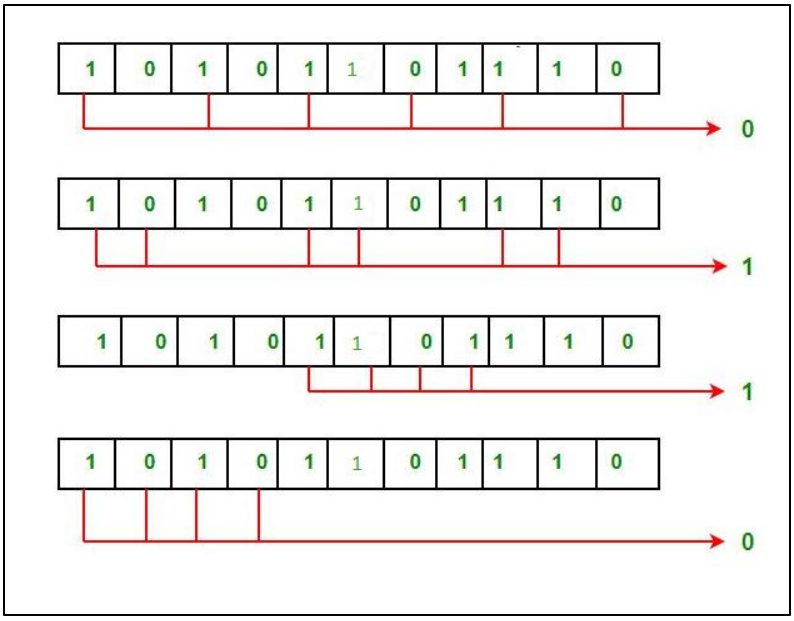
- R2 จะตรวจสอบบิตที่ 3, 6, 7, 10, 11 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ ให้ R2 เป็น 0 หากเป็นจำนวนคี่ ให้ R2 เป็น 1
- R3 จะตรวจสอบบิตที่ 5, 6, 7 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ ให้ R3 เป็น 0 หากเป็นจำนวนคี่ ให้ R3 เป็น 1
- R4 จะตรวจสอบบิตที่ 9, 10, 11 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ ให้ R4 เป็น 0 หากเป็นจำนวนคี่ ให้ R4 เป็น 1

Hamming Code นี้ เรียกว่า Hamming (11, 7) จะตรวจสอบการส่งข้อมูลผิดพลาดและแก้ไขได้ 1 บิต

การตรวจสอบว่าบิตใดที่มีการส่งข้อมูลผิดพลาดให้ตรวจสอบย้อนกลับดังนี้

- บิตที่ 1 ตรวจสอบบิตที่ 1, 3, 5, 7, 9, 11 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ให้เป็น 0 หากเป็นจำนวนคี่ให้เป็น 1
- บิตที่ 2 ตรวจสอบบิตที่ 2, 3, 6, 7, 10, 11 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ให้เป็น 0 หากเป็นจำนวนคี่ให้เป็น 1
- บิตที่ 3 ตรวจสอบบิตที่ 4, 5, 6, 7 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ให้เป็น 0 หากเป็นจำนวนคี่ให้เป็น 1
- บิตที่ 4 ตรวจสอบบิตที่ 8, 9, 10, 11 ว่ามีเลข 1 อยู่รวมกันกี่ตัว หากเป็นจำนวนคู่ให้เป็น 0 หากเป็นจำนวนคี่ให้เป็น 1

เช่น จากรูปด้านล่าง นำบิตทั้ง 4 มารวมกันได้ 0110 จึงรู้ว่าบิตที่ 6 ผิดพลาด ต้องเปลี่ยน 1 กลับเป็น 0



ที่มา - https://www.geeksforgeeks.org/hamming-code-in-computer-network/

จงเขียนโปรแกรมเพื่อจำลองการเข้ารหัสแบบ Hamming Code จากวิธีการด้านบน โดยส่งไฟล์นามสกุล .py แยกตามข้อดังนี้

1.1 เลือกค่าที่จะส่งข้อมูลตั้งแต่ 0 - 127 แบบสุ่ม (เพื่อให้ง่ายต่อการจำลอง) แปลงเป็นเลขฐาน 2 หลังจากนั้นให้เข้ารหัสแบบ Hamming Code (11, 7) แสดงค่าที่เข้ารหัสออกทางหน้าจอ หลังจากนั้น สุ่มบิต 1 บิตเพื่อเปลี่ยนเลข และหาเลขบิตที่ถูกเปลี่ยนโดยวิธี Hamming Code และแสดงออกทางหน้าจอ

Deadline: 23 October 2023

Floating Point Arithmetic vs Fixed Point Arithmetic

เลขทศนิยมในคอมพิวเตอร์สามารถเขียนแทนด้วย 2 รูปแบบ ได้แก่ Floating Point Arithmetic และ Fixed Point Arithmetic ซึ่งแตกต่างกันที่วิธีการเก็บข้อมูล การเก็บข้อมูลแบบ Floating Point Arithmetic มีข้อดีที่สามารถเก็บข้อมูลทศนิยมจำนวนละเอียดมาก ๆ ได้โดยที่ใช้พื้นที่ในการจัดเก็บน้อย แต่มีข้อเสียที่สามารถเกิดความคลาดเคลื่อนของข้อมูลได้ง่าย ขณะที่ Fixed Point Arithmetic จะสามารถเก็บข้อมูลได้แม่นยำ เนื่องจากใช้จำนวนเต็มแล้วเลื่อนจุดทศนิยม แต่มีข้อเสียที่ต้องใช้พื้นที่ในการเก็บข้อมูลเยอะ หากต้องการใช้ข้อมูลทศนิยมที่ละเอียดมาก ๆ

ในแล็บนี้ เราจะทดลองการคำนวณ Floating Point Arithmetic และ Fixed Point Arithmetic พร้อมทั้งขนาดหน่วยความจำที่เก็บตัวแปร จงเขียนโปรแกรมเพื่อคำนวณทศนิยม โดยส่งไฟล์นามสกุล .py แยกตามข้อดังนี้

2.1 Floating Point Arithmetic ให้พิมพ์โค้ดตามด้านล่าง แล้วเปรียบเทียบการคำนวณและหน่วยความจำที่ใช้ พร้อมทั้งลองการคำนวณอื่น เช่น ลบ คูณ หาร

```python
from sys import getsizeof

float_a = 0.2
float_b = 0.1

float_c = float_a + float_b

print(float_c)
print(getsizeof(float_a))
print(getsizeof(float_b))
print(getsizeof(float_c))
```

2.2 Fixed Point Arithmetic ให้พิมพ์โค้ดตามด้านล่าง แล้วเปรียบเทียบการคำนวณและหน่วยความจำที่ใช้ พร้อมทั้งลองการคำนวณอื่น เช่น ลบ คูณ หาร

```python
from sys import getsizeof
from decimal import Decimal

dec_a = Decimal('0.2')
dec_b = Decimal('0.1')

dec_c = dec_a + dec_b

print(dec_c)
print(getsizeof(dec_a))
print(getsizeof(dec_b))
print(getsizeof(dec_c))
```

Deadline: 23 October 2023