

อัลกอริทึมการแบ่งแยกและเอาชนะ  
(Divide and Conquer algorithm)  
Part 1

อ.ดร.ลือพล พิพานเมฆาภรณ์

luepol.p@sci.kmutnb.ac.th

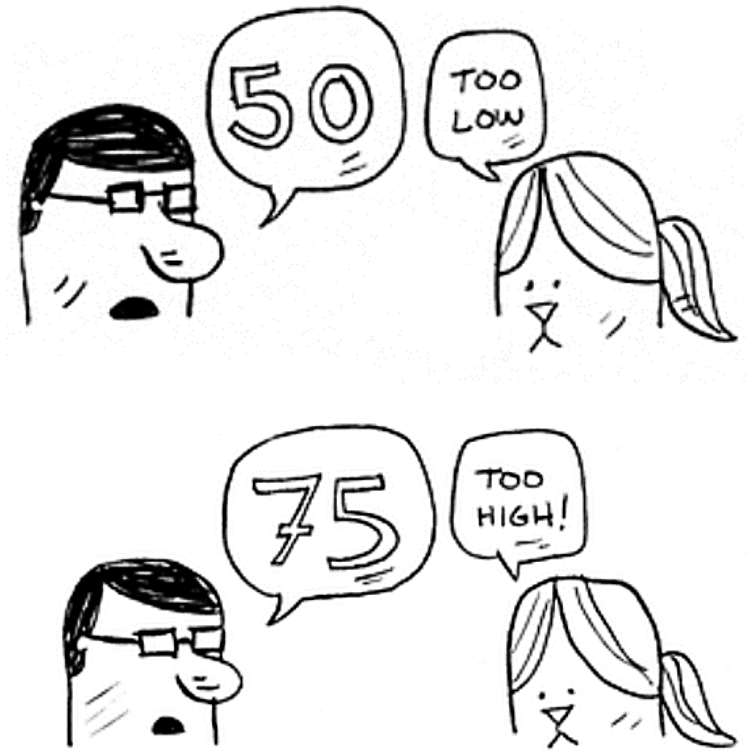
# ขนาดปัญหา (problem size)

- ปัญหาด้านคอมพิวเตอร์มักต้องมีการประมวลผลข้อมูล ซึ่งหากมีข้อมูลมากก็จะใช้เวลาในการประมวลผลเพื่อหาคำตอบนาน แต่หากข้อมูลมีน้อยก็จะใช้เวลาในการประมวลผลเร็วกว่า
- เรามักเรียกขนาดของข้อมูลที่ต้องการประมวลผลว่า **ขนาดของปัญหา** (problem size)
- มักใช้ตัวแปร  $n$  แทนขนาดของปัญหา ตัวอย่างเช่น เรียงข้อมูลในอาร์เรย์ขนาด  $n$  ตัว หรือค้นหาข้อมูลจากอาร์เรย์ขนาด  $n$  ตัว
- โดยทั่วไปปัญหาเหล่านี้มักถูกแก้ด้วย**วิธีตะลุยกาคำตอบ** (brute-force method) ซึ่งมักใช้เวลาในการหาคำตอบนาน เมื่อขนาดของอินพุตมีมาก

# ทายเบอร์มือถือ 2 ตัวท้าย

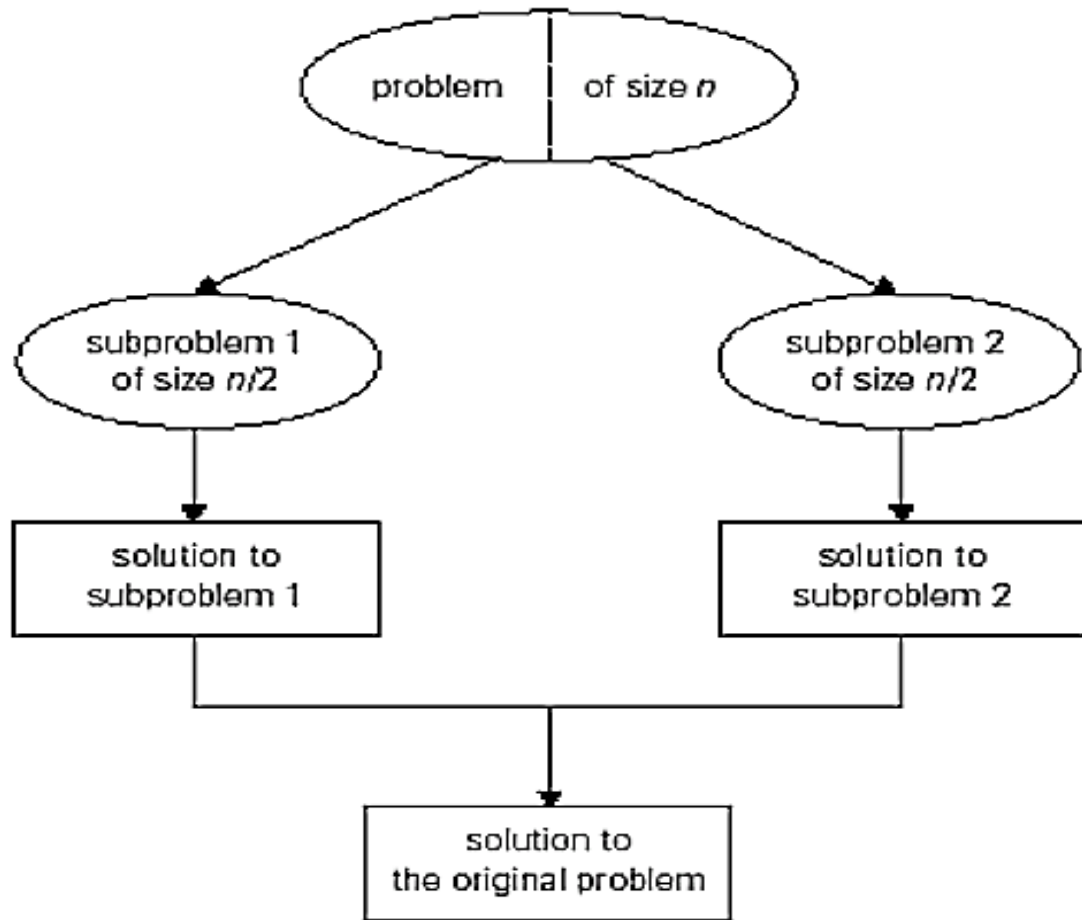


จะต้องถามทั้งหมดไม่เกิน **100** ครั้ง



จะต้องถามทั้งหมดไม่เกิน **??** ครั้ง

# หลักการแบ่งแยกและเอาชนะ (Divide and Conquer)



- เป็นเทคนิคแก้ปัญหาด้วยกระบวนการทางคอมพิวเตอร์ ซึ่งมีแนวคิดมาจากฟังก์ชันรีเคอร์ซีฟ (recursive function) ที่แบ่งปัญหาออกเป็นปัญหาย่อย (sub-problem) ไปเรื่อยๆ จนกระทั่งปัญหาย่อยเหล่านี้นี้เล็กเพียงพอที่จะใช้เวลาหาคำตอบไม่นาน
- หากจำเป็นก็ต้องทำการรวมคำตอบของปัญหาย่อย เพื่อสร้างคำตอบของปัญหาหลัก
- สิ่งที่คาดหวังจาก DAC ก็คือเวลาที่ใช้ในการแก้ปัญหาย่อยๆ น่าจะเร็วกว่าเวลาที่ใช้ในการแก้ปัญหาโดยตรง

# อัลกอริทึม Divide and Conquer

```
procedure divide_and_conquer
begin
  if base case then
    solve problem
  else
    partition problem into subproblems  $L$  and  $R$ 
    solve problem  $L$  using divide-and-conquer
    solve problem  $R$  using divide-and-conquer
    combine solutions to problems  $L$  and  $R$ 
  endif
end
```

เครื่องมือที่สำคัญในการแบ่งปัญหาใหญ่เป็นปัญหาย่อยๆ ก็คือ recursive function

# ทบทวน recursive function

จงเขียนฟังก์ชันรีเคอร์ซีพเพื่อหาค่ามากที่สุดในอาร์เรย์ด้านล่าง

34	3	47	91	32	0
----	---	----	----	----	---

# ตัวอย่างของอัลกอริทึมที่ใช้แนวคิดการแบ่งแยกและเอาชนะ

- Binary Search
  - แบ่งปัญหาโดยการคำนวณค่า mid
  - ไม่มีการรวมคำตอบ
- Merge Sort
  - แบ่งอาร์เรย์ออกเป็น 2 ส่วน
  - รวมคำตอบโดยฟังก์ชัน merge()
- Quick Sort
  - แบ่งอาร์เรย์ออกเป็น 2 ส่วน โดย partition()
  - ไม่มีการรวมคำตอบ

# ตัวอย่างปัญหาที่เหมาะสมของอัลกอริทึม DAC

1. การนับข้อมูลซ้ำ (count duplicate number)
2. การแบ่งกลุ่มข้อมูล (clustering)
3. การคูณเลขจำนวนเต็ม (integer multiplication)
4. การคูณเมตริกซ์ (matrix multiplication)
5. เซตของจุดที่ครอบคลุม (maxima set)



# การนับข้อมูลซ้ำจากตัวเลขที่กำหนด

กำหนดให้อาร์เรย์จำนวนเต็ม  $N$  ตัว เรียงลำดับไว้จากน้อยไปมากแล้ว จงพัฒนาอัลกอริทึมที่มีประสิทธิภาพเพื่อนับจำนวนซ้ำกันของตัวเลขที่กำหนด (target)

2 5 5 5 6 6 8 9 9 9

target: 5



# Workshop: DAC

จงออกแบบอัลกอริทึมแบ่งแยกและเอาชนะ (Divide and Conquer) เพื่อแก้ปัญหา count duplicate number พร้อมเขียนฟังก์ชัน

2 5 5 5 6 6 8 9 9 9

# การลำดับน้อยที่สุดอันดับ $k$

ต้องการหาเลขน้อยที่สุดอันดับ  $k$  ในอาร์เรย์เลขจำนวนเต็มขนาด  $n$  ตัว

1, 5, 10, 4, 8, 2, 6

ค่าน้อยที่สุดอันดับ 3 ของอาร์เรย์นี้ คือ 4

จงเขียนโปรแกรมที่มีประสิทธิภาพในการค้นหาค่าน้อยที่สุดอันดับ  $k$

# การลำดับน้อยที่สุดอันดับ k

วิธีการ brute force

1, 5, 10, 4, 8, 2, 6

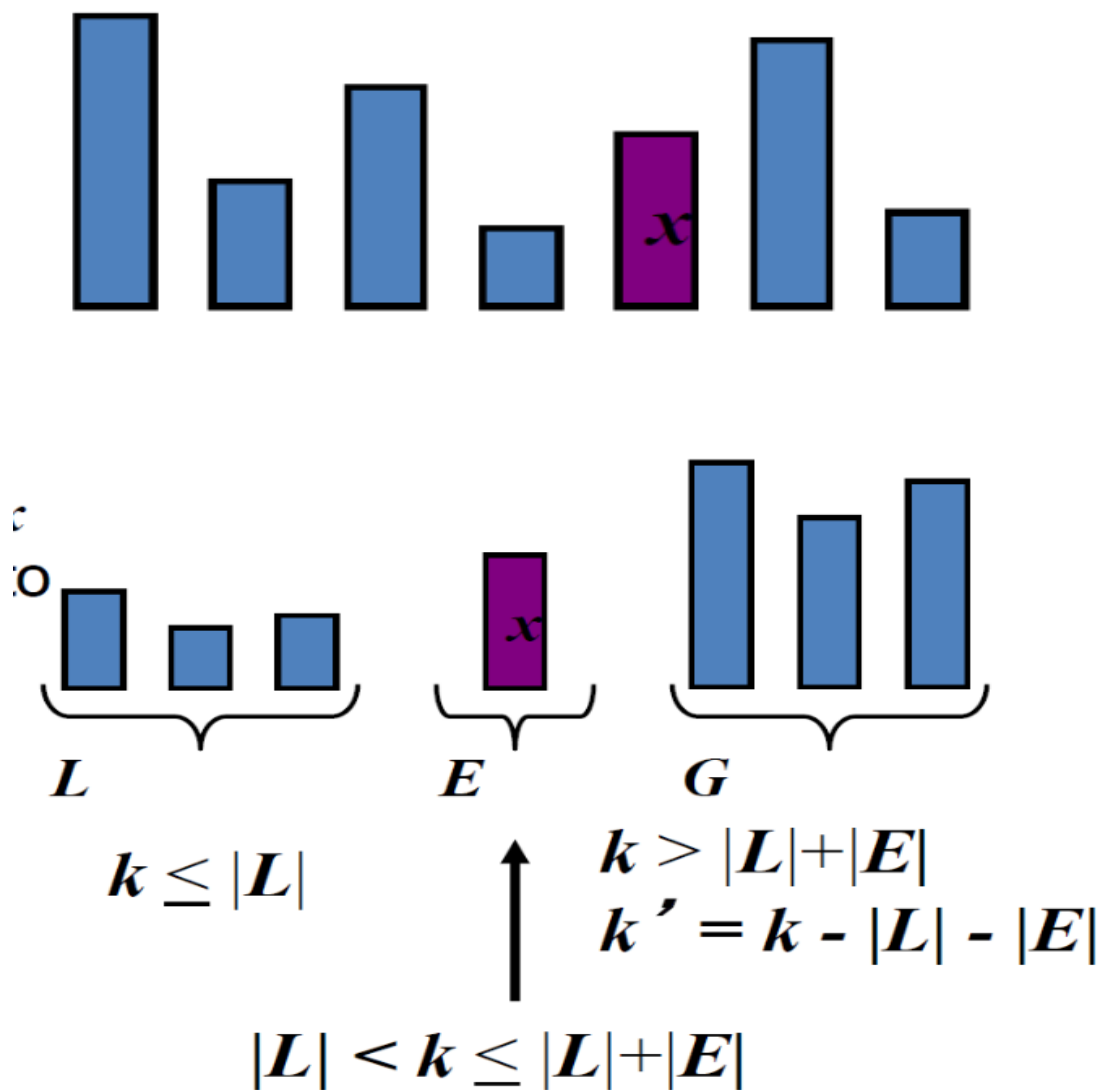


sort

1, 2, 4, 5, 6, 8, 10

หากกำหนดให้  $k = 3$  ก็จะได้คำตอบคือ 4 ปัญหาคือ เราทราบอยู่แล้วว่า sorting algorithm ใช้เวลาเรียงข้อมูล ระหว่าง  $O(n)$  to  $O(n^2)$

# Quick Select algorithm



1. สุ่มหยิบสมาชิก  $x$  เพื่อแบ่งข้อมูลออกเป็น 3 กลุ่ม

- $L$  บรรจุสมาชิก  $< x$
- $E$  เท่ากับ  $x$
- $G$  บรรจุสมาชิก  $> x$

2. ตัดสินใจตามกฎดังนี้

2.1 หาก  $k = |L| + 1$  จบการทำงาน ได้  $x$  เป็นคำตอบ

2.2 หาก  $k \leq |L|$  ทำ partition ในฝั่ง  $L$

2.3 หาก  $k > |L| + 1$  ทำการปรับปรุ้ค่า  $k$  โดยให้  $k' = k - (|L| + 1)$  แล้วทำ partition ฝั่ง  $G$

# ตัวอย่าง Quick Select $k = 3$

1, 5, 10, 4, 8, 2, 6

1, 5, 4, 2, 6, 10, 8  
L E G

1, 2, 5, 4  
 $|L| = 1$

4, 5

- หybrid 6 เป็น pivot เพื่อแบ่งข้อมูลออกเป็น 3 ส่วน
- เนื่องจาก  $3 < 4$  ดังนั้นคำตอบอยู่ใน L  
จึง partition ฝั่ง L
- hybrid 2 เป็น pivot แบ่ง L ออกเป็น 3 ส่วน  
เนื่องจาก  $3 > 1 + 1$  ปรับปรุงค่า  $k' = 3 - (1+1) = 1$   
จากนั้นทำ partition ฝั่ง G
- hybrid 4 เป็น pivot แบ่ง G เป็น 3 ส่วน เนื่องจาก 4  
เป็นสมาชิกตัวแรก จึงเป็นตำแหน่งเดียวกับ  $k'$  ดังนั้นจึง  
สามารถจบการทำงานและส่งกลับค่า 4





# Workshop

จงเขียนฟังก์ชัน Quick Select ตามแนวคิดการแบ่งแยกและเอาชนะเพื่อแก้ปัญหาลำดับ  $k$  กำหนดให้การหีบ pivot ใช้ตำแหน่งสุดท้ายของอาร์เรย์เสมอ

# Integer multiplication with Brute force

ในการคำนวณตัวเลข การคูณ (multiplication) และการหาร (division) ถือเป็นงานที่ต้องใช้ทรัพยากรในการคำนวณสูงเมื่อเปรียบเทียบกับ การบวก (addition) และการลบ (subtraction) โดยเฉพาะอย่างยิ่งหากตัวเลขมีจำนวนหลายๆ หลัก

วิธีการคูณเลข 2 จำนวน ที่เรารู้กันเคยมากที่สุด คือ *pen-and-pencil algorithm* โดยเริ่มจากการพิจารณานำตัวคูณในแต่ละหลักไปคูณหลักของตัวตั้ง จากนั้นเลื่อนตัวคูณไปหลักถัดไป และทำซ้ำจนกระทั่งครบทุกหลัก และนำมาผลลัพธ์บวกกัน

$$\begin{array}{r} 23958233 \\ \times \quad 5830 \\ \hline 00000000 \\ 71874699 \\ 191665864 \\ + 119791165 \\ \hline 139676498390 \end{array}$$

จากตัวอย่างจะเห็นได้ว่า หากตัวตั้งและตัวคูณมีจำนวนหลักเท่ากันคือ  $n$

จำนวนครั้งในการคูณทั้งหมดคือ  $n^2$

# Integer multiplication with divide & conquer

เพื่อลดจำนวนครั้งในการคูณเราอาจใช้เทคนิคการแบ่งแยกและเอาชนะสำหรับแก้ปัญหานี้ได้ แต่เราจะต้องหาวิธีในการแบ่งปัญหาและการรวมปัญหาให้ได้ ซึ่งในปัญหานี้อาจทำได้ง่าย ดังนี้

สมมติให้เลขจำนวนเต็ม 2 หลักคือ 23 และ 14 เราสามารถเขียนในอีกรูปแบบดังนี้

$$23 = 2 \cdot 10^1 + 3 \cdot 10^0$$

$$14 = 1 \cdot 10^1 + 4 \cdot 10^0$$

การแบ่งปัญหาย่อย



$$23 * 14 = (2 \cdot 10^1 + 3 \cdot 10^0) * (1 \cdot 10^1 + 4 \cdot 10^0) \quad \text{การรวมคำตอบ}$$

$$23 * 14 = (2 * 1)10^2 + (2 * 4 + 3 * 1)10^1 + (3 * 4)10^0 = 322$$



## Integer multiplication with divide & conquer

ดังนั้นเลขจำนวนเต็ม 2 ตัว คือ X และ Y ขนาด n หลัก สามารถแบ่งปัญหาย่อยได้ครั้งละ 4 เทอม คือ a, b, c และ d สำหรับตัวตั้งและตัวคูณ ดังนี้

$$x \begin{array}{|c|c|} \hline a & b \\ \hline \end{array}$$

$$x = a.10^{n/2} + b$$

$$y \begin{array}{|c|c|} \hline c & d \\ \hline \end{array}$$

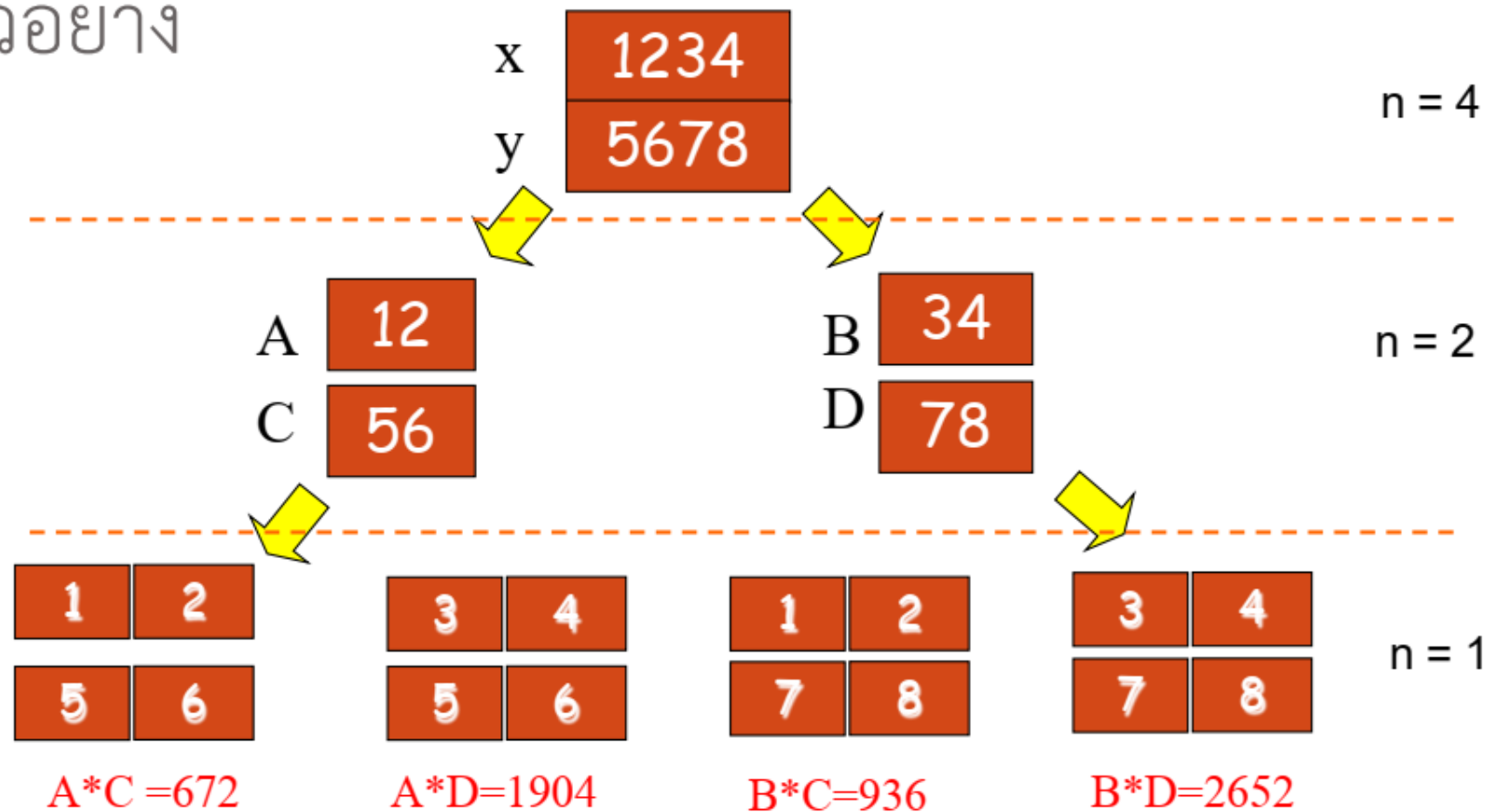
$$y = c.10^{n/2} + d$$

และการรวมคำตอบของปัญหาย่อยคือ

$$x * y = (a.10^{n/2} + b) * (c.10^{n/2} + d)$$

$$x * y = (a * c).10^n + (a * d + b * c).10^{n/2} + b * d$$

# ตัวอย่าง



$$\begin{aligned}
 1234 * 5678 &= (a*c)10^n + (a*d + b*c)10^{n/2} + (b*d) \\
 &= 672*10^4 + (1,904+936)*10^2 + (2,652) \\
 &= 7,006,652
 \end{aligned}$$

## Karatsuba method (1962)

จากหลักการ divide & conquer จะเห็นได้ว่าจำนวนครั้งในการคูณไม่ได้ลดลงไปจากวิธีการ brute force คือ  $n^2$

ต่อมาในปี ค.ศ. 1962 Anatoly Karatsuba จะได้นำเสนอการแยกตัวประกอบในเทอมกลางเพื่อลดจำนวนครั้งในการคูณดังนี้

$$x * y = (a * c).10^n + (a * d + b * c).10^{n/2} + b * d$$

โดยพยายามแยกตัวประกอบในเทอมกลางเพื่อให้เกิดเทอมที่ซ้ำกันออกมา

$$x * y = (a * c).10^n + ((a + b) * (c + d) - a * c - b * d).10^{n/2} + b * d$$

จะเห็นว่าด้วยวิธีการแยกตัวประกอบนี้ จะทำให้สามารถลดจำนวนครั้งในการคูณกัน จาก 4 ครั้งในแต่ละรอบของการแบ่งเหลือเพียง 3 ครั้งเท่านั้น





```

long int multiply (string X, string Y)
{
    int n = makeEqualLength(X, Y);

    if (n == 1) return multiplySingleBit(X, Y);

    int fh = n/2;
    int sh = (n-fh);
    string Xl = X.substr(0, fh);           // a
    string Xr = X.substr(fh, sh);          // b

    string Yl = Y.substr(0, fh);           // c
    string Yr = Y.substr(fh, sh);          // d

    long int P1=multiply (Xl, Yl);          // a*c
    long int P2=multiply (Xr, Yr);          // b*d
    long int P3=multiply (addBitStrings(Xl, Xr),
                           addBitStrings(Yl, Yr));

    return P1*(1<<(2*sh)) + (P3 - P1 - P2) * (1<<sh) + P2;
}

```