

Algorithm Notes

by Ham Kittichet

May 24, 2025

► Table of Contents

บทที่ 1. Divide-and-Conquer	1
► 1.1. ปัญหา Maximum Subarray	1
► 1.2. อัลกอริทึมการคูณเมทริกซ์ของ Strassen	2
► 1.3. ความสัมพันธ์เวียนเกิด	2

บทที่ 1 | Divide-and-Conquer

► 1.1. ปัญหา Maximum Subarray

สมมติเรามีลำดับของจำนวนจริง (array) ชุดหนึ่ง และเราต้องการหาลำดับย่อยที่เรียงติดกันที่มีผลรวมมากที่สุด (จะเรียกว่าเป็น *maximum subarray*) เราอาจจะทำตรง ๆ เลยโดยเช็คทุก ๆ ลำดับย่อยที่เป็นไปได้ซึ่งถ้าลำดับนี้มีจำนวนสมาชิกอยู่ n ตัว จะทำให้ต้องเช็คลำดับย่อยทั้งหมด $\binom{n}{2}$ ชุด จึงต้องใช้เวลา $\Omega(n^2)$

อีกวิธีที่ดีกว่าคือการใช้ recursion โดยเราจะแบ่ง array ที่ได้รับมานี้ออกเป็น 2 subarray (โดยจะเก็บ index ไว้สามตัวคือ *low*, *mid*, และ *high*) ไปเรื่อย ๆ และหา maximum subarray ของ subarray ซ้าย, subarray ขวา, และ maximum subarray ที่ข้ามจุดแบ่ง (crossing subarray) จากนั้นเลือกค่าที่มากที่สุดในสามกรณีนี้

สังเกตว่าเราสามารถหา maximum crossing subarray ของ array A ขนาด n ที่ผ่าน *mid* ได้โดยการหา maximum subarray ของครึ่งซ้ายรวมกับของครึ่งขวา:

(1.1) →

การหา Maximum Crossing Subarray.

```
1 Function findMaxCrossingSubarray( $A$ ,  $low$ ,  $mid$ ,  $high$ )
2    $leftSum = -\infty$ 
3    $sum = 0$ 
4   for  $i = mid$  downto  $low$  do
5      $sum = sum + A[i]$ 
6     if  $sum > leftSum$  then
7        $leftSum = sum$ 
8        $maxLeft = i$ 
9   do similarly for the right, i.e., find maximum  $rightSum$  and  $maxRight$  index
10  return ( $maxLeft$ ,  $maxRight$ ,  $leftSum + rightSum$ )
```

ซึ่งใช้เวลา $\Theta(n)$

ดังนั้นจะได้อัลกอริทึมในการหา maximum subarray โดยการ *divide-and-conquer*:

(1.2)

อัลกอริทึม Maximum Subarray.

```

1 Function findMaxSubarray(A, low, high)
2   if low = high then
3     return (low, high, A[low])
4   mid =  $\lfloor (\textit{low} + \textit{high}) / 2 \rfloor$ 
5   (lefti)i=13 = findMaxSubarray(A, low, mid)
6   (midi)i=13 = findMaxCrossingSubarray(A, low, mid, high)
7   (righti)i=13 = findMaxSubarray(A, mid + 1, high)
8   maxSubarray = choose x ∈ {left, mid, right} with maximum x3
9   return maxSubarray

```

โดยเราจะเรียก findMaxSubarray(*A*, 1, *A.length*) เมื่อต้องการหา maximum subarray ของ *A*

ถ้ากำหนดให้อัลกอริทึมนี้ทำงานได้ในเวลา $T(n)$ ก็จะได้ความสัมพันธ์

(1.3)

$$T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n)$$

(เพราะการหา *left* และ *right* เป็นการเรียกฟังก์ชันเดิมซ้ำ โดยที่ array มีขนาดลดลงครึ่งหนึ่ง ใช้เวลา $T(\lfloor n/2 \rfloor)$, การหา *mid* ใช้เวลา $\Theta(n)$, และที่เหลือทั้งหมดใช้เวลา $\Theta(1)$) โดยในส่วนตัวัด ๆ ไปเราจะแก้ได้ว่าความสัมพันธ์เวียนเกิดนี้มีคำตอบ $T(n) = \Theta(n \lg n)$ ซึ่งเร็วกว่าการทำตรง ๆ

► 1.2. อัลกอริทึมการคูณเมทริกซ์ของ Strassen

► 1.3. ความสัมพันธ์เวียนเกิด