Algorithm Design and Analysis

Week11: Closest Pair of Points

Adisak Supeesun

24 February 2022

Review

- การทำงานของอัลกอริทึม divide and conquer แบ่งเป็น 3 ขั้นตอน ได้แก่

1. Divide: แบ่งปัญหาที่ต้องการจะแก้ ออกเป็นปัญหาย่อยๆ

2. Conquer: แก้ปัญหาย่อยแบบ recursive

3. Combine: นำคำตอบของปัญหาย่อยมาสร้างเป็นคำตอบของปัญหาตั้งต้น

- การคิดเวลาในการทำงานของอัลกอริทึม divide and conquer
  - เขียน recurrence ของเวลาในการทำงานของอัลกอริทึม
  - แก้ recurrence เพื่อหารูปแบบปิด (closed form)

เช่น Mergesort

$O(n)$ { • divide : แบ่ง input ออกเป็น 2 ส่วน ส่วนละ $\frac{n}{2}$

$T(\frac{n}{2})+T(\lceil\frac{n}{2}\rceil)$ { • conquer : เรียงข้อมูลแต่ละส่วนโดยการ recursive

$O(n)$ { • combine : Merge ผลลัพธ์ที่ได้จากการ recursive

ให้ $T(n)$ เป็นเวลาที่ mergesort ใช้ในการเรียงข้อมูล n ตัว

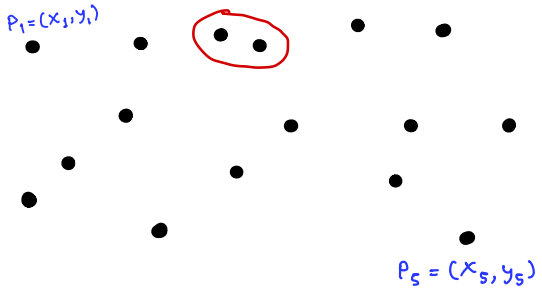
$$T(n) = \begin{cases} 2T(\frac{n}{2}) + O(n) & \text{if } n > 1 \\ O(1) & \text{if } n = 1 \end{cases}$$

ซึ่งเราสามารถแก้ recurrence นี้ด้วยวิธีการใช้ recursion-tree ได้เป็น $T(n) = O(n \log n)$

Closest Pair of Points

ให้ $A = \{p_1, p_2, \ldots, p_n\}$ เป็นเซตของจุด n จุดบนระนาบ 2 มิติ โดยที่จุด p_i ใดๆ ระบุด้วยพิกัด (x_i, y_i)

ต้องการหาระยะระหว่างคู่ของจุดใน A ที่อยู่ใกล้กันที่สุด

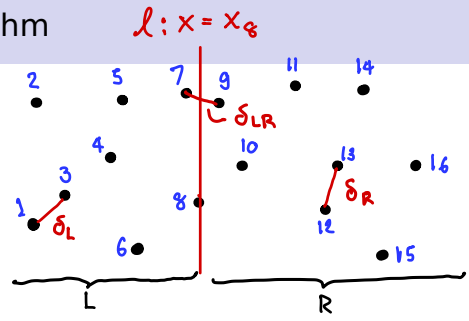


เราสามารถ Brute force ได้โดย
หาระยะห่างระหว่างทุกคู่ของจุด
แล้วตอบคู่ที่ระยะห่างน้อยที่สุด
$\therefore$ เวลาที่ใช้ = #คู่ของจุด $\times O(1)$
$= \binom{n}{2} \times O(1)$
$= O(n^2)$

Note ระยะห่างระหว่างจุด p_i และ p_j ใดๆ คำนวณจาก $d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ — คำนวณได้ใน $O(1)$

"Euclidean distance"

Designing Algorithm



สมมติว่า ค่าพิกัด x และพิกัด y ของทุกจุด แตกต่างกัน

1. Divide: แบ่งจุด ทุกจุด ออกเป็น 2 กลุ่ม $\begin{cases} L \\ R \end{cases}$ กลุ่มละ $\frac{n}{2}$ ตัว
2. Conquer: หาคู่ของจุดที่ใกล้กันที่สุดของกลุ่ม L และ R โดย recursive
   (สมมติได้คำตอบเป็นคู่ของจุดที่ใกล้กัน δ_L และ δ_R ตามลำดับ)
3. Combine: หาคู่ของจุดที่ใกล้กันที่สุดที่อยู่คนละฝั่งของเส้น ℓ
   (สมมติคำตอบที่ได้ เป็นคู่ของจุดที่ห่างกัน δ_{LR})

เปรียบเทียบค่า $\delta_L, \delta_R, \delta_{LR}$ ตอบคู่ที่ห่างกันเพียงกับ ของ 3 ค่าดังกล่าว

ClosestDistance($A = [p_1, p_2, \ldots, p_n]$) // return ระยะห่างระหว่างคู่ของจุดใน A ที่ใกล้กันที่สุด

if $n \leq 3$
  return ระยะห่างระหว่างคู่ของจุดใน P ที่อยู่ใกล้กันที่สุด } $O(1)$
end if

divide {
$A \leftarrow$ เรียงจุดใน A ตามพิกัดในแนวแกน x จากน้อยไปมาก — $O(n \log n)$
$L \leftarrow A[p_1, \ldots, p_{\lfloor \frac{n}{2} \rfloor}]$
$R \leftarrow A[p_{\lfloor \frac{n}{2} \rfloor + 1}, \ldots, p_n]$ } $O(n)$

conquer {
$\delta_L \leftarrow$ ClosestDistance (L) $T(\lfloor \frac{n}{2} \rfloor)$
$\delta_R \leftarrow$ ClosestDistance (R) $T(\lceil \frac{n}{2} \rceil)$

combine {
$\delta_{LR} \leftarrow$ ระยะระหว่างคู่ของจุดใน L และ R ที่อยู่ใกล้กันที่สุด — $O(n^2)$ (เขียนแบบง่ายๆ ก่อน)
return $\min\{\delta_L, \delta_R, \delta_{LR}\}$

Running Times

ให้ $T(n)$ เป็นเวลาที่อัลกอริทึมใช้ในการหาคำตอบ เมื่อ input มี n จุด

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + \underbrace{O(n \log n) + O(n^2)}_{O(n^2)} & \text{if } n > 3 \\ O(1) & \text{if } n \leq 3 \end{cases}$$

สมมติ n เป็นกำลังของ 2 (หารด้วย 2 ลงตัวเรื่อยๆ) จะได้ว่า $T(\lfloor \frac{n}{2} \rfloor) = T(\lceil \frac{n}{2} \rceil) = T(\frac{n}{2})$

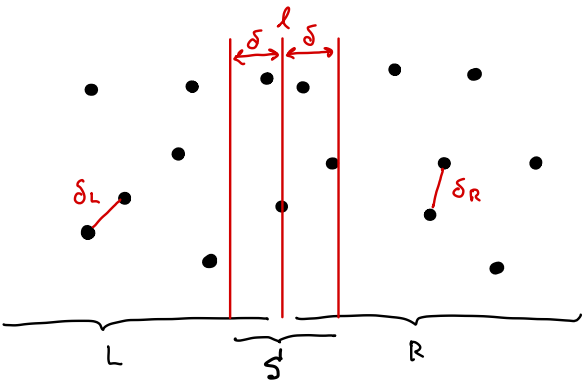
$$\therefore T(n) = \begin{cases} 2T(\frac{n}{2}) + O(n^2) & \text{if } n > 3 \\ O(1) & \text{if } n \leq 3 \end{cases}$$

แต่เวลาส่วนนี้ก็เท่ากับ bruteforce แล้ว ช้าไป!

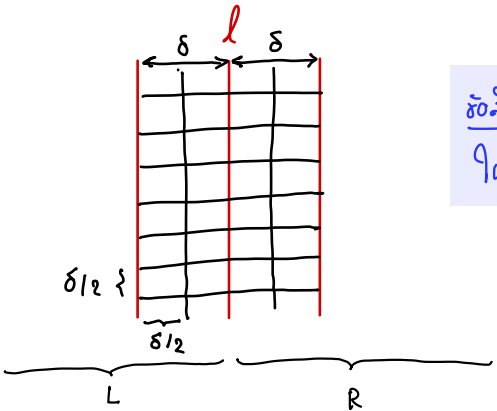
ต้องหาวิธีที่ดีกว่านี้ให้ได้

การหาระยะห่างที่ใกล้ที่สุด ระหว่างคู่ของจุดซึ่งอยู่คนละฝั่ง

ให้ $\delta = \min\{\delta_L, \delta_R\}$ และ S เป็นเซตของจุดที่อยู่ห่างจากเส้นแบ่ง ℓ ไม่เกิน δ



ไอเดีย: เราไม่จำเป็นต้องเช็คระยะของทุกคู่ข้ามฝั่ก เช็คแค่คู่ที่อยู่ใกล้ๆ กับเส้น ℓ ก็พอ



ข้อสังเกต 1

ในแต่ละช่องตารางต้องมีไม่เกิน 1 รูป

มุมแทยงห่างกัน

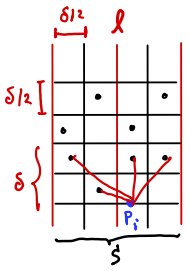
$$\sqrt{\left(\frac{\delta}{2}\right)^2 + \left(\frac{\delta}{2}\right)^2}$$

$$= \frac{\sqrt{2}}{2} \times \delta$$

$$> \delta$$

ต้องหาระยะห่างคู่ของจุดที่ใกล้กันที่สุดในคู่ของ l ที่มีระยะไม่เกิน δ

สำหรับจุด $p_i=(x_i,y_i)$ ใดๆใน S, พิจารณาจุดใน S ที่ค่าพิกัดในแกน y ของมันมากกว่า y_i (จุดที่อยู่เหนือ p_i)



มีแค่จุดใน 2 แถวนี้เท่านั้นที่มีโอกาสเป็นจุดที่ห่างจาก p_i ไม่เกิน δ

จากข้อสังเกต 1 ทุกช่องในตารางมีไม่เกิน 1 จุด จริงๆ

ข้อสังเกต 2

จุดใน S ที่อยู่เหนือ p_i ที่มีโอกาสที่จะอยู่ห่างจาก p_i ไม่เกิน δ มีไม่เกิน 7 จุด

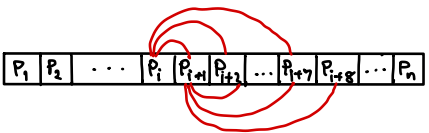
จากข้อสังเกตุ 2 เราสามารถหาระยะห่างระหว่างคู่ของจุดที่อยู่คนละฝั่งของเส้น l ที่มีระยะไม่เกิน δ ได้ดังนี้

- เรียงจุดใน S ตามค่าพิกัดแกน y จากน้อยไปมาก — $O(n \log n)$
- $\delta_{LR} \leftarrow \infty$ — $O(1)$
- for $i = 1, 2, \ldots, n$

ทวน $O(1)$ {

if $\min\{d(p_i,p_{i+1}), d(p_i,p_{i+2}), \ldots, d(p_i,p_{i+7})\} < \delta_{LR}$

$\delta_{LR} \leftarrow \min\{d(p_i,p_{i+1}), d(p_i,p_{i+2}), \ldots, d(p_i,p_{i+7})\}$



ใช้ความยาวแต่ละจุด p_i กับ 7 จุดถัดจากมัน

รวมเวลา $O(n \log n) + O(1) + \underbrace{n \times O(1)}_{O(n)} = O(n \log n)$

ด้วยวิธีการข้างต้น เราสามารถปรับ recurrence ของเวลาการทำงานในสไลด์หน้า 8/23 ได้ดังนี้

$$T(n) = \begin{cases} T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + \underbrace{O(n \log n) + O(n)}_{O(n \log n)} & \text{if } n > 3 \\ O(1) & \text{if } n \leq 3 \end{cases}$$

$O(n \log n)$

ถ้า n เป็นกำลังของ 2 (หารด้วย 2 ลงตัวเรื่อยๆ) จะได้ว่า $T(\lfloor \frac{n}{2} \rfloor) = T(\lceil \frac{n}{2} \rceil) = T(\frac{n}{2})$

$$\therefore T(n) = \begin{cases} 2T(\frac{n}{2}) + O(n \log n) & \text{if } n > 3 \\ O(1) & \text{if } n \leq 3 \end{cases}$$

แก้ recurrence เพื่อหารูปแบบปิดของ $T(n)$

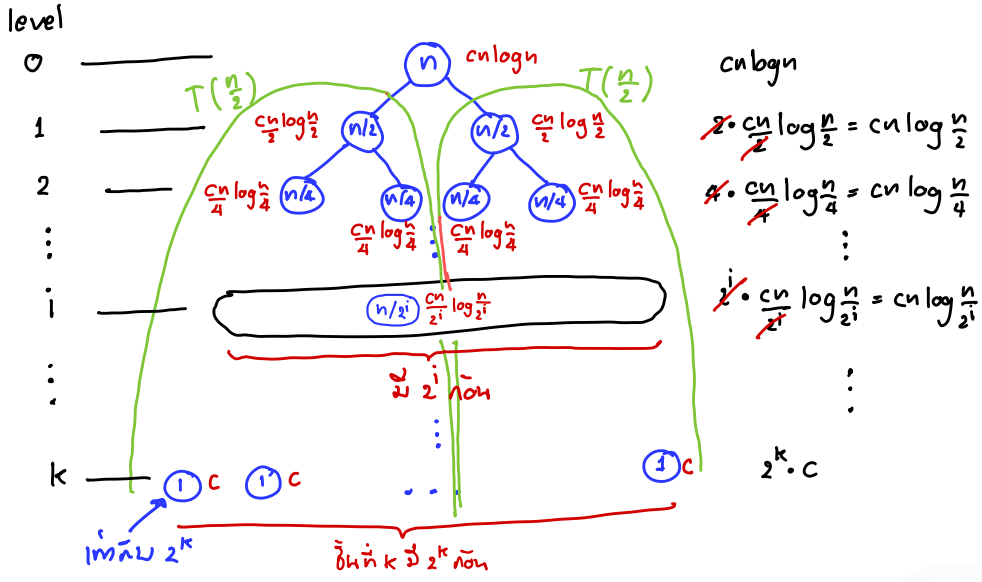
$$T(n) = \begin{cases} 2T(\frac{n}{2}) + O(n\log n) & \text{if } n > 3 \\ O(1) & \text{if } n \leq 3 \end{cases}$$

$O(n\log n)$ → $cn\log n$

$O(1)$ → c

เพื่อความง่าย ตอนแก้ เราจะแทก recursion tree ไปจนถึงชั้นที่ $n = 1$ เลย

(recursion tree จริงๆ ก็ลึกกว่านี้ แต่ทำให้เวลาที่แก้ออกมาจะมากกว่าเดิมเล็กน้อย)


level
0
1
2
i
k
n
cn log n
T(n/2)
T(n/2)
cn/2 log n/2
cn/2 log n/2
n/2
n/2
cn/4 log n/4
cn/4 log n/4
cn/4 log n/4
cn/4 log n/4
n/4
n/4
n/4
n/4
n/2^i
cn/2^i log n/2^i
มี 2^i ตัว
1 c
1 c
1 c
เท่ากับ 2^k
ไปเรื่อยๆ k มี 2^k ตัว
cn log n
2 · cn/2 log n/2 = cn log n/2
4 · cn/4 log n/4 = cn log n/4
2^i · cn/2^i log n/2^i = cn log n/2^i
2^k · c

$$T(n) = \underbrace{\left[\sum_{i=0}^{k-1} cn \log \frac{n}{2^i} \right]}_{\text{ผลบวกของชั้นที่ 0 ถึง } k-1} + \underbrace{2^k \cdot c}_{\text{ชั้นที่ } k}$$

$$= cn \sum_{i=0}^{k-1} (\log n - \underbrace{\log 2^i}_{\log \frac{a}{b} = \log a - \log b}) + 2^k \cdot c$$

$$= cn \left(\sum_{i=0}^{k-1} \log n - \sum_{i=0}^{k-1} \log 2^i \right) + 2^k \cdot c$$

$$= cn \left(k \log n - \underbrace{\sum_{i=0}^{k-1} i \cdot \log 2}_{\log a^k = k \log a} \right) + 2^k \cdot c$$

หา k

จาก $\frac{n}{2^k} = 1$

$\Rightarrow n = 2^k$

$\Rightarrow \log_2 n = \log_2 2^k$

$= k \cdot \log_2 2$ $(\log_b a^c = c \log_b a)$

$= k$ $(\log_b b = 1)$

$\therefore k = \log_2 n$

$$= cn\left(k\log n - \log 2 \cdot \underbrace{\sum_{i=0}^{k-1} i}\right) + 2^k \cdot c$$

$$0+1+2+\dots+(k-1) = \frac{k(k-1)}{2} = \frac{k^2-k}{2}$$

$$= cn\left(k\log n - \log 2 \cdot \left(\frac{k^2-k}{2}\right)\right) + 2^k \cdot c$$

แทนค่า $k = \log_2 n$ จะได้

$$= cn\left((\log_2 n \cdot \log n) - \log 2 \cdot \left(\frac{(\log_2 n)^2 - \log_2 n}{2}\right)\right) + 2^{\log_2 n} \cdot c$$

$\log 2 \cdot \log_2 n$

$\because \log_b a = \frac{\log_k a}{\log_k b}$

จะได้ $\log_2 n = \frac{\log n}{\log 2}$

$\therefore \log n = \log 2 \cdot \log_2 n$

n (เพราะ $b^{\log_b a} = a$)

$$= cn\left(\log 2 \cdot (\log_2 n)^2 - \frac{\log 2 \cdot (\log_2 n)^2}{2} + \frac{\log 2 \cdot \log_2 n}{2}\right) + cn$$

$$= \frac{cn \log 2}{2} \cdot (\log_2 n)^2 + \frac{cn \log 2}{2} \cdot \log_2 n + cn$$

main

$\frac{c \log 2}{2} \cdot n$ ← main x n

$$= O(n(\log_2 n)^2) + O(n \log_2 n) + O(n)$$

$$= O(n(\log n)^2)$$

สังเกตว่าการหา δ_{LR} ตามวิธีการในสไลด์หน้า 11/23 นั้นจะต้อง

* เรียงลูกใน S ตามค่าพิกัดแกน y ก่อน * ← ส่วนนี้ถูกทำทุกครั้งที่ recursive เท่ากับ recurrece เราจึงได้ค่า $O(n \log n)$

จากนั้นจึงวนลูปหาค่า δ_{LR}

* เราสามารถปรับปรุงการทำงานให้เร็วขึ้นได้ โดยการเรียงลูกตามค่าพิกัดแกน y * แค่ครั้งเดียว (เรียงแค่ตอนหลังจาก input ไม่ต้องไปเรียงทุกครั้งที่ recursive)

นอกฟังก์ชัน:

รับ input: $A \leftarrow [p_1, \ldots, p_n]$

$A_x \leftarrow$ เรียงจุดใน A ตามพิกัดในแนวแกน x จากน้อยไปมาก — $O(n \log n)$

$A_y \leftarrow$ เรียงจุดใน A ตามพิกัดในแนวแกน y จากน้อยไปมาก — $O(n \log n)$

ClosestDistance(A_x, A_y) // A_x คือ list ของจุดที่เรียงตามพิกัดแกน x จากน้อยไปมาก
// A_y คือ list ของจุดที่เรียงตามพิกัดแกน y จากน้อยไปมาก

$O(1)$

if $n \leq 3$
  return ระยะห่างระหว่างคู่ของจุดใน A_x ที่อยู่ใกล้กันที่สุด
end if

divide

$\ell \leftarrow$ พิกัดในแนวแกน x ของจุดที่ $\lfloor \frac{n}{2} \rfloor$ ของ A_x — $O(1)$

$L_x \leftarrow$ เลือกจุดจาก A_x เอาเฉพาะจุดที่อยู่ฝั่งซ้ายของ ℓ ($A_x[1, \ldots, \lfloor \frac{n}{2} \rfloor]$)

$R_x \leftarrow$ เลือกจุดจาก A_x เอาเฉพาะจุดที่อยู่ฝั่งขวาของ ℓ ($A_x[\lfloor \frac{n}{2} \rfloor + 1, \ldots, n]$)

$L_y \leftarrow$ เลือกจุดจาก A_y เอาเฉพาะจุดที่อยู่ฝั่งซ้ายของ ℓ ($A_y[1, \ldots, \lfloor \frac{n}{2} \rfloor]$)

$R_y \leftarrow$ เลือกจุดจาก A_y เอาเฉพาะจุดที่อยู่ฝั่งขวาของ ℓ ($A_y[\lfloor \frac{n}{2} \rfloor + 1, \ldots, n]$)

$O(n)$

conquer

$\delta_L \leftarrow$ ClosestDistance (L_x, L_y) — $T(\frac{n}{2})$

$\delta_R \leftarrow$ ClosestDistance (R_x, R_y) — $T(\frac{n}{2})$

combine

$\delta \leftarrow \min\{\delta_L, \delta_R\}$ — $O(1)$

$S \leftarrow$ เลือกจุดจาก A_y เอาเฉพาะจุดที่อยู่ห่างจาก ℓ ไม่เกิน δ // S คือ list ของจุดที่ห่างตามเส้น l ไม่เกิน δ โดยเรียงตามค่าพิกัดแกน y ตามน้อยไปมาก

$\delta_{LR} \leftarrow \infty$

for each p_i in S

  $\delta_{LR} \leftarrow \min\{\delta_{LR}, d(p_i, p_{i+1}), d(p_i, p_{i+2}), \ldots d(p_i, p_{i+7})\}$ } $O(n)$

end for

return $\min\{\delta, \delta_{LR}\}$

สรุปเวลาการทำงาน

- นอกฟังก์ชัน $O(n \log n)$
- ในฟังก์ชัน

$$T(n) = \begin{cases} 2T(\frac{n}{2}) + O(n) & \text{if } n > 3 \\ O(1) & \text{if } n \leq 3 \end{cases}$$

ลดลงเมื่อเทียบกับหน้า 22/23

เหมือนของ merge sort

$\therefore$ แก้ได้เป็น $O(n \log n)$

รวมเป็น $O(n \log n)$

Logarithm and Geometric Series

- **Logarithm** : เราจะกล่าวว่า $x = \log_b n$ ก็ต่อเมื่อ $b^x = n$

  คุณสมบัติของ logarithm

  1. $\log_b b = 1$
  2. $\log_b nm = \log_b n + \log_b m$
  3. $\log_b \frac{n}{m} = \log_b n - \log_b m$
  4. $\log_b n^c = c \log_b n$
  5. $\log_b n = \frac{log_k n}{log_k b}$
  6. $b^{\log_b n} = n$

- **Geometric Series** : สำหรับจำนวนจริง $r \neq 1$ และจำนวนเต็ม $n \geqslant 0$ ใดๆ

  $$r^0 + r^1 + r^2 + \ldots + r^n = \frac{r^{n+1}-1}{r-1}$$