

CONVEX HULL

CH Graham เกรแฮม
กลุ่ม ไออิช

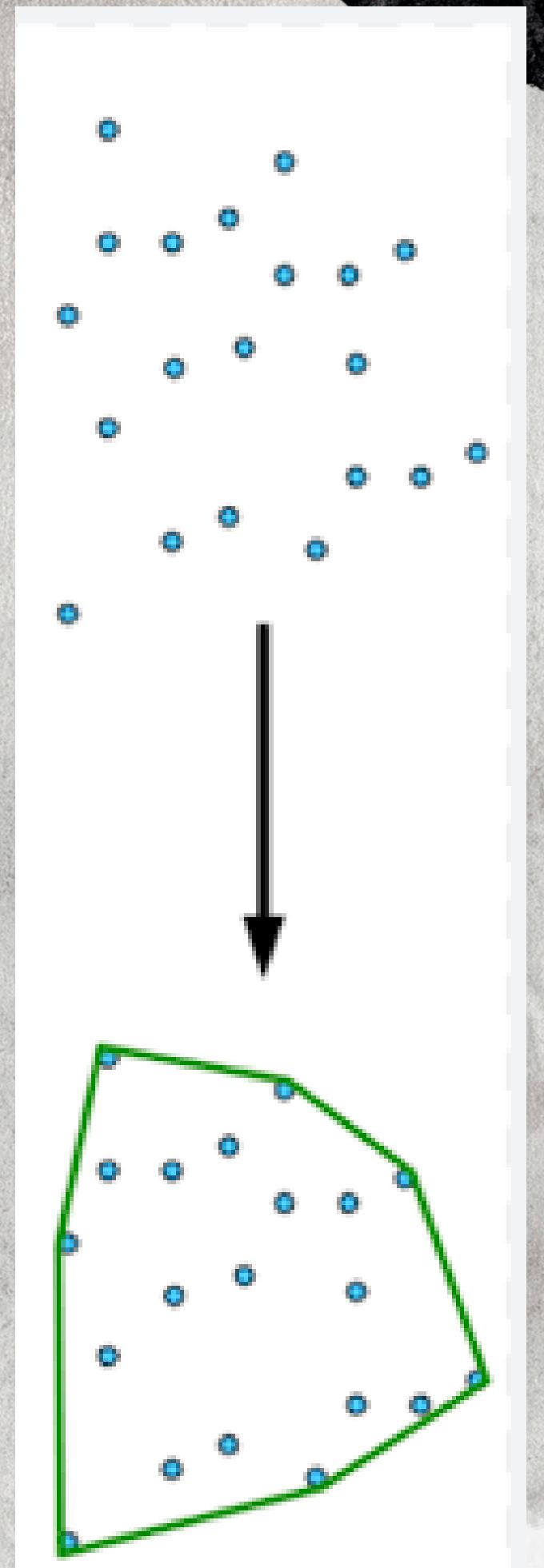
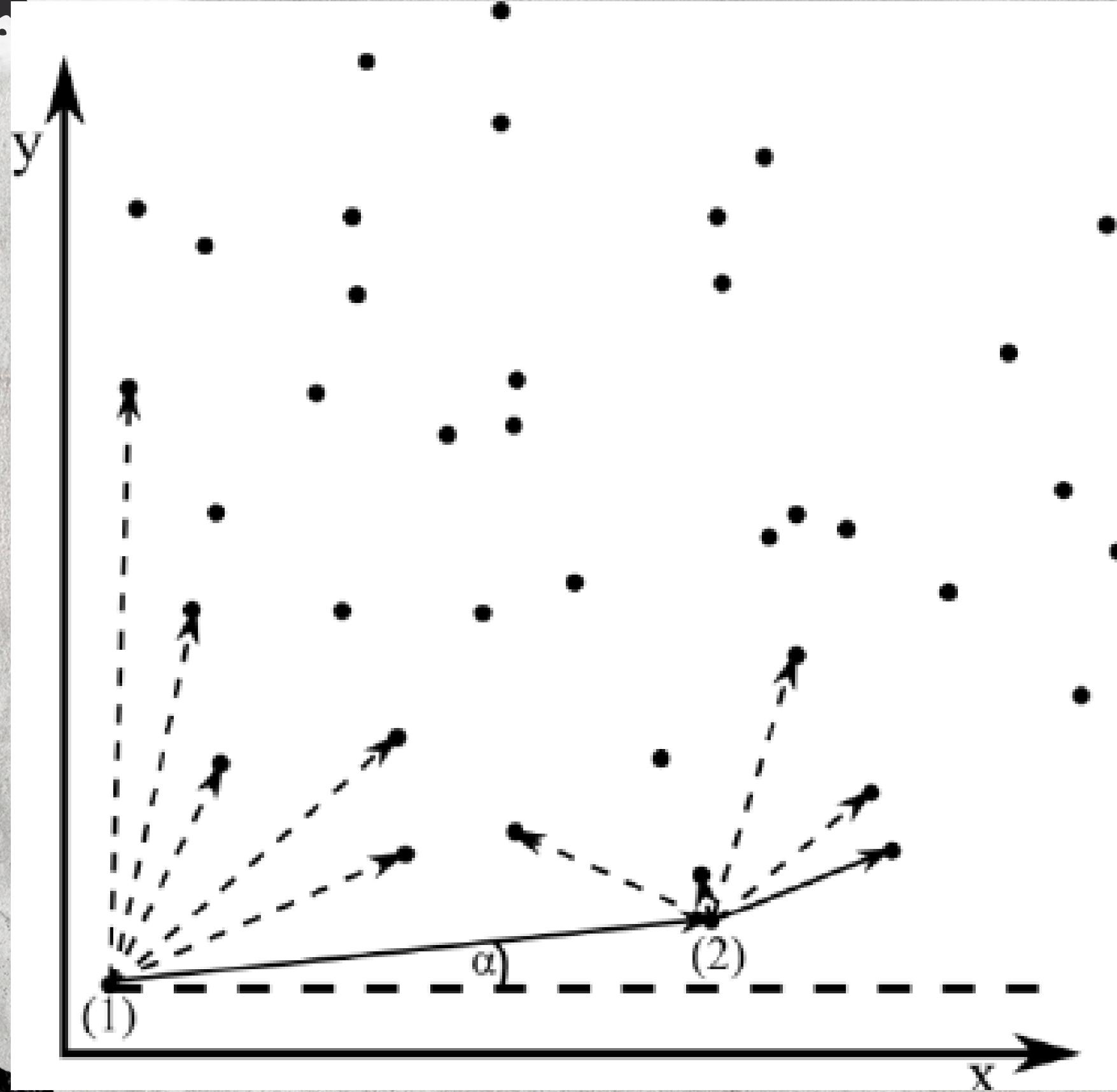
หัวหน้ากลุ่ม 1. นายคชา สวัสดิพิศาล 6710301054
สมาชิก 2. นายเจตนา 6710301022
สมาชิก 3. นายปั่นพงศ์ บุสุวรรณ 6710301023

GRAHAM'S SCAN คืออะไร?

CH Graham หรือ Graham's Scan Algorithm เป็นอัลกอริธึมที่ใช้ในการหาขอบเขตนอกสุด (Convex Hull) ของชุดข้อมูลจุดในระนาบ 2 มิติ (2D plane) โดยมีขั้นตอนที่มีประสิทธิภาพในเชิงเวลา $O(N \log N)$

Graham's Scan ບໍາຍມໃຊ້

1. การประมวลผลภาพ (Image Processing)
2. การວິເຄຣະໜີເສີງພື້ນຖານ (Geospatial Analysis)
3. การຈັດກຸລຸມຂ້ອມູລ (Data Clustering)
4. ມູນຍບຕີແລະການນຳກາງ (Robotics and Navigation)



Data structure ในการเก็บข้อมูล

- 1. Stack (สแต็ค)
 - Data Structure ที่สำคัญที่สุดที่ใช้ในการเก็บข้อมูลคือ Stack ซึ่งเป็นโครงสร้างข้อมูลแบบ LIFO (Last In First Out) หมายความว่า ข้อมูลที่ถูกเพิ่มเข้ามาล่าสุดจะถูกดึงออกมาก่อน
 - ในขั้นตอนของการหาขอบเขตวนกสุด (Convex Hull) เราจะใช้ Stack เพื่อเก็บจุดที่เป็นส่วนหนึ่งของขอบเขต
 - Stack นี้จะถูกใช้ในการเก็บจุดที่เป็น vertices ของ Convex Hull ที่เกิดจากการทดสอบว่าแต่ละจุดใหม่ที่เพิ่มเข้ามาทำให้ขอบเขตเปลี่ยนแปลงไปหรือไม่
- 2. Array (อาร์เรย์)
 - Array หรือ List ใช้ในการเก็บข้อมูลกึ่งหมดของจุดต่างๆ ที่เราใช้ในการหาขอบเขตวนกสุด (Convex Hull)
 - ในที่นี้จะมีการใช้ อาร์เรย์ P [] เพื่อเก็บจุดต่างๆ ที่เป็น input หรือจุดที่ใช้ในการคำนวณหา Convex Hull
 - อาร์เรย์นี้จะเก็บจุดในลักษณะของพิกัด (x , y) ที่ใช้ในการคำนวณหาว่าจุดไหนเป็นส่วนหนึ่งของ Convex Hull

3. Integer Variables

- ตัวแปร i : ใช้เพื่อเก็บตำแหน่งของจุดในอาร์เรย์ $P[]$ ที่จะพิจารณาถัดไป
- ตัวแปร N : ขนาดของอาร์เรย์ $P[]$ ที่บ่งบอกจำนวนของจุดที่ต้องพิจารณา

4. Comparison Functions (ฟังก์ชันเปรียบเทียบ)

- ฟังก์ชัน $ccw(A, B, C)$: ใช้เพื่อเปรียบเทียบว่า จุดสามจุด A, B, C อยู่ในลำดับทวนเข็มนาฬิกาหรือไม่ (Counterclockwise) ซึ่งจะใช้ในการตรวจสอบการหมุนของเส้นตรงที่ลากจากจุด A ไป B ไป C
- ฟังก์ชันนี้ช่วยในการตัดสินใจว่าจะเก็บหรือไม่เก็บจุดในสแต็ค

ข้อดี

1. ประสิทธิภาพดี ($O(n \log n)$):

- ใช้เวลาประมาณผลเป็น $O(n \log n)$ ซึ่งมีประสิทธิภาพสูงสำหรับการคำนวณ Convex Hull โดยเฉพาะเมื่อเปรียบเทียบกับวิธีที่ซับซ้อนกว่า

2. เหมาะกับชุดข้อมูลขนาดใหญ่:

- การจัดเรียงข้อมูลในขั้นตอนแรกช่วยลดความซับซ้อนของการคำนวณในขั้นต่อไป

3. ง่ายต่อการนำไปใช้งาน :

- โครงสร้างของอัลกอริทึมชัดเจนและสามารถนำไปเขียนโค้ดได้ง่าย

4. ผลลัพธ์เชื่อถือได้ :

- การคำนวณแบบเชิงเรขาคณิตช่วยให้ได้ผลลัพธ์ที่ถูกต้องและน่าเชื่อถือ

ข้อเสีย

1. ต้องจัดเรียงข้อมูลก่อน :

- การเรียงลำดับข้อมูลก่อนเริ่มคำนวณอาจเพิ่มเวลาในการขั้นตอนแรก โดยเฉพาะถ้ามีข้อมูลจำนวนมาก

2. ไม่เหมาะสมกับข้อมูลบางประเภท :

- ถ้าชุดข้อมูลมีจุดจำนวนมากที่เรียงตัวอยู่ในแนวเดียวกันหรือใกล้เคียงกัน อัลกอริทึมอาจไม่ประสิทธิภาพเท่ากับค่าวันเดียว

3. ต้องการการคำนวณทางเรขาคณิตที่แม่นยำ :

- การตรวจสอบ "turning direction" ของเส้นอาจเกิดความผิดพลาดถ้าการคำนวณทางเรขาคณิตมีความคลาดเคลื่อน เช่น จากข้อจำกัดของการคำนวณเลขศูนย์

4. ไม่เหมาะสมกับการขยายไปในมิติที่สูงขึ้น :

- Graham's Scan ถูกออกแบบมาสำหรับปัญหาในมิติที่สองเท่านั้น การนำไปใช้กับปัญหาในมิติที่สูงขึ้นต้องการการปรับปรุงเพิ่มเติมหรือใช้อัลกอริทึมอื่นแทน เช่น Quickhull หรือ Chan's Algorithm

เป้าหมายของ Algorithm คือการหาขอบเขตนอกสุด (Convex Hull) คือการค้นหาชุดของจุดที่สร้างขอบเขตนอกสุด ของกลุ่มจุดที่อยู่ในระนาบ 2 มิติ (2D plane) ซึ่งสามารถอธิบายได้ดังนี้:

1. การหาขอบเขตนอกสุด (Convex Hull)

- เป้าหมายหลักของอัลกอริธึมนี้คือการ หาเส้นขอบที่เชื่อมต่อจุดในกลุ่มจุดที่เป็น "ขอบเขต" ซึ่งเส้นขอบนี้จะต้องเป็นเส้นที่ ห่อหุ้มจุดทั้งหมด ในกลุ่มจุดที่ให้มา และไม่สามารถเส้นผ่านจุดในกลุ่มได้
- อัลกอริธึมจะทำการคัดเลือกเฉพาะจุดที่อยู่ บนขอบเขตนอกสุด โดยจุดที่อยู่ภายในขอบเขตนี้จะไม่ได้ถูกเลือก

2. การหาจุดที่อยู่บนขอบเขต

- เป้าหมายของอัลกอริธึมคือการระบุจุดที่สร้างขอบเขตภายนอก (Convex Hull) ซึ่งจะประกอบด้วยจุดที่ ห่อหุ้มจุดทั้งหมด โดยไม่ให้จุดอื่นๆ อยู่ภายในขอบเขตนี้
- ในการนี้จะใช้โครงสร้างข้อมูล stack ในการเก็บจุดที่ถูกเลือก เพื่อสร้างขอบเขตที่เป็นรูปหลายเหลี่ยมที่ห่อหุ้มจุดทั้งหมด

3. การหาขอบเขตที่มีความคงที่ (Convexity)

- ขอบเขตที่ได้จากการหาจุดเหล่านี้จะต้องเป็น Convex (คงที่) , หมายความว่า ถ้าคุณ เชื่อมจุดที่อยู่ในขอบเขตแล้ว จะไม่มีจุดใดที่หลุดออกจากขอบเขต
- อัลกอริธึมเช่น Graham's Scan หรือ Jarvis's March จะใช้การทดสอบ ลำดับ ทวนเข็มนาฬิกา (Counterclockwise) หรือ ตามเข็มนาฬิกา (Clockwise) เพื่อยืนยันว่าเส้นเชื่อมจากจุดหนึ่งไปยังจุดถัดไปจะเป็นส่วนหนึ่งของขอบเขตหรือไม่

4. การทำงานที่มีประสิทธิภาพ

- เป้าหมายที่สำคัญคือการ ทำให้การคำนวน Convex Hull มีความเร็วและมีประสิทธิภาพ สูง ซึ่งทำได้โดยการใช้โครงสร้างข้อมูลที่เหมาะสม เช่น stack เพื่อช่วยในการเพิ่มและลบ จุดออกจากขอบเขตอย่างมีระเบียบ
- การเลือกใช้ อัลกอริธึมที่มีเวลาในการคำนวนเป็น $O(N \log N)$ เช่น Graham's Scan ทำให้สามารถหาขอบเขตบนอุปกรณ์สุดได้เร็วขึ้นเมื่อจำนวนจุดในกลุ่มมาก

ขั้นตอนของ **GRAHAM'S SCAN**

- ขั้นตอนที่ 1: หาจุด Pivot
- จุดที่ ต่ำที่สุด (ถ้าซ้ำ ให้เลือก น้อยที่สุด)
- 2: เรียงลำดับจุด :
- เรียงลำดับจุดกึ่งหมดตามมุมโปลาเมื่อเทียบกับ Pivot

ขั้นตอนของ

GRAHAM'S SCAN

- ขั้นตอนที่ 3: สร้าง Convex Hull ด้วย Stack
- ใส่จุดแรก 3 จุดใน Stack
- ໄລ່ພິຈານາແຕ່ລະຈຸດ :
- ຂາກ "ເລື້ອງຊ້າຍ": ใส่ຈຸດໃນ Stack
- ຂາກ "ເລື້ອງຂວາ": เອາຈຸດບນສຸດຂອງ Stack ອອກ
- ກຳຈັນມົດຖຸກຈຸດ

4. โค้ดอธิบาย โค้ด PYTHON (GRAHAM'S SCAN)

cpp

```
// Initial setup
push P[N-1], P[0], P[1] into stack S;

while (i < N) { // i = 2, N = size of P

    if (ccw(S.size() - 2], S[S.size() - 1], P[i])) {
        // Push P[i] into stack and increase i
        push P[i] into stack;
        i++;
    } else {
        // Pop the top element from the stack
        pop from S;
    }
}

// S is the result
```

- อธิบายโค้ด:
- Pivot Point: หาจุดเริ่มต้นที่ต่ำที่สุด
- Polar Angle Sorting: เรียงลำดับจุดตามมุมกีวัดจาก Pivot
- ใช้ Stack:
- ตรวจสอบ "เลี้ยวซ้าย" ด้วย Cross Product:
- ถ้าเลี้ยวขวา (ค่า Cross Product): นำจุดล่าสุดออกจาก Stack
- ถ้าเลี้ยวซ้าย: เพิ่มจุดใน Stack

**THANK
YOU**