โจทย์สำหรับแข่งขันเขียนโปรแกรม

จุดประสงค์ของโจทย์

- เพื่อทดสอบความสามารถในการแก้ปัญหา
  - การตีความโจทย์
  - การออกแบบอัลกอริทึมที่มีประสิทธิภาพ

ระดับที่เหมาะสม

- ผู้เริ่มต้น
  - การอ่านอินพุตอาจต้องการการคิดแก้ปัญหาเชิงอัลกอริทึมก็ได้
- ระดับกลาง
  - การใช้ต้นไม้ การเขียน (implement) ลิงก์ลิสต์
- ระดับสูง + (ที่มี STL ให้ใช้)
  - การใช้ balanced binary search tree เป็นเรื่องพื้นฐาน (ใช้จาก library โดยตรง) ไม่สมควรวัดโดยตรง

ข้อสมมติสำหรับวันนี้

- ผู้เข้าแข่งขันเขียนโปรแกรมได้ระดับหนึ่งแล้ว
- อาจารย์เลือก "ระดับ" ที่สนใจได้เอง

ประเภทโจทย์

- Batch
- Reactive
- Output-only

องค์ประกอบ

- ตัวปัญหา
  - เงื่อนไข/ขอบเขต
- เนื้อเรื่อง
- ข้อมูลชุดทดสอบ

ตัวปัญหา

- ปัญหาเชิงอัลกอริทึม

ตัวปัญหา

- จะ**ไม่**เน้น (อาจมีข้อมูลทดสอบบ้าง แต่น้อย)
  - การอ่านอินพุต/การเขียนเอาท์พุต/การจัดรูปแบบ

ตัวปัญหา

- จะ**ไม่**เน้น (อาจมีข้อมูลทดสอบบ้าง แต่น้อย)
  - ตัวความรู้เชิงเทคนิค เช่น
    - อัลกอริทึมที่ซับซ้อน, รู้แล้วทำได้โดยตรงเลย

ตัวปัญหา

- จะ**ไม่**เน้น (อาจมีข้อมูลทดสอบบ้าง แต่น้อย)
  - รายละเอียดปลีกย่อยอื่น ๆ
    - เช่น ขอบเขตชนิดข้อมูล

อัลกอริทึมที่ทำงานได้จริง

- ต้องเขียนได้ และทำงานได้จริงภายในเวลาที่กำหนด
  - โดยปกติ 1 วินาที ทำงานพื้นฐานได้ประมาณ 1,000,000,000 หน่วย

ตัวปัญหาที่ดี?

- ”เรารู้ว่าปัญหานี้ดี เวลาเราเห็น”
- เป้าหมายของช่วงเช้า:
  - จะ ”ทดลอง” อธิบายลักษณะทั่วไปของปัญหาที่ผมคิดว่าดี

ลักษณะของตัวปัญหาที่ดี (1)

- ใช้ความรู้พื้นฐานน้อย
  - ในการทำความเข้าใจ
  - ในการแก้ปัญหา

ความรู้พื้นฐาน (1)

- คณิตศาสตร์พื้นฐาน
  - concept ทั่วไป
  - ทฤษฎีกราฟ (กราฟ เส้นทาง การระบายสี)
  - เรขาคณิตพื้นฐาน (เส้น จุด วงกลม เส้นตัดกัน)
  - ทฤษฎีจำนวนพื้นฐาน (จำนวนเฉพาะ)

ความรู้พื้นฐาน (2)

- อัลกอริทึม/โครงสร้างข้อมูล
  - การจัดเรียง/การสืบค้น
    - binary search trees, augmented binary search trees
    - hashing,
    - union-find
  - เทคนิคการออกแบบ:
    - อัลกอริทึมเชิงละโมบ, การโปรแกรมเชิงพลวัติ, การแบ่งแยกแล้วเอาชนะ

ความรู้พื้นฐาน (3)

- อัลกอริทึม/โครงสร้างข้อมูล
  - อัลกอริทึมบนกราฟ
    - การค้นในกราฟ, เส้นทางสั้นสุด, ปัญหาการไหล
  - อัลกอริทึมเชิงเรขาคณิต
    - convex hull, sweep-line technique
- (ดู IOI Sylabus)

ลักษณะของตัวปัญหาที่ดี (1)

- ใช้ความรู้พื้นฐานน้อย
- แต่ใช้การพลิกแพลง
  - นำความรู้หลายเรื่องมาประยุกต์ใช้

ลักษณะของตัวปัญหาที่ดี (1)

- ใช้ความรู้พื้นฐานน้อย
- แต่ใช้การพลิกแพลงที่ **ไม่ธรรมดา**

"ไม่ธรรมดา"

- อัลกอริทึม (พื้นฐาน) ที่คาดไม่ถึง
  - "ปัญหาลักษณะนี้ไม่น่าใช้อัลกอริทึมกลุ่มนี้"
  - ต้องใช้การตีความและวิเคราะห์โจทย์

"ไม่ธรรมดา"

- เงื่อนไขที่คาดไม่ถึง
  - ใช้วิธีตรงไปตรงมาจะไม่ได้ผลลัพธ์ในเวลาที่กำหนด
  - แต่เงื่อนไขบางอย่างเพิ่มโครงสร้างให้กับปัญหา
  - ทำให้สามารถแก้ปัญหานั้นได้

ตัวอย่าง "ไม่ธรรมดา"

- ข้อ mravi ในการแข่งขัน Croatian Olympiad in Informatics 2004
  - อาจจะ "ไม่ธรรมดา" มากเกินไป

ลักษณะของตัวปัญหาที่ดี (2)

- มีคำตอบที่เป็นไปได้มาก
  - มีประสิทธิภาพแตกต่างกัน
  - (เพื่อการวัดผลที่ "แยก" ผู้เข้าแข่งขันได้)

ตัวปัญหา:เงื่อนไข/ขอบเขต

- จำเป็นมากในการออกแบบอัลกอริทึม
  - ความถูกต้อง
  - ประสิทธิภาพ

ตัวปัญหา:เงื่อนไข/ขอบเขต

- จำเป็นมากในการออกแบบอัลกอริทึม
- และยังเป็น "คำใบ้" ของประสิทธิภาพที่ต้องการ
  - ย้ำ: 1,000,000,000 คำสั่งย่อย
    - $n = 20$; อัลกอริทึมทำงานใน $O(2^n)$
    - $n = 1{,}000$; อัลกอริทึมทำงานใน $O(n^2)$
    - $n = 100{,}000$; อัลกอริทึมทำงานใน $O(n \log n)$
    - $n = 10{,}000{,}000$; อัลกอริทึมทำงานใน $O(n)$

ตัวอย่างขอบเขต

- N <= 100,000 นอกจากนี้ ใน 30% ของข้อมูลชุดทดสอบ N <= 1,000
  - หมายความว่า "ถ้าใช้อัลกอริทึม $O(N \log N)$ จะมีโอกาสได้คะแนนเต็ม แต่ถ้าใช้ $O(N^2)$ จะได้คะแนน 30%"

เนื้อเรื่อง

- เพื่อความเพลิดเพลิน/น่าสนใจ
  - ประโยชน์ด้านวัฒนธรรม
- วัดความสามารถในการ "ตี" ปัญหา

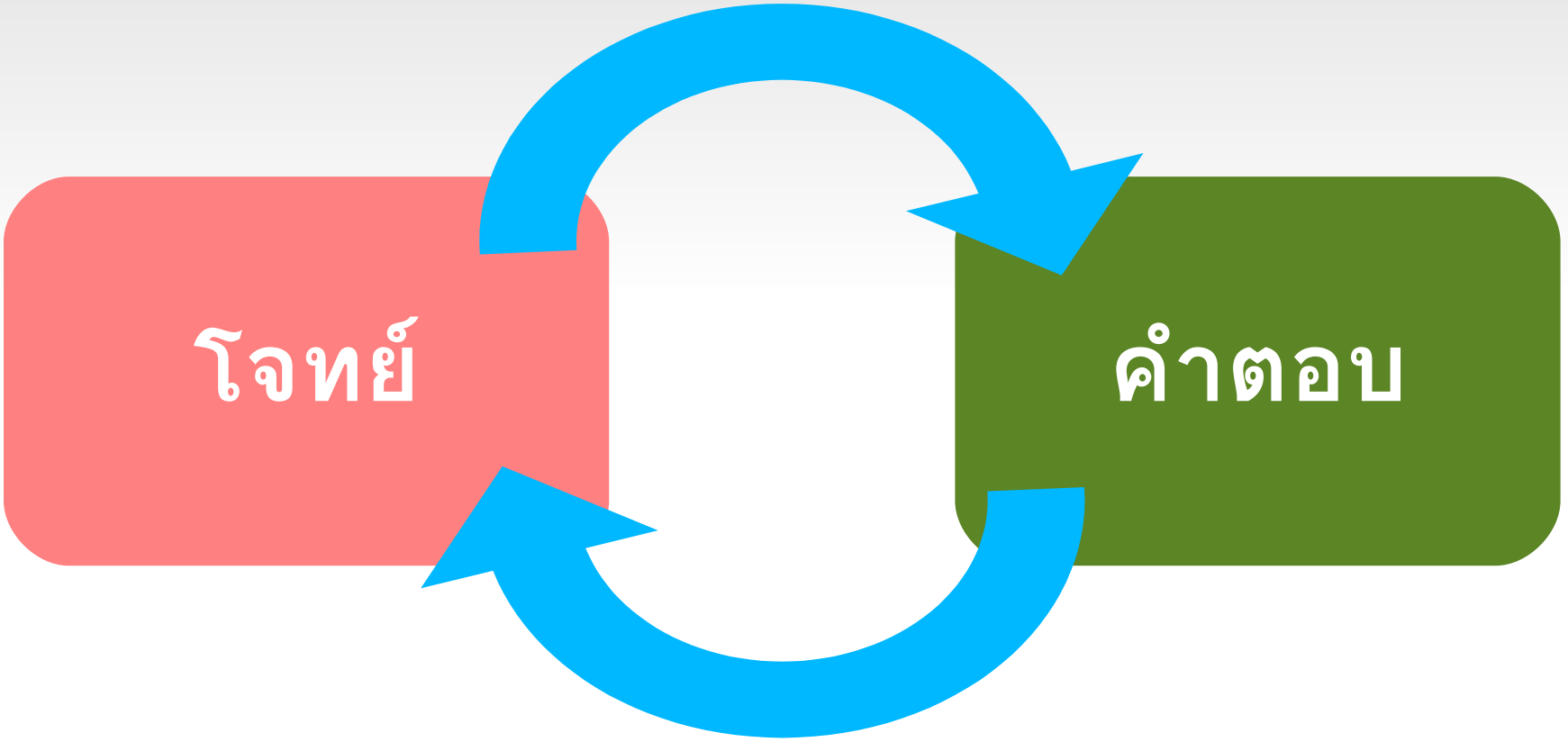
ข้อมูลชุดทดสอบ

- วัดความถูกต้องของโปรแกรม
- วัดประสิทธิภาพของโปรแกรม
  - แยกอัลกอริทึมที่ประสิทธิภาพแตกต่างกันได้

ตัวอย่างโจทย์

- Wires and Switches
  - IOI'95, NL
- Aliens
  - IOI'07, Croatia
- The Tournament
  - Polish Olympiad in Informatics 03-04, stage II
- Birthday
  - IOI'05, Poland

การออกแบบโจทย์



การออกแบบโจทย์

- จากปัญหาไปคำตอบ
  - ค่อนข้างยาก หลายครั้งหาคำตอบไม่ได้
  - บางครั้งคำตอบที่ดีก็ง่ายเกินไป
- จากคำตอบไปปัญหา
  - บางครั้งได้ปัญหาที่ไม่น่าสนใจ หรือเป็นปัญหาที่ ”ปลอม” มากเกินไป

ตัวอย่างสมมติ

- ปัญหาไปคำตอบ
  - โจทย์การพับกระดาษ
- คำตอบไปปัญหา
  - เริ่มจาก binary search
  - เริ่มจาก augmented binary search tree

คำตอบไปปัญหา

- เริ่มจากคำตอบ: binary search
  - เราทราบอะไรเกี่ยวกับ binary search บ้าง?

การค้นหาแบบทวิภาค

- ข้อมูลเรียงลำดับ/เข้าถึงแบบ random
- ต้องสามารถเปรียบเทียบข้อมูลสองตัวได้
- ใช้เวลา O(log n) สำหรับกรณีที่มีข้อมูล n ตัว

การค้นหาแบบทวิภาค: เงื่อนไข

- รายการข้อมูลเรียงเป็นเส้น/เข้าถึงแบบ random
- เมื่อพิจารณาข้อมูลค้น x กับข้อมูล y ใด ๆ ในลำดับ สามารถระบุได้ว่า x จะต้องอยู่ในลำดับก่อนหน้าหรือถัดจาก y
- ใช้เวลา O(log n) สำหรับกรณีที่มีข้อมูล n ตัว

จุดระเบิด: การค้นหาแบบทวิภาค

- ข้อมูลในรายการ
  - จะใช้อะไร? ตัวเลข สตริง เซต รูปภาพ?
- รายการข้อมูลเรียงเป็นเส้น/เข้าถึงแบบ random
  - ไม่เรียงได้หรือไม่?, ไม่เป็นเส้นได้หรือไม่?, ไม่สามารถเข้าถึงแบบ random ได้หรือไม่?
- เปรียบเทียบ x กับ y ใด ๆ ในลำดับ
  - เปรียบเทียบอย่างไร? ไม่เปรียบเทียบได้หรือไม่? เปรียบเทียบผิดพลาด?

การบิดดัดเงื่อนไข (1)

- ข้อมูล n ตัว, ไม่เรียงทั้งหมด
  - ต้องค้นใน O(n) แน่ ๆ

การบิดดัดเงื่อนไข (2)

- เรียงบางส่วน? เช่น
  - มีข้อมูลบางตัวไม่เรียง
    - ข้อมูล k ตัวใน n ตัว ถ้าลบออกไปจะทำให้ข้อมูลเรียงลำดับ
  - โดยทั่วไปไม่เรียง แต่ว่าค่อนข้างเรียง????
    - ระบุค่า k
    - ข้อมูลตัวที่ i จะต้องน้อยกว่าข้อมูลตัวที่ i+k เสมอ

- **ปัญหา:** ยังใช้ binary search ได้อยู่หรือไม่?

การบิดดัดเงื่อนไข (3)

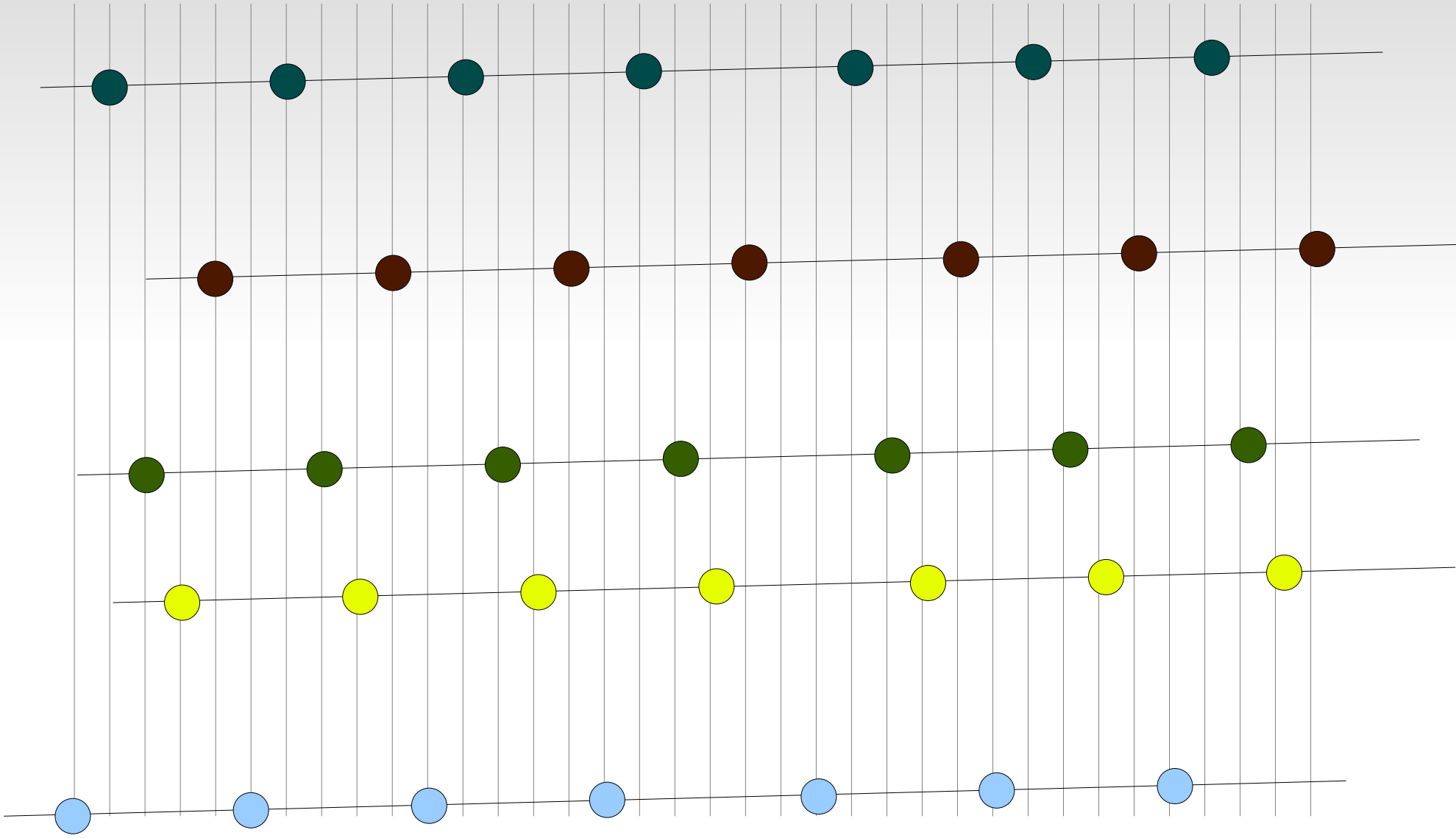
- ถ้าตัวที่ i ต้องน้อยกว่า (หรือเท่ากับ) ตัวที่ i+k เสมอ
  - ถ้านำข้อมูลตัวที่ 1, 1+k, 1+2k, 1+3k, ... มา จะเป็นรายการที่เรียงลำดับ
  - binary search บนข้อมูลตัวแทนเหล่านี้ได้

- **ปัญหา:** แล้วจะหาข้อมูลที่ต้องการอย่างไร?

คิดตัวอย่างอันตราย (1)

- มีรายการที่เรียงลำดับ k รายการ ที่ไม่เกี่ยวข้องกันเลย
  - นำมาผสานกันเป็นรายการเดียว
    - ตัวที่ 1, 1+k, 1+2k,... มาจากรายการแรก
    - ตัวที่ 2, 2+k, 2+2k,... มาจากรายการที่สอง
    - ตัวที่ 3, 3+k, 3+2k,... มาจากรายการที่สาม
  - ถ้าเริ่ม search จากรายการแรก จะไม่ได้ข้อมูลเกี่ยวกับข้อมูลในรายการอื่น ๆ เลย

รูปตัวอย่างอันตราย



คิดตัวอย่างอันตราย (2)

- ทางออก: เพิ่มเงื่อนไขเกี่ยวกับตัวที่ i กับตัวอื่น ๆ ที่ไม่ใช่ i+k
  - ข้อมูลตัวที่ i จะน้อยกว่า i+k, i+k+1, i+k+2, ...
  - เมื่อ binary search ในตัวแทนแล้ว ใช้ sequential search ในกลุ่มสองกลุ่ม (จำนวน 2k ตัว) ที่ติดกับตัวแทนนั้นได้
  - เวลาการทำงาน $O(\log (n/k) + k)$

- **ข้อสังเกต:** ถ้าไม่เพิ่มเงื่อนไข แต่ใช้วิธีที่อธิบายมา โปรแกรมจะให้ผลลัพธ์ที่ผิดพลาด

คิดตัวอย่างอันตราย (3)

- ทางออก 2 (ได้จากการคิดตัวอย่างอันตราย)
  - ไม่เพิ่มเงื่อนไข
  - ทำ binary search ในทุก ๆ กลุ่ม (รวม k รอบ)
    - binary search บนรายการของข้อมูลที่ i, i+k, i+2k,... สำหรับ i=1,2,...,k
  - เวลาการทำงาน O(k log (n/k))

คิดตัวอย่างอันตราย (4)

- ทางออก 3
  - ไม่ต้องเพิ่มเงื่อนไข
  - เรียงข้อมูลก่อน binary search !!!
  - เวลาการทำงาน สำหรับการค้น q ครั้ง:
    - O(n log n + q log n)

- **ข้อสังเกต:** ถ้านี่ไม่ใช่คำตอบที่ต้องการ ต้องกำหนดขอบเขตในโจทย์ให้ไม่สามารถทำได้

การบิดดัดเงื่อนไข (4)

- สมมติว่าเราเลือกที่จะไม่ปรับเงื่อนไข
- เวลาในการค้น q ครั้ง
  - วิธีที่ 2: $O(qk \log (n/k))$
  - วิธีที่ 3: $O((n + q) \log n)$
- ต้องเลือกขอบเขตที่วิธีที่ 2 ใช้ได้ แต่วิธีที่ 3 ไม่ผ่าน
- **หมายเหตุ**: ยังไม่นับเวลาการอ่านข้อมูล

การบิดดัดเงื่อนไข (5)

- บางครั้งการใช้เวลาไม่สามารถแยกความแตกต่างได้
  - 10000 คำสั่ง กับ 20000 คำสั่ง แยกด้วยเวลาไม่ได้
- สามารถบังคับให้ใช้ library (reactive task)
  - ในการเปรียบเทียบข้อมูล ห้ามอ่านข้อมูลโดยตรง
  - กำหนดจำนวนครั้งที่มากที่สุดในการเรียกใช้ library ได้

ข้อมูลชุดทดสอบ (1)

- ข้อมูลเล็ก ทดสอบความผิดพลาดทั่วไป
- ข้อมูลชุดทดสอบ สำหรับความผิดพลาดที่เอาวิธี 1 มาใช้
- ต้องป้องกันวิธีที่ 3

ข้อมูลชุดทดสอบ (2)

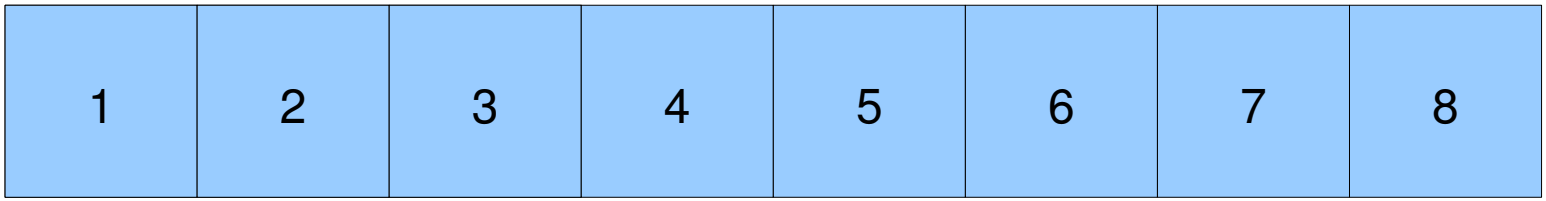
- ออกแบบคะแนน
  - เขียนทำงานได้ทั่วไป: 10%
    - linear search / quadratic-time sort / etc.
  - ใช้วิธีที่ 1 (วิธีที่ผิด): 30%
  - ใช้วิธีที่ 3: 50%
  - ใช้วิธีที่ 2: 100%

ข้อมูลชุดทดสอบ (3)

- ชุดพื้นฐาน 10%
- ชุดง่าย (วิธีที่ 1 ก็ทำงานได้) 20%
- ชุดเล็ก (สำหรับวิธี 3 แต่ถ้าใช้วิธีที่ 1 ต้องทำงานผิด) 20%
- ชุดใหญ่ (แบบต่าง ๆ ขนาดต่าง ๆ ถ้าใช้วิธีที่ 3 ต้องทำงานไม่ทัน และวิธีที่ 1 ต้องทำงานผิด) 50%

ปัญหาไปคำตอบ: การพับกระดาษ

- การพับกระดาษ
  - พับอะไร?
- เอากระดาษยาว ๆ มาพับ
  - กระดาษยาว $1 \times N$ มาพับตามรอยช่อง
  - พับแล้วได้อะไร?



พับกระดาษ

- ทราบผลลัพธ์ของการพับ ถามว่าพับอย่างไร?
  - ต้องอธิบาย/นิยาม
    - ผลลัพธ์ของการพับ
    - ขั้นตอนการพับ

ผลลัพธ์ของการพับ

- ผลลัพธ์ของการพับ
  - พับจนกลายเป็นชิ้นกระดาษจตุรัส 1 x 1
  - ใส่เลขให้กับแต่ละชิ้นจตุรัส
  - ได้ลำดับของชิ้นย่อยเมื่ออ่านจากบนลงล่าง
    - ชิ้นย่อยกลับบน/ล่าง มองเป็นชิ้นเดียวกัน

กระบวนการพับ

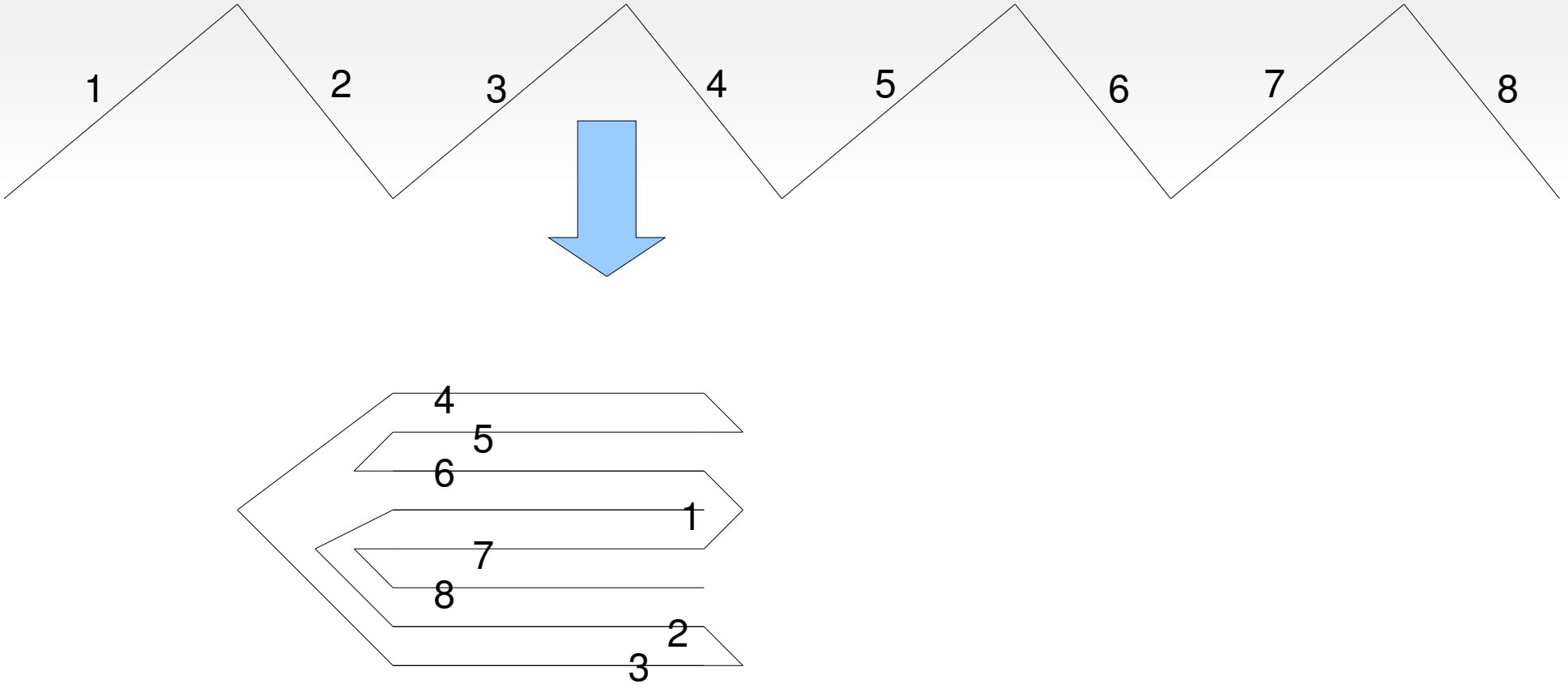
- พับอย่างไร?
  - ระบุวิธีการพับยาก
- เปลี่ยนเป็นถามว่าพับได้หรือไม่?

โครงร่างปัญหาพับกระดาษ

- ได้โครงร่างของปัญหา
  - ทราบลำดับของหมายเลขจากแผ่นบนสุด
  - ถามว่า ถ้านำกระดาษยาว $1 \times N$ มีหมายเลข 1-N มาพับตามรอยช่องจนเป็นจตุรัส จะพับได้ตามต้องการหรือไม่?

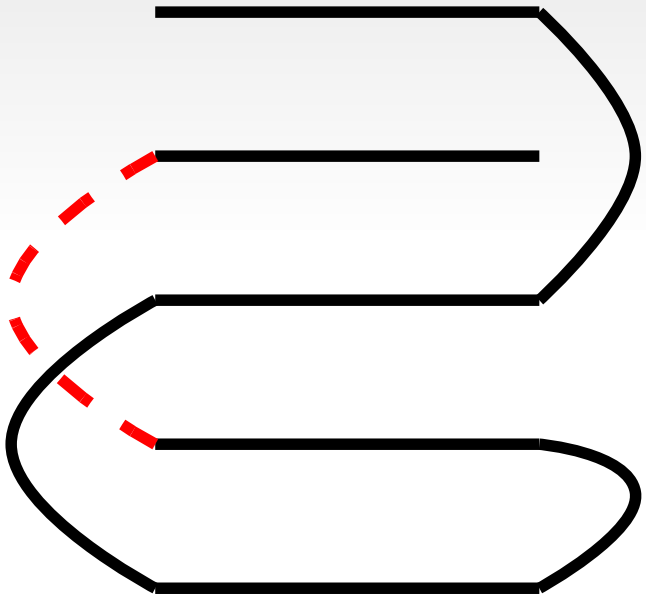
พับกระดาษ: คำตอบ

- ทราบ "ผลลัพธ์" ของการพับ



ตัวอย่างที่พับไม่ได้

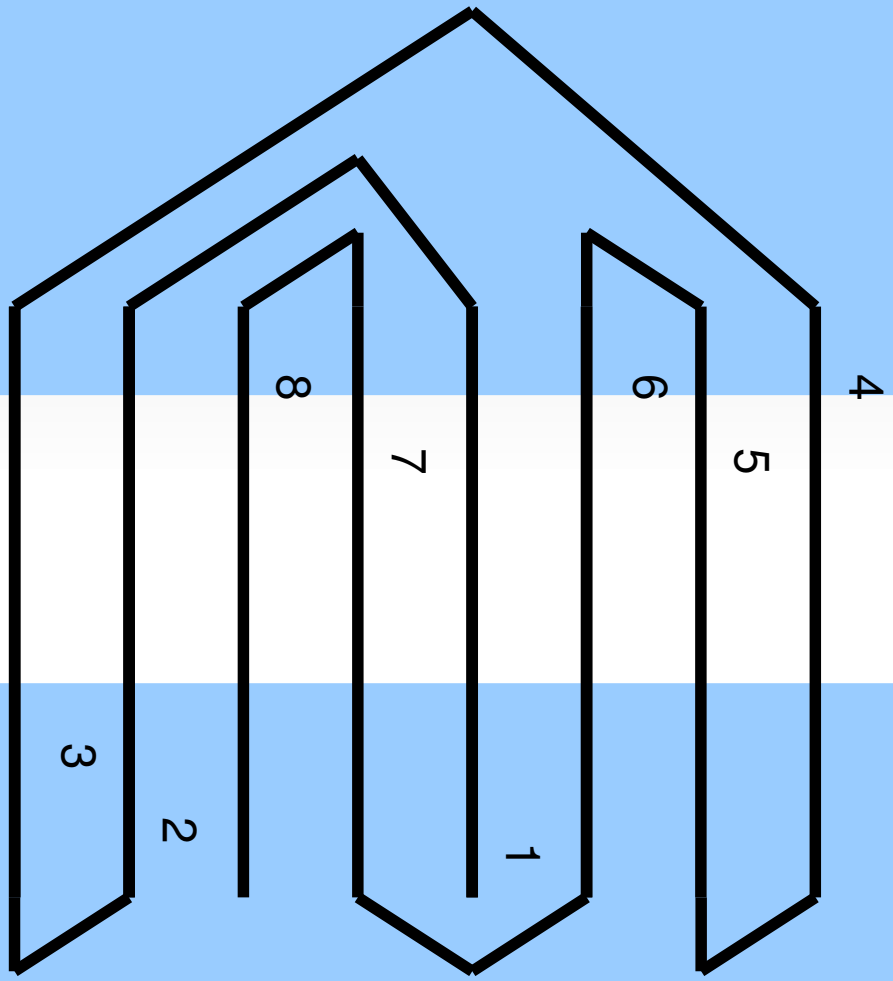
- ถ้าลำดับเป็น
  - 1
  - 5
  - 2
  - 4
  - 3



พับกระดาษ: โครงสร้างของคำตอบ

- กระดาษต้องต่อเนื่องกัน
  - แผ่น 1 ต่อกับแผ่น 2 ต่อกับแผ่น 3,4,...
- มุมที่พับจะต้องอยู่ขอบซ้าย-ขวาสลับกัน
  - 1 -> 2 พับทางด้านซ้าย, 2 -> 3 พับทางด้านขวา, ...

พับกระดาษ: คำตอบ



เป็นปัญหาตรวจโครงสร้างวงเล็บ

สองปัญหา

พับกระดาษ: กรณีพิเศษ

- หลายรูปแบบแม้จะ "พับได้" ตามเงื่อนไขที่เราตรวจสอบ
  - กระดาษต่อกัน/พับซ้ายขวา
- แต่อาจไม่สามารถพับได้จริง ๆ ด้วยมือ
  - ซ้อนเข้าซ้อนออกหลายชั้นมาก ๆ
- **ต้องระบุในโจทย์ให้ชัดเจนว่าเครื่องพับพิเศษ**

พับกระดาษ: โจทย์

- มีกองกระดาษ N แผ่น แต่ละแผ่นระบุหมายเลข
- ทราบลำดับของหมายเลขจากแผ่นบนสุด
- ถามว่า
  - ถ้านำกระดาษยาว $1 \times N$ มีหมายเลข 1-N มาพับด้วย "เครื่องพับพิเศษ" ตามรอยช่องจนเป็นจตุรัส แล้วตัดขอบให้กลายเป็นกระดาษ N แผ่น
  - จะทำให้เป็นกองกระดาษแรกได้หรือไม่

พับกระดาษ: เนื้อเรื่อง

- มีกระดาษยาว 1 x N นำมาพับด้วยเครื่องพิเศษ
- ตัดขอบเป็นกระดาษ N แผ่น
- ลมพัดปลิว แล้วรีบเก็บมา
- อยากทราบว่าเก็บผิดหรือไม่?
  - ถ้าดูลำดับแล้วพับ-ตัดไม่ได้ แสดงว่าเก็บผิดแน่นอน