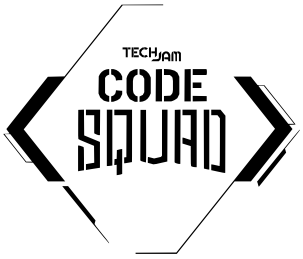
บทนำ



TECHJAM 2018 เป็นการแข่งขันที่จัดขึ้นโดย KBTG เพื่อเฟ้นหาสุดยอดขุนพลแห่งอนาคตด้าน *Coding*, *Data Science* และ *Design* ที่ผู้เข้าร่วมแข่งขันเปรียบเสมือนขุนพลจากทั่วประเทศที่จะนำทักษะความสามารถทางเทคโนโลยีและการออกแบบ ซึ่งเป็นเครื่องมือสำคัญของยุคสมัยแห่งอนาคต มาประชันกันเพื่อหาสุดยอดฝีมือ

ในการแข่งขันของกลุ่ม Code Squad นั้นเราต้องการค้นหาสุดยอดฝีมือด้านการเขียนโปรแกรมและพัฒนาซอฟต์แวร์ที่มีความสามารถรอบด้าน เราเชื่อว่านักพัฒนาซอฟต์แวร์ที่ดีนอกจากจำเป็นจะต้องมีทักษะการเขียนโปรแกรมที่เก่งกาจแล้ว ยังต้องมีความสามารถที่จะเข้าใจและวิเคราะห์ปัญหาในเชิงนามธรรมและระดับโครงสร้างได้ และยังต้องสามารถเลือกใช้วิธีแก้ปัญหาที่เหมาะสมกับเงื่อนไขและตัวแปรข้อจำกัดต่าง ๆ ได้ดีอีกด้วย

ในการแข่งขัน Code Squad นั้น เราจึงออกแบบโจทย์ให้มีความหลากหลาย ท้าทาย และอาจไม่เคยพบเห็นที่ใดมาก่อน โจทย์แต่ละข้อจะวัดทักษะการเขียนโปรแกรมหรือความรู้ที่เกี่ยวกับการแก้ปัญหาเชิงคำนวณด้วยคอมพิวเตอร์ ตลอดจนไปถึงการนำทักษะและความรู้ดังกล่าวมาแก้ปัญหาที่สลับซับซ้อนมากขึ้น ในขณะเดียวกันนี้เราพยายามปรับสมดุลโจทย์ให้เข้าใจได้ไม่ยาก แต่ยังคงไว้ซึ่งความเข้มข้นของการไขโจทย์ไว้อย่างครบเครื่อง

ทีมงานได้คัดเลือกโจทย์ที่ใช้แข่งขันของกลุ่ม Code Squad จากงาน **TECHJAM 2018** นำมารวบรวมและเรียบเรียงไว้ในเอกสารฉบับนี้ เพื่อเป็นประโยชน์แก่นักเรียน นักศึกษา และบุคคลทั่วไปที่สนใจเรื่องการพัฒนาซอฟต์แวร์และการศึกษาที่เกี่ยวกับวิทยาการคำนวณและวิทยาการคอมพิวเตอร์ อันเป็นแรงบันดาลใจให้นักเรียน นักศึกษา นักเขียนโปรแกรมและผู้ที่สนใจในเทคโนโลยีได้ศึกษาค้นคว้าความรู้ต่อไป

หวังว่าผู้อ่านจะสนุกและเรียนรู้ไปพร้อมกับการแก้ปัญหาโจทย์ปัญหาที่ท้าทายเหล่านี้

—— ทีมงาน Code Squad, **TECHJAM 2018**

♦ โจทย์ที่ใช้แข่งขันมีลักษณะอย่างไร?

โจทย์ที่ใช้แข่งขันในกลุ่ม Code Squad นั้นแบ่งออกได้เป็นสามประเภทใหญ่ ๆ คือ

1. **Quickfire** เป็นโจทย์ที่ต้องใช้ไหวพริบแล้วความรวดเร็วในการตอบคำถาม ภายในเวลา 10–60 วินาที ผู้เข้าแข่งขัน*ไม่*สามารถใช้คอมพิวเตอร์เพื่อช่วยแก้ปัญหาเหล่านี้ได้

2. **Ponder** เป็นโจทย์ที่ต้องใช้เวลาในการพิจารณาคำถามและค้นหาคำตอบภายใน 5 นาที ผู้แข่งแข่งขันสามารถใช้คอมพิวเตอร์หรือเทคนิคใด ๆ ก็ได้เพื่อช่วยแก้ปัญหาเหล่านี้ได้ หากผู้เข้าแข่งขันสามารถค้นหาคำตอบได้ถูกต้องและรวดเร็วกว่าคนอื่น ๆ ก็จะได้คะแนนสูงขึ้นตามไปด้วย

3. **Coding** เป็นโจทย์ปัญหาเชิงอัลกอริทึมที่ผู้เข้าแข่งขันต้องเขียนโปรแกรมเพื่อช่วยค้นหาคำตอบอย่างมีประสิทธิภาพ ตามสเปกที่กำหนด

♦ โจทย์เหล่านี้มีเฉลยหรือไม่?

ถึงแม้ว่าโจทย์เหล่านี้จะมีคำตอบที่ถูกต้องชัดเจนอยู่แล้วทุกข้อ (เว้นคำถามข้อสุดท้าย) แต่ในขณะนี้ทางทีมงานยังไม่มีแผนที่จะจัดและเผยแพร่เฉลยของโจทย์เหล่านี้อย่างเป็นทางการ

อย่างไรก็ดีทางทีมงานมีความยินดีเป็นอย่างยิ่ง หากผู้อ่านท่านใดเลือกที่จะนำเสนอวิธีที่ใช้ไขปัญหาโจทย์ข้อต่าง ๆ ในรูปแบบของบทความหรือสื่ออื่นใด อันจะเป็นประโยชน์แก่สาธารณะได้

© สงวนลิขสิทธิ์ 2561–2562 กสิกร บิซิเนส–เทคโนโลยี กรุ๊ป
ตามพระราชบัญญัติลิขสิทธิ์ พุทธศักราช 2537

เอกสารฉบับนี้ถูกเผยแพร่ภายใต้สัญญาอนุญาต Creative Commons Attribution–NonCommercial 4.0 International ซึ่งอนุญาตให้ทำซ้ำ แจกจ่าย หรือแสดงและนำเสนอเอกสารฉบับนี้ และสร้างงานดัดแปลงจากเอกสารฉบับนี้ โดยต้องให้เครดิตและแสดงที่มาและไม่ใช้เพื่อการค้า

อ่านเงื่อนไขฉบับเต็มได้ที่ https://creativecommons.org/licenses/by-nc/4.0/

พัฒนาโดยใช้ XeLaTeX เมื่อ 2019-05-11 เวลา 00:02 (เวอร์ชัน v1.1)
ติดต่อสอบถามหรือติดตามรายละเอียดเพิ่มเติมได้ที่ https://techjam.tech

หมวดที่ 1
โจทย์ Quickfire

โจทย์ Quickfire เป็นโจทย์ประเภทถามเร็ว-ตอบเร็ว ผู้เข้าแข่งขันมีเวลา 10–60 วินาที ที่จะตอบคำถามให้ถูกต้อง โดยไม่สามารถใช้คอมพิวเตอร์เพื่อช่วยหาคำตอบได้

หมายเหตุ โจทย์บางข้อที่ปรากฏอาจมีลักษณะเป็นคำถามปรนัย (มีตัวเลือก) เนื่องจากเป็นข้อจำกัดของระบบในการแข่งขันระดับ Audition

คำถามที่ 1. จงพิจารณาโปรแกรมดังต่อไปนี้

```
function mystery_1(A[0 ... n-1]):
    # A คือ 0-indexed array ของจำนวนเต็ม
    n := A.length()
    for i := 0 to n-2 do:
        if A[i] > A[i+1] then:
            swap A[i] with A[i+1]
        end
    end
    return A
end
```

จากโปรแกรมข้างต้นนี้ ประพจน์ใดกล่าวถูกต้องบ้าง?

☐ **ประพจน์ P:** มีค่าค่าหนึ่งใน output array ซึ่งเมื่อลบออกจาก array ดังกล่าว จะทำให้ array ผลลัพธ์เป็น array ที่เรียงลำดับ

☐ **ประพจน์ Q:** สำหรับ input array A ใด ๆ ที่มีขนาด N จะได้ว่า `f(A)[N-1] = max(A)`

☐ **ประพจน์ R:** สำหรับ input array A ใด ๆ ที่มีขนาด N จะได้ว่า `f(A)[0] = min(A)`

คำถามที่ 2. ให้พิจารณาลำดับจำนวน Fibonacci ที่มีความยาวเป็นอนันต์ จงหาอัตราส่วนของจำนวนเลขคู่ต่อจำนวนเลขคี่ในลำดับจำนวนนี้

คำถามที่ 3. ธนาคารกสิกรจัดประเพณีวิ่งควายที่จังหวัดชลบุรี มีควายร่วมเข้าแข่งขันวิ่งจำนวน 66 ตัว แต่ภายในงานมีลูให้ควายวิ่งได้แค่รอบละ 6 ตัวเท่านั้น

หากธนาคารต้องการหาควายที่วิ่งเร็วที่สุด 1 ตัว เราจะต้องนำควายมาจัดวิ่งแข่งกันทั้งหมดอย่างน้อยกี่รอบ?[1]

[1] **เงื่อนไข** สมมติว่าควายวิ่งด้วยอัตราเร็วเท่าเดิมตลอดทุกรอบ และเราไม่สามารถใช้นาฬิกาจับเวลาได้ แต่สามารถเปรียบเทียบได้ว่าในการแข่งขันรอบหนึ่ง ควายตัวใดเข้าเส้นชัยก่อนหรือหลังได้

คำถามที่ 4. ธนาคารกสิกรจัดประเพณีวิ่งควายที่จังหวัดชลบุรี มีควายร่วมเข้าแข่งขันวิ่งจำนวน 45 ตัว แต่ภายในงานมีลูให้ควายวิ่งได้แค่รอบละ 5 ตัวเท่านั้น

หากธนาคารต้องการหาควายที่วิ่งเร็วที่สุดและช้าที่สุด อย่างละ 1 ตัว เราจะต้องนำควายมาจัดวิ่งแข่งกันทั้งหมดอย่างน้อยกี่รอบ?

คำถามที่ 5. กำหนดให้มีต้นไม้ค้นหาแบบทวิภาค (Binary search tree) ที่ว่างเปล่าอยู่ต้นหนึ่ง จากนั้นเรา insert ข้อมูลจำนวนเต็ม 9, 2, 3, 4, 18, 11, 20 เข้าไปในต้นไม้นี้ตามลำดับ โดยที่ไม่เกิดกระบวนการ tree rebalancing ขึ้น

จงหาความสูง (height) ของต้นไม้ผลลัพธ์ (**หมายเหตุ:** กำหนดให้ Binary search tree ที่มี node เดียวมีความสูงเท่ากับ 0)

คำถามที่ 6. สมมติว่ามี array ของจำนวนเต็มที่เรียงลำดับแล้ว ขั้นตอนวิธีใดต่อไปนี้กระทำได้เร็วที่สุด?

- ○ ค้นหาตัวเลข 42 ว่าอยู่ใน array หรือไม่
- ○ นับว่ามีจำนวนตัวเลขที่แตกต่างกันภายใน array ทั้งหมดกี่ตัว
- ○ หาค่าเฉลี่ยเลขคณิต (Arithmetic mean) ของตัวเลขทั้งหมดใน array
- ○ นับว่ามีเลขคู่ทั้งหมดกี่ตัวใน array
- ○ ทุกคำตอบข้างต้นใช้เวลาเป็น linear เท่ากันหมด ดังนั้นจึงใช้เวลาพอ ๆ กัน

คำถามที่ 7. จงพิจารณาโปรแกรมดังต่อไปนี้ที่เขียนขึ้นเพื่อนับว่า มีจำนวนคู่กี่จำนวนในบรรดาจำนวนตั้งแต่จำนวนเต็ม m จนถึงจำนวนเต็ม n (สมมติว่า $m \leq n$) โปรแกรมนี้จะให้ผลลัพธ์ที่ถูกต้องภายใต้เงื่อนไขใดของ input บ้าง?

```
1  function count_even(m, n):
2      # m, n คือจำนวนเต็มซึ่ง m ⩽ n
3      count := 0
4      i := m
5      while i ⩽ n do:
6          count := count + 1
7          i := i + 2
8      end
9      return count
10 end
```

- ☐ กรณีที่ m และ n เป็นจำนวนคู่
- ☐ กรณีที่ m เป็นจำนวนคู่ n เป็นจำนวนคี่
- ☐ กรณีที่ m เป็นจำนวนคี่ n เป็นจำนวนคู่
- ☐ กรณีที่ m และ n เป็นจำนวนคี่

คำถามที่ 8. ธนาคารกสิกรไทยสาขาหนึ่งเปิดให้บริการตั้งแต่เวลา 8:30 จนถึง 15:30 ของวันเดียวกัน อยากทราบว่าในช่วงเวลาดังกล่าวจะมีเหตุการณ์ที่เข็มสั้นและเข็มยาวของนาฬิกาทำมุมตั้งฉากกันพอดีทั้งหมดกี่ครั้ง?[2]

[2] **เงื่อนไข** กำหนดให้เข็มสั้นและเข็มยาวขยับอย่างต่อเนื่องตลอดเวลา

คำถามที่ 9. จงพิจารณาโปรแกรมดังต่อไปนี้

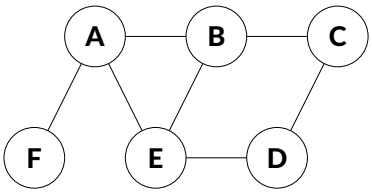
```
1  function mystery_9(I):
2     # I คือ input character stream
3     S := empty stack
4     for each character c in stream I do:
5        if c == '(' then:
6           S.push('(')
7        elseif c == ')' and S ≠ empty then:
8           S.pop()
9        else:
10          return "unbalanced"
11    return "balanced"
12 end
```

ค่า input ในข้อใดต่อไปนี้บ้างที่ทำให้โปรแกรมข้างต้นให้ output เป็น `"balanced"`?

☐ `I = "()()()())"` ☐ `I = ")()("` ☐ `I = "()(()())"` ☐ `I = "((()))()"`

☐ `I = "((()))"` ☐ `I = "())(()"` ☐ `I = "(())(()))("`

คำถามที่ 10. จากกราฟที่กำหนดให้ดังรูปทางด้านขวามือ ข้อใดต่อไปนี้เป็นลำดับของ node ที่เป็นไปได้จากการใช้อัลกอริทึม Breadth-first search เพื่อ traverse กราฟนี้



○ ABCDEF ○ EABDFC ○ BEADCF ○ EABDCF ○ FAEDBC

คำถามที่ 11. จงศึกษาตัวอย่างโจทย์ดังต่อไปนี้ แล้วจึงแก้โจทย์ในช่วงท้ายของคำถาม

ตัวอย่าง จากนิพจน์ (expression) ทางคณิตศาสตร์ต่อไปนี้

$$5 - 2 + 8 - 4 - 3$$

หากเราเติมวงเล็บในนิพจน์ข้างต้นกี่คู่ก็ได้เพื่อเปลี่ยนกลุ่มการบวกหรือการลบเท่านั้น[3] จะทำให้ผลลัพธ์สุดท้ายที่ได้ต่างออกไป แต่วิธีการเติมวงเล็บที่ทำให้ผลลัพธ์สุดท้ายที่ค่ามากที่สุดคือ

$$5 - 2 + 8 - (4 - 3) = 10$$

โจทย์ จงพิจารณานิพจน์ที่เป็นโจทย์ดังต่อไปนี้แล้วตอบว่า จะเติมวงเล็บลงในนิพจน์นี้เพื่อเปลี่ยนกลุ่มการลบ และให้ผลลัพธ์สุดท้ายมีค่ามากที่สุด จะได้ค่าเท่าใด?

$$0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10$$

[3] **ยกตัวอย่าง** เช่น

- $5 - (2 + 8) - 4 - 3 = -12$
- $5 - (2 + 8 - 4 - 3) = 2$
- $5 - (2 + 8 - (4 - 3)) = -4$

หมายเหตุ แต่ไม่อนุญาตให้มีการคูณเกิดขึ้นจากการเติมวงเล็บในนิพจน์ที่กำหนดให้ เช่น

- $5 - 2 + 8(-4)(-3) = 99$

คำถามที่ 12. จงพิจารณานิพจน์ที่เป็นโจทย์ดังต่อไปนี้แล้วตอบว่า จะเติมวงเล็บลงในนิพจน์นี้เพื่อเปลี่ยนกลุ่มการบวกหรือการลบ และให้ผลลัพธ์สุดท้ายมีค่ามากที่สุด จะได้ค่าเท่าใด?

$$10 - 9 - 8 - 7 - 6 + 5 - 4 - 3 - 2 - 1$$

คำถามที่ 13. จงพิจารณาโปรแกรมดังต่อไปนี้

```
function mystery_13(A[0 ... n-1])
    # A คือ 0-indexed array
    n := A.length()
    for i := 0 to n-1
        for j := n downto i + 1
            swap values between A[i] and A[j]
        end
    end
    return A
end
```

จงหา array A ที่เมื่อป้อนเป็น input ให้แก่โปรแกรมข้างต้นแล้ว จะได้ output ออกมาเป็น array ที่มีค่า `[1, 2, 3, 4]`

คำถามที่ 14. มีกล่องอยู่ 75 ใบ

- กล่องแต่ละใบถูกล็อกไว้ด้วยแม่กุญแจที่แตกต่างกันทั้งหมด
- กุญแจสำหรับแม่กุญแจแต่ละอัน ถูกบรรจุซ่อนอยู่ในกล่องใบอื่น ๆ (แปลว่ากุญแจที่จะเปิดกล่อง X จะอยู่ในกล่องอื่นซึ่งไม่ใช่ X)
- นอกจากนั้น แต่ละกล่องมีกุญแจอยู่ข้างในเพียงดอกเดียว
- การเปิดกล่องหนึ่งกล่องมีได้สองวิธีคือ 1. ทำลายแม่กุญแจ หรือ 2. ใช้กุญแจไข

หากเราต้องการเปิดกล่องทุกใบออกทั้งหมด โดยที่ทำลายแม่กุญแจให้น้อยที่สุด ในกรณีที่เลวร้ายที่สุดจะต้องทำลายแม่กุญแจอย่างน้อยกี่กล่องจึงจะเปิดกล่องได้ครบ 75 ใบ?

คำถามที่ 15. เรานิยาม **Proper binary tree** คือต้นไม้ทวิภาคที่ทุกโหนดมีโหนดลูกเป็นจำนวน 0 หรือ 2 โหนดพอดีเท่านั้น (กล่าวคือทุกโหนดที่เป็น Internal/Non-leaf Node จะต้องมีโหนดลูก 2 โหนดพอดี)

สมมติว่าเรามี Programming Library ตัวหนึ่งที่สามารถจัดการข้อมูลที่มีโครงสร้างเป็น Proper binary tree ซึ่งมีรายละเอียดดังต่อไปนี้

Data type. โหนดแต่ละโหนดของ Proper binary tree จะมี data type ที่มีชื่อว่า `Node` ซึ่งแบ่ง subtype ได้ 2 แบบย่อย ดังนี้

- `Leaf{value: Integer}`
  เป็น Leaf node ซึ่งประกอบด้วยข้อมูล 1 อย่าง ซึ่งก็คือจำนวนเต็มภายในโหนด
- `Internal{left: Node, right: Node, value: Integer}`
  เป็น Non-leaf node ซึ่งประกอบด้วยข้อมูลทั้งสิ้น 3 อย่าง ได้แก่ 1. Subtree ทางซ้าย 2. Subtree ทางขวา และ 3. ข้อมูลจำนวนเต็มภายในโหนด ตามลำดับ

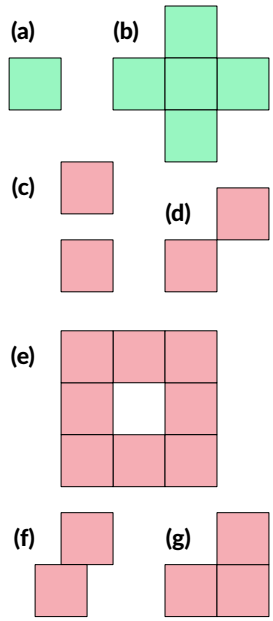
Library function. กำหนดให้ฟังก์ชัน `compute` รับข้อมูลต้นไม้ประเภท `Node` และฟังก์ชันอื่น ๆ อีก 2 ฟังก์ชัน (ดูโค้ดด้านล่างประกอบ) เพื่อประมวลผลข้อมูลต้นไม้ดังกล่าว ซึ่งสามารถเขียนเป็น pseudocode ได้ดังต่อไปนี้

```
function compute(tree: Node,
                 leaf_transformer: function,        # signature: Integer → T
                 internal_transformer: function):   # signature: T, T, Integer → T
    if tree matches Leaf{value} then:
        return leaf_transformer(value)
    else if tree matches Internal{left, right, value} then:
        return internal_transformer(
            compute(left, leaf_transformer, internal_transformer),
            compute(right, leaf_transformer, internal_transformer),
            value
        )
    end
end
```

โจทย์ สมมติว่าเรามีข้อมูล Proper binary tree ที่เก็บอยู่ในตัวแปรชื่อ `data` แล้วโปรแกรมในข้อใดต่อไปนี้จะนับจำนวนโหนดทั้งหมดภายในต้นไม้ (ทั้ง Leaf และ Non-leaf) ภายในตัวแปร `data` ดังกล่าวได้ถูกต้อง?[4]

[4] **หมายเหตุ** ยกตัวอย่าง syntax ของฟังก์ชันนิรนาม เช่น `lambda(x, y) → (x + y)/2` คือ ฟังก์ชันที่หาค่าเฉลี่ยของตัวเลข input argument สองตัว

- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r)`
- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r) * 2`
- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r) * 2 + 1`
- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r) * 2 - 1`
- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r + v)`
- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r + v) * 2`
- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r + v) * 2 + 1`
- ○ `compute(data, lambda(v) → v, lambda(l, r, v) → l + r + v) * 2 - 1`
- ○ `compute(data, lambda(v) → 1, lambda(l, r, v) → l + r) * 2`
- ○ `compute(data, lambda(v) → 1, lambda(l, r, v) → l + r) * 2 + 1`
- ○ `compute(data, lambda(v) → 1, lambda(l, r, v) → l + r) * 2 - 1`
- ○ `compute(data, lambda(v) → 1, lambda(l, r, v) → l + r + 1) * 2`
- ○ `compute(data, lambda(v) → 1, lambda(l, r, v) → l + r - 1) * 2`
- ○ `compute(data, lambda(v) → 0, lambda(l, r, v) → l + r) * 2`
- ○ `compute(data, lambda(v) → 0, lambda(l, r, v) → l + r) * 2 + 1`
- ○ `compute(data, lambda(v) → 0, lambda(l, r, v) → l + r) * 2 - 1`
- ○ `compute(data, lambda(v) → 0, lambda(l, r, v) → l + r + 1) * 2`
- ○ `compute(data, lambda(v) → 0, lambda(l, r, v) → l + r - 1) * 2`



คำถามที่ 16. สมมติว่าเราต้องการสร้างรูปเรขาคณิตใหม่ โดยนำชิ้นส่วนจัตุรัสขนาด 1 ตารางหน่วยอย่างน้อย 1 ชิ้นมาประกอบกัน โดยมีเงื่อนไขดังต่อไปนี้

- รูปเรขาคณิตใหม่จะต้องไม่มีชิ้นส่วนที่แยกจากกัน (ยกตัวอย่างเช่นรูป **(c)** ใช้ไม่ได้)
- ชิ้นส่วนจัตุรัสสองชิ้นที่แตะกันเฉพาะตำแหน่งมุม ถือว่าเป็นชิ้นส่วนที่แยกจากกัน (รูปตัวอย่าง **(d)** ใช้ไม่ได้)
- รูปเรขาคณิตใหม่จะต้องไม่มีรูหรือช่องว่างภายในรูปปิด (ตัวอย่างเช่นรูป **(e)** ใช้ไม่ได้)
- รูปเรขาคณิตใหม่จะต้องมีทุกด้านยาว 1 หน่วยเท่านั้น (เช่นรูป **(f)** หรือ **(g)** ใช้ไม่ได้ เพราะมีบางด้านยาว 0.5 หน่วย หรือ 2 หน่วย)

สังเกตว่า หากเรามีชิ้นส่วนจัตุรัส 1 ชิ้นพอดี หรือ 5 ชิ้นพอดี (จากรูปตัวอย่าง **(a)** และ **(b)** ตามลำดับ) เราสามารถสร้างรูปเรขาคณิตที่สอดคล้องกับเงื่อนไขข้างต้นได้ อย่างไรก็ดี เราไม่สามารถสร้างรูปเรขาคณิตตามเงื่อนไขดังกล่าวได้เลย หากเรามีชิ้นส่วนจัตุรัสเพียง 2 หรือ 3 หรือ 4 ชิ้นเท่านั้น

จงหาจำนวนชิ้นส่วนจัตุรัสที่มากที่สุดที่ไม่สามารถนำมาประกอบเป็นรูปเรขาคณิตตามเงื่อนไขข้างต้นได้

คำถามที่ 17. สมมติว่ามีไพ่สำรับมาตรฐาน 2 สำรับ สำรับละ 52 ใบ รวมทั้งสิ้น 104 ใบ วางกองไว้รวมกันโดยไม่เรียงลำดับ

- ไพ่ 1 สำรับ มีชุดตัวเลขทั้งสิ้น 13 ชุด (2, 3, 4, ..., 10, J, Q, K, A) ชุดละ 4 ใบ
- ไพ่ตอง คือ ไพ่สามใบที่ชุดตัวเลขเหมือนกันเช่น 2, 2, 2 หรือ K, K, K

เราจะต้องหยิบไพ่ในกองดังกล่าวอย่างสุ่มอย่างน้อยกี่ใบ จึงจะรับประกันว่าในบรรดาไพ่ที่เราหยิบมาจะมีไพ่ตองอย่างน้อย 1 ชุด?

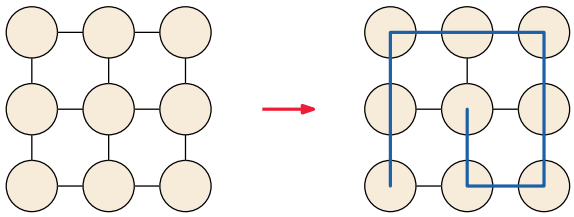
คำถามที่ 18. จงพิจารณาโปรแกรมดังต่อไปนี้

```
function mystery_18(A[0 ... n-1]):
    # A คือ 0-indexed array ของจำนวนเต็ม
    n = A.length()
    result = 0
    for i := 0 to n-1 do:
        if A[i] < result then:
            result := A[i]
        end
    return result
end
```

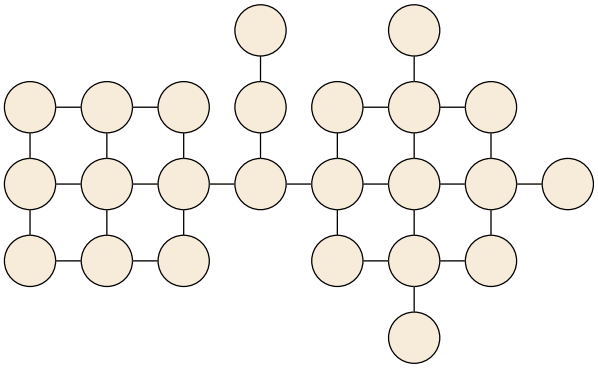
จงหาผลลัพธ์ของการรันโปรแกรม `mystery_18([9, 43, 214, 1, 5])`

คำถามที่ 19. คำถามเกี่ยวกับอัลกอริทึม Depth-first search (DFS)

ตัวอย่าง พิจารณากราฟที่มีลักษณะเป็น grid ขนาด 3×3 ดังที่แสดงทางด้านขวา หากเราใช้อัลกอริทึม Depth-first search (DFS) กับกราฟนี้โดยเริ่มต้นจากโหนดใดก็ได้ เราจะได้ Maximum recursion depth เท่ากับ 9 ดังที่แสดงด้วยเส้นสีฟ้า



โจทย์ จงหา Maximum recursion depth ที่เป็นไปได้ หากเราสามารถใช้อัลกอริทึม Depth-first search จากโหนดใดก็ได้ในกราฟต่อไปนี้

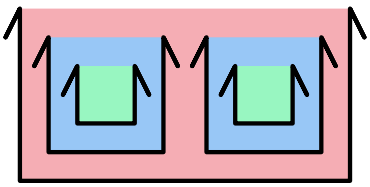


คำถามที่ 20. เรากำหนดให้มีกล่องจำนวนหนึ่งวางอยู่บนโต๊ะ ซึ่งกล่องบางใบอาจจะถูกซ้อนอยู่ภายในกล่องใบอื่นก็ได้ และมีเงื่อนไขเพื่อความชัดเจนว่า

- ถ้ากล่อง X ถูกซ้อนในกล่อง Y และกล่อง Y ถูกซ้อนในกล่อง Z แล้วกล่อง X จะถือว่าถูกซ้อนในกล่อง Z ด้วย
- ถ้ากล่อง X ถูกซ้อนในกล่อง Y แล้วกล่อง Y จะไม่ถูกซ้อนในกล่อง X

นอกจากเงื่อนไขข้างต้นนี้ เรากำหนดนิยามเพิ่มเติมดังต่อไปนี้

> **นิยาม** หากกำหนดให้ B เป็นสัญลักษณ์ที่สื่อถึงรูปแบบของการจัดวางกล่องบนโต๊ะตามเงื่อนไขที่กำหนดให้ แล้วสัญลักษณ์ $B[n]$ จะหมายถึง “จำนวนของกล่องทั้งหมดบนโต๊ะที่มีกล่องอื่น ๆ จำนวน n ใบซ้อนอยู่ภายใน”
>
> ยกตัวอย่างเช่น สมมติว่า B_0 คือการจัดวางกล่อง 5 ใบบนโต๊ะรูปแบบหนึ่ง ดังที่แสดงทางด้านขวามือนี้ เราสามารถบอกได้ว่า $B_0[1] = 2$ (กล่องสีฟ้า), $B_0[2] = 0$, $B_0[3] = 0$ และ $B_0[4] = 1$ (กล่องสีแดง) ฯลฯ



รูปตัวอย่างการจัดวางกล่อง B_0

โจทย์ จงคำนวณว่าสำหรับการจัดวางกล่อง B_1 ในรูปแบบใด ๆ ที่สอดคล้องกับเงื่อนไขว่า $B_1[1] = 5$, $B_1[2] = 4$, $B_1[3] = 3$, $B_1[4] = 2$ และ $B_1[5] = 1$ จะต้องมีจำนวนกล่องรวมทั้งหมดอย่างน้อยกี่ใบ?

คำถามที่ 21. กำหนดให้มีฟังก์ชันหนึ่ง `sort_between(L, p, q)` ซึ่งมีสเปกดังนี้

- ข้อมูล input `L` เป็น 0-indexed array ของจำนวนชุดหนึ่ง
- ข้อมูล input `p` และ `q` เป็น index ภายใน array `L` โดยที่มีเงื่อนไขว่า `0 ⩽ p ⩽ q < L.length()`
- ข้อมูล output ของ `sort_between(L, p, q)` จะเป็น array ใหม่ ซึ่งเกิดจากการจัดเรียงจำนวนบางจำนวนใน array `L` เดิม ตั้งแต่ตำแหน่ง `p` ถึงตำแหน่ง `q` จากน้อยไปมาก และจำนวนในตำแหน่งอื่น ๆ นอกเหนือจากนี้ยังคงเดิม

ยกตัวอย่าง ถ้ากำหนดให้ `L_0 = [4, 2, 7, 3, 8, 1, 5]` แล้วเมื่อเรียกฟังก์ชัน `sort_between(L_0, 2, 5)` จะได้ output เป็น `[4, 2, 1, 3, 7, 8, 5]`

โจทย์ สมมติว่าเรามี array `L_1` ซึ่งประกอบไปด้วยจำนวนทั้งสิ้น 250 จำนวน โปรแกรมในข้อใดต่อไปนี้ไม่รับประกันว่าสามารถเรียงลำดับจำนวนทุกจำนวนภายใน array ได้ทั้งหมด?

○ `sort_between(L_1, 0, 249)`

○ `sort_between(L_1, 0, 199)`
`sort_between(L_1, 150, 249)`
`sort_between(L_1, 0, 199)`

○ `sort_between(L_1, 0, 199)`
`sort_between(L_1, 50, 249)`
`sort_between(L_1, 0, 149)`

○ `sort_between(L_1, 100, 249)`
`sort_between(L_1, 0, 149)`
`sort_between(L_1, 50, 249)`

○ `sort_between(L_1, 150, 249)`
`sort_between(L_1, 0, 199)`
`sort_between(L_1, 150, 249)`

คำถามที่ 22. กำหนดให้มีฟังก์ชันหนึ่ง `sort_between(L, p, q)` ซึ่งมีสเปกเหมือนกับคำถามที่แล้ว (ดูคำถามที่ 21)

เราจะนำฟังก์ชัน `sort_between` ดังกล่าวนี้มาใช้งานเพื่อเขียนฟังก์ชันใหม่ที่มีชื่อว่า `slider_sort_between` ซึ่งมีกระบวนการทำงานดังต่อไปนี้

```
1  function slider_sort_between(L, k):
2      n := L.length()
3      for i := 0 to n-k do:
4          # เรียงลำดับจำนวนที่ติดกัน k จำนวนใน array L
5          # ตั้งแต่ตำแหน่งที่ i จนถึงตำแหน่งที่ i-k+1
6          sort_inplace(L, i, i+k-1)
7      end
8  end
```

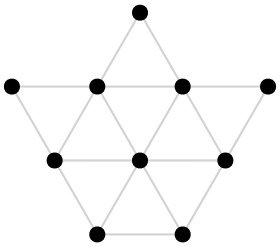
สมมติว่าเรามี array `L_1` ซึ่งประกอบไปด้วยจำนวนทั้งสิ้น 250 จำนวน เป้าหมายคือเราต้องการเรียงลำดับจำนวนทุกจำนวนภายใน array นี้ด้วยการเรียกใช้งานคำสั่ง

```
L_1 := slider_sort_between(L_1, k=25)
```

นี้ซ้ำ ๆ กันอย่างต่อเนื่อง จนกว่าจำนวนใน array `L_1` จะเรียงลำดับทั้งหมด อยากทราบว่าเราจะต้องรันคำสั่งข้างต้นนี้อย่างน้อยกี่รอบ จึงเพียงพอที่จะรับประกันว่าค่าทั้งหมดของ array `L_1` เรียงลำดับจากน้อยไปมาก?

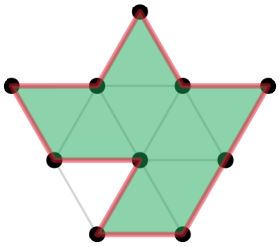
คำถามที่ 23. คำถามเกี่ยวกับการลากเส้นรูปปิดตามโครงสร้างที่กำหนดให้

ตัวอย่าง จากรูปที่แสดงทางขวามือนี้ มีโครงสร้างเรขาคณิตที่เรียงตัวเป็นสามเหลี่ยมด้านเท่าย่อย ๆ หลายรูปประกอบกัน โดยที่สามเหลี่ยมย่อยแต่ละรูปมีพื้นที่รูปละ 1 ตารางหน่วย



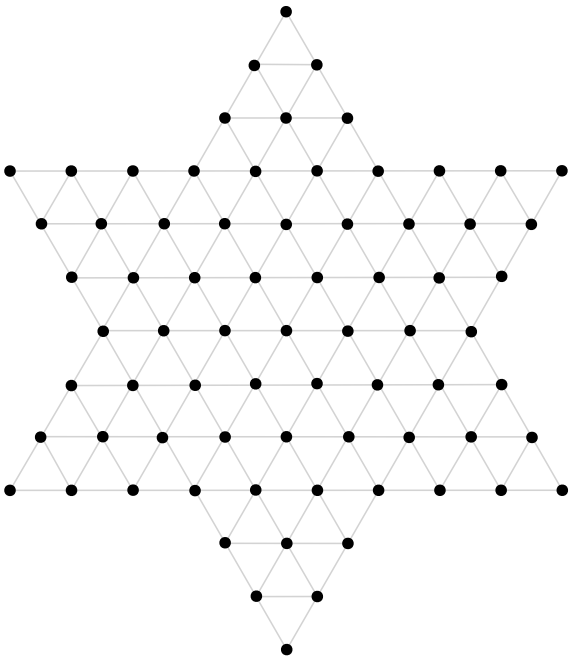
จากโครงสร้างเรขาคณิตรูปนี้ เราจะพยายามลากเส้นภายในโครงสร้างที่กำหนดให้ข้างต้น โดยมีเงื่อนไขว่า

- เราจะต้องลากเส้นเพียงเส้นเดียว ผ่านจุดยอดให้ครบทุกจุด แล้วกลับมายังจุดเริ่มต้นกลายเป็นรูปปิด
- จุดยอดแต่ละจุดจะต้องถูกเยือน 1 ครั้งพอดี ไม่ขาดและไม่เกิน
- เส้นทุกเส้นที่ลากผ่านจะต้องมีเค้าโครงเดิมจากเส้นสีเทาที่กำหนดให้จากรูปดั้งเดิม
- รูปปิดที่เกิดขึ้นจะต้องมีพื้นที่ภายในมากที่สุดเท่าที่เป็นไปได้

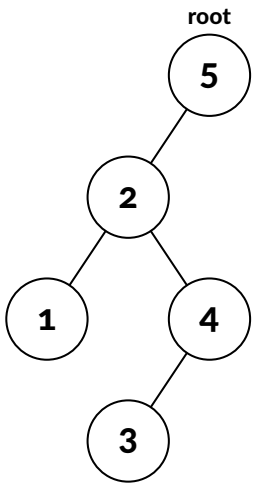


รูปที่สองที่ปรากฏทางด้านขวามือนี้ แสดงหนึ่งในวิธีที่เราสามารถลากเส้นรูปปิดภายในโครงสร้างเรขาคณิต (ดังที่แสดงในรูปแรก) ที่สอดคล้องกับเงื่อนไขที่กล่าวมาข้างต้นได้ทั้งหมด ซึ่งรูปปิดดังกล่าวจะได้พื้นที่ภายในรวม 8 ตารางหน่วย (**หมายเหตุ:** อาจมีวิธีลากเส้นวิธีอื่น ๆ ที่ทำให้ได้พื้นที่ขนาดเท่ากัน)

โจทย์ จงหาว่าในรูปต่อไปนี้ เราจะลากเส้นสร้างรูปปิดตามเงื่อนไขเดียวกันให้ได้พื้นที่ภายในมากที่สุด จะได้พื้นที่เท่าใด?



ตัวอย่าง เช่นจากรูปต้นไม้ตัวอย่างนี้



- จะมี Pre-order traversal ordering เป็น $[5, 2, 1, 4, 3]$
- และจะมี Breadth-first search ordering ได้สองรูปแบบ ได้แก่ $[5, 2, 1, 4, 3]$ และ $[5, 2, 4, 1, 3]$

คำถามที่ 24. จงพิจารณานิยามของลำดับ (ordering) ในการท่องต้นไม้ค้นหาแบบทวิภาค (Binary search tree) ที่แตกต่างกันดังต่อไปนี้

- **Pre-order traversal** ของโครงสร้างข้อมูลต้นไม้ค้นหาทวิภาค (Binary search tree) คือการใช้อัลกอริทึม Depth-first search (DFS) เพื่อประมวลผลโหนดแต่ละโหนดของต้นไม้หนึ่งตามลำดับดังต่อไปนี้ (โดยนับเริ่มต้นจากราก (root) ของต้นไม้)

  1. compute โหนดตัวเอง
  2. recursively compute โหนดในกลุ่ม Subtree ทางซ้ายของโหนดตัวเอง
  3. recursively compute โหนดในกลุ่ม Subtree ทางขวาของโหนดตัวเอง

- **Breadth-first search ordering** ของโครงสร้างข้อมูลต้นไม้ทวิภาค (Binary tree) คือลำดับการท่องต้นไม้ตามอัลกอริทึม Breadth-first search (BFS) (**หมายเหตุ:** โดยมีเงื่อนไขกำหนดให้เริ่มต้นจากรากของต้นไม้)

โจทย์ สมมติว่ามีต้นไม้ค้นหาแบบทวิภาค (Binary search tree) T อยู่ต้นหนึ่ง ซึ่งพบว่ามีลำดับ Pre-order traversal เป็น $[7, 2, 1, 4, 3, 6, 5, 9, 8, 10, 11]$ อยากทราบว่าต้นไม้ T นี้จะมี Breadth-first search (BFS) ordering ที่เริ่มต้นจากรากได้ต่างกันทั้งหมดกี่รูปแบบ?

คำถามที่ 25. สมมติว่าครอบครัวหนึ่ง มีลูกทั้งสิ้น 5 คน ได้แก่ Dijkstra (D), Hopper (H), Lovelace (L), Neumann (N) และ Shannon (S) ซึ่งไม่เรียงลำดับใด ๆ ทั้งสิ้น
ต่อไปนี้จะเป็นคำบอกเล่าของแม่ 7 ประโยค ซึ่งมี 1 ประโยคเท่านั้นที่เป็นเท็จ

1. Hopper มีอายุน้อยกว่า Shannon
2. Dijkstra มีอายุน้อยกว่า Lovelace
3. Shannon มีอายุน้อยกว่า Neumann
4. Hopper มีอายุน้อยกว่า Dijkstra
5. Lovelace มีอายุน้อยกว่า Shannon
6. Shannon มีอายุน้อยกว่า Dijkstra
7. Neumann มีอายุน้อยกว่า Lovelace

จงตอบคำถามว่า 1. ประโยคข้างต้นใดเป็นเท็จ และ 2. ใครเป็นพี่คนโต

หมวดที่ 2
โจทย์ Ponder

↶ สำหรับโจทย์ประเภท Ponder ผู้เข้าแข่งขันจะมีเวลาวิเคราะห์โจทย์และค้นหาคำตอบภายในเวลา 5 นาที สามารถใช้คอมพิวเตอร์เพื่อช่วยหาคำตอบได้อย่างเต็มที่

หมายเหตุ โจทย์บางข้อที่ปรากฏอาจมีลักษณะเป็นคำถามปรนัย (มีตัวเลือก) เนื่องจากเป็นข้อจำกัดของระบบในการแข่งขันระดับ Audition

คำถามที่ 26. ธนาคารกสิกรไทยสาขา **TECHJAM 2018** ได้ก่อสร้างอาคารสูง 1000 ชั้นเสร็จหมาด ๆ เจ้าหน้าที่ต้องการติดป้ายบอกชั้นตั้งแต่ชั้นที่ 1 ถึงชั้นที่ 1000 ชั้นละ 1 ชุด โดยเลขแต่ละชุดเกิดจากการนำแผ่นกระดานที่มีเลขโดด 0 ถึง 9 หลายแผ่นมาประกอบกันแล้วนำไปติดตามชั้นต่าง ๆ[1]

อยากทราบว่าเจ้าหน้าที่ต้องเตรียมแผ่นกระดานที่มีเลขโดด 9 ทั้งหมดกี่แผ่น เพื่อนำมาติดป้ายให้ครบทุกชั้น?

[1] **ตัวอย่าง** เช่นชั้นที่ 37 จะต้องใช้แผ่นกระดานเลขโดด 3 และ 7 อย่างละ 1 แผ่น

คำถามที่ 27. กำหนดให้มีต้นไม้ค้นหาแบบทวิภาค (Binary search tree) อยู่ต้นหนึ่ง ซึ่งประกอบไปด้วยจำนวนบางจำนวนที่อยู่ระหว่าง 1 กับ 100 ปรากฏอยู่

หากเราต้องการค้นหาข้อมูลซึ่งคือจำนวน 56 ภายในต้นไม้ข้างต้นนี้ ข้อใดต่อไปนี้เป็นลำดับการเกิด tree traversal ที่เริ่มต้นจากราก (root) เพื่อค้นห้าข้อมูลดังกล่าว ที่ไม่สามารถเกิดขึ้นได้?

○ 7, 82, 46, 66, 43, 58, 56
○ 13, 77, 62, 41, 59, 57, 56
○ 92, 13, 66, 34, 61, 41, 56
○ 77, 11, 72, 59, 13, 52, 56

คำถามที่ 28. ในบรรดาจำนวนเต็มตั้งแต่ 1 ถึง 1000 มีกี่จำนวนซึ่งมีผลรวมเลขโดดที่ไม่มีเลขโดด 1 ปรากฏภายในผลรวมเลขโดดนั้นเลย?[2]

[2] **ตัวอย่าง** เพื่อความชัดเจน
- เลข 123 มีผลรวมเลขโดดคือ $1 + 2 + 3 = 6$ ซึ่งไม่มีเลขโดด 1 อยู่ใน 6
- เลข 67 มีผลรวมเลขโดดคือ $6 + 7 = 13$ ซึ่งมีเลขโดด 1 อยู่ใน 13

คำถามที่ 29. กำหนดให้มีจำนวน 6 จำนวน ได้แก่ 1, 2, 5, 6, 7, 9 ให้นำจำนวนเหล่านี้มา “บวก–ลบ–คูณ–หาร” ให้ได้จำนวน 258 โดยที่มีเงื่อนไขดังนี้

- จะใช้ $+, -, \times, \div$ อย่างไรก็ได้ และจะจัดกลุ่มหรือใส่วงเล็บอย่างไรก็ได้
- จะใช้ตัวเลขในลำดับใดก็ได้ แต่ต้องใช้ตัวเลขทุกตัว ตัวละหนึ่งครั้งพอดี

จงหาวิธีการจัดนิพจน์ให้กับจำนวนทั้งหมดข้างต้น เพื่อให้ได้ผลลัพธ์ตามที่ต้องการ?

คำถามที่ 30. จงพิจารณาโปรแกรมดังต่อไปนี้[3]

```
function mystery_30(n):
   # n เป็นจำนวนเต็มที่ไม่ติดลบ
   lb := 0
   ub := n
   loop:
      attempt := floor((lb + ub) / 2)
      if n < attempt^2:
         ub := attempt - 1
      elseif n ⩾ (attempt + 1)^2:
         lb := attempt + 1
      else:
         break loop and return attempt
      end
   end
end
```

โปรแกรมข้างต้นทำหน้าที่ตามที่ระบุในข้อใดต่อไปนี้?

- ○ ฟังก์ชัน square root แต่ปัดเศษเป็นจำนวนเต็มที่ใกล้ที่สุดเสมอ (round to nearest integer)
- ○ ฟังก์ชัน square root แต่ปัดเศษทิ้งเป็นจำนวนเต็มเสมอ (round down)
- ○ ฟังก์ชัน square root แต่ปัดเศษขึ้นเป็นจำนวนเต็มเสมอ (round up)
- ○ ฟังก์ชัน square root แต่เศษอาจถูกปัดขึ้นหรือลงอย่างไรก็ได้ ไม่สามารถคาดเดาได้
- ○ ฟังก์ชันติด infinite loop ไม่รู้จบ

[3] **หมายเหตุ** `floor` คือฟังก์ชันที่ปัดเศษของจำนวนเต็มทิ้งให้กลายเป็นจำนวนเต็มที่มากที่สุดที่น้อยกว่าหรือเท่ากับจำนวนเดิม

คำถามที่ 31. กำหนดให้ A เป็น Array ของเลขจำนวนเต็มดังต่อไปนี้

```
A = [133, 60, 96, 130, 125, 65, 482, 88, 220, 165, 25, 45]
```

จงเลือกจำนวน 3 จำนวนที่ไม่ซ้ำกันจาก A แล้วหาผลคูณที่ลงท้ายด้วยเลขโดด 0 เยอะที่สุด ผลคูณดังกล่าวมีค่าเท่าใด?

คำถามที่ 32. กำหนดให้ `rand7()` เป็นฟังก์ชันที่สุ่มจำนวนเต็มในช่วงตั้งแต่ 1 ถึง 7 ด้วยความน่าจะเป็นเท่า ๆ กัน

ให้พิจารณานิพจน์ดังต่อไปนี้[4] ที่สุ่มจำนวนเต็ม 1 จำนวนในช่วงตั้งแต่ 1 ถึง 11

```
((rand7() + rand7()) mod 11) + 1
```

อยากทราบว่า output ของนิพจน์ข้างต้นที่เป็นไปได้มากที่สุด มีค่าเท่าใด?

[4] **หมายเหตุ** binary operator `mod` คือการหารที่เอาเฉพาะเศษเป็นผลลัพธ์ของนิพจน์นั้น เช่น `13 mod 3 = 1`

คำถามที่ 33. กำหนดให้มี input ทั้งสิ้น 1 จำนวน ได้แก่จำนวนจริง x

จาก input ข้างต้นนี้ เป้าหมายคือการคำนวณค่าของ x^n โดยที่ n เป็นจำนวนเต็มบวก โดยใช้ operation การคูณเป็นจำนวนครั้งน้อยที่สุด ภายใต้เงื่อนไขดังต่อไปนี้

- อนุญาตให้ใช้เฉพาะ operation การคูณ
- อนุญาตให้นำผลคูณที่เกิดขึ้นก่อนหน้านั้นระหว่างการคำนวณ มาใช้เป็นตัวตั้งหรือตัวคูณของการคูณครั้งถัดไปได้[5] (หมายความว่า เรามีการจดบันทึกผลการคูณที่เกิดขึ้นทั้งหมด คล้ายกับ history tape ในเครื่องคิดเลขของนักบัญชี)

โจทย์ การคำนวณหาค่าของ x^n ในกรณีที่ $n = 125$ (นั่นคือให้คำนวณค่าของ x^{125}) จะต้องใช้ operation การคูณเป็นจำนวนน้อยที่สุดกี่ครั้ง? และการคูณในแต่ละขั้นนั้นจะคำนวณ $x^{???}$ อะไรบ้างตามลำดับ? (ให้ตอบมา 1 วิธี)

[5] **ยกตัวอย่าง** สมมติว่าเราต้องการคำนวณ x^n ในกรณีที่ $n = 6$ เราจะใช้การคูณน้อยที่สุดเพียง 3 ครั้งเท่านั้น เขียนเป็นขั้นตอนวิธีได้ดังนี้

```
r_1 := x * x # ⇒ x^2
r_2 := r_1 * r_1 # ⇒ x^4
r_3 := r_1 * r_2 # ⇒ x^6
```

สังเกตว่า จากข้อมูล x ที่เราทราบ เราจะคำนวณหาค่าของ x^2, x^4 และ x^6 ตามลำดับ นอกเหนือจากวิธีนี้ ยังมีวิธีอื่นอีก เช่น การคำนวณหา x^2, x^3 และ x^6 ตามลำดับ

คำถามที่ 34. กำหนดให้มี input ทั้งสิ้น 3 จำนวน ได้แก่จำนวนจริง x, y และ z

จาก input ข้างต้นนี้ เป้าหมายคือการคำนวณค่าของนิพจน์ $ax + by + cz$ โดยที่ a, b และ c เป็นค่าคงที่จำนวนเต็มที่ไม่ติดลบ โดยใช้ operation การบวกเป็นจำนวนครั้งน้อยที่สุด ภายใต้เงื่อนไขดังต่อไปนี้

- อนุญาตให้ใช้เฉพาะ operation การบวก
- อนุญาตให้นำผลบวกที่เกิดขึ้นก่อนหน้านั้นระหว่างการคำนวณ มาใช้เป็นตัวตั้งหรือตัวบวกของการบวกครั้งถัดไปได้(หมายความว่า เรามีการจดบันทึกผลการบวกที่เกิดขึ้นทั้งหมด คล้ายกับ history tape ในเครื่องคิดเลขของนักบัญชี)

ตัวอย่าง สมมติว่าเราต้องการคำนวณ $x + 2y + 3z$ ในเคสทั่วไป เราอาจคำนวณตามลำดับในสมการ $x + y + y + z + z + z$ ซึ่งใช้การบวกทั้งสิ้น 5 ครั้ง แต่เนื่องจากเงื่อนไขอนุญาตให้นำผลบวกก่อนหน้ามาใช้งานได้ เราสามารถใช้การบวกน้อยที่สุดเพียง 4 ครั้งเท่านั้น ซึ่งเขียนเป็นขั้นตอนวิธีได้ดังนี้

```
1  r_1 := y + z        # ⇒ y + z
2  r_2 := r_1 + r_1    # ⇒ 2y + 2z
3  r_3 := r_2 + z      # ⇒ 2y + 3z
4  r_4 := r_3 + x      # ⇒ x + 2y + 3z
```

โจทย์ การคำนวณหาค่าของ $x + 4y + 9z$ จาก input x, y และ z จะต้องใช้ operation การบวกเป็นจำนวนน้อยที่สุดกี่ครั้ง?

คำถามที่ 35. สมมติว่าเรามีกองแท่งไม้ 5000 แท่ง แท่งไม้แต่ละแท่งมีความยาว 1, 2, 3, ..., 5000 หน่วยตามลำดับ เราต้องการสร้างสามเหลี่ยมที่มีพื้นที่ภายในมากกว่าศูนย์[6] โดยการหยิบแท่งไม้จากกองดังกล่าวมา 3 แท่งมาประกอบกัน

อยากทราบว่าเราจะมีวิธีเลือกหยิบแท่งไม้ 3 แท่งจากกองดังกล่าว ประกอบให้กลายเป็นสามเหลี่ยมที่มีพื้นที่มากกว่าศูนย์ได้กี่วิธี?

หมายเหตุ ข้อนี้อาจต้องใช้ 64-bit integer

[6] สามเหลี่ยมใด ๆ ที่มีพื้นที่มากกว่าศูนย์จะต้องมีผลรวมความยาวของด้าน 2 ด้านใด ๆ ยาวกว่าด้านที่สามเสมอ เช่น

- 3, 4, 5 ประกอบเป็นความยาวด้านของสามเหลี่ยมได้
- 1, 3, 5 ประกอบเป็นความยาวด้านของสามเหลี่ยมไม่ได้ เพราะ $1 + 3 < 5$
- 1, 2, 3 ประกอบเป็นความยาวด้านของสามเหลี่ยมได้ แต่จะมีพื้นที่เท่ากับ 0

คำถามที่ 36. จงพิจารณาโปรแกรมดังต่อไปนี้

```
function f(n):
    if n ≤ 1 then:
        return 1
    else:
        x = f(n/2)
        y = f(n/2)
        return g(n, x, y)
    end
end
```

หากกำหนดให้ time complexity ของฟังก์ชัน `g(n, ·, ·)` คือ $O(n)$ แล้วจงคำนวณหา time complexity ของฟังก์ชัน `f(n)`

คำถามที่ 37. จงพิจารณาโปรแกรมดังต่อไปนี้

```
function foo(A):
    n := A.length()
    bar(A, 0, n-1)
end

function bar(A, lo, hi):
    if hi - lo ≤ 31 then:
        return spin(A, lo, hi)
    end
    mid := select an integer strictly between
           lo and hi uniformly at random
    return bar(A, lo, mid) + bar(A, mid, hi)
end

function spin(A, lo, hi):
    count := 0
    for i := lo to hi-1 do:
        for j := i + 1 to hi do:
            if A[i] > A[j] then count := count + 1
        end
    end
    return count
end
```

หากกำหนดให้ฟังก์ชัน `foo(A)` ในโปรแกรมข้างต้นมี input argument คือ array `A` ของจำนวนเต็มที่มีความยาว `n` ตัวแล้ว จงวิเคราะห์เพื่อคำนวณหา Worst-case time complexity และ Average-case time complexity ของฟังก์ชัน `foo` ในรูปของ `n`

คำถามที่ 38. นายกสิกรเขียนอักขระภาษาอังกฤษ 1 ตัว ลงบนแต่ละหน้าของเหรียญ 4 เหรียญ โดยไม่มีตัวอักขระตัวใดซ้ำกันเลย จากนั้นเขาโยนเหรียญทั้ง 4 เหรียญลงบนโต๊ะอย่างสุ่ม แล้วนำอักขระบนเหรียญด้านที่หงายมาเรียงเป็นคำ ทำอย่างนี้ทั้งหมด 3 ครั้ง สะกดได้คำว่า “BOAT”, “NODE” และ “RANT” ตามลำดับ

อยากทราบว่า คำใดบ้างต่อไปนี้ที่ไม่สามารถสะกดจากอักขระที่ปรากฏบนเหรียญได้?

☐ BART ☐ DONE ☐ BORE ☐ NEAR

คำถามที่ 39. นายกสิกรเข้าร่วมในงานประมูลทะเบียนรถหมวด “ขศ” ซึ่งมีการเปิดประมูลเลขทะเบียนรถทุกหมายเลข ตั้งแต่ 1 จนถึง 9999

นายกสิกรเคยได้ยินจากหมอดูหลายท่านว่า หากต้องการให้ชีวิตมั่งคั่งร่ำรวย จะต้องมี “ผลรวมเลขโดดสุดท้าย”[7] เป็น 8 เช่น

- ป้ายทะเบียน 7595 เป็นป้ายทะเบียนมั่งคั่ง
  เพราะ $7+5+9+5=26 \implies 2+6=8$
- แต่ป้ายทะเบียน 7777 ไม่มั่งคั่ง
  เพราะ $7+7+7+7=28 \implies 2+8=10 \implies 1+0=1 \neq 8$

จงคำนวณว่า ในบรรดาป้ายทะเบียนในเปิดประมูลทั้งหมดในหมวด “ขศ” ในวันนี้ จะมีป้ายทะเบียนมั่งคั่งเป็นจำนวนกี่ป้าย?

[7] ผลรวมเลขโดดสุดท้าย คือการหาผลรวมเลขโดดของจำนวนจำนวนหนึ่ง ซ้ำ ๆ กันไปเรื่อย ๆ จนกว่าจะเหลือเลขโดดเพียงตัวเดียว

คำถามที่ 40. จงหาผลรวมของจำนวนทุกจำนวนที่ปรากฏในตารางดังต่อไปนี้

$$\begin{bmatrix} 1 & 2 & 3 & \cdots & \cdots & 15000 \\ 2 & 3 & 4 & \cdots & \cdots & 15001 \\ 3 & 4 & 5 & \cdots & \cdots & 15002 \\ \vdots & \vdots & \vdots & \ddots & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & \vdots \\ 15000 & 15001 & 15002 & \cdots & \cdots & 29999 \end{bmatrix}_{15000\times 15000}$$

หมายเหตุ ข้อนี้อาจต้องใช้ 64-bit integer

คำถามที่ 41. กำหนดให้มีนาฬิกาดิจิทัลอยู่เรือนหนึ่ง มีลักษณะเป็น 24-hour clock ที่แสดงผลในรูปแบบ `HH:MM:SS` ตั้งแต่เวลา `00:00:00` ไปจนถึง `23:59:59`

นิยาม เวลา ณ วินาทีหนึ่ง ๆ จะเป็น “เวลาเลขสวย” ก็ต่อเมื่อการแสดงผลบนหน้าปัดนาฬิกาประกอบไปด้วยเลขโดดที่แตกต่างกันไม่เกิน 2 ตัวเท่านั้น ยกตัวอย่างเช่น

- `13:31:11` เป็นเวลาเลขสวย เนื่องจากแสดงผลได้ด้วยเลขโดด 1 และ 3 เพียงสองตัว
- `11:11:11` เป็นเวลาเลขสวย เพราะใช้เลขโดดเพียงตัวเดียว (ยังไม่เกิน 2 ตัว)
- `23:00:00` ไม่เป็นเลขสวย เนื่องจากต้องใช้ตัวเลขโดดถึง 3 ตัวในการแสดงผล

โจทย์ จงหาว่า ในช่วงเวลาตั้งแต่ `01:30:00` ไปจนถึง `21:00:00` นาฬิกาดิจิทัลจะปรากฏเวลาเลขสวยทั้งหมดกี่ครั้ง?

คำถามที่ 42. กำหนดให้ $S = \{1, 2, 3, \ldots, 10\}$ เป็นเซตของจำนวนเต็ม

หากเราต้องการคัดเลือกจำนวนบางจำนวนจากเซต S ที่ไม่ซ้ำกันอย่างน้อย 1 จำนวน โดยที่ผลรวมของจำนวนที่เลือกมากนั้นจะต้องหารด้วย 10 ลงตัวพอดี[8]

อยากทราบว่าจะกระทำได้ทั้งหมดกี่วิธี?

[8] **ยกตัวอย่าง** บางวิธีที่กระทำได้ เช่น 10, $1 + 9$, $2 + 3 + 5$, $2 + 5 + 6 + 7$ เป็นต้น

คำถามที่ 43. เรากำหนดนิยามของ “จำนวนรูปงาม” ดังนี้

> **นิยาม** “จำนวนรูปงาม” คือจำนวนเต็มที่สามารถเขียนในรูปของ string concatenation ของสตริงย่อย ๆ อย่างน้อยหนึ่งตสตริงได้ และสตริงย่อยแต่ละตัวจะต้องเป็นจำนวนเฉพาะที่ไม่ขึ้นต้นด้วย `0` เท่านั้น
>
> ยกตัวอย่างเช่น
>
> - จำนวนเฉพาะทุกตัว (ได้แก่ `"2"`, `"3"`, `"5"`, `"7"`, `"11"`, `"13"`, ...) เป็นจำนวนรูปงามโดยปริยาย
> - จำนวน `"1012"` เป็นจำนวนรูปงาม เพราะอยู่ในรูปของ `"101" + "2"` ซึ่งแต่ละพจน์เป็นจำนวนเฉพาะ
> - จำนวน `"5932"` เป็นจำนวนรูปงาม เพราะอยู่ในรูปของ `"59" + "3" + "2"` ซึ่งแต่ละพจน์เป็นจำนวนเฉพาะ
> - จำนวน `"7352"` เป็นจำนวนรูปงาม เพราะอยู่ในรูปของ `"7" + "3" + "5" + "2"` ซึ่งแต่ละพจน์เป็นจำนวนเฉพาะ
> - จำนวน `"1149"` ไม่เป็นจำนวนรูปงามเพราะไม่ว่าจะแบ่งสตริงดังกล่าวอย่างไร ก็ไม่สามารถเขียนในรูปของ string concatenation ของจำนวนเฉพาะได้
> - จำนวน `"7007"` ไม่เป็นจำนวนรูปงามเพราะไม่ว่าจะแบ่งสตริงดังกล่าวอย่างไร ก็ไม่สามารถเขียนในรูปของ string concatenation ของจำนวนเฉพาะที่ไม่ขึ้นต้นด้วย 0 ได้

โจทย์ จงหาว่าจำนวนเต็ม 5 หลักที่จัดว่าเป็น “จำนวนรูปงาม” มีทั้งหมดกี่จำนวน?

คำถามที่ 44. จงคำนวณหาลำดับของจำนวนเฉพาะที่เรียงอยู่ติดกันที่ยาวที่สุด ที่มีผลรวมเท่ากับ 1 ล้าน (1,000,000) พอดี

1. ลำดับดังกล่าวมีความยาวเท่าใด?
2. จำนวนเฉพาะตัวแรกและตัวสุดท้ายของลำดับดังกล่าวคือจำนวนใด?

คำถามที่ 45. กำหนดให้มีเหรียญบาทอยู่เหรียญหนึ่ง เมื่อโยนแต่ละครั้งแล้วจะให้ผลลัพธ์เป็นหัว (head) ด้วยความน่าจะเป็น 49% และเป็นก้อย (tail) ด้วยความน่าจะเป็น 51%

เป้าหมายของเราคือ เราจะใช้เหรียญบาทดังกล่าวในการเขียนโปรแกรมเพื่อสุ่มตัวเลข 0 หรือ 1 อย่างเที่ยงธรรมที่สุดเท่าที่เป็นไปได้ (นั่นคือโอกาสในการออกหัวหรือก้อยต้องเท่ากันหรือใกล้เคียงกันให้มากที่สุด)

โดยโปรแกรมดังกล่าวจะมี API ที่ต่อเข้าถึงการโยนเหรียญบาทข้างต้น ด้วยการเรียกใช้ฟังก์ชัน `toss()` ซึ่งจะคืนค่าสตริง `"head"` หรือ `"tail"` เท่านั้น)

จงเรียงลำดับความยุติธรรมของโปรแกรม A, B, C, D ต่อไปนี้ จากยุติธรรมมากที่สุดไปยังน้อยที่สุด

△ โปรแกรม A

```
repeat until terminate
    count_heads := 0
    for i := 1 to 100 do:
        if toss() == "head" then count_heads := count_heads + 1
    end
    if count_heads < 49 then
        output 0 and terminate
    else if count_heads > 49
        output 1 and terminate
    end
end
```

△ โปรแกรม B

```
repeat until break
    result := toss()
    if result == "tail" then
        break  # break from loop
    end
end
if toss() := "head" then
    output 0
else
    output 1
end
terminate
```

△ โปรแกรม C

```
repeat until terminate
    count_heads := 0
    if toss() == "head" then count_heads := count_heads + 1
    if toss() == "head" then count_heads := count_heads + 1
    if count_heads ≠ 1 then
        output count_heads/2 and terminate
    end
end
```

△ โปรแกรม D

```
repeat until terminate
    count_heads := 0
    if toss() == "head" then count_heads := count_heads + 1
    result := count_heads
    if toss() == "head" then count_heads := count_heads + 1
    if count_heads == 1 then
        output result and terminate
    end
end
```

คำถามที่ 46. สมมติว่านายกสิกรมีโปรแกรมที่เขียนขึ้นมาเองอยู่โปรแกรมหนึ่ง เพื่อใช้ประมวลผลข้อมูลขนาดใหญ่มหาศาล ซึ่งโปรแกรมนี้อาจจะต้องใช้เวลาทำงานต่อเนื่องหลายชั่วโมง และไม่อาจคาดเดาได้ว่าโปรแกรมนี้จะใช้เวลาประมวลผลกี่ชั่วโมง (จนกว่าโปรแกรมจะรันเสร็จสิ้นเท่านั้น)

นายกสิกรตัดสินใจใช้บริการคลาวด์แห่งหนึ่งเพื่อรันโปรแกรมของตัวเอง โดยคลาวด์ดังกล่าวมีนโยบายการคิดค่าบริการดังนี้

- หากเช่าเป็นชั่วโมง คิดชั่วโมงละ 1,200 บาท
- ระหว่างที่โปรแกรมของนายกสิกรกำลังรันอยู่และค่าใช้บริการถูกคิดเป็นค่าเช่ารายชั่วโมงอยู่นั้น นายกสิกรสามารถเลือก upgrade บริการคลาวด์ให้คิดค่าบริการแบบ Flat rate เมื่อใดก็ได้ โดยคิดเหมาจ่ายในราคา 10,000 บาท และจะไม่ได้ค่าเช่ารายชั่วโมงก่อนหน้านั้นคืน (กล่าวคือ upgrade เร็วย่อมคุ้มค่ากว่า upgrade ช้า)

เมื่อพิจารณาตัวอย่างสถานการณ์ที่ปรากฏทางด้านข้างแล้ว สังเกตว่าปัญหามีอยู่สองส่วนคือ

1. นายกสิกร ไม่สามารถคาดเดาระยะเวลาที่ Program จะใช้ประมวลผลได้ล่วงหน้า
2. ถ้าเรา upgrade เร็วหรือช้าเกินไป ค่าใช้จ่ายในกรณี worst-case อาจจะสูงเกินกว่าที่ควรจะเป็น อันเนื่องมาจากสาเหตุข้อแรก

นายกสิกรต้องการต้องการคิดกลยุทธ (Strategy) เพื่อวางแผนใช้บริการคลาวด์ดังกล่าวให้คุ้มค่าทุกบาททุกสตางค์มากที่สุดเท่าที่เป็นไปได้ แม้ว่านายกสิกรจะคาดเดาจำนวนชั่วโมงที่โปรแกรมจะใช้รันไม่ได้เลย กล่าวคือค่าใช้จ่ายจริง (Actual cost) จะต้องมีปริมาณน้อยที่สุดเมื่อเทียบอัตราส่วนกับค่าใช้จ่ายที่ถูกที่สุดที่เป็นไปได้ (Optimal cost)

อยากทราบว่านายกสิกรควรวางแผนเช่าหรือ upgrade อย่างไรจึงจะ minimize ค่าของ $\frac{\text{Actual cost}}{\text{Optimal cost}}$ ให้เหลือน้อยที่สุด ไม่ว่าโปรแกรมของนายกสิกรจะใช้เวลารันกี่ชั่วโมงก็ตาม?

○ นายกสิกรควร upgrade เป็นการคิดค่าบริการแบบ Flat rate ทันทีโดยไม่เสียค่าเช่ารายชั่วโมง (แปลว่านายกสิกรจะเสียเงิน 10,000 บาทเสมอ ไม่ว่าโปรแกรมจะรันกี่ชั่วโมง)

○ นายกสิกรควรตัดสินใจเสียค่าเช่ารายชั่วโมงตลอดไป ไม่ควร upgrade เลย

*○ นายกสิกรควรเสียค่าเช่ารายชั่วโมงเป็นเวลาไม่เกิน h ชั่วโมง แล้วจึง upgrade หากโปรแกรมยังรันไม่เสร็จเมื่อเวลาผ่านไป h ชั่วโมงพอดี
(**หมายเหตุ:** หากเลือกตัวเลือกนี้ กรุณาระบุจำนวนเต็ม h ดังกล่าวด้วย)

ตัวอย่าง ลองพิจารณาสถานการณ์สมมติดังต่อไปนี้

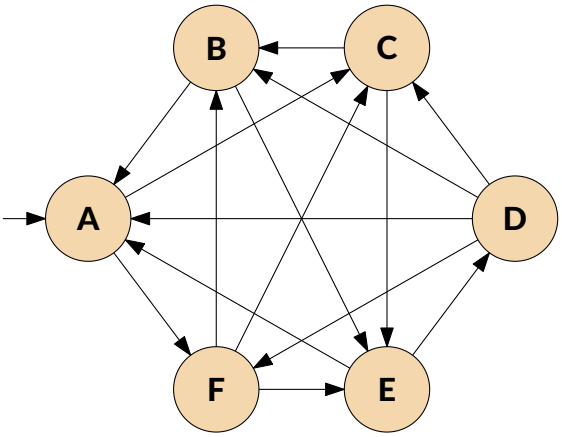
- สมมติว่านายกสิกรเริ่มนำโปรแกรมนี้ไปรันในคลาวด์นี้แบบคิดค่าเช่ารายชั่วโมง เมื่อเวลาผ่านไป 4 ชั่วโมงพบว่าโปรแกรมนี้ยังประมวลผลไม่เสร็จ นายกสิกร จึงตัดสินใจเลือก upgrade เป็น Flat rate แต่จากนั้นเมื่อเวลาผ่านไปอีก 2 ชั่วโมงโปรแกรมจึงรันเสร็จสิ้น
  จึงเท่ากับว่าค่าใช้จ่ายที่ถูกที่สุดที่เป็นไปได้ในทางทฤษฎี (Optimal cost) ในกรณีนี้คือ 7,200 บาท แต่นายกสิกรต้องเสียเงินจริง (Actual cost) ไปถึง 14,800 บาท ซึ่งคิดเป็น 2.056 เท่าของ Optimal cost
- ในอีกเหตุการณ์หนึ่ง สมมติว่านายกสิกรรันโปรแกรมดังกล่าวจนครบ 6 ชั่วโมง พบว่ายังประมวลผลไม่เสร็จสิ้น จึงลองเสี่ยง upgrade เป็น Flat rate ดู แต่สุดท้ายแล้วโปรแกรมนี้ใช้เวลาถึง 20 ชั่วโมงในการรันจนเสร็จสิ้น
  จึงเท่ากับว่านายกสิกรเสีย Actual cost ไป 17,200 บาท ซึ่งคิดเป็น 1.72 เท่าของ Optimal cost ที่เกิดจากการเหมาจ่ายตั้งแต่แรกที่ 10,000 บาท

คำถามที่ 47. ชุดของจำนวนชุดหนึ่งจะมี 3-term arithmetic progression (**3TAP**) หากมีจำนวน 3 จำนวนในชุดดังกล่าวที่เรียงเป็นลำดับเลขคณิต ยกตัวอย่างเช่น

- ในชุดของจำนวน 1, 2, 4, 5, 7, 11 จะมี **3TAP** เนื่องจากมีจำนวน 3 จำนวน ซึ่งได้แก่ 1, 4, 7 เป็นลำดับเลขคณิต
- ในชุดของจำนวน 1, 2, 5, 7, 10, 11 นั้นไม่มี **3TAP** อยู่เลย

โจทย์ จงหาชุดของจำนวนเต็มจากจำนวนเต็มตั้งแต่ 1 ไปจนถึง 30 ที่มีจำนวนสมาชิกมากที่สุด และไม่มี **3TAP** อยู่เลย

คำถามที่ 48. จงพิจารณากราฟที่กำหนดให้ต่อไปนี้



ให้พิจารณากราฟที่กำหนดให้ เราจะเขียนโปรแกรมเพื่อเดิน (traverse) บนกราฟดังกล่าว โดยมีเงื่อนไขดังนี้

1. เราจะเริ่มต้นการเดินจากโหนด A
2. จากโหนดหนึ่ง ๆ เราจะเดินไปยังโหนดถัดไปเฉพาะโหนดที่มีลูกศรชี้ไปหาเสมอ

   หากมีลูกศรชี้ไปยังโหนดอื่น ๆ มากกว่า 1 โหนด ให้เลือกเดินไปโหนดถัดไปโหนดใดก็ได้จากตัวเลือกนั้น ๆ (เช่น จากโหนด B เราสามารถเดินไปยังโหนด A หรือ E โหนดใดก็ได้)
3. เราจะเดินบนกราฟนี้ไปเรื่อย ๆ ไม่มีหยุดอยู่กับที่ (เสมือนว่าโปรแกรมของเรามี infinite loop)

จากการเดินบนกราฟอย่างไม่สิ้นสุดตามคำอธิบายข้างต้น ข้อใดต่อไปนี้ถูกต้องบ้าง?

☐ **ประพจน์ P:** รับประกันว่าในการเดินดังกล่าว “เราจะเดินพบโหนด A อนันต์ครั้ง” หรือไม่เช่นนั้น “เราจะเดินพบโหนด D อนันต์ครั้ง”

☐ **ประพจน์ Q:** รับประกันว่าในการเดินดังกล่าว “เราจะเดินพบโหนด B อนันต์ครั้ง” หรือไม่เช่นนั้น “เราจะเดินพบโหนด E อนันต์ครั้ง”

☐ **ประพจน์ R:** รับประกันว่าในการเดินดังกล่าว “เราจะเดินพบโหนด C อนันต์ครั้ง” หรือไม่เช่นนั้น “เราจะเดินพบโหนด F อนันต์ครั้ง”

คำถามที่ 49. ให้พิจารณาสมการดทางด้านขวาต่อไปนี้ โดยมีเงื่อนไขว่า

- แต่จะพจน์ (term) ของการบวกคือจำนวนเต็มที่เลขโดดแต่ละตัวถูกเขียนแทนด้วยอักขระภาษาอังกฤษ 1 ตัว
- อักขระตัวเดียวกันแทนเลขโดดเดียวกัน อักขระที่ต่างกันแทนเลขโดดที่ไม่ซ้ำกัน
- ไม่มีพจน์ใดที่อักขระตัวแรกแทนเลขโดด 0

อยากทราบว่าคำว่า **BRACKET** แทนจำนวนใด?

TEACH
KBTG + TECH
HACKER

คำถามที่ 50. กำหนดให้มีจำนวนเต็ม ได้แก่ 1, 2, 3, ..., 12

ให้แบ่งกลุ่มของจำนวนเต็มดังกล่าวออกเป็น 2 กลุ่ม โดยแต่ละกลุ่มจะต้องมีจำนวนเต็มอย่างน้อย 1 จำนวน จากนั้นให้หาผลคูณของจำนวนในแต่ละกลุ่ม

โจทย์ อยากทราบว่าผลต่างของผลคูณจากทั้งสองกลุ่ม มีค่าน้อยที่สุดเท่าใด?

คำถามที่ 51. จงศึกษาตัวอย่างโจทย์ต่อไปนี้ แล้วจึงแก้โจทย์ในช่วงท้ายของคำถาม

ตัวอย่าง สมมติว่ามีลูกอมยี่ห้อหนึ่ง ถูกบรรจุขายเป็นแพ็คถุง 2 ขนาด คือแบบถุงละ 3 ลูก หรือ 5 ลูก สังเกตว่า

- เราสามารถซื้อลูกอมให้มีลูกอมเป็นจำนวนรวม 8 ลูกได้ (คือซื้อทั้งสองแบบ แบบละ 1 ถุง แล้วนำลูกอมมาเทรวมกัน)
- แต่เราไม่มีวิธีที่สามารถซื้อลูกอมเป็นจำนวนรวม 7 ลูกพอดีได้เลย

โจทย์ สมมติว่ามีลูกอมอีกยี่ห้อหนึ่ง ถูกบรรจุขายเป็นแพ็ค 3 ขนาด คือแบบถุงละ 6 ลูก หรือ 15 ลูก หรือ 40 ลูก ตามลำดับ แล้วจำนวนลูกอมที่มากที่สุดที่เราไม่สามารถหาวิธีซื้อให้พอดีจากการผสมแพ็คลูกอมทั้ง 3 แบบนี้ มีจำนวนลูกอมเท่าใด?

คำถามที่ 52. ในปัจจุบัน เงินตราที่นิยมใช้กันแพร่หลายในประเทศไทยประกอบไปด้วยเหรียญกษาปณ์หรือธนบัตรชนิดราคา 1 บาท, 2 บาท, 5 บาท, 10 บาท, 20 บาท, 50 บาท, 100 บาท, 500 บาท และ 1000 บาท ตามลำดับ

สมมติว่าเราต้องการชำระยอดหนี้ก้อนหนึ่งซึ่งมีมูลค่า d บาท โดยมี*เป้าหมาย*ว่าจะต้องใช้เงินตราเป็นจำนวนน้อยที่สุดเพื่อชำระหนี้ดังกล่าวให้พอดี สังเกตว่าเราสามารถใช้ **Greedy algorithm** ดังต่อไปนี้ เพื่อบรรลุ*เป้าหมาย*ดังกล่าวได้[9] ไม่ว่ายอดหนี้ d จะมีมูลค่ากี่บาทก็ตาม

> **Greedy algorithm.** เราจะเลือกเหรียญกษาปณ์หรือธนบัตรที่มีมูลค่ามากที่สุดที่เป็นไปได้ที่ไม่เกินยอดหนี้ นำไปหักจากยอดหนี้ ทำเช่นนี้ไปเรื่อย ๆ จนกว่ายอดหนี้จะลดลงเหลือ 0 บาท
>
> ยกตัวอย่างเช่น หากเราต้องการชำระหนี้มูลค่า $d = 94$ บาท เราสามารถจ่ายด้วยเหรียญกษาปณ์หรือธนบัตรที่มีมูลค่า $50 + 20 + 20 + 2 + 2$ ตามลำดับ ซึ่งหมายความว่าเราใช้จำนวนเงินตรา 5 อัน ซึ่งน้อยที่สุดที่เพียงพอจะชำระหนี้ดังกล่าวพอดี

อย่างไรก็ดี สมมติว่าวันหนึ่ง ประเทศไทยจะเพิ่มเหรียญกษาปณ์หรือธนบัตรชนิดราคาใหม่จำนวน 1 ชนิดราคาเข้ามาในระบบ สังเกตว่า

- ถ้าสมมติว่าประเทศไทยตัดสินใจเพิ่มเหรียญกษาปณ์ชนิดราคา 4 บาทเข้ามาในระบบ แล้ว **Greedy algorithm** ข้างต้นจะไม่รับประกันว่าจะให้ผลลัพธ์ที่ optimal เสมอไป (เช่น หากต้องการชำระเงิน $d = 8$ บาท **Greedy algorithm** จะจ่ายด้วยเหรียญ $5 + 2 + 1$ บาทตามลำดับ แทนการใช้วิธี $4 + 4$ บาทที่ใช้จำนวนเงินตราน้อยกว่า)
- แต่ถ้าเราเพิ่มเหรียญกษาปณ์ชนิดราคา 3 บาทแล้ว **Greedy algorithm** จะยังคงให้จำนวนเงินตราที่ optimal อยู่ไม่ว่ายอดหนี้ d จะมีมูลค่าเท่าใดก็ตาม

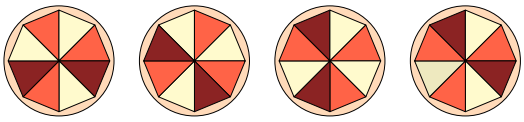
จงหาเงินตราที่มีชนิดราคาสูงที่สุด 1 ชนิดราคา ที่เมื่อเพิ่มเข้ามาในระบบแล้วจะทำให้ **Greedy algorithm** ให้ผลลัพธ์ไม่เป็น optimal สำหรับยอดหนี้บางจำนวน[10]

[9] หมายความว่า **Greedy algorithm** ให้ผลลัพธ์เป็น *optimal* สำหรับเงินตราที่ระบุไว้ข้างต้น

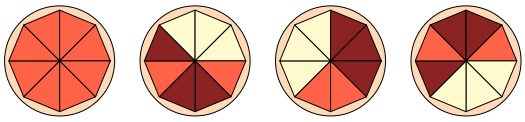
[10] พร้อมทั้งยกตัวอย่างค้านว่า ยอดหนี้ d มูลค่าเท่าใดที่ทำให้ **Greedy algorithm** ให้ผลลัพธ์ที่ไม่เป็น optimal

คำถามที่ 53. พิซซ่าถาดหนึ่งถูกตัดแบ่งด้วยรัศมีออกเป็นพิซซ่าชิ้นย่อย ๆ 8 ชิ้นที่มาขนาดเท่า ๆ กัน ต่อจากนั้นพิซซ่าแต่ละชิ้นย่อยทุกชิ้นจะถูกทาด้วยซอส 1 ใน 3 ชนิด (ได้แก่ ซอสขาว ซอสแดง หรือซอสน้ำตาล)

นิยาม พิซซ่าสองถาดจะมีหน้าตาแบบเดียวกัน ก็ต่อเมื่อ หากสามารถหมุนถาดพิซซ่าถาดหนึ่งให้มีหน้าตาเหมือนกันพิซซ่าอีกถาดหนึ่งได้



ตัวอย่างของพิซซ่าที่หน้าตาเหมือนกัน



ตัวอย่างของพิซซ่าที่หน้าตาแตกต่างกันทั้งหมด

โจทย์ อยากทราบว่าเราจะได้ถาดพิซซ่าที่หน้าตามีการทาซอสออกมาแตกต่างกันทั้งหมดกี่แบบ?

คำถามที่ 54. จงเติมจำนวนเต็ม 1 ถึง 16 ลงในตารางขนาด 4×4 มาอย่างน้อย 1 วิธี ซึ่งสอดคล้องกับเงื่อนไขดังต่อไปนี้

?	?	?	?
?	?	?	?
?	?	?	?
?	?	?	?

- จำนวนในแต่ละช่องต้องไม่ซ้ำกัน
- จำนวนในแต่ละแถว จะต้องเรียงลำดับจากน้อยไปมาก จากซ้ายไปขวา
- จำนวนในแต่ละคอลัมน์ จะต้องเรียงลำดับจากน้อยไปมาก จากบนลงล่าง
- สำหรับทุก $k \in \{1, 2, 3, 4\}$ ผลรวมของจำนวนในแถวที่ k (นับจากบน) จะต้องเท่ากับผลรวมของจำนวนในคอลัมน์ที่ k (นับจากซ้าย)

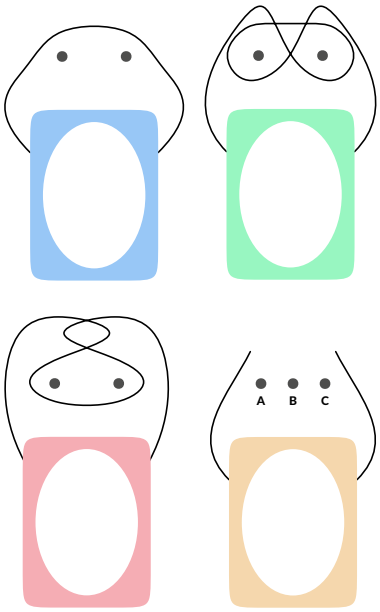
คำถามที่ 55. กำหนดให้ `validate_array(A)` คือฟังก์ชันที่รับ input argument เป็น 0-indexed array `A` ของจำนวนเต็ม และให้ output result เป็นค่า boolean ที่เป็น `true` หรือ `false` เท่านั้น ฟังก์ชันดังกล่าว สามารถเขียนเป็น pseudocode ได้ดังนี้

```
1  function validate_array(A[0 ... n-1]):
2      return (0 ≤ A[i] ≤ n-1 and A[i] == A[A[i]] for each i := 0 to n-1)
                 and (A[i-1] ≤ A[i] for each i := 1 to n-1)
3  end
```

ตัวอย่างของการเรียกใช้ฟังก์ชันข้างต้น

- `validate_array(A = [0, 1, 1, 3])  # ⇒ true`
- `validate_array(A = [2, 2, 2, 2])  # ⇒ true`
- `validate_array(A = [1, 2, 3, 3])  # ⇒ false`
- `validate_array(A = [3, 1, 1, 3])  # ⇒ false`

โจทย์ จงหาจำนวนรูปแบบทั้งหมดของ input array `A` ที่มีจำนวนสมาชิก 20 ตัวที่ทำให้ `validate_array(A)` คืนค่าออกมาเป็น `true`?



คำถามที่ 56. เราต้องการแขวนกรอบรูปอันหนึ่งด้วยเชือก 1 เส้นที่ร้อยจากขอบซ้ายของรูป คล้องเหนือหมุดบนกำแพงบางอัน แล้วมาร้อยเชื่อมกับขอบขวาของรูป

ตัวอย่าง หมุด 2 ตัว — สมมติว่าเราเลือกที่จะแขวนรูปโดยการร้อยเชือกกับหมุด 2 ตัวในรูปแบบต่าง ๆ ดังนี้

- **รูปสีฟ้า:** หากเราแขวนรูปด้วยวิธีปกติ (ดังรูปสีฟ้า) เราพบว่าหากหมุดตัวใดตัวหนึ่งถูกดึงออกไป หมุดอีกตัวหนึ่งจะยังสามารถรั้งกรอบภาพไม่ให้ตกตามแรงโน้มถ่วงได้
- **รูปสีเขียว:** แต่หากเราแขวนอีกแบบหนึ่ง (ดังรูปสีเขียว) เราพบว่ากรอบภาพจะยังคงแขวนได้เช่นกัน แต่หากหมุดตัวใดตัวหนึ่งถูกดึงออกไป กรอบภาพจะหลุดลงมาทันทีแม้ว่าหมุดอีกตัวจะยังยึดกำแพงอยู่ก็ตาม

โจทย์ หมุด 3 ตัว — หากเรามีหมุดสามอัน คือ A, B, C เรียงจากซ้ายไปขวา (ดู**รูปสีส้ม**ประกอบ) เราต้องการร้อยเชือกรอบหมุดให้สอดคล้องกับเงื่อนไขต่อไปนี้

- หากดึงหมุด A หรือหมุด C อันใดอันหนึ่ง รูปจะยังรั้งไว้ได้ ไม่หล่นลงมา
- หากดึงหมุด B หมุดเดียว รูปจะหล่นลงมาทันที
- หากดึงทั้งหมุด A และหมุด C ทั้งสองหมุด รูปจะหล่นลงมาเช่นกัน

จงหาวิธีแขวน**รูปสีส้ม**ที่สอดคล้องกับเงื่อนไขข้างต้น[11]

[11] นอกจากนี้ ยังมีข้อห้ามเกี่ยวกับการแขวนรูปดังนี้
- ห้ามใช้เชือกคล้องกันเอง (ดังเช่น**รูปสีแดง**) ในรูปนี้ให้ถือว่าเชือกขดทับกันแต่ไม่ไขว้กัน ซึ่งแปลว่ารูปจะไม่ถูกแขวนได้สำเร็จแต่แรก
- นอกจากนั้น ห้ามมัดเชือกกันเองเป็นปมเพื่อแขวนรูป

คำถามที่ 57. มีตู้เซฟอยู่ตู้หนึ่ง ตู้เซฟนี้ถูกล็อกด้วยหมายเลขปริศนา Q ซึ่งมีความยาว 6 หลัก (เราเรียก Q ว่า*รหัสผ่านจริง*)

ทุก ๆ ครั้งที่เราป้อนรหัสเซฟเพื่อเปิดตู้เซฟตู้นี้ หากเราป้อนรหัสไม่ถูกต้อง ตู้เซฟจะมีเสียงร้องพร้อมทั้งยังมีข้อความตอบกลับ (response message) ว่า

> เลขโดดที่อยู่ติดกันที่ยาวที่สุดที่ปรากฏทั้งในรหัสผ่านจริง Q
> และ*รหัสผ่านที่ป้อนผิด*นั้นมีความยาวเท่าใด[12]

ต่อไปนี้คือประวัติของการลองป้อนรหัสเซฟแก่ตู้เซฟนี้ทั้งสิ้น 9 ครั้ง พร้อมทั้ง response message ในแต่ละครั้ง

รหัสผ่านที่ป้อน A	ข้อความตอบกลับ LCS(A, Q)
027292	1
135135	0
257015	2
362447	1
470619	3
560968	1
674669	1
822642	1
903287	3

จงหารหัสผ่านจริง Q ของตู้เซฟนี้จากข้อมูลข้างต้น

[12] พูดอีกนัยหนึ่งคือ response message จะเป็นความยาวของ **Longest common substring** (LCS) ระหว่างรหัสผ่านจริง Q กับ*รหัสผ่านที่ป้อนผิด*

ตัวอย่าง สมมติว่ารหัสผ่านจริงของตู้เซฟคือ

```
Q = 123456
```

แต่เราป้อนรหัสตู้เซฟเป็น

```
A = 134579
```

แล้วตู้เซฟจะมี response message ตอบกลับออกมาเป็น

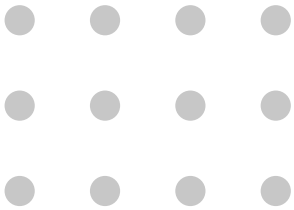
```
LCS(A, Q) = 3
```

คำถามที่ 58. ปัญหา **Traveling salesperson problem** (TSP) คือปัญหากราฟที่มีเป้าหมายเพื่อค้นหาเส้นทางจากโหนดหนึ่ง ๆ ไปเยือนโหนดอื่น ๆ ทุกโหนด ก่อนกลับมายังโหนดเริ่มต้น โดยใช้ระยะทางรวมน้อยที่สุด

สำหรับโจทย์ข้อนี้ให้พิจารณา Greedy algorithm เพื่อแก้ปัญหา TSP ข้างต้น ซึ่งมีขั้นตอนวิธีดังนี้[13]

1. เริ่มต้นจากโหนดใดก็ได้*
2. จากโหนดปัจจุบัน ให้เลือกไปเยือนโหนดถัดไปที่อยู่ใกล้ที่สุดที่ยังไม่เคยเดินทางไปเยือน
   หมายเหตุ: หากมีหลายโหนดที่สอดคล้องกับเงื่อนไขข้างต้น ให้เลือกไปเยือนโหนดใดก็ได้ในบรรดาโหนดเหล่านั้น*
3. เมื่อกระทำตามข้อ 2 จนเยือนครบทุกโหนดแล้ว ให้เดินกลับไปยังโหนดที่เป็นจุดเริ่มต้น

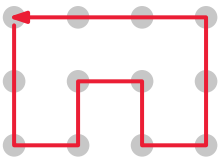
อย่างไรก็ดี Greedy algorithm นี้ไม่รับประกันว่าจะให้คำตอบที่ดีที่สุด (หรือ optimal solution) เสมอไป โจทย์ข้อนี้เราจะพิจารณาหาตัวอย่างค้านที่ Greedy algorithm ข้างต้นให้ผลลัพธ์ที่แย่มากเป็นพิเศษ



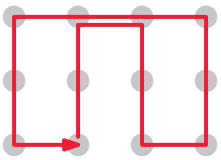
เคสตัวอย่าง ให้พิจารณากราฟข้างต้น ซึ่งประกอบด้วยจุดบน Euclidean space ที่เรียงเป็นตารางสี่เหลี่ยมผืนผ้า 3 แถว แถวละ 4 จุด นอกจากนั้นแต่ละแถวและแต่ละคอลัมน์ห่างกัน 1 หน่วย

- หากเราโชคดีหน่อย Greedy algorithm อาจจะค้นพบเส้นทาง (a) ที่แสดงทางด้านขวามือ เส้นทางดังกล่าวมีระยะทางเท่ากับ 12 หน่วย ซึ่งเป็น optimal solution
- แต่หากโชคไม่ค่อยดี Greedy algorithm มีโอกาสพบเส้นทางอื่นเช่น (b) หรือ (c) ซึ่งมีระยะทางรวมเท่ากับ 14 หน่วย และ $11 + \sqrt{5} \approx 13.236$ หน่วยตามลำดับ

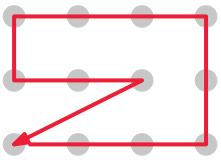
จงหาเส้นทางที่ Greedy algorithm นี้มีโอกาสค้นพบ ที่มีระยะทางมากกว่า $\frac{160}{9}$ หน่วย



(a)



(b)



(c)

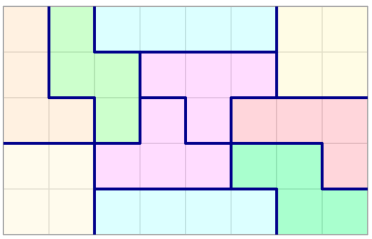
คำถามที่ 59. จงพิจารณาระบบการถ่ายเทพลังงานในระบบห่วงโซ่อาหารของสัตว์กลุ่มหนึ่ง โดยที่สัตว์ทุกตัวในกลุ่มนี้มีโอกาสเป็นทั้งผู้ล่าและเหยื่อกับสัตว์ตัวอื่น ๆ ได้ทุกตัว

- กำหนดให้สัตว์ทุกตัวมีพลังงานสะสมเริ่มต้นในร่างกาย 1,000,000 KCal
- เมื่อเกิดการล่าเหยื่อขึ้น หากสัตว์ A จับสัตว์ B เป็นอาหารแล้วพบว่า B จะเสียชีวิตไป และ A จะได้รับถ่ายทอดพลังงานสะสมครึ่งหนึ่งจาก B[14]
- สัตว์ทุกตัวในระบบจะจับเหยื่อตัวอื่น ๆ เป็นอาหารได้ไม่เกิน 3 ตัว

หากในตอนแรกมีสัตว์ในระบบนิเวศนี้ทั้งสิ้น 1,000 ตัว แล้วจึงเกิดการล่ากันเองขึ้นจนเหลือสัตว์ที่อยู่รอดตัวเดียว สัตว์ตัวดังกล่าวนี้จะมีพลังงานสะสมตอนท้ายอย่างน้อยที่สุดและอย่างมากที่สุดกี่ KCal? (ให้ตอบเป็นจำนวนเต็มโดยปัดเศษทิ้ง)

[13] อัลกอริทึมนี้มีความเป็น non-deterministic เนื่องจากอัลกอริทึมมีการตัดสินใจแบบ arbitrary ได้เมื่อมีตัวเลือกปรากฏขึ้น ซึ่งมีผลทำให้ได้คำตอบที่แตกต่างออกไป
ขั้นตอนของอัลกอริทึมดังกล่าวที่มีความเป็น non-deterministic จะมีเครื่องหมายดอกจันกำกับ

[14] **หมายเหตุ** เราสมมติว่าสัตว์แต่ละตัวจะไม่เสียพลังงานใด ๆ กับกิจกรรมอื่น ๆ เลย พลังงานจะหายไปจากระบบจากการที่เหยื่อถูกบริโภคเท่านั้น



[15] **หมายเหตุ** ความยาวของเส้นปากกา เรานับเฉพาะเส้นที่คั่นระหว่างชิ้นส่วน TETRIS® สองชิ้น และไม่นับเส้นกรอบ

คำถามที่ 60. จงศึกษาตัวอย่างโจทย์ต่อไปนี้ แล้วจึงแก้โจทย์ในช่วงท้ายของคำถาม

ตัวอย่าง พิจารณารูปที่ปรากฏทางซ้ายมือ เป็นพื้นที่สี่เหลี่ยมผืนผ้าขนาด 8×5 หน่วย ซึ่งถูกตัดแบ่งออกเป็นชิ้นส่วน TETRIS® ประกอบกัน 10 ชิ้น ด้วยการขีดเส้นปากกาดังรูป (แสดงด้วยเส้นสีน้ำเงินทึบภายในกรอบสี่เหลี่ยม) พบว่าความยาวเส้นปากกาดังกล่าวคือ 35 หน่วย[15]

สังเกตว่า หากพื้นที่สี่เหลี่ยมผืนผ้านี้ถูกแบ่งเป็นชิ้นส่วน TETRIS® ในรูปแบบที่ต่างออกไป อาจจะต้องขีดเส้นปากกาที่มีความยาวรวมมากกว่าหรือน้อยกว่า 35 หน่วยก็ได้

โจทย์ ให้พิจารณาพื้นที่สี่เหลี่ยมผืนผ้าขนาด 12×9 หน่วย หากเราใช้ปากกขีดเส้นภายในกรอบสี่เหลี่ยม เพื่อแบ่งให้พื้นที่ว่างในกรอบเป็นชิ้นส่วน TETRIS® จำนวน 27 ชิ้น

จงคำนวณหาว่า เส้นปากกาที่ขีดในกรอบสี่เหลี่ยมดังกล่าวจะมีความยาวรวมน้อยที่สุดและมากที่สุดกี่หน่วย?

คำถามที่ 61. ในโรงเรียนประถมศึกษาแห่งหนึ่ง มีห้องเรียนชั้น ป.1 อยู่ 2 ห้อง ได้แก่ ห้องทานตะวัน และห้องกุหลาบ

เด็กชาย K เรียนอยู่ในห้องทานตะวัน และมีเพื่อนสนิทอยู่ 8 คน ได้แก่ A, B, C, D, E, F, G และ H บางคนเรียนอยู่ห้องทานตะวันเช่นเดียวกับเด็กชาย K ส่วนบางคนเรียนอยู่ห้องกุหลาบ คนละห้องกับเด็กชาย K

ต่อไปนี้คำบอกกล่าวของครูใหญ่ทั้งสิ้น 14 ประโยค

1. A กับ B เรียนอยู่คนละห้องกัน
2. B กับ C เรียนอยู่คนละห้องกัน
3. C กับ D เรียนอยู่คนละห้องกัน
4. D กับ E เรียนอยู่คนละห้องกัน
5. E กับ A เรียนอยู่คนละห้องกัน
6. A กับ F เรียนอยู่คนละห้องกัน
7. F กับ G เรียนอยู่คนละห้องกัน
8. G กับ H เรียนอยู่คนละห้องกัน
9. H กับ D เรียนอยู่คนละห้องกัน
10. D กับ B เรียนอยู่คนละห้องกัน
11. B กับ E เรียนอยู่คนละห้องกัน
12. E กับ C เรียนอยู่คนละห้องกัน
13. C กับ K เรียนอยู่คนละห้องกัน
14. K กับ G เรียนอยู่คนละห้องกัน

ในบรรดา 14 ประโยคข้างต้น มี 2 ประโยคที่ไม่เป็นความจริง จงพิจารณาข้อมูลข้างต้น แล้วตอบคำถามต่อไปนี้

- ประโยคใดเป็นเท็จบ้าง?
- ใครเรียนอยู่ห้องทานตะวันเช่นเดียวกับเด็กชาย K บ้าง?

หมวดที่ 3
โจทย์ Coding

โจทย์ประเภท Coding เป็นโจทย์สไตล์ competitive programming ที่ผู้เข้าแข่งขันจะต้องทำความเข้าใจโจทย์ปัญหาเชิงคำนวณ แล้วเลือกโครงสร้างข้อมูลหรืออัลกอริทึมที่เหมาะสมมาแก้ปัญหาดังกล่าว ผู้เข้าแข่งขันจะต้องเขียนโปรแกรมด้วยภาษาที่ตนเองถนัดมารับ input ไปประมวลผล แล้วจึง output ผลลัพธ์ที่ประมวลผลได้

คำถามที่ 62. Island Counting

Background

ในเกมย้อนยุคเกมหนึ่ง มีการวาดแผนที่ด้วย ASCII Art โดยประกอบไปด้วยอักขระ ASCII ที่เรียงตัวกันเป็นสี่เหลี่ยมผืนผ้าจำนวน R แถว แถวละ C ตัว

อักขระแต่ละตัวจะแทนช่อง 1 ช่องและมีค่าได้ 2 แบบ ได้แก่

- ผืนดิน ซึ่งเราจะขอแทนด้วย '#'
- ผืนน้ำ ซึ่งเราจะขอแทนด้วย '.'

นอกจากนี้เราจะสมมติว่า รอบนอกสี่เหลี่ยมผืนผ้านี้จะมีแต่น้ำทะเลอันกว้างใหญ่ไพศาลและไม่มีผืนดินอยู่เลย

ตัวอย่าง* นี่คือแผนที่ตัวอย่าง ซึ่งมีขนาด $R = 6$ และ $C = 18$

```
.................##
..#####...##......
.##...##..##...###
.#..#..#.......#..
.#....##..#.#..#.#
.######....#...#.#
```

What is an island?

จากแผนที่ในลักษณะข้างต้น เรานิยามภูมิประเทศทีของเกาะดังนี้

- หากเราเริ่มต้นจากตำแหน่งผืนดินใด ๆ ในแผนที่ แล้วสามารถเดินทางไปยังผืนดินอื่น ๆ ข้างเคียงได้ด้วยการเดิน ขึ้น–ลง–ซ้าย–ขวา ไปเรื่อย ๆ ได้ ให้ถือว่าผืนดินที่เดินไปถึงทั้งหมดเหล่านั้นเป็นส่วนหนึ่งของเกาะเดียวกัน

- มากไปกว่านั้น จากการเดินข้างต้น หากเราเริ่มเดินจนวกกลับมาที่ผืนดินเริ่มต้น ให้ถือว่าบริเวณที่ถูกรายล้อมด้วยการเดินข้างต้นเป็นส่วนหนึ่งของเกาะเดียวกันเช่นกัน
  - สังเกตว่าอาจมีทะเลสาบที่ถูกรายล้อมด้วยผืนดินของเกาะเกาะหนึ่ง ซึ่งผืนน้ำดังกล่าวจะถูกนับไปส่วนหนึ่งของเกาะนั้นด้วย
  - ไม่เพียงแค่นั้น ผืนดินที่ซ้อนอยู่ภายในทะเลสาบดังกล่าว ก็ยังถือว่าเป็นส่วนของเกาะภายนอกด้วย ไม่นับเป็นเกาะแยกต่างหาก

ตัวอย่าง* จากแผนที่ตัวอย่างเดิมข้างต้น เราสามารถเขียนใหม่โดยส่วนของเกาะเดียวกันเขียนด้วยตัวอักษรเดียวกันได้ดังนี้ (สังเกตได้ว่าแผนที่นี้จะมีเกาะทั้งสิ้น 8 เกาะ)

```
.............AA
..BBBBB...CC.....
.BBBBBBB..CC...DDD
.BBBBBBB.......D..
.BBBBBBB..E.F..D.G
.BBBBBB....H...D.G
```

Problem Statement

จากแผนที่ภายในเกมที่กำหนดให้ มีเกาะทั้งสิ้นกี่เกาะ?

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดที่ 1: มีจำนวนเต็มสองตัว R, C คั่นด้วยช่องว่าง
- อีก R บรรทัดถัดมา บรรทัดที่ $i+1$ จะมีสตริงความยาว C ที่ประกอบไปด้วย '`.`' หรือ '`#`' (ซึ่งบอกข้อมูลแถวนั้น ๆ ของแผนที่)

```
  1   R C
  2   M[1,1 ... C]
  3   M[2,1 ... C]
 ...
R+1   M[R,1 ... C]
```

หมายเหตุ: ตัวแปร `M` ข้างต้น คือแผนที่ซึ่งเขียนในรูปของ 1-indexed array สองมิติ

Output Format

- คำตอบประกอบด้วยจำนวนเต็มตัวเดียว ซึ่งระบุจำนวนเกาะในแผนที่ที่กำหนดให้

Data Examples

Example Input	Example Output
6 18 ....## ..#####...##...... .##...##..##...### .#..#..#.......#.. .#....##..#.#..#.# .######....#...#.#	8
7 7 ....####. .#...#. .#.#.#. .#...#. .#####.	2

Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

- test cases ชุดเล็กจะมีเงื่อนไขว่า $1 \leq R, C \leq 200$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า $1 \leq R, C \leq 3000$

คำถามที่ 63. Shipping Dools

Problem Statement

โรงงานแห่งหนึ่งรับจ้างผลิตตุ๊กตาแบบสั่งทำพิเศษ อยู่มาวันหนึ่งมีลูกค้า A มาติดต่อจ้างให้ผลิตตุ๊กตาทั้งสิ้น N ตัว ตุ๊กตาแต่ละตัวมีหมายเลขกำกับ $i = 1, 2, \ldots, N$ นอกจากนั้นตุ๊กตาตัวที่ i จะมีน้ำหนัก w_i กรัม ซึ่งอาจเท่ากันหรือต่างกันก็ได้

เมื่อโรงงานแห่งนี้ผลิตตุ๊กตาเสร็จเป็นที่เรียบร้อยแล้ว โรงงานจะต้องขนส่งตุ๊กตาทั้งหมดนี้ให้ลูกค้า A โรงงานสามารถเลือกขนส่งตุ๊กตาแต่ละตัวได้ 2 วิธี คือ (1) บรรจุตุ๊กตาลงในกล่องพัสดุที่จำกัดน้ำหนัก หรือ (2) บรรจุตุ๊กตาลงถุงกระสอบที่จำกัดจำนวนตุ๊กตา โดยมีเงื่อนไขว่า

- การขนส่งอาจใช้กล่องหลายใบก็ได้ กล่องแต่ละใบจุของน้ำหนักรวมไม่เกิน L กรัม
- การขนส่งสามารถใช้ถุงกระสอบได้เพียงถุงเดียว และใส่ตุ๊กตาได้ไม่เกิน M ตัว (ไม่จำกัดน้ำหนัก)
- หากตุ๊กตาหมายเลขที่ s และตุ๊กตาหมายเลขที่ t จะถูกบรรจุลงในกล่องใบเดียวกันแล้ว ตุ๊กตาหมายเลขที่ i แต่ละตัวซึ่งมีหมายเลขอยู่ระหว่าง s กับ t จะต้องถูกบรรจุในกล่องใบเดียวกันด้วย หรือจะต้องถูกบรรจุในถุงกระสอบเท่านั้น

Main Goal

โรงงานต้องการขนส่งตุ๊กตาทั้งหมดให้ลูกค้า A โดยใช้จำนวนกล่องให้น้อยที่สุด จะต้องใช้กล่องทั้งหมดกี่ใบ?

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดที่ 1: มีจำนวนเต็มสามตัว N, L, M คั่นด้วยช่องว่าง
- อีก N บรรทัดถัดมา บรรทัดที่ $i + 1$ จะมีจำนวนเต็ม w_i ระบุน้ำหนักของตุ๊กตาตัวที่ i

```
1   N L M
2   w_1
3   w_2
...
N+1 w_N
```

Output Format

- คำตอบประกอบด้วยจำนวนเต็มตัวเดียว ซึ่งระบุจำนวนกล่องที่น้อยที่สุดที่สามารถใช้ขนส่งตุ๊กตาทั้งหมดตามเงื่อนไขโจทย์ข้างต้น

Data Example

Example Input	Example Output
6 5 1 1 2 3 2 1 4	2

อธิบายตัวอย่าง: หยิบตุ๊กตาตัวที่ $i = 3$ ซึ่ง $w_i = 3$ ใส่ถุง จากนั้นหยิบตุ๊กตาตัวที่ $1, 2, 4$ ใส่กล่องใบแรก และตัวที่ $5, 6$ ใส่กล่องใบที่สอง

Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

- test cases ชุดเล็กจะมีเงื่อนไขว่า $1 \leq N \leq 250$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า $1 \leq N \leq 2500$
- นอกจากนั้นกำหนดให้ $1 \leq L \leq 10^8;\ 0 \leq M \leq N$ และตุ๊กตาแต่ละตัวมีน้ำหนัก $1 \leq w_i \leq L$

คำถามที่ 64. Descending Drills

Background

ผืนดินแห่งหนึ่งมีสมบัติซ่อนอยู่ใต้ดินมากมาย เนื่องด้วยเทคโนโลยี Remote Sensing ในปัจจุบัน ทำให้เราสามารถสำรวจมูลค่าของสมบัติที่อยู่ใต้ดินในบริเวณต่าง ๆ ได้ โดยที่เราไม่ต้องขุดสมบัติออกจากดินเพื่อมาตีราคาแต่อย่างใด

เราจะมองชั้นดินที่เต็มไปด้วยสมบัติดังกล่าวเป็นพื้นที่หน้าตัดรูปสี่เหลี่ยมผืนผ้า ซึ่งเราจะแบ่งสี่เหลี่ยมผืนผ้าดังกล่าวเป็นชั้นดินลึก R ชั้น ชั้นละ C ช่อง ดินแต่ละช่องจะมีมูลค่าของสมบัติกำกับไว้ด้วยซึ่งเป็นจำนวนเต็มที่อาจเป็นบวก ลบ หรือศูนย์ก็ได้

ตัวอย่าง รูปต่อไปนี้คือตัวอย่างข้อมูลของสมบัติในชั้นดินที่มี $R = 6$ และ $C = 6$

3	−1	1	2	3	7
4	−2	7	4	4	−5
6	−5	1	−9	−2	6
8	7	−9	8	6	−8
1	9	2	6	−4	1
4	2	−9	4	−5	−3

Drilling Constraints

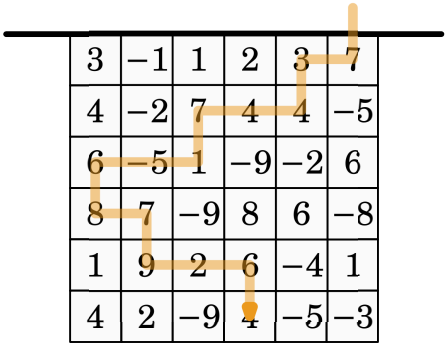
เราต้องการจะเจาะผืนดินเพื่อล่าสมบัติที่อยู่ในดินให้ได้ผลรวมมากที่สุด แต่เนื่องด้วยขีดจำกัดของนวัตกรรมการขุดเจาะที่ยังมีราคาแพง ทำให้เรามีโอกาสเดียวเท่านั้นในการขุดเจาะผืนดินดังกล่าว ลักษณะเส้นทางของการขุดดินจะมีเงื่อนไขดังนี้

- เราสามารถเริ่มต้นขุดเจาะจากผิวดิน เหนือช่องคอลัมน์ใดก็ได้
- ตลอดการขุดเจาะในครั้งนี้ เราสามารถขุดเจาะดินในแนวดิ่ง เผื่อลงไปยังชั้นดินชั้นต่อไปก็ได้ หรือจะขุดเจาะในแนวราบไปทางซ้ายหรือขวาในชั้นดินระดับเดียวกันก็ได้ แต่ไม่สามารถเจาะสวนทางแรงโน้มถ่วงในทิศทางชี้สู่ผิวดินได้
- สำหรับการขุดเจาะแนวราบนั้น เมื่อเราขุดเจาะลงสู่ชั้นดินหนึ่ง ๆ เครื่องขุดเจาะอาจจะเลือกขุดเจาะไปทางซ้ายหรือทางขวา ทิศทางใดทิศทางหนึ่งเท่านั้น (หรือจะไม่ขยับในแนวราบก็ได้) และการขุดแนวราบดังกล่าว จะขยับจากจุดเริ่มต้นได้ไม่เกิน K ช่อง
- เครื่องขุดเจาะไม่สามารถเดินถอยหลังไปยังช่องดินที่เคยขุดเจาะไปแล้วได้ ไม่ว่าจะเป็นแนวดิ่งหรือแนวราบก็ตาม
- การขุดเจาะจะสิ้นสุดที่ช่องใดก็ได้ และมูลค่ารวมของสมบัติที่เก็บสะสมได้ คือผลรวมของมูลค่าของสมบัติทุกช่องที่เครื่องขุดเจาะนี้แทรกผ่าน
- ไม่จำเป็นว่าจะต้องขุดเจาะถึงชั้นผิวดินแถวล่างสุดเสมอไป

- สมบัติที่มีมูลค่าติดลบที่ค้นพบระหว่างทางจะต้องถูกนำมารวมในผลรวมด้วยเสมอ
- หากไม่มีรูปแบบการขุดเจาะที่ทำให้ผลรวมสมบัติเป็นบวกเลย สามารถตอบ `0` ได้

ตัวอย่าง รูปต่อไปนี้มีเส้นสีส้มแสดงเส้นทางการขุดเจาะชั้นดิน เพื่อล่าสมบัติที่อยู่ในดิน โดยมีเงื่อนไขว่า $K = 2$ สังเกตว่าไม่มีการขยับในแนวราบเกิน 2 ช่องเลยในทุกระดับชั้นดิน

ผลรวมมูลค่าสมบัติที่ขุดเจาะตามเส้นทางตัวอย่างนี้คือ 63 หน่วย ซึ่งเป็นเส้นทางที่ดีที่สุดสำหรับรูปตัวอย่างนี้



Problem Statement

กำหนดให้มูลค่าของสมบัติในดินเป็นตารางสี่เหลี่ยมผืนผ้าขนาด R แถวและ C คอลัมน์ จงหามูลค่าสมบัติรวมที่มากที่สุดที่เกิดจากการขุดเจาะด้วยโอกาสเพียง 1 ครั้งตามเงื่อนไขข้างต้น

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดที่ 1: มีจำนวนเต็มสามตัว R, C, K คั่นด้วยช่องว่าง
- อีก R บรรทัดถัดมา บรรทัดที่ $i+1$ จะมีจำนวนเต็ม C จำนวน คั่นด้วยช่องว่าง แทนมูลค่าของสมบัติในชั้นดินที่ i เรียงจากซ้ายไปขวา

```
R C K
v[1, 1] v[1, 2] ... v[1, C]
v[2, 1] v[2, 2] ... v[2, C]
...
v[R, 1] v[R, 2] ... v[R, C]
```

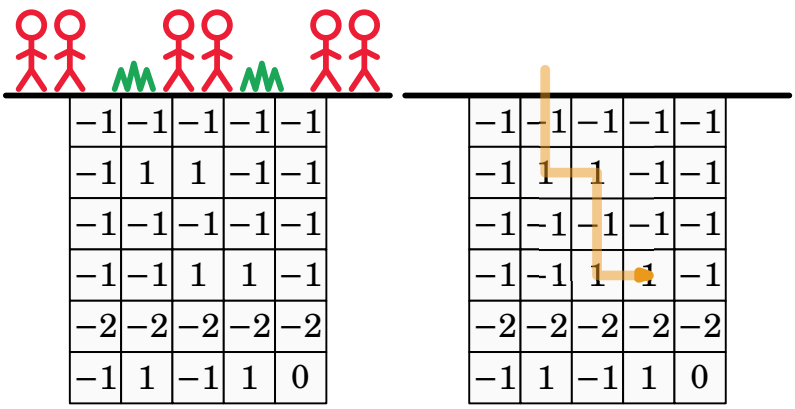
Output Format

- คำตอบประกอบด้วยจำนวนเต็มเพียงหนึ่งตัว ซึ่งระบุผลรวมของสมบัติที่มากที่สุดที่สามารถหาได้จากการขุดเจาะเพียงครั้งเดียวตามเงื่อนไขที่กำหนดไว้

Data Examples

Example Input	Example Output
6 6 2 3 -1 1 2 3 7 4 -2 7 4 4 -5 6 -5 1 -9 -2 6 8 7 -9 8 6 -8 1 9 2 6 -4 1 4 2 -9 4 -5 -3	63
6 5 1 -1 -1 -1 -1 -1 -1 1 1 -1 -1 -1 -1 -1 -1 -1 -1 -1 1 1 -1 -2 -2 -2 -2 -2 -1 1 -1 1 0	2

อธิบายตัวอย่างที่ 2: โปรดพิจารณารูปตัวอย่างต่อไปนี้ประกอบข้อมูลตัวอย่างข้างต้น



Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

- test cases ชุดเล็กจะมีเงื่อนไขว่า ขนาดของตารางจะสอดคล้องกับเงื่อนไขที่ว่า $1 \leq R, C \leq 200$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า จำนวนช่องในตารางจะสอดคล้องกับเงื่อนไขที่ว่า $1 \leq RC \leq 2 \cdot 10^6$
- สำหรับทุก test cases จะมีเงื่อนไขว่า จำนวนช่องที่ขยับได้ในแนวราบในแถว ๆ หนึ่งจะสอดคล้องกับเงื่อนไข $0 \leq K < C$ และมูลค่าสมบัติแต่ละช่องจะมีค่าที่สอดคล้องกับเงื่อนไข $-1000 \leq$ `v[i, j]` ≤ 1000

คำถามที่ 65. Polymer Chain

Background

ในห้องปฏิบัติการวิจัยเคมีแห่งหนึ่ง มีการทดลอง Simulation การเปลี่ยนแปลงโครงสร้างสาย polymer ชื่อ polymer K ซึ่งประกอบไปด้วยอะตอมหลากหลายชนิดเรียงตัวเป็นเส้นตรง

ในคอมพิวเตอร์ เราจะแทนอะตอมแต่ละชนิดด้วยอักขระ `"A"` ถึง `"Z"` และเราจะแทน polymer `K` ที่มีความยาว N ด้วยสตริงที่ประกอบไปด้วยอักขระ `"A"` ถึง `"Z"` ความยาว N ตัว

ในห้องปฏิบัติการดังกล่าว ยังมีเครื่องจัดเรียง polymer ที่มีชื่อว่า

```
manipulate_polymer_once(K[0 ... N-1], p)
```

ซึ่งมี Specification ในการทำงานดังต่อไปนี้

> หาก input string `K` ของฟังก์ชัน `manipulate_polymer_once` คือลำดับของอักขระ `K[0]`, `K[1]`, ..., `K[N-1]` ตามลำดับ และ `p` คือ index ของสตริง `K` โดยที่ `0 ⩽ p ⩽ N-1` แล้ว ฟังก์ชันนี้จะ return ค่าสตริงซึ่งประกอบด้วยอักขระต่อไปนี้ตามลำดับ
>
> $$\underbrace{\texttt{K[p-1], K[p-2], ..., K[1], K[0]}}_{\text{exists if } p > 0}\texttt{, K[p], }\underbrace{\texttt{K[N-1], K[N-2], ..., K[p+2], K[p+1]}}_{\text{exists if } p < N-1}$$
>
> ยกตัวอย่างเช่น ถ้า input ของ `manipulate_polymer_once` ได้แก่ `K = "ASDFGHJKL"` และ `p = 3` จะได้ output string เป็น `"DSAFLKJHG"` สังเกตว่า `F` จะอยู่ในตำแหน่งเดิมไม่เปลี่ยนแปลง แต่สตริงย่อยที่อยู่ข้างหน้าและข้างหลังจะถูกเรียงกลับหลัง

ในการทำการทดลอง Simulation จริง เราจะนำ polymer สายหนึ่งมาจัดเรียงใหม่ไปเรื่อย ๆ อย่างต่อเนื่องเป็นจำนวน M ครั้ง โดยปรับเปลี่ยน parameter ไปเรื่อย ๆ เช่น ถ้า polymer ตั้งต้นคือ `K = "ASDFGHJKL"` และค่า parameter ของการจัดเรียง $M = 3$ ครั้งอย่างต่อเนื่องคือ `p_1 = 3`, `p_2 = 6`, `p_3 = 0` เราจะได้ Polymer ผลลัพธ์เป็น

```
ASDFGHJKL → DSAFLKJHG → KLFASDJGH → KHGJDSAFL
```

Problem Statement

จงเขียนโปรแกรมเพื่อทำ Simulation ของการจัดเรียง polymer สายหนึ่งด้วยลำดับของพารามิเตอร์ที่กำหนดให้ แล้วหาว่า polymer ผลลัพธ์สุดท้ายมีหน้าตาเป็นอย่างไร

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดที่ 1: มีจำนวนเต็มสองจำนวน N และ M คั่นด้วยช่องว่าง
- บรรทัดที่ 2: มีสตริงความยาว N ซึ่งระบุข้อมูลสาย Polymer เริ่มต้นก่อนการทดลอง
- อีก M บรรทัดถัดมา บรรทัดที่ $i+2$ จะมีค่า p_i ซึ่งเป็น Parameter ของคำสั่งการจัดเรียง Polymer คำสั่งที่ i ที่ต้องกระทำดับ Polymer K ตามลำดับ

```
1   N M
2   K[0 ... N-1]
3   p_1
4   p_2
...
M+2 p_M
```

Output Format

- คำตอบประกอบตัวสตริง 1 ตัว ซึ่งก็คือสาย Polymer สุดท้ายหลังจากจัดเรียง Polymer ตามคำสั่งทั้งหมด M ตามที่กำหนดให้ใน input

Data Example

Example Input	Example Output
9 3 ASDFGHJKL 3 6 0	KHGJDSAFL

Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

- test cases ชุดเล็กจะมีเงื่อนไขว่า ความยาว polymer จะสอดคล้องกับเงื่อนไข $1 \leq N \leq 300$ และจำนวนครั้งที่เรียกใช้งานเครื่องจัดเรียง polymer สอดคล้องกับเงื่อนไข $0 \leq M \leq 30{,}000$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า ความยาว polymer จะสอดคล้องกับเงื่อนไข $1 \leq N \leq 300{,}000$ และจำนวนครั้งที่เรียกใช้งานเครื่องจัดเรียง polymer สอดคล้องกับเงื่อนไข $0 \leq M \leq 300{,}000$

คำถามที่ 66. Lobbying Tollway

Background

บริษัทขนส่งสินค้าแห่งหนึ่ง จำเป็นต้องวางแผนการลำเลียงส่งสินค้าระหว่างเมืองสองเมืองในดินแดนที่มีเมืองทั้งสิ้น N เมือง และมีโครงข่ายของถนน M สายที่เชื่อมเมืองเหล่านี้ให้เดินทางไปมาหาสู่กันได้ทั้งหมด เมืองแต่ละเมืองจะมีหมายเลข 1 ถึง N ส่วนถนนแต่ละสายจะมีหมายเลข 1 ถึง M ตามลำดับ

สำหรับแต่ละ $i = 1, 2, \ldots, M$ ถนนสายที่ i จะเป็นถนนวิ่งทางเดียว (one-way road) ที่เชื่อมการเดินทางจากเมือง u_i ไปยังเมือง v_i เสมอ ($1 \leq u_i, v_i \leq N$) นอกจากนั้นอาจจะมีค่าผ่านทาง p_i บาทที่คนใช้ถนนสายนี้ต้องจ่ายเพื่อใช้งาน ($p_i \geq 0$) นอกจากนั้นกำหนดว่าถ้าถนนสายไหนไม่มีค่าผ่านทาง นั่นแปลว่า $p_i = 0$

พึงทราบว่า อาจมีถนนวิ่งทางเดียวที่เชื่อมจากเมืองหนึ่งไปยังอีกเมืองหนึ่ง มากกว่า 1 สายก็ได้ นอกจากนั้นอาจมีถนนที่เชื่อมระหว่างเมืองสองเมือง ไป-กลับ โดยที่ถนนเหล่านี้เก็บค่าผ่านทางที่ไม่เท่ากันก็ได้

โดยปกตินั้น บริษัทนี้ได้สำรวจเส้นทางทั้งหมดที่เป็นไปได้ เพื่อใช้ลำเลียงสินค้าจากเมืองหมายเลข 1 ไปยังเมืองหมายเลข N โดยเส้นทางเหล่านี้ล้วนแต่เป็นเส้นทางที่เสียค่าผ่านทางรวมน้อยที่สุดทั้งสิ้น

ในเวลาต่อมา บริษัทนี้ต้องการเปิดเส้นทางการลำเลียงสินค้าเพิ่มขึ้นอย่างน้อย 1 เส้นทาง โดยมีเงื่อนไขต่อไปนี้

- บริษัทจะไปล็อบบี้กับผู้บริหารของเครือข่ายถนน เพื่อให้ลดค่าผ่านทางของถนนเพียง 1 สายเท่านั้น
- ค่าผ่านทางใหม่นั้นจะติดลบไม่ได้
- ค่าผ่านทางใหม่นั้นจะต้องลดลงจากค่าผ่านทางเดิม เป็นปริมาณเงินน้อยที่สุดเท่าที่เป็นไปได้
- เส้นทางการลำเลียงสินค้าเดิมที่เคยสำรวจไว้จะต้องไม่กระทบ กล่าวคือเส้นทางเดิมแต่ละเส้นทางจะยังคงใช้งานได้เช่นเดิม และมีค่าผ่านทางรวมเท่าเดิม ไม่เพิ่มขึ้นหรือลดลง
- จะต้องมีเส้นทางใหม่การลำเลียงสินค้าเกิดขึ้นอย่างน้อย 1 เส้นทาง และจะต้องไม่ซ้ำกันเส้นทางเดิมที่บริษัทเคยสำรวจไว้ และราคาค่าผ่านทางรวมของเส้นทางใหม่นี้จะต้องเท่ากับราคาค่าผ่านทางรวมของเส้นทางเดิมอื่น ๆ ของบริษัทด้วย

Problem Statement

จงรับข้อมูลเครือข่ายถนนในดินแดนแห่งหนึ่ง รวมถึงค่าผ่านทางของถนนแต่ละสาย แล้วหาว่าบริษัทนี้จะต้องไปล็อบบี้เพื่อลดค่าผ่านทางของถนนสายใด 1 สาย และเป็นปริมาณเงินลดลงน้อยที่สุดเท่าใด จึงจะสามารถเปิดเส้นทางใหม่เพื่อใช้ลำเลียงสินค้าจากเมือง 1 ไปเมือง N ได้ โดยเส้นทางใหม่ที่เกิดขึ้นนี้จะมีค่าผ่านทางรวมถูกที่สุด และถูกเท่า ๆ กับเส้นทางอื่น ๆ ที่เคยมีการสำรวจมาก่อนหน้านี้แล้ว

หากมีถนนที่เป็นไปได้หลายสายที่สามารถล็อบบี้ให้ลดราคาลงเป็นปริมาณที่น้อยที่สุดได้ ให้ตอบหมายเลขของถนนทุกสายด้วย

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดที่ 1: มีจำนวนเต็มสองจำนวน N และ M คั่นด้วยช่องว่าง
- อีก M บรรทัดถัดมา บรรทัดที่ $i+1$: จะมีจำนวนเต็มสามจำนวน u_i, v_i, p_i (คั่นด้วยช่องว่าง) ระบบข้อมูลของถนนหมายเลข i ซึ่งเป็นถนนวิ่งทางเดียวจากเมืองหมายเลข u_i ไปยังเมืองหมายเลข v_i และเก็บค่าผ่านทาง p_i บาท

```
  1   N M
  2   u_1 v_1 p_1
  3   u_2 v_2 p_2
 ...
M+1   u_M v_M p_M
```

หมายเหตุ: ข้อมูล Input จะรับประกันว่า มีเส้นทางที่เชื่อมจากเมืองหมายเลข 1 ไปเมืองหมายเลข N เสมอ

Output Format

- บรรทัดที่ 1: จะต้องเขียนจำนวนเต็มสองจำนวน D และ K คั่นด้วยช่องว่างหนึ่งช่อง โดยที่ D จะระบุปริมาณค่าผ่านทางที่ลดลงน้อยที่สุดที่เป็นไปได้ และ K คือจำนวนถนนทั้งหมดที่สามารถล็อบบี้ให้ลดค่าผ่านทางได้
- อีก K บรรทัดถัดมา แต่ละบรรทัดจะมีจำนวนเต็ม 1 จำนวน ซึ่งแต่ละจำนวนจะระบุหมายเลขถนนที่สามารถล็อบบี้ได้ นอกจากนั้น หมายเลขถนนทั้งหมดจะต้องเรียงจากน้อยไปมาก

  หมายเหตุ: ในกรณีที่บริษัทไม่สามารถใช้วิธีล็อบบี้ใด ๆ เพื่อเปิดเส้นทางใหม่ได้เลย ให้ตอบว่า $D = 0$ และ $K = 0$ เป็นกรณีพิเศษ

First Data Example

Example Input	Example Output
7 10 1 2 8 1 3 6 1 4 6 1 5 3 1 6 12 2 7 8 3 7 5 4 7 7 5 7 8 6 7 1	2 3 3 5 8

อธิบายตัวอย่างที่ 1:

- จากตัวอย่างข้อมูลนี้ พบว่าจะมีเส้นทางลำเลียงที่ใช้ค่าผ่านทางรวมน้อยที่สุด 11 บาท ซึ่งมี 2 เส้นทาง ได้แก่ (1) เส้นทางที่ใช้ถนนหมายเลข $2\,\&\,7$ และอีกเส้นทางที่ใช้ถนนหมายเลข $4\,\&\,9$
- หากเราล็อบบี้ให้มีการลดค่าผ่านทาง 2 บาท ให้แก่ถนน 1 สายในบรรดาถนน 3 สาย สายได้ก็ได้ (ซึ่งได้แก่ถนนหมายเลข 3, 5 และ 8) แล้วจะทำให้มีเส้นทางลำเลียงสินค้าเส้นทางใหม่ที่ใช้เงินรวม 11 บาทเช่นกัน

Second Data Example

Example Input	Example Output
4 5 1 2 2 1 3 3 2 3 1 2 4 3 3 4 2	0 0

Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

- test cases ชุดเล็กจะมีเงื่อนไขว่า จำนวนเมืองทั้งหมดจะสอดคล้องกับเงื่อนไข $3 \leq N \leq 50$ และจำนวนถนนทั้งหมดจะสอดคล้องกับเงื่อนไข $1 \leq M \leq 2{,}000$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า จำนวนเมืองทั้งหมดจะสอดคล้องกับเงื่อนไข $3 \leq N \leq 100{,}000$ และจำนวนถนนทั้งหมดจะสอดคล้องกับเงื่อนไข $1 \leq M \leq 200{,}000$
- สำหรับทุก test cases จะมีเงื่อนไขว่า ค่าผ่านทางเริ่มต้นของถนนทุกสายจะสอดคล้องกับเงื่อนไข $0 \leq p_i \leq 5{,}000$

คำถามที่ 67. Intercept Meteor

Background

องค์การบริหารการบินและอวกาศแห่งโลกได้ตรวจพบอุกกาบาตขนาดใหญ่ที่กำลังพุ่งชนโลก จึงได้เตรียมแผนการรับมือโดยการส่งนักขุดเจาะฝีมือดีขึ้นไปวางระเบิดที่แกนของอุกกาบาต เพื่อระเบิดอุกกาบาตออกเป็นชิ้นเล็ก ๆ ก่อนจะตกลงสู่พื้นโลก จากการคาดการณ์ เศษอุกกาบาตส่วนใหญ่จะตกลงสู่ทะเล มีเพียงเกาะโคะเกาะเดียวเท่านั้นที่มีคนอาศัยอยู่และได้รับผลกระทบจากเศษอุกกาบาต

เพื่อปกป้องเกาะโคะที่มีอารยธรรมโบราณอันมีคุณค่า องค์การบริหารการบินและอวกาศแห่งโลกจึงได้ติดตั้งฐานยิงจรวดมิสไซล์ไว้ที่เกาะโคะเพื่อยิงเศษอุกกาบาตก่อนจะตกถึงพื้น ข้อเสียของการใช้จรวดมิสไซล์คือ การยิงเศษอุกกาบาตแต่ละครั้งจะสร้างมลพิษสู่ชั้นบรรยากาศเป็นจำนวนมาก องค์การบริหารการบินและอวกาศแห่งโลกจึงต้องการยิงเศษอุกกาบาตให้น้อยที่สุด โดยจะยิงเฉพาะเศษอุกกาบาตที่มีจุดตกอยู่บนพื้นเกาะโคะเท่านั้น

ความพิเศษของเกาะโคะคือมีลักษณะเป็น convex polygon (เป็นรูปหลายเหลี่ยม และทุกคู่จุดใด ๆ บนเกาะสามารถเดินเป็นเส้นตรงถึงกันได้โดยไม่มีทะเลมาขวาง)

Problem Statement

องค์การบริหารการบินและอวกาศแห่งโลกต้องการให้คุณเขียนโปรแกรมช่วยคำนวนว่าจากจุดตกของเศษอุกกาบาตแต่ละลูก มีลูกไหนตกบนพื้นเกาะโคะ, ตกตรงขอบเกาะโคะ และตกนอกเกาะโคะบ้าง

ให้พิจารณาเฉพาะจุดตกเท่านั้น ไม่ต้องสนใจขนาดของเศษอุกกาบาต

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดแรก เป็นจำนวนเต็มบวก N แทนจำนวนมุมของ convex polygon C ที่แสดงขอบเขตของเกาะโคะ
- N บรรทัดต่อมา แต่ละบรรทัดเป็นจำนวนเต็ม x y คั่นด้วยช่องว่าง แทนพิกัดจุดมุมแต่ละจุดของ C เรียงตามเข็มนาฬิกา โดยเริ่มจากจุดที่มีค่า x น้อยที่สุด หากมีจุดที่มีค่า x น้อยที่สุดมากกว่าหนึ่งจุดจะเริ่มต้นจากจุดที่มีค่า y น้อยที่สุด (รับประกันว่าไม่มีพิกัดใดซ้ำกันเลย และ ไม่มี 3 จุดมุมใดๆที่เรียงต่อเนื่องกันเป็นเส้นตรง)
- บรรทัดต่อมา เป็นจำนวนเต็มบวก K แทนจำนวนเศษอุกกาบาตที่ต้องการตรวจสอบ
- K บรรทัดต่อมา แต่ละบรรทัดเป็นจำนวนเต็ม x y คั่นด้วยช่องว่าง แทนพิกัดของจุดตกของเศษอุกกาบาตที่ต้องการตรวจสอบแต่ละลูก

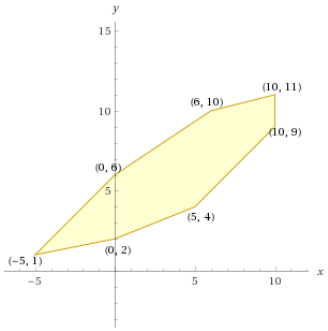
Output Format สำหรับเศษอุกกาบาตที่ต้องการตรวจสอบแต่ละลูก ให้แสดงข้อความในหนึ่งบรรทัด โดยข้อความจะเป็น

- `"Inside"` หากจุดตกอยู่ภายในเกาะโคะ หรือ
- `"Outside"` หากจุดตกอยู่ภายนอกเกาะโคะ หรือ
- `"On the boundary"` หากจุดตกอยู่บนเส้นรอบรูปของเกาะโคะพอดี (อยู่บนขอบหรืออยู่ที่จุดมุมก็ได้)

Data Example

Example Input	Example Output
7 -5 1 0 6 6 10 10 11 10 9 5 4 0 2 6 0 5 -4 9 5 4 10 10 10 3 5 5	Inside Outside On the boundary On the boundary Outside Inside

อธิบายตัวอย่าง: รูปล่างของเกาะโคะมีลักษณะดังที่ปรากฏทางขวามือ นอกจากนั้นมี 6 คำถามดังต่อไปนี้



- คำถามแรก จุด (0, 5) อยู่ภายใน C ตอบว่า `"Inside"`
- คำถามที่สอง จุด (-4, 9) อยู่ภายนอก C ตอบว่า `"Outside"`
- คำถามที่สาม จุด (5, 4) เป็นจุดมุม ตอบว่า `"On the boundary"`
- คำถามที่สี่ จุด (10, 10) อยู่บนขอบ ตอบว่า `"On the boundary"`
- คำถามที่ห้า จุด (10, 3) อยู่ภายนอก C ตอบว่า `"Outside"`
- คำถามที่หก จุด (5, 5) อยู่ภายใน C ตอบว่า `"Inside"`

Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

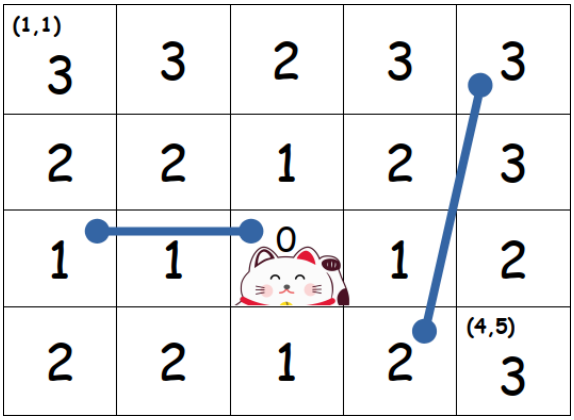
- test cases ชุดเล็กจะมีเงื่อนไขว่า จำนวนมุมของ convex polygon ที่แสดงขอบเขตของเกาะโคะสอดคล้องกับเงื่อนไข $3 \leq N \leq 1{,}000$ และจำนวนเศษอุกกาบาตที่ต้องการตรวจสอบสอดคล้องกับเงื่อนไข $1 \leq K \leq 1{,}000$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า จำนวนมุมของ convex polygon ที่แสดงขอบเขตของเกาะโคะสอดคล้องกับเงื่อนไข $3 \leq N \leq 10^5$ และจำนวนเศษอุกกาบาตที่ต้องการตรวจสอบสอดคล้องกับเงื่อนไข $1 \leq K \leq 10^5$
- สำหรับทุก test cases จะมีเงื่อนไขว่า พิกัดจุดมุมของ convex polygon ที่แสดงขอบเขตของเกาะโคะ และ พิกัดจุดตกของเศษอุกกาบาตสอดคล้องกับเงื่อนไข $-10^9 \leq x, y \leq 10^9$

คำถามที่ 68. Finding Peep Chan

Background

เด็กหญิงเกดเป็นเด็กน้อยแสนน่ารักที่เลี้ยงแมวเหมียวชื่อว่าปิ๊บจัง วันหนึ่งเจ้าปิ๊บจังหายไปเลย ออกไปตามหา เด็กหญิงเกดได้เบาะแสมาว่าเจ้าปิ๊บจังไปติดอยู่ในถ้ำแห่งหนึ่ง ถ้ำมีลักษณะเป็น grid 2 มิติ แบ่งเป็นห้องขนาด $N \times M$ โดยห้องซ้ายบนสุดคือพิกัด $(1,1)$ และห้องขวาล่างสุดคือพิกัด (N,M)

แต่ละห้องจะสามารถเดินไปมาหาสู่กันได้เฉพาะห้องที่ติดกันในทิศทาง บน ล่าง ซ้าย ขวา หรือก็คือ ห้อง (x,y) สามารถเดินไปยังห้อง $(x-1,y)$, $(x+1,y)$, $(x,y-1)$ และ $(x,y+1)$ นอกจากนี้ในถ้ำยังมีทางลัดอีก K เส้นทาง โดยเส้นทางที่ i จะสามารถเดินทางไปมาหาสู่กันระหว่างห้อง (x_{i1},y_{i1}) และ (x_{i2},y_{i2}) ได้ การเดินทางไปยังช่องที่ติดกันในทิศทาง บน ล่าง ซ้าย ขวา หรือ การใช้ทางลัด แต่ละครั้งจะใช้เวลาในการเดินทาง 1 วัน (ขอเรียกระยะการเดินของแมวใน 1 วันว่า 1 แมวเดิน)



จากรูปแสดงระยะทางในหน่วยแมวเดิน จากการเดินของแมวที่อยู่ที่ห้องตรงกลาง ลูกศรคือเส้นทางลัด

เด็กหญิงเกดรู้สึกจนปัญญามากเพราะไม่รู้ว่าแมวอยู่ห้องไหนในถ้ำกันแน่ ดังนั้นจะถือว่าตอนแรกแมวมีโอกาสที่จะอยู่ในแต่ละห้องด้วยความน่าจะเป็นที่เท่า ๆ กัน (ซึ่งมีค่าเท่ากับ $\frac{1}{N \times M}$) เด็กหญิงเกดจึงไปร้องขอต่อเทพเจ้าให้ช่วยเหลือ

เทพเจ้ามีพลังวิเศษ 2 อย่าง คือ

- **หยั่งรู้** — เทพเจ้าจะเลือกห้องห้องหนึ่ง หลังจากนั้นพระเจ้าจะรู้ว่าแมวอยู่ห่างจากห้องที่เลือกเป็น “ระยะทางที่ใกล้ที่สุด” กี่แมวเดิน แต่ว่าสามารถใช้ได้วันละ 1 ครั้งเท่านั้น
- **ช่วยเหลือ** — เทพเจ้าจะเลือกห้องห้องหนึ่งเพื่อช่วยเหลือแมว ถ้าแมวอยู่ในห้องที่เลือกจะถือว่าช่วยเหลือสำเร็จ แต่ว่าสามารถใช้ได้แค่ครั้งเดียวเท่านั้น

เนื่องจากถ้ำมีอากาศอยู่เบาบางมาก แมวจะสามารถมีชีวิตอยู่ได้แค่ 2 วัน เพราะฉะนั้นการช่วยเหลือแมวจะมีโอกาสแค่วันนี้กับพรุ่งนี้เท่านั้น

เทพเจ้าตอบรับคำร้องขอของเด็กหญิงเกด และดำเนินการช่วยเหลือตามขั้นตอนต่อไปนี้

1. เทพเจ้าจะใช้พลัง **“หยั่งรู้”** เพื่อเลือกห้องห้องหนึ่งในถ้ำ
2. เทพเจ้าจะพิจารณาว่า จะใช้พลัง **“ช่วยเหลือ”** ในวันนี้เลยหรือไม่ แน่นอนว่าถ้าใช้แล้วจะไม่สามารถดำเนินการในข้อต่อไปได้ เพราะไม่สามารถใช้พลัง **“ช่วยเหลือ”** ได้อีกต่อไป
3. ในกรณีที่เทพเจ้าไม่ได้ใช้พลัง **“ช่วยเหลือ”** เทพเจ้าจะนอน 1 วันเพื่อพักผ่อน ในขณะที่เทพเจ้านอนพัก แมวจะเคลื่อนที่ 1 ครั้ง โดยการเคลื่อนที่นี้แมวจะสามารถเดินไปยังห้องที่ติดกันหรือห้องที่มีทางลัดไปหากันได้ โดยแมวจะเดินไปยังห้องที่ไปได้ทั้งหมดด้วยโอกาสที่เท่าๆกัน
4. ในวันถัดไป เทพเจ้าก็จะใช้พลัง **“หยั่งรู้”** อีกครั้ง
5. เทพเจ้าก็จะใช้พลัง **“ช่วยเหลือ”** เพราะว่าเป็นโอกาสสุดท้ายที่จะช่วยเหลือแมวแล้ว

Problem Statement

เทพเจ้าอยากทราบว่า ความน่าจะเป็นที่มากที่สุดที่เป็นไปได้ที่จะช่วยเหลือแมวได้สำเร็จเป็นเท่าไหร่ ถ้าดำเนินการช่วยเหลืออย่าง optimal ที่สุด

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดแรกประกอบด้วยจำนวนเต็ม N, M แทนขนาดของถ้ำ และ K แทนจำนวนเส้นทางลัด
- K บรรทัดถัดมา บรรทัดที่ i จะประกอบด้วย จำนวนเต็ม 4 จำนวน x_{i1}, y_{i1}, x_{i2} และ y_{i2} คั่นด้วยช่องว่าง ซึ่งระบุเส้นทางลัดเส้นที่ i

```
N M K
x_{1,1} y_{1,1} x_{1,2} y_{1,2}
x_{2,1} y_{2,1} x_{2,2} y_{2,2}
...
x_{K,1} y_{K,1} x_{K,2} y_{K,2}
```

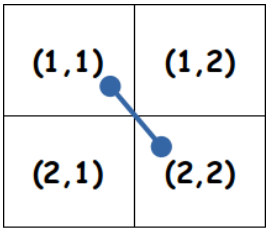
หมายเหตุ: รับประกันว่า $1 \leq x_{i1}, x_{i2} \leq N$ และ $1 \leq y_{i1}, y_{i2} \leq M$ และจะไม่มีเส้นทางลัดซ้ำกัน และไม่มีทางลัดของห้องที่อยู่ติดกัน

Output Format

- ตอบเป็นจำนวนจริง 1 จำนวน แทนความน่าจะเป็นที่มากที่สุดที่เป็นไปได้ของการช่วยเหลือแมว ถ้าเทพเจ้าทำการช่วยเหลืออย่าง Optimal ที่สุด
  (**หมายเหตุ:** คำตอบจะถูกต้องถ้าคาดเคลื่อนไม่มากกว่าคำตอบจริงเกิน 10^{-6})

First Data Example

Example Input	Example Output
2 2 1 1 1 2 2	0.9166666667



อธิบายตัวอย่างที่ 1: จากตัวอย่างของข้อมูลนำเข้าข้างต้น สามารถแสดงให้เห็นตามรูปที่ปรากฏทางด้านข้าง โดยที่ลูกศรแสดงเส้นทางลัดระหว่างห้องสองห้อง

หนึ่งในการวิธีช่วยเหลือที่ optimal ขอเทพเจ้ามีดังต่อไปนี้

1. วันแรก ใช้พลัง **"หยั่งรู้"** ที่ห้อง $(1,2)$
   (a) ความน่าจะเป็น $\frac{1}{4}$ ที่ระยะทางที่ใกล้ที่สุด จะเป็น 0 แมวจะอยู่ที่ห้อง $(1,2)$ กรณีนี้สามารถใช้พลัง **"ช่วยเหลือ"** เพื่อช่วยแมวได้ทันที
   (b) ความน่าจะเป็น $\frac{2}{4}$ ที่ระยะทางที่ใกล้ที่สุด จะเป็น 1 แมวอาจจะอยู่ที่ห้อง $(1,1)$ หรือ $(2,2)$ ก็ได้ กรณีนี้ควรรอวันที่ 2
   (c) ความน่าจะเป็น $\frac{1}{4}$ ที่ระยะทางที่ใกล้ที่สุด จะเป็น 2 แมวจะอยู่ที่ห้อง $(2,1)$ กรณีนี้สามารถใช้พลัง **"ช่วยเหลือ"** เพื่อช่วยแมวได้ทันที

2. จากข้อ 1(b) ระหว่างวันในขณะที่เทพเจ้านอนพัก การเคลื่อนที่ของแมวเป็นดังนี้
   - ความน่าจะเป็นที่แมวจะเดินไปห้อง $(1,2)$ คือ $\frac{2}{4} \times \frac{1}{3} = \frac{2}{12}$ ← เดินจาก $(1,1)$ หรือ $(2,2)$
   - ความน่าจะเป็นที่แมวจะเดินไปห้อง $(1,1)$ คือ $\frac{1}{4} \times \frac{1}{3} = \frac{1}{12}$ ← เดินจาก $(2,2)$
   - ความน่าจะเป็นที่แมวจะเดินไปห้อง $(2,2)$ คือ $\frac{1}{4} \times \frac{1}{3} = \frac{1}{12}$ ← เดินจาก $(1,1)$
   - ความน่าจะเป็นที่แมวจะเดินไปห้อง $(2,1)$ คือ $\frac{2}{4} \times \frac{1}{3} = \frac{1}{12}$ ← เดินจาก $(1,1)$ หรือ $(2,2)$

3. วันถัดมาวันที่ ใช้พลัง "หยั่งรู้" ที่ห้อง $(1,2)$ เช่นเดิม
   (a) ความน่าจะเป็น $\frac{2}{12}$ ที่ระยะทางที่ใกล้ที่สุดจะเป็น 0 แมวจะอยู่ที่ห้อง $(1,2)$ กรณีนี้สามารถชใช้พลัง **"ช่วยเหลือ"** แมวได้แน่นอน
   (b) ความน่าจะเป็น $\frac{2}{12}$ ที่ระยะทางที่ใกล้ที่สุดจะเป็น 2 แมวจะอยู่ที่ห้อง $(2,1)$ กรณีนี้สามารถใช้พลัง **"ช่วยเหลือ"** แมวได้แน่นอน
   (c) ความน่าจะเป็น $\frac{1}{12} + \frac{1}{12} = \frac{2}{12}$ ที่ระยะทางที่ใกล้ที่สุดจะเป็น 1 แมวจะอาจจะอยู่ที่ห้อง $(1,1)$ หรือ $(2,2)$ ก็ได้
   กรณีนี้ การเลือกห้องใดห้องหนึ่งเพื่อใช้พลัง **"ช่วยเหลือ"** จะมีโอกาสช่วยแมวสำเร็จด้วยความน่าจะเป็น $\frac{2}{12} \times \frac{1}{2} = \frac{1}{12}$

4. สรุปว่า ความน่าจะเป็นที่เทพเจ้าจะสามารถช่วยเหลือแมวได้ ตามข้อ 1(a), 1(c), 3(a), 3(b) และ 3(c) คือ

$$\frac{1}{4} + \frac{1}{4} + \frac{2}{12} + \frac{2}{12} + \frac{1}{12} = \frac{11}{12} \approx 0.9166666667$$

Second Data Example

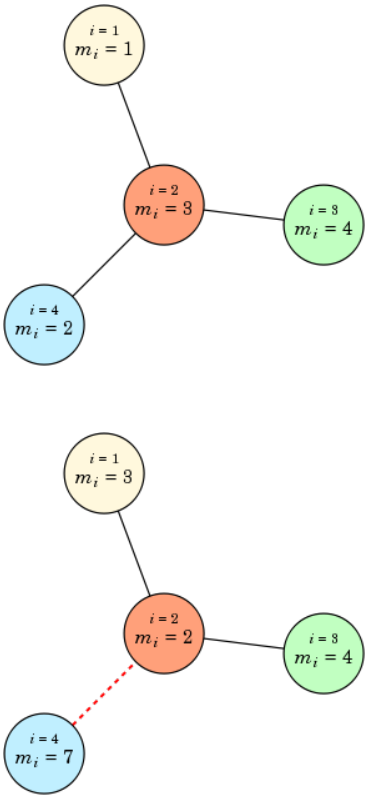
Example Input	Example Output
3 3 3 3 1 1 2 3 1 2 3 1 2 3 1	0.7259259259

Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

- test cases ชุดเล็กจะมีเงื่อนไข ขนาดของถ้ำสอดคล้องกับเงื่อนไข $1 \leq N, M \leq 20$ และ $N \times M \leq 200$ และจำนวนเส้นทางลัดสอดคล้องกับเงื่อนไข $1 \leq K \leq 100$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า ขนาดของถ้ำสอดคล้องกับเงื่อนไข $1 \leq N, M \leq 20$ และจำนวนเส้นทางลัดสอดคล้องกับเงื่อนไข $1 \leq K \leq 70,000$

คำถามที่ 69. Stable Molecule



Background

ในห้องปฏิบัติการวิจัยเคมีแห่งหนึ่ง คุณกุ้งกำลังศึกษาโครงสร้างของโมเลกุลชนิดหนึ่ง ซึ่งประกอบไปด้วยอะตอมหลากหลายชนิดมาประกอบกันด้วยพันธะที่เชื่อมระหว่างอะตอมบางคู่ จนมีโครงสร้างเป็นกราฟต้นไม้

กล่าวคือโครงสร้างโมเลกุลนี้จะประกอบด้วยอะตอมทั้งสิ้น N ชนิด ชนิดละ 1 อนุภาค (เรียกสั้น ๆ ว่า “ลูก”) และมีพันธะทั้งสิ้น $N-1$ พันธะที่เชื่อมอะตอมเหล่านี้เข้าด้วยกัน (เราจะเรียกอะตอมแต่ละลูกว่าลูกที่ i สำหรับ $i = 1, 2, \ldots, N$)

คุณกุ้งสามารถกำหนดค่ามวลของอะตอมแต่ละลูกได้ โดยที่มวลของอะตอมลูกที่ i จะกำหนดด้วยตัวแปร m_i ซึ่งเป็นจำนวนเต็มบวก กล่าวคือมีเงื่อนไขว่า $m_i \geq 1$

โครงสร้างโมเลกุลนี้จะ **“เสถียร”** ก็ต่อเมื่อ แรงดึงดูดระหว่างอะตอม 2 ลูกที่เป็นพันธะต่อกันจะมีค่าไม่เกิน P โดยแรงดึงดูดระหว่างมวลสามารถคำนวณได้จากผลคูณของมวลของอะตอมแต่ละลูก (นั่นแปลว่า $m_u \times m_v \leq P$ สำหรับทุกคู่อะตอม u และ v ที่มีพันธะต่อกัน)

ตัวอย่าง เมื่อพิจารณาโครงสร้างโมเลกุลที่เกิดจากอะตอม 4 ลูกที่เชื่อมกันดังรูปทางซ้าย และกำหนดให้ $P = 12$ แล้วพบว่าโครงสร้างโมเลกุลรูปบนจะเสถียร แต่โครงสร้างอันล่างจะไม่เสถียร

Problem Statement

จงเขียนโปรแกรมเพื่อรับข้อมูลที่กำหนดโครงสร้างโมเลกุล และค่าขีดจำกัดสูงสุดของแรงดึดดูดระหว่างอะตอม P แล้วหาว่าคุณกุ้งจะสามารถกำหนดค่ามวลให้แก่อะตอมแต่ละลูกให้แตกต่างกันได้ทั้งหมดกี่รูปแบบ โดยที่โมเลกุลจะยังเสถียรอยู่

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดที่ 1: มีจำนวนเต็มสองจำนวน N และ P คั่นด้วยช่องว่าง
- อีก $N-1$ บรรทัดถัดมา บรรทัดที่ $j+1$ จะมีจำนวนเต็ม u_j และ v_j คั่นด้วยช่องว่าง ซึ่งระบุว่ามีพันธะระหว่างอะตอมลูกที่ u_j และอะตอมลูกที่ v_j

```
1   N P
2   u_1 v_1
3   u_2 v_2
...
N+1 u_N v_N
```

หมายเหตุ: กำหนดให้ $1 \leq u_j, v_j \leq N$ และ $u_j \neq v_j$

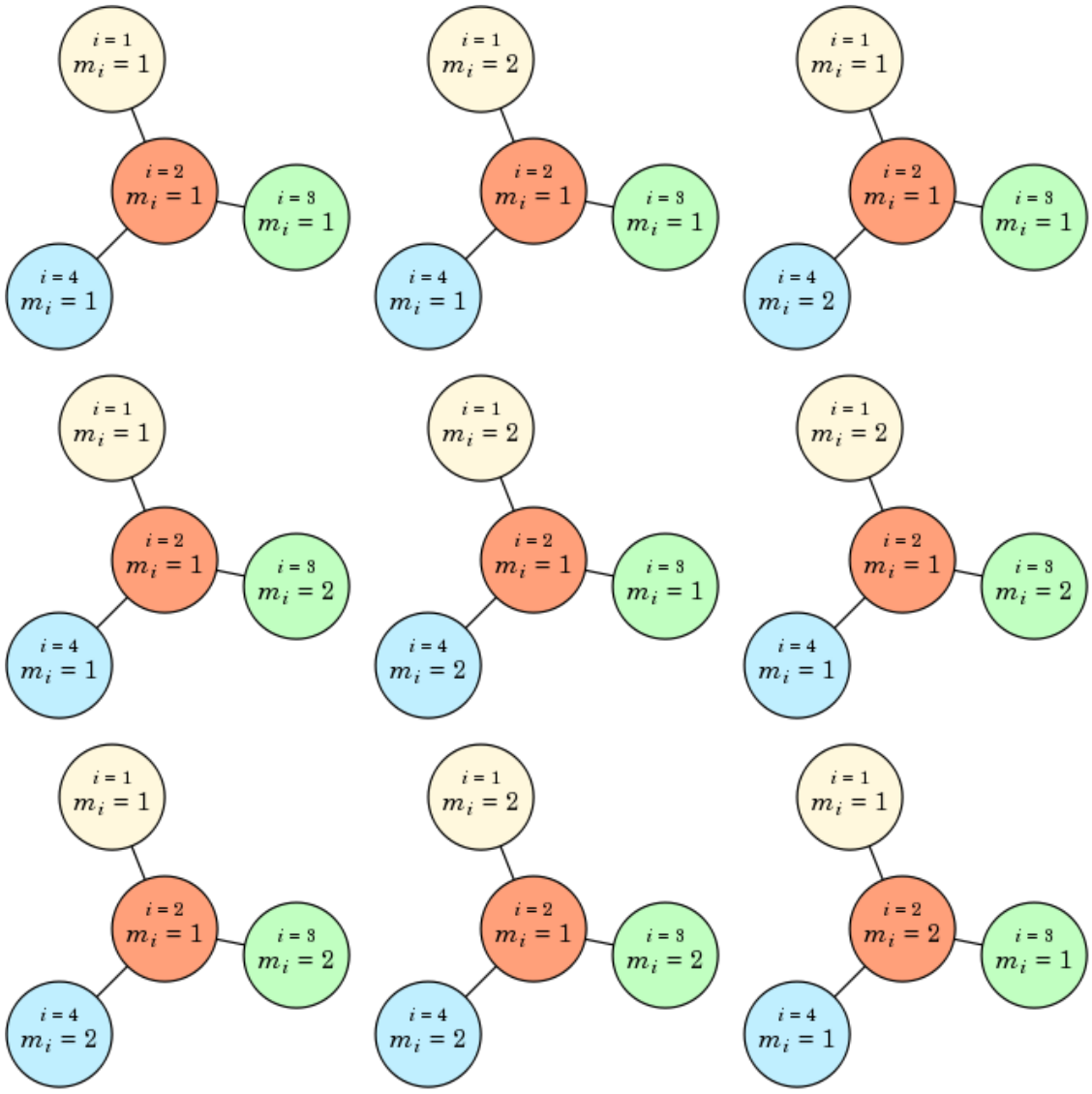
Output Format

- คำตอบประกอบด้วยจำนวนเต็ม 1 ตัว ระบุจำนวนรูปแบบของโมเลกุลที่คุณกุ้งสามารถกำหนดมวลให้อะตอมแต่ละลูกได้ โดยคำตอบจะต้องอยู่ในรูปของเศษที่เกิดจากการหารด้วย $1{,}000{,}000{,}007$

First Data Example

Example Input	Example Output
4 2 1 2 2 3 4 2	9

อธิบายตัวอย่างที่ 1: เราสามารถแจกแจงรูปแบบของการกำหนดมวลให้อะตอมแต่ละลูกในโมเลกุลได้ดังนี้



Second Data Example

Example Input	Example Output
5 3 4 2 3 2 1 3 5 3	51

Constraints

โปรแกรมของคุณจะถูกทดสอบกับ test cases สองชุด (เรียกว่าชุดเล็ก และชุดใหญ่)

- test cases ชุดเล็กจะมีเงื่อนไขว่า ค่าขีดจำกัดสูงสุดของแรงดึดดูดระหว่างอะตอมจะสอดคล้องกับเงื่อนไข $1 \leq P \leq 10^3$
- test cases ชุดใหญ่จะมีเงื่อนไขว่า ค่าขีดจำกัดสูงสุดของแรงดึดดูดระหว่างอะตอมจะสอดคล้องกับเงื่อนไข $1 \leq P \leq 10^9$
- สำหรับทุก test cases จะมีเงื่อนไขว่า จำนวนของอะตอมจะสอดคล้องกับเงื่อนไข $1 \leq N \leq 1{,}000$

คำถามที่ 70. Rotate Sort (Optimization Problem)

*โจทย์ Coding ข้อนี้ ปรากฏในการแข่งขัน **TECHJAM 2018** รอบชิงชนะเลิศระดับประเทศ โดยเป็นโจทย์ที่ผู้เข้าแข่งขันจะต้อง optimize เพื่อหาคำตอบที่ดีที่สุดในบรรดาผู้เข้าแข่งขันทั้งหมด หากโปรแกรมของผู้เข้าแข่งขันให้ผลลัพธ์ที่ดีมากเท่าใด ก็จะยิ่งมีโอกาสได้คะแนนสูงมากขึ้นเท่านั้น*

กติการที่ปรากฏในส่วนของ Submission and Scoring นั้นมีนัยยะสำคัญเฉพาะภายในการแข่งขันดังกล่าวเท่านั้น โปรดใช้เนื้อหาดังกล่าวเพื่อการอ้างอิงเท่านั้น

Background

กำหนดให้ operation ชื่อว่า `rotate_3way` รับ input argument อยู่ 3 อย่าง ได้แก่

- array A ของจำนวนเต็ม
- ดัชนี p และ q ภายใน array A ซึ่ง $1 \leq p \leq q \leq |A|$
  (เมื่อ $|A|$ คือความยาวของ array A)

เมื่อเรียกใช้งาน `rotate_3way(A, p, q)` จะได้ output value เป็น array ที่มีลักษณะเป็นดังนี้

$$\texttt{rotate_3way(A, p, q)} = \left[\underbrace{\texttt{A[q+1], A[q+2], …, A[N],}}_{\text{can be empty}}\texttt{A[p], A[p+1],…,A[q],}\underbrace{\texttt{A[1], A[2], …, A[p-1]}}_{\text{can be empty}}\right]$$

ยกตัวอย่างเช่น

- `rotate_3way([5, 2, 1, 0], 2, 3) = [0, 2, 1, 5]`
- `rotate_3way([5, 2, 1, 0], 2, 2) = [1, 0, 2, 5]`
- `rotate_3way([5, 2, 1, 0], 1, 2) = [1, 0, 5, 2]`
- `rotate_3way([5, 2, 1, 0], 1, 4) = [5, 2, 1, 0]`

Problem Statement

โจทย์ข้อนี้ โปรแกรมของผู้เข้าแข่งขันจะได้รับข้อมูลนำเข้าเป็น array X ของจำนวนเต็มที่ไม่เรียงลำดับ กำหนดให้ $N = |X|$ คือความยาวของ array X

เป้าหมายของโปรแกรมของผู้เข้าแข่งขันคือ จะต้องเรียงลำดับจำนวนใน array X โดยเรียกใช้งาน operation `rotate_3way` เป็นจำนวนครั้งให้ได้น้อยที่สุดเท่าที่ผู้เข้าแข่งขันสามารถทำได้

กล่าวคือ เมื่อโปรแกรมของผู้เข้าแข่งขันได้รับข้อมูล array X แล้วจะต้องระบุว่าจะเรียกใช้งาน `rotate_3way` ทั้งสิ้นกี่ครั้ง และในแต่ละครั้ง จะกำหนดค่าดัชนี p และ q เท่าใดตามลำดับ

สำหรับเกณฑ์การให้คะแนนในข้อนี้ โปรดดูหัวข้อ Submission and Scoring

Program Specification

โปรแกรมที่คุณเขียนจะต้องอ่านข้อมูลจาก stardard input และเขียนคำตอบลง standard output โดยข้อมูลจะมีฟอร์แมตดังต่อไปนี้

Input Format

- บรรทัดที่ 1: จะมีจำนวนเต็มหนึ่งจำนวน ระบุ N ซึ่งเป็นความยาวของ array X
- อีก N บรรทัดถัดมา บรรทัดที่ $i+1$ จะระบุจำนวน $X[i]$ ของ array X

```
1   N
2   X[1]
3   X[2]
...
N+1 X[N]
```

Output Format

- บรรทัดที่ 1: จะมีจำนวนเต็ม K หนึ่งจำนวน ระบุจำนวนครั้งที่ operation `rotate_3way` จะถูกเรียกใช้งาน
- อีก K บรรทัดถัดมา บรรทัดที่ $j+1$ จะระบุจำนวนเต็มสองจำนวน p_j และ q_j คั่นด้วยช่องว่าง ซึ่งเป็นดัชนี p และ q ของการเรียกใช้งาน operation `rotate_3way` ครั้งที่ j

Data Example

Example Input	Example Output A	Example Output B
4 5 2 1 0	3 2 2 2 2 3 4	4 2 4 2 4 3 3 3 3

อธิบายตัวอย่าง: สังเกตว่าลำดับของ operation ที่แสดงในทั้งสองคำตอบข้างต้น สามารถทำให้ array X เรียงลำดับได้ถูกต้อง แต่คำตอบ A ใช้จำนวน operation น้อยกว่า B

Submission and Scoring

สำหรับโจทย์ข้อนี้ พึงทราบว่า test case แต่ละอันจะมีคะแนนเต็ม 18 คะแนน นอกจากนั้น คะแนนรวมตอนท้ายสำหรับโจทย์ข้อนี้จะคิดจากค่าเฉลี่ยของคะแนนจากแต่ละ test case จึงทำให้คะแนนรวมสูงสุดที่เป็นไปได้สำหรับข้อนี้คือ 18 คะแนน เช่นกัน (เศษทศนิยมของคะแนนจะถูกปัดทิ้งให้เหลือทศนิยม 6 ตำแหน่ง)

Minimum requirement

สำหรับ test case แต่ละอัน โปรแกรมของผู้เข้าแข่งขันจะถือว่าให้ผลลัพธ์ที่ถูกต้อง ก็ต่อเมื่อ

1. โปรแกรมให้ผลลัพธ์ตรงกับ Program Specification ที่โจทย์กำหนดให้
2. ผลลัพธ์ของโปรแกรมเรียกใช้งาน operation `rotate_3way` ที่ทำให้ array X เรียงลำดับจากน้อยไปมากได้ถูกต้อง
3. ผลลัพธ์ของโปรแกรมเรียกใช้งาน operation `rotate_3way` ไม่เกิน $100N$ ครั้ง

หมายเหตุ: หากโปรแกรมของคุณให้ผลลัพธ์ไม่ถูกต้อง จะได้ 0 คะแนนทันที

Scoring scheme for correct answers

เมื่อโปรแกรมของคุณให้ผลลัพธ์ที่ถูกต้อง กำหนดให้

- โปรแกรมของคุณเรียกใช้งาน operation `rotate_3way` เป็นจำนวน K ครั้งสำหรับ test case นี้
- ในบรรดาผู้เข้าแข่งขันทั้งหมด จำนวน operation ที่น้อยที่สุดที่มีผู้เข้าแข่งขันทำได้คือ K^* ครั้ง สำหรับ test case เดียวกัน

แล้วคะแนนของคุณสำหรับ test case อันนี้จะคำนวณจากสูตรดังต่อไปนี้

$$\begin{cases} 18 & \text{if } K = K^* \\ 6 + 9 \cdot \dfrac{K^*}{K} & \text{if } K > K^* \end{cases}$$

สังเกตว่าเมื่อโปรแกรมให้ผลลัพธ์ที่ถูกต้อง รับประกันว่าจะได้คะแนนอย่างน้อย 6 คะแนนสำหรับ test case นั้น ๆ และกำหนดให้เศษที่เกิดขึ้นจากการหาร จะถูกปัดทิ้งให้เหลือทศนิยม 6 ตำแหน่ง

Constraints

แต่ละ test case จะมีเงื่อนไขดังต่อไปนี้

- ความยาวของ array X จะสอดคล้องกับเงื่อนไข $2 \leq N \leq 100$
- จำนวนที่พบใน array X จะอยู่ในช่วง 0 ถึง 1000 ซึ่งอาจมีบางจำนวนซ้ำกันได้