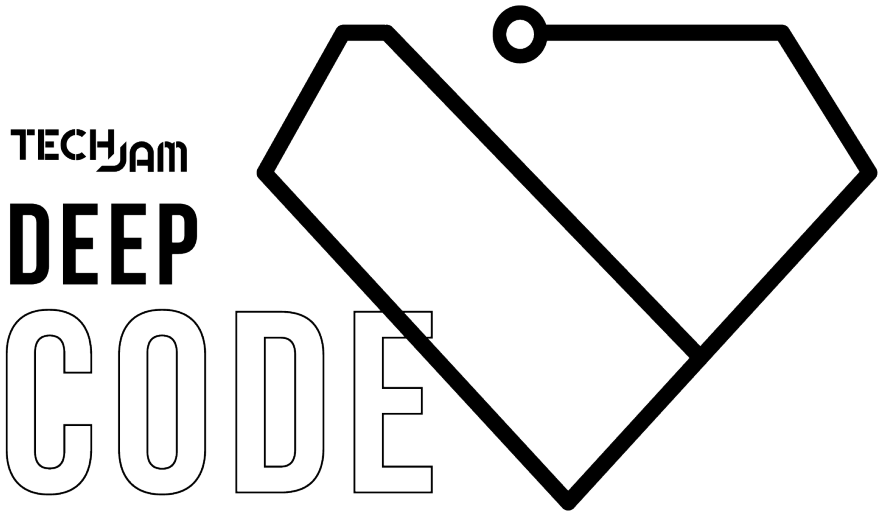


รวบรวมโจทย์แข่งขัน

TechJam 2019
DEEP CODE

จัดการแข่งขันโดย

KBTG

สารบัญ

ติดต่อสอบถามหรือติดตามรายละเอียดเพิ่มเติมได้ที่ https://techjam.tech

บทที่ 1
เกี่ยวกับการแข่งขัน

กสิกร บิซิเนส–เทคโนโลยี กรุ๊ป (KBTG) ได้จัดการแข่งขัน **TechJam** ขึ้นทุกปีเพื่อค้นหาสุดยอดผู้เชี่ยวชาญทางด้านเทคโนโลยี ทั้งในด้าน Code · Data · Design ที่จะมาผสานกันเป็นเครื่องจักรกำลังสูงที่จะขับเคลื่อนวงการเทคโนโลยีในประเทศไทยไปข้างหน้าต่อไป

สำหรับ **TechJam Code Track** นั้น เรามุ่งเน้นที่จะค้นหาโปรแกรมเมอร์ที่มีทักษะและความสามารถรอบด้าน ทั้งในเรื่องของการคิดวิเคราะห์ แก้ปัญหาเชิงคำนวณ ในเรื่องความรู้ความสามารถเกี่ยวกับคอมพิวเตอร์ โครงสร้างข้อมูล และอัลกอริทึม และในเรื่องของทักษะในการพัฒนาซอฟต์แวร์ที่ดีและได้มาตรฐาน ซอฟต์แวร์ที่ดีนั้นสามารถมองได้หลายปัจจัย เช่น

- ซอฟต์ทำงานได้อย่างถูกต้อง สามารถประมวลผลข้อมูลได้แม่นยำ และทำงานได้สอดคล้องกับสเปกที่วางไว้
- ซอฟต์แวร์ทำงานได้อย่างมีประสิทธิภาพ เช่น ประมวลผลได้เร็ว ใช้ทรัพยากรได้อย่างคุ้มค่า เป็นต้น
- ซอฟต์แวร์มีการวางโครงสร้างที่ดี มีความยืดหยุ่น ทำให้สามารถต่อเติมหรือแก้ไขโค้ดได้ง่ายเมื่อต้องการ และบำรุงดูแลรักษาได้ง่ายในระยะยาว

การแข่งขัน **TechJam Code Track** ในปีก่อนๆ จะเน้นการแก้ปัญหาเชิงอัลกอริทึมเป็นหลัก แต่สำหรับการแข่งขัน **TechJam 2019 DEEP CODE** ในปีนี้ เราปรับเปลี่ยนรูปแบบการแข่งขันเพื่อทดสอบความสามารถของโปรแกรมเมอร์ในมุมมองที่กว้างขึ้น กล่าวคือเราได้สร้างการแข่งขันรูปแบบใหม่ นั่นก็คือการพัฒนาแอปพลิเคชันตามโจทย์ที่กำหนดให้ภายใต้สถานการณ์จำลองเสมือนจริง เพื่อทดสอบความสามารถของผู้เข้าแข่งขันในการรับมือโจทย์ที่มีการเปลี่ยนแปลง Requirement หรือ Specification ที่เกิดขึ้นหลายครั้งระหว่างการแข่งขัน

อย่างไรก็ตาม KBTG มิได้คาดหวังที่จะค้นหาสุดยอดโปรแกรมเมอร์เพียงอย่างเดียว เราหวังว่าการแข่งขัน **TechJam** จะจุดประกายความสนใจทางด้านเทคโนโลยีให้แก่ชุมชนนักพัฒนาซอฟต์แวร์และคนรุ่นใหม่ที่จะมาเป็นกำลังสำคัญในวงการเทคโนโลยีต่อไป

—— ทีมงาน **TechJam 2019 DEEP CODE**

ลักษณะของโจทย์

โจทย์ที่ใช้ในการแข่งขัน **TechJam 2019 DEEP CODE** มี 3 รูปแบบดังนี้

โจทย์ปัญหาเชิงอัลกอริทึม

ผู้เข้าแข่งขันจะต้องเขียนโปรแกรมเพื่อรับข้อมูลนำเข้า (Input Data) ไปประมวลผลและคืนออกมาเป็นข้อมูลส่งออก (Output Data) โจทย์จะระบุว่าข้อมูลนำเข้ามีลักษณะอย่างไรและต้องการให้นำไปประมวลผลออกมาเป็นข้อมูลส่งออกในลักษณะใด ผู้เข้าแข่งขันจะต้องคิดวิเคราะห์สารจากโจทย์แล้วจึงเลือกสรรโครงสร้างข้อมูลและอัลกอริทึมที่เหมาะสมกับงานมาเขียนโปรแกรมให้ทำงานตามที่โจทย์ระบุ

โจทย์ปริศนาลับสมอง

ผู้เข้าแข่งขันจะต้องไขปริศนาหาคำตอบที่ถูกต้อง โดยจะใช้เทคนิคและเครื่องมือใดในการช่วยค้นหาคำตอบ

โจทย์เขียน Containerized Application

ผู้เข้าแข่งขันจะต้องเขียน Web Service Application ตาม Requirement และ Specification ที่กำหนดให้ แต่โจทย์จะสามารถเปลี่ยนแปลงและถูกแก้ไขได้ตลอดระยะเวลาในการแข่งขัน ผู้เข้าแข่งขันไม่ทราบว่าล่วงหน้าว่าจะมี Requirement ส่วนใดเปลี่ยนบ้าง และจะมีการประกาศเปลี่ยนแปลงเมื่อใด

รูปแบบการแข่งขัน

การแข่งขัน **TechJam 2019 DEEP CODE** แบ่งออกเป็น 3 รอบหลัก ๆ ได้แก่

บททดสอบคัดกรองรอบลงทะเบียน

ผู้เข้าแข่งขันจะต้องเขียนโปรแกรมแก้ปัญหาเชิงอัลกอริทึมอย่างง่ายจำนวน 3 ข้อผ่านระบบตรวจออนไลน์ ผู้เข้าแข่งขันที่ทำคะแนนรวมได้ 50% ขึ้นไปจะทีสิทธิแข่งขันใน First Round ต่อไป

First Round

ผู้เข้าแข่งขันจะต้องเขียนโปรแกรมแก้ปัญหาเชิงอัลกอริทึมแสนท้าทายจำนวน 3 ข้อผ่านระบบตรวจออนไลน์ ผู้เข้าแข่งขันที่ทำคะแนนรวมได้ดีที่สุดประมาณ 30 ทีมจะได้รับเชิญให้มาแข่งใน Final Round

Final Round

เป็นการแข่งขันเขียนโปรแกรมเต็มรูปแบบที่อาคาร KBTG แบ่งออกเป็นการแข่งขันภาคเช้าและภาคบ่าย ในช่วงเช้าจะเป็นการแก้ปัญหาเชิงอัลกอริทึมผสมกับการไขโจทย์ปริศนาลับสมอง ภายในเวลา 3 ชั่วโมง ส่วนในช่วงบ่ายจะเป็นการเขียนซอฟต์แวร์เป็น Containerized Web Service Application ตาม Requirement ที่กำหนดภายในเวลา 4 ชั่วโมง คะแนนของการแข่งขันแต่ละส่วนถูกแบ่งออกเป็นดังนี้

- [90 คะแนน] **โจทย์ปัญหาเชิงอัลกอริทึม** 3 ข้อ ข้อละ 30 คะแนน
- [30 คะแนน] **โจทย์ปริศนาลับสมอง** 5 ข้อ ข้อละ 6 คะแนน
- [80 คะแนน] **โจทย์เขียน Containerized Application** 1 ตัว

บทที่ 2
โจทย์อัลกอริทึม First Round

2.1 Emergency Stress Test

Problem Statement

องค์การบริการกู้ภัยฉุกเฉิน (Emergency Response Service Authority; ERSA) กำลังจะจัดงานซ้อมทดสอบกู้ภัยประจำปี ซึ่งจัดขึ้น ณ ถนนทางหลวงทางตรงสายหนึ่ง เพื่อทดสอบขีดความสามารถในการตอบสนองต่อภัยที่เกิดขึ้นที่อาจเกิดขึ้นอย่างฉับพลันและพร้อมกันในหลาย ๆ จุด สมาชิกหน่วยกู้ภัยฉุกเฉินแต่ละคนจากทั้งหมด n คนจะประจำการอยู่ที่จุดต่าง ๆ บนทางหลวงสายนี้ ทางหลวงสายนี้เปรียบเสมือนเส้นจำนวนในแกน X และจุดต่าง ๆ บนทางหลวงสามารถแทนได้ด้วยพิกัดบนแกน X นอกจากนี้ระยะทาง 1 เมตรบนทางหลวงสายนี้มีค่าเท่ากับระยะ 1 หน่วยบนแกน X

การทดสอบจะเริ่มขึ้นเมื่อมีเสียงนกหวีด เมื่อเสียงนกหวีดดังขึ้น นาฬิกาจับเวลา m เรือนจะปรากฏขึ้นตามสถานที่ตำแหน่งต่าง ๆ บนทางหลวง นอกจากนั้นนาฬิกาแต่ละเรือนจะเริ่มนับเวลาถอยหลังจากค่าเริ่มต้นที่แตกต่างกัน เป้าหมายของทีมหน่วยกู้ภัยโดยรวมคือจะต้องส่งสมาชิกของทีมตัวเองไปเคลมนาฬิกาจับเวลาให้ได้จำนวนมากที่สุดก่อนนาฬิกาเรือนนั้น ๆ หมดเวลาลงเสียก่อน ตลอดเวลาที่ทำการทดสอบนี้ สมาชิกทีมกู้ภัยแต่ละคนสามารถเคลมนาฬิกาได้อย่างมากที่สุด 1 เรือน และนาฬิกาเรือนดังกล่าวจะถูกเคลมได้หากสมาชิกคนนั้นวิ่งไปถึงนาฬิกาเรือนดังกล่าวก่อนหมดเวลาหรือทันเวลาฉิวเฉียดพอดีเท่านั้น

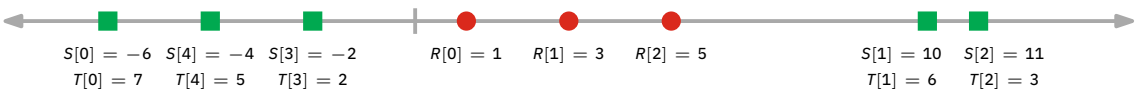
หมายเหตุ กฎความปลอดภัยของทีมหน่วยกู้ภัยคือ สมาชิกแต่ละคนสามารถเคลื่อนไหวด้วยอัตราเร็วสูงสุด 1 เมตรต่อวินาทีเท่านั้น

Situation Example

ลองพิจารณาตัวอย่างสถานการณ์ดังนี้ สมมติว่า ทีมหน่วยกู้ภัยมีสมาชิกทั้งสิ้น $n = 3$ คน แต่ละคนประจำการอยู่ที่ตำแหน่ง $x = 1$, $x = 3$ และ $x = 5$ ตามลำดับ

เมื่อเสียงนกหวีดดังขึ้น นาฬิกาจับเวลา $m = 5$ เรือนปรากฏขึ้นดังต่อไปนี้

- นาฬิกาเรือนที่ 1 ปรากฏที่ตำแหน่ง $x = -6$ และเริ่มต้นนับเวลาถอยหลังที่ 7 วินาที
- นาฬิกาเรือนที่ 2 ปรากฏที่ตำแหน่ง $x = 10$ และเริ่มต้นนับเวลาถอยหลังที่ 6 วินาที
- นาฬิกาเรือนที่ 3 ปรากฏที่ตำแหน่ง $x = 11$ และเริ่มต้นนับเวลาถอยหลังที่ 3 วินาที
- นาฬิกาเรือนที่ 4 ปรากฏที่ตำแหน่ง $x = -2$ และเริ่มต้นนับเวลาถอยหลังที่ 2 วินาที
- นาฬิกาเรือนที่ 5 ปรากฏที่ตำแหน่ง $x = -4$ และเริ่มต้นนับเวลาถอยหลังที่ 5 วินาที



ในกรณีนี้ ทีมหน่วยกู้ภัยจะสามารถเคลมนาฬิกาจับเวลาได้มากที่สุดเพียง 2 เรือนเท่านั้น นั่นคือ

- สมาชิกที่ประจำอยู่ที่ตำแหน่ง $x = 1$ เคลมนาฬิกาเรือนที่ 1 อย่างทันฉิวเฉียด
- สมาชิกที่ประจำอยู่ที่ตำแหน่ง $x = 5$ เคลมนาฬิกาเรือนที่ 2 ก่อนหมดเวลา 1 วินาทีพอดี

Objectives

จงเขียนโปรแกรมเพื่อรับ Input Data ต่อไปนี้

- ข้อมูลเกี่ยวกับสมาชิกทีมหน่วยกู้ภัยทั้งสิ้น n ราย (โดยที่ $1 \le n \le 200{,}000$)
  กล่าวคือสมาชิกกู้ภัยคนที่ i สำหรับ $i = 0, 1, \ldots, n-1$ จะมีข้อมูลดังต่อไปนี้
  - $R[i]$ คือตำแหน่งเริ่มต้นบนแกน X ของสมาชิกคนที่ i
    (โดยที่ $-1{,}000{,}000{,}000 \le R[i] \le 1{,}000{,}000{,}000$)
- ข้อมูลของนาฬิกาทั้งสิ้น m เรือนที่ปรากฏขึ้นเมื่อเริ่มเสียงสัญญาณนกหวีด (โดยที่ $1 \le m \le 200{,}000$)
  กล่าวคือนาฬิกาเรือนที่ j สำหรับ $j = 0, 1, \ldots, m-1$ จะมีข้อมูลดังต่อไปนี้
  - $S[j]$ คือตำแหน่งที่นาฬิกาเรือนที่ j ปรากฏบนแกน X
    (โดยที่ $-1{,}000{,}000{,}000 \le S[j] \le 1{,}000{,}000{,}000$)
  - $T[j]$ คือเวลาเริ่มต้นของนาฬิกาเรือนที่ j ในหน่วยวินาที
    (โดยที่ $0 \le T[j] \le 1{,}000{,}000{,}000$)

แล้วจึงคำนวณจำนวนนาฬิกาที่สามารถเคลมได้มากที่สุดเท่าที่เป็นไปได้ และคืนค่าคำตอบดังกล่าวเป็น Output Data ของโปรแกรม

Interfaces and Data Format

โปรแกรมที่เขียนขึ้นจะต้องรับ Input Data ผ่าน Standard Input ซึ่งมีรูปแบบดังต่อไปนี้

- บรรทัดแรกมีจำนวนเต็มสองจำนวน n และ m
- บรรทัดที่ $i+2$ สำหรับ $i = 0, 1, \ldots, n-1$ จะมีจำนวนเต็มหนึ่งจำนวน ซึ่งก็คือ $R[i]$
- บรรทัดที่ $n+j+2$ สำหรับ $j = 0, 1, \ldots, m-1$ จะมีจำนวนเต็มสองจำนวนที่ถูกคั่นด้วยช่องว่าง
  ซึ่งก็คือ $S[j]$ และ $T[j]$

```
  1   n m
  2   R[0]
  3   R[1]
 ...
n+1   R[n-1]
n+2   S[0] T[0]
n+3   S[1] T[1]
 ...
n+m+1 S[m-1] T[m-1]
```

โปรแกรมที่เขียนขึ้นจะต้องคืน Output Data ผ่าน Standard Output เป็นจำนวนเต็ม 1 จำนวน ซึ่งเป็นคำตอบของโจทย์ตามที่ระบุไว้ในหัวข้อ Objectives ข้างต้น

Example Input	Example Output
3 5 1 3 5 -6 7 10 6 11 3 -2 2 -4 5	2

Scoring

โปรแกรมของคุณจะถูกทดสอบกับ Test Cases ที่มีเงื่อนไขต่าง ๆ ดังนี้

SMALL (คะแนน 20%)

รับประกันว่าจำนวนของสมาชิกกู้ภัยและจำนวนของนาฬิกาจะสอดคล้องกับเงื่อนไข $1 \le n,m \le 1,000$ และค่าพิกัดแกน X จะอยู่ในช่วงตั้งแต่ $-100,000$ จนถึง $100,000$

MEDIUM (คะแนน 35%)

รับประกันว่าค่าพิกัดแกน X จะอยู่ในช่วงตั้งแต่ $-100,000$ จนถึง $100,000$

LARGE (คะแนน 45%)

ไม่มีเงื่อนไขเพิ่มเติม

Limitations

โปรแกรมจะถูกจำกัดเวลาอยู่ที่ 1.0 วินาทีต่อ Test Case (baseline) และถูกจำกัดหน่วยความจำอยู่ที่ 512 MB

- สำหรับโปรแกรมที่เขียนด้วยภาษา C หรือ C++ จะถูกจำกัดเวลาเท่ากับค่า baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา Go หรือ Java จะถูกจำกัดเวลาอยู่ที่ 1.5 เท่าของ baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา JavaScript หรือ Python จะถูกจำกัดเวลาอยู่ที่ 2.5 เท่าของ baseline ข้างต้น

2.2 Irregular Sequence

Problem Statement

กำหนดให้ `S` เป็น Sequence ของจำนวนเต็ม `S[0]`, `S[1]`, ..., `S[n-1]` ซึ่งมีค่าอยู่ในช่วง `1` ถึง `k` Sequence `S` จะถือว่าเป็น **Irregular Sequence** ก็ต่อเมื่อใน Sequence ดังกล่าวไม่ปรากฏรูปแบบที่มีความซ้ำซ้อนบางประการ กล่าวคือ Sequence `S` ดังกล่าวจะมีสมบัติ 2 ข้อดังต่อไปนี้ จึงจะถือว่าเป็น Irregular Sequence

1. ไม่มีจำนวนที่ซ้ำกันอยู่ติดกัน (หมายถึง `S[i-1] ≠ S[i]` สำหรับ `i = 1, 2, ..., n-1`)
2. ไม่มี Subsequence (ลำดับย่อย) ความยาว 4 จำนวนชุดใด ๆ ที่เป็น Alternating Subsequence (หรือกล่าวอีกนัยหนึ่งคือสำหรับ `w`, `x`, `y`, `z` ใด ๆ ซึ่ง `0 ≤ w < x < y < z ≤ n-1` จะพบว่า `S[w] = S[y] ≠ S[x] = S[z]` ไม่เป็นความจริง)

ลองพิจารณาตัวอย่างต่อไปนี้โดยสมมติว่า `k = 5`

- Sequence `[1, 2, 3, 4, 5]` จะเป็น Irregular Sequence
- Sequence `[1, 2, 3, 1, 4]` จะเป็น Irregular Sequence เช่นกัน
- Sequence `[1, 2, 3, 3, 4]` จะไม่เป็น Irregular Sequence เพราะปรากฏเลข `3` ติดกันใน Sequence ดังกล่าว
- Sequence `[1, 2, 3, 1, 3]` จะไม่เป็น Irregular Sequence เพราะมี Alternating Subsequence `[1, 3, 1, 3]` ปรากฏอยู่

สำหรับโจทย์ข้อนี้ จะมี Input Data เป็น Irregular Sequence `P` ซึ่งมีขนาดเล็ก เราต้องการเติมจำนวนเต็มจากช่วง `1` ถึง `k` ต่อท้าย Sequence `P` นี้ให้ยาวที่สุดโดยที่ Sequence ผลลัพธ์ยังคงเป็น Irregular Sequence อยู่เช่นเดิม กล่าวอีกนัยหนึ่งคือ เราต้องการหา Irregular Sequence `S` ที่ยาวที่สุดที่มี Sequence `P` เป็น Prefix หากมีคำตอบ `S` หลายคำตอบ ให้ตอบ Sequence อันแรกสุดตามลำดับของ Lexicographical Ordering

Lexicographical Ordering

กำหนดให้ S_1 และ S_2 คือ Sequence สองอันที่มีความยาวเท่ากัน แล้ว S_1 จะมาก่อน S_2 ใน Lexicographical Ordering ก็ต่อเมื่อมีจำนวนเต็ม j ซึ่ง $S_1[i] = S_2[i]$ สำหรับทุก ๆ $0 \le i < j$ และ $S_1[j] < S_2[j]$

Situation Example

สมมติว่า `k = 5` และ Input Sequence `P` มีค่าดังต่อไปนี้

```
P = [1, 3, 5, 2, 3]
```

แล้วจะพบว่ามี Irregular Sequence `S` ที่เป็นส่วนต่อขยายของ Input Sequence `P` ได้สองรูปแบบคือ

```
S = [1, 3, 5, 2, 3, 4, 3, 1]  # Possibility #1
S = [1, 3, 5, 2, 3, 1, 4, 1]  # Possibility #2
```

ในบรรดารคำตอบที่เป็นไปได้ข้างต้น โปรแกรมควรให้คำตอบที่สอง `S = [1, 3, 5, 2, 3, 1, 4, 1]` เนื่อง จากเป็นคำตอบที่ปรากฏแรกสุดในลำดับ Lexicographical Ordering

Objectives

จงเขียนโปรแกรมเพื่อรับ Input Data ต่อไปนี้

- จำนวนเต็ม `k` (โดยที่ `2 ≤ k ≤ 200,000`) ระบุช่วงจำนวนที่สามารถปรากฏใน Sequence ได้
- Sequence `P` (ประกอบด้วยจำนวนเต็ม `P[0]`, `P[1]`, ..., `P[m-1]` โดยที่ `1 ≤ m ≤ 200,000` และสมาชิก `P[i]` แต่ละจำนวนจะมีค่าสอดคล้องกับเงื่อนไข `1 ≤ P[i] ≤ k`)

แล้วจึงคำนวณ Irregular Sequence `S` ที่ยาวที่สุด ซึ่ง

- ประกอบไปด้วยจำนวนเต็มในช่วง `1` ถึง `k`
- มี Input Sequence `P` เป็น Prefix
- และเป็นคำตอบที่พบแรกสุดเพื่อพิจารณาตามลำดับ Lexicographical Ordering

และคืนค่า Sequence `S` ดังกล่าวเป็น Output Data ของโปรแกรม

Interfaces and Data Format

โปรแกรมที่เขียนขึ้นจะต้องรับ Input Data ผ่าน Standard Input ซึ่งมีรูปแบบดังต่อไปนี้

- บรรทัดแรกมีจำนวนเต็มสองจำนวน `k` และ `m`
- บรรทัดที่ `i+2` สำหรับ `i = 0, 1, ..., m-1` จะมีจำนวนเต็มหนึ่งจำนวน ซึ่งก็คือ `P[i]`

```
  1   k m
  2   P[0]
  3   P[1]
 ...
m+1   P[m-1]
```

โปรแกรมที่เขียนขึ้นจะต้องคืนค่า Sequence `S` เป็น Output Data (ซึ่งเป็นคำตอบตามที่ระบุไว้ในหัวข้อ Objectives ข้างต้น) ผ่าน Standard Output ซึ่งมีรูปแบบดังต่อไปนี้

- บรรทัดแรกมีจำนวนเต็มหนึ่งจำนวน `n`
- บรรทัดที่ `j+2` สำหรับ `j = 0, 1, ..., n-1` จะมีจำนวนเต็มหนึ่งจำนวน ซึ่งก็คือ `S[j]`

```
  1   n
  2   S[0]
  3   S[1]
 ...
n+1   S[n-1]
```

Example Input	Example Output
5 5 1 3 5 2 3	8 1 3 5 2 3 1 4 1

Scoring

โปรแกรมของคุณจะถูกทดสอบกับ Test Cases ที่มีเงื่อนไขต่าง ๆ ดังนี้

SMALL (คะแนน 20%)

รับประกันว่าความยาวของ Input Sequence จะสอดคล้องกับเงื่อนไข $1 \le m \le 100$

และสมาชิกของ Irregular Sequence จะมี Upper Bound ที่สอดคล้องกับเงื่อนไข $2 \le k \le 100$

MEDIUM (คะแนน 35%)

รับประกันว่าความยาวของ Input Sequence จะสอดคล้องกับเงื่อนไข $1 \le m \le 10,000$

และสมาชิกของ Irregular Sequence จะมี Upper Bound ที่สอดคล้องกับเงื่อนไข $2 \le k \le 10,000$

LARGE (คะแนน 45%)

ไม่มีเงื่อนไขเพิ่มเติม

Limitations

โปรแกรมจะถูกจำกัดเวลาอยู่ที่ 0.6 วินาทีต่อ Test Case (baseline) และถูกจำกัดหน่วยความจำอยู่ที่ 512 MB

- สำหรับโปรแกรมที่เขียนด้วยภาษา C หรือ C++ จะถูกจำกัดเวลาเท่ากับค่า baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา Go หรือ Java จะถูกจำกัดเวลาอยู่ที่ 1.5 เท่าของ baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา JavaScript หรือ Python จะถูกจำกัดเวลาอยู่ที่ 2.5 เท่าของ baseline ข้างต้น

2.3 Reconstruct Transaction List

Problem Statement

นายกสิกรเป็นคนที่ชอบจดบันทึกรายรับ-รายจ่ายของตนเอง ในตอนต้นเดือนของทุก ๆ เดือนเขาจะนำเงินที่เหลือค้างจากเดือนก่อนหน้าออกจากกระเป๋าตังค์ของเขาทั้งหมด และเริ่มต้นเดือนใหม่ด้วยเงิน 0 บาทในกระเป๋าตังค์ นอกจากนั้น เขาจะนำสมุดเปล่าสำหรับจดบันทึกรายการรายรับ-รายจ่ายใส่ไว้ในกระเป๋าเป้ไว้ 2 เล่ม เพื่อเอาไว้บันทึกรายรับ-รายจ่ายสำหรับเดือนใหม่

ตลอดทั้งเดือนนี้ ทุก ๆ ครั้งที่นายกสิกรมีเงินเข้าหรือออกจากกระเป๋าตังค์ (กล่าวคือมี **Transaction** รายการใหม่เกิดขึ้น) เขาจะสุ่มเลือกหยิบสมุดบันทึกรายการรายรับ-รายจ่ายจากกระเป๋าเป้ออกมา 1 เล่ม แล้วจากนั้นเขาจะจดบันทึก Transaction ดังกล่าวต่อท้ายรายการก่อนหน้าที่เคยจดไว้แล้วในสมุดเล่มนั้น

เมื่อถึงเวลาสิ้นเดือน นายกสิกรจะนำบันทึกรายการ Transaction จากสมุดทั้งสองเล่มมาเขียนรวมกันกลายเป็นบันทึกรายการรายรับ-รายจ่ายฉบับเต็ม (**Consolidated Transaction List**) โดยที่เงื่อนไขดังต่อไปนี้

1. ลำดับของ Transaction ที่ปรากฏในบันทึกฉบับเต็ม จะต้องสอดคล้องกับลำดับของ Transaction ที่ปรากฏในสมุดแต่ละเล่ม
2. เมื่อพิจารณารายการรายรับ-รายจ่าย จากรายการแรกไปยังรายการสุดท้าย ยอดคงเหลือสะสม ณ เวลาใด ๆ ไม่สามารถมีค่าติดลบได้ (นั่นก็เพราะว่าไม่มีเวลาใด ๆ ที่นายกสิกรใช้จ่ายเงินมากกว่าเงินที่คงเหลือในกระเป๋าตังค์ของเขา)

Situation Example

สมมติว่า `A` และ `B` คือลำดับของ Transaction ที่ปรากฏภายในสมุดทั้งสองเล่มตามลำดับ ซึ่งมีค่าดังต่อไปนี้

```
A = [10, -7, -4]
B = [-8, 6, 8, -4]
```

รายการ `A` และ `B` ข้างต้นสามารถนำมารวมกันเป็น Consolidated Transaction List ได้ 5 วิธี ได้แก่

วิธีที่ 1. `[ 10 A, -8 B, 6 B, -7 A, 8 B, -4 A, -4 B ]`
วิธีที่ 2. `[ 10 A, -8 B, 6 B, -7 A, 8 B, -4 B, -4 A ]`
วิธีที่ 3. `[ 10 A, -8 B, 6 B, 8 B, -7 A, -4 A, -4 B ]`
วิธีที่ 4. `[ 10 A, -8 B, 6 B, 8 B, -7 A, -4 B, -4 A ]`
วิธีที่ 5. `[ 10 A, -8 B, 6 B, 8 B, -4 B, -7 A, -4 A ]`

Objectives

จงเขียนโปรแกรมเพื่อรับ Input Data ต่อไปนี้

- ลำดับของ Transaction `A` (ประกอบด้วยจำนวนเต็ม `A[0]`, `A[1]`, ..., `A[n-1]` โดยที่ `1 ≤ n ≤ 2,000` และ Transaction `A[i]` แต่ละรายการจะสอดคล้องกับเงื่อนไข `-500,000 ≤ A[i] ≤ 500,000`) ซึ่งจะแสดงยอดรายรับหรือรายจ่ายแต่ละ Transaction ที่ถูกบันทึกไว้ภายในสมุดเล่มแรกในหนึ่งเดือน
- ลำดับของ Transaction `B` (ประกอบด้วยจำนวนเต็ม `B[0]`, `B[1]`, ..., `B[m-1]` โดยที่ `1 ≤ m ≤ 2,000` และ Transaction `B[j]` แต่ละรายการจะสอดคล้องกับเงื่อนไข `-500,000 ≤ B[j] ≤ 500,000`) ซึ่งจะแสดงยอดรายรับหรือรายจ่ายแต่ละ Transaction ที่ถูกบันทึกไว้ภายในสมุดเล่มที่สองในเดือนเดียวกัน

แล้วจึงคำนวณจำนวนวิธีที่นายกสิกรสามารถควบรวมบันทึกรายการ Transaction จากสมุดทั้งสองเล่มให้กลายเป็น Consolidated Transaction List ตอนท้ายเดือน และคืนค่าคำตอบดังกล่าวเป็น Output Data ของโปรแกรม

หากคำตอบข้างต้นมีค่ามากกว่าหรือเท่ากับ 1,000,000,007 ให้ตอบคำตอบในรูปเศษที่เกิดจากการหารค่าดังกล่าวด้วยจำนวนเต็ม 1,000,000,007

Interfaces and Data Format

โปรแกรมที่เขียนขึ้นจะต้องรับ Input Data ผ่าน Standard Input ซึ่งมีรูปแบบดังต่อไปนี้

- บรรทัดแรกมีจำนวนเต็มหนึ่งจำนวน ซึ่งก็คือ `n`
- บรรทัดที่ `i+2` สำหรับ `i = 0, 1, ..., n-1` จะมีจำนวนเต็มหนึ่งจำนวน ซึ่งก็คือ `A[i]`
- บรรทัดที่ `n+2` มีจำนวนเต็มหนึ่งจำนวน ซึ่งก็คือ `m`
- บรรทัดที่ `n+j+3` สำหรับ `j = 0, 1, ..., m-1` จะมีจำนวนเต็มหนึ่งจำนวน ซึ่งก็คือ `B[j]`

```
  1   n
  2   A[0]
  3   A[1]
 ...
n+1   A[n-1]
n+2   m
n+3   B[0]
n+4   B[1]
 ...
n+m+2 B[m-1]
```

โปรแกรมที่เขียนขึ้นจะต้องคืน Output Data ผ่าน Standard Output เป็นจำนวนเต็ม 1 จำนวน ซึ่งเป็นคำตอบของโจทย์ตามที่ระบุไว้ในหัวข้อ Objectives ข้างต้น

Example Input	Example Output
3 10 -7 -4 4 -8 6 8 -4	5

Scoring

โปรแกรมของคุณจะถูกทดสอบกับ Test Cases ที่มีเงื่อนไขต่าง ๆ ดังนี้

SMALL (คะแนน 20%)

รับประกันว่าความยาวของ Input Transaction List แต่ละอันจะสอดคล้องกับเงื่อนไข $1 \le n,m \le 100$ และ Transaction แต่ละรายการจะสอดคล้องกับเงื่อนไข $-1 \le A[i] \le 1$ สำหรับ $i = 0, 1, ..., n-1$ และ $-1 \le B[j] \le 1$ สำหรับ $j = 0, 1, ..., m-1$

MEDIUM (คะแนน 35%)

รับประกันว่าความยาวของ Input Transaction List แต่ละอันจะสอดคล้องกับเงื่อนไข $1 \le n,m \le 100$

LARGE (คะแนน 45%)

ไม่มีเงื่อนไขเพิ่มเติม

Limitations

โปรแกรมจะถูกจำกัดเวลาอยู่ที่ 1.0 วินาทีต่อ Test Case (baseline) และถูกจำกัดหน่วยความจำอยู่ที่ 512 MB

- สำหรับโปรแกรมที่เขียนด้วยภาษา C หรือ C++ จะถูกจำกัดเวลาเท่ากับค่า baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา Go หรือ Java จะถูกจำกัดเวลาอยู่ที่ 1.5 เท่าของ baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา JavaScript หรือ Python จะถูกจำกัดเวลาอยู่ที่ 2.5 เท่าของ baseline ข้างต้น

บทที่ 3
โจทย์อัลกอริทึม Final Round

3.1 Protein Synthesis

Problem Statement

ณ ศูนย์วิจัยทางวิทยาศาสตร์ชีวโมเลกุลแห่งหนึ่ง มีนักวิจัยที่กำลังทำวิจัยเรื่องสมบัติของ Mixture (สารผสม) ของสาย Polypeptide หลากหลายประเภท โดยปกติแล้วสาย Polypeptide จะประกอบไปด้วยกรดอะมิโนอย่างน้อย 1 ตัวถูกจับมาเรียงตัวกันเป็นเส้นตรงผ่านพันธะเอไมด์ (Amide Bond) แต่สำหรับโจทย์ในข้อนี้ เราจะสนใจกรดอะมิโนเพียงสองชนิด ได้แก่ Serine (`'S'`) และ Threonine (`'T'`)

ในการเตรียมสารนี้ นักวิจัยต้องการให้สารดังกล่าวเป็น Mixture ของสาย Polypeptide ทั้งหมด `n` ชนิด โดยสาย Polypeptide สองประเภทใด ๆ ไม่ควรเป็น Prefix ต่อกันและกัน (เพราะความคล้ายคลึงกันของสาย Polypeptide แบบดังกล่าวอาจจะทำให้การวัดอ่านค่าในการทดลองมีความสับสนได้) เช่น

- สาย Polypeptide `A = "STSST"` และ `B = "STS"` ไม่ควรนำมาผสมกันใน Mixture เพราะ `B` เป็น Prefix ของ `A`
- สาย Polypeptide `A = "TTS"` และ `B = "TSSTTS"` สามารถนำมาผสมอยู่ใน Mixture เดียวกันได้ เพราะไม่มีสาย Polypeptide ใดที่เป็น Prefix ของ Polypeptide อีกสายหนึ่ง

การสังเคราะห์สาย Polypeptide แต่ละประเภทจะมีต้นทุนที่ต่างกัน กล่าวคือในการสร้างสาย Polypeptide หนึ่งจะต้องใช้ต้นทุน `Cs` หน่วยต่อ Serine หนึ่งตัวที่ปรากฏในสาย Polypeptide ดังกล่าว และจะต้องใช้ต้นทุนอีก `Ct` หน่วยต่อ Threonine หนึ่งตัวที่ปรากฏในสาย Polypeptide ดังกล่าว เช่น การสร้างสาย Polypeptide `"STTSSTT"` จะมีต้นทุนเท่ากับ $3\texttt{Cs} + 4\texttt{Ct}$ หน่วย

อยากทราบว่าเราจะสามารถสังเคราะห์สาย Polypeptide ทั้งหมด `n` ชนิดจากกรดอะมิโนสองชนิดข้างต้น โดยที่ไม่มีสาย Polypeptide สองประเภทใด ๆ ที่เป็น Prefix ต่อกันและกัน เพื่อมาผสมกันเป็น Mixture ที่ต้องการ โดยใช้ต้นทุนเป็นมูลค่าต่ำที่สุดเท่าใด

Objectives

จงเขียนโปรแกรมเพื่อรับ Input Data ดังต่อไปนี้

- จำนวนสาย Polypeptide `n` ประเภทที่ต้องการสังเคราะห์ (โดยที่ $1 \le \texttt{n} \le 100{,}000{,}000$)
- ราคาของกรดอะมิโนต่อตัวสำหรับ Serine (`Cs`) และ Threonine (`Ct`)
  (โดยที่ $0 \le \texttt{Cs}, \texttt{Ct} \le 100{,}000$)

แล้วจึงคำนวณหาต้นทุนที่ต่ำที่สุดสำหรับสร้างสาย Polypeptide ทั้งหมด `n` ประเภทที่ไม่เป็น Prefix ต่อกันและกัน ตามราคาของกรดอะมิโนที่กำหนดเพื่อผสมเป็น Mixture

Interfaces and Data Format

โปรแกรมที่เขียนขึ้นจะต้องรับ Input Data ผ่าน Standard Input ซึ่งมีรูปแบบดังต่อไปนี้

- บรรทัดเดียวมีจำนวนเต็มสามจำนวนคือ `n` `Cs` และ `Ct` คั่นด้วยช่องว่าง

```
n Cs Ct
```

โปรแกรมที่เขียนขึ้นจะต้องคืน Output Data ผ่าน Standard Output ซึ่งเป็นมูลค่าที่ต่ำที่สุดที่สามารถใช้ผสม Mixture ตามที่กำหนดใน Objectives ข้างต้น

Example Input	Example Output
`4 1 2`	`12`

หมายเหตุ: ในตัวอย่างข้างต้น เราสามารถสร้างสาย Polypeptide `"SS"` (ต้นทุน 2 หน่วย), `"ST"` (ต้นทุน 3 หน่วย), `"TS"` (ต้นทุน 3 หน่วย), และ `"TT"` (ต้นทุน 4 หน่วย) ได้ ดังนั้นแล้วต้นทุนรวมของ Mixture นี้เท่ากับ $2 + 3 + 3 + 4 = 12$ หน่วย ซึ่งน้อยที่สุดที่เป็นไปได้ในกรณีนี้

Scoring

โปรแกรมของคุณจะถูกทดสอบกับ Test Cases ที่มีเงื่อนไขต่าง ๆ ดังนี้

SMALL (คะแนน 20%)

รับประกันว่าจำนวนของประเภทสาย Polypeptide ที่ต้องการจะสอดคล้องกับเงื่อนไข $1 \le n \le 100$ และราคาของกรดอะมิโนต่อตัวในสาย Polypeptide จะสอดคล้องกับเงื่อนไข $0 \le Cs, Ct \le 100$

MEDIUM (คะแนน 35%)

รับประกันว่าจำนวนของประเภทสาย Polypeptide ที่ต้องการจะสอดคล้องกับเงื่อนไข $1 \le n \le 100{,}000$

LARGE (คะแนน 45%)

ไม่มีเงื่อนไขเพิ่มเติม

Limitations

โปรแกรมจะถูกจำกัดเวลาอยู่ที่ 0.4 วินาทีต่อ Test Case (baseline) และถูกจำกัดหน่วยความจำอยู่ที่ 512 MB

- สำหรับโปรแกรมที่เขียนด้วยภาษา C หรือ C++ จะถูกจำกัดเวลาเท่ากับค่า baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา Go หรือ Java จะถูกจำกัดเวลาอยู่ที่ 1.5 เท่าของ baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา JavaScript หรือ Python จะถูกจำกัดเวลาอยู่ที่ 2.5 เท่าของ baseline ข้างต้น

3.2 Sensors and Radio Tower

Problem Statement

สำนักสำรวจวนอุทยาน (State Forest Survey Bereau) ต้องการทราบข้อมูลทางอุตุนิยมวิทยา (จำพวก เช่น อุณหภูมิ ความชื้น ความกดอากาศ ทิศทางลม ปริมาณฝุ่น PM 2.5 เป็นต้น) ทางสำนักสำรวจจึงมีโครงการที่จะติดตั้ง Sensor (มาตรวัดข้อมูล) ที่ตำแหน่งต่าง ๆ ภายในวนอุทยานแห่งหนึ่งซึ่งเป็นพื้นราบสองมิติ และเพื่อความสะดวกรวดเร็วในการเก็บเกี่ยวข้อมูลจาก Sensor ที่ติดตั้งนั้น สำนักสำรวจยังจะติดตั้ง Radio Tower (เสาสื่อสัญญาณ) 1 ต้น ณ ตำแหน่งใดตำแหน่งหนึ่งภายในวนอุทยานเดียวกันนี้ โดยอาจเป็นตำแหน่งเดียวกับ Sensor บางตัวก็ได้

Radio Tower ต้นดังกล่าวนี้จำเป็นต้องใช้พลังในการทำงานเพื่อรับและส่งสัญญาณกับ Sensor ที่ติดตั้งไว้ทุกตัวอย่างต่อเนื่องตลอดเวลา ในแต่ละวินาทีนั้น Radio Tower ซึ่งติดตั้งอยู่ที่ตำแหน่ง $t = (x^*, y^*)$ จะต้องใช้พลังงานติดต่อกับ Sensor หนึ่งตัวที่ติดตั้งอยู่ที่ตำแหน่ง $s_i = (x_i, y_i)$ ซึ่งสอดคล้องกับสมการต่อไปนี้

$$e(t, s_i) := \text{พลังงานที่ใช้ต่อวินาที}(t, s_i) = \max\left\{|x^* - x_i|, |y^* - y_i|\right\}$$

และพลังงานทั้งหมดที่ Radio Tower ดังกล่าวต้องการใช้ในหนึ่งวินาที จะมีค่าเท่ากับผลรวมของพลังงานที่ Radio Tower ใช้ติดต่อไปยัง Sensor $i \in S$ ทุกตัวที่ถูกติดตั้ง ซึ่งสามารถสรุปความได้ด้วยสมการดังต่อไปนี้

$$E(t) := \text{พลังงานรวมต่อวินาที(t)} = \sum_{i \in S} \text{พลังงานที่ใช้ต่อวินาที}(t, s_i) = \sum_{i \in S} e(t, s_i)$$

ในช่วงเริ่มต้นโครงการ ทางสำนักสำรวจยังไม่แน่ใจว่าควรติดตั้ง Sensor ณ ตำแหน่งใดบ้าง สำนักสำรวจจึงวางแผนโครงการตามลำดับดังนี้

1. สำนักสำรวจเลือกติดตั้ง Sensor ตัวแรก ($i = 1$) ณ ตำแหน่ง $s_1 = (x_1, y_1)$
2. จากนั้นสำนักสำรวจจึงนำ Radio Tower ไปวางในตำแหน่งที่ใช้พลังงานรวมต่อวินาทีต่ำที่สุด (ซึ่งในตอนนี้จะเป็นตำแหน่งเดียวกับ Sensor ตัวแรก ซึ่งจะทำให้ใช้พลังงาน 0 หน่วยต่อวินาที)
3. เมื่อสำนักสำรวจทราบข้อมูลเพิ่มเติมแล้ว สำนักสำรวจจะกำหนดตำแหน่งที่จะติดตั้ง Sensor ตัวที่ $i = 2, 3, \ldots$ ต่อไป และทุก ๆ ครั้งที่มีการติดตั้ง Sensor ตัวใหม่ตัวที่ ทางสำนักสำรวจอาจจำเป็นต้องย้ายตำแหน่งที่ตั้งของ Radio Tower ไปยังตำแหน่งใหม่เพื่อให้ได้พลังงานรวมต่อวินาทีค่าใหม่ที่ต่ำที่สุด

จงคำนวณหาตำแหน่งที่ควรติดตั้ง Radio Tower ที่จะทำให้ค่าพลังงานรวมต่อวินาทีที่ต่ำที่สุดที่เพียงพอที่จะให้ Radio Tower ทำงานได้ หลังจากที่ติดตั้ง Sensor แต่ละตัว (ตัวที่ $i = 1, 2, \ldots, n$) เป็นที่เรียบร้อยแล้ว หากมีตำแหน่งที่สามารถติดตั้ง Radio Tower ได้หลายตำแหน่ง ให้ตอบตำแหน่งใดก็ได้

Situation Example

จงพิจารณาเหตุการณ์ที่เกิดขึ้นต่อไปนี้

1. สำนักสำรวจเลือกติดตั้ง Sensor ตัวที่ $i = 1$ ณ ตำแหน่ง $s_1 = (1, 2)$

   ในกรณีนี้ตำแหน่งที่ดีที่สุดสำหรับ Radio Tower ตำแหน่งหนึ่งที่เป็นไปได้คือ ณ ตำแหน่ง $t = (1, 2)$ ซึ่งเป็นตำแหน่งเดียวกับ Sensor ตัวเดียวที่มีอยู่ พลังงานที่ใช้งานสำหรับแผนนี้คือ 0 หน่วยต่อวินาที

2. สำนักสำรวจเลือกติดตั้ง Sensor ตัวที่ $i = 2$ ณ ตำแหน่ง $s_2 = (1, 5)$

   ในกรณีนี้ตำแหน่งที่ดีที่สุดสำหรับ Radio Tower ตำแหน่งหนึ่งที่เป็นไปได้คือ ณ ตำแหน่ง $t = (1, 2)$ ซึ่งเป็นตำแหน่งเดิมของ Radio Tower ไม่เปลี่ยนแปลง พลังงานที่ใช้งานต่อวินาทีสำหรับแผนนี้คือ

   $$\begin{aligned} E(t) &= e(t, s_1) + e(t, s_2) \\ e(t, s_1) &= \max\{|1 - 1|, |2 - 2|\} = \max\{0, 0\} = 0 \\ e(t, s_2) &= \max\{|1 - 1|, |2 - 5|\} = \max\{0, 3\} = 3 \\ \therefore \quad E(t) &= 0 + 3 = 3 \end{aligned}$$

   ซึ่งมีค่าต่ำที่สุดเท่าที่เป็นไปได้

3. สำนักสำรวจเลือกติดตั้ง Sensor ตัวที่ $i = 3$ ณ ตำแหน่ง $s_3 = (5, 7)$

   ในกรณีนี้ตำแหน่งที่ดีที่สุดสำหรับ Radio Tower ตำแหน่งหนึ่งที่เป็นไปได้คือ ณ ตำแหน่ง $t = (2, 4)$ พลังงานที่ใช้งานต่อวินาทีสำหรับแผนนี้คือ

   $$\begin{aligned} E(t) &= e(t, s_1) + e(t, s_2) + e(t, s_3) \\ e(t, s_1) &= \max\{|2 - 1|, |4 - 2|\} = \max\{1, 2\} = 2 \\ e(t, s_2) &= \max\{|2 - 1|, |4 - 5|\} = \max\{1, 1\} = 1 \\ e(t, s_3) &= \max\{|2 - 5|, |4 - 7|\} = \max\{3, 3\} = 3 \\ \therefore \quad E(t) &= 2 + 1 + 3 = 6 \end{aligned}$$

   ซึ่งมีค่าต่ำที่สุดเท่าที่เป็นไปได้

4. สำนักสำรวจเลือกติดตั้ง Sensor ตัวที่ $i = 4$ ณ ตำแหน่ง $s_4 = (7, 1)$

   ในกรณีนี้ตำแหน่งที่ดีที่สุดสำหรับ Radio Tower ตำแหน่งหนึ่งที่เป็นไปได้คือ ณ ตำแหน่ง $t = (2, 4)$ เช่นเดิม พลังงานที่ใช้งานต่อวินาทีสำหรับแผนนี้คือ

   $$\begin{aligned} E(t) &= e(t, s_1) + e(t, s_2) + e(t, s_3) + e(t, s_4) \\ e(t, s_1) &= \max\{|2 - 1|, |4 - 2|\} = \max\{1, 2\} = 2 \\ e(t, s_2) &= \max\{|2 - 1|, |4 - 5|\} = \max\{1, 1\} = 1 \\ e(t, s_3) &= \max\{|2 - 5|, |4 - 7|\} = \max\{3, 3\} = 3 \\ e(t, s_3) &= \max\{|2 - 7|, |4 - 1|\} = \max\{5, 3\} = 5 \\ \therefore \quad E(t) &= 2 + 1 + 3 + 5 = 11 \end{aligned}$$

   ซึ่งมีค่าต่ำที่สุดเท่าที่เป็นไปได้

Objectives

จงเขียนโปรแกรมเพื่อรับ Input Data เป็นพิกัดตำแหน่งที่ติดตั้ง Sensor รวมทั้งสิ้น n ตัว หลังจากที่โปแกรมของผู้เข้าแข่งขันพิจารณาข้อมูลตำแหน่งของ Sensor แต่ละตัวตามลำดับแล้ว ให้ตอบพิกัดของ Radio Tower ที่ใช้พลังงานรวมต่อวินาทีน้อยที่สุดเป็น Output Data ก่อนที่จะพิจารณา Sensor ตัวถัดไป

จำนวน Sensor ทั้งหมดที่จะติดตั้งในวนอุทยานจะสอดคล้องกับเงื่อนไข $1 \le n \le 200{,}000$ นอกจากนั้นค่าพิกัดที่ปรากฏใน Input Data ทั้งหมดจะเป็นข้อมูลจุดพิกัด (x, y) บนระนาบ 2 มิติ ซึ่งสอดคล้องกับเงื่อนไข $-10^9 \le x, y \le 10^9$

Interfaces and Data Format

โปรแกรมที่เขียนขึ้นจะต้องรับ Input Data ผ่าน Standard Input ดังต่อไปนี้

- บรรทัดแรกมีจำนวนเต็มหนึ่งจำนวนคือ n
- บรรทัดที่ $i+1$ สำหรับ $i = 1, 2, \ldots, n$ จะมีจำนวนเต็มสองจำนวน
  ซึ่งก็คือ x_i และ y_i ซึ่งระบุพิกัดของ Sensor ตัวที่ i

```
1     n
2     x_1  y_1
3     x_2  y_2
...
n+1   x_n  y_n
```

โปรแกรมที่เขียนขึ้นจะต้องคืน Output Data ผ่าน Standard Output ซึ่งประกอบด้วยข้อมูลดังต่อไปนี้

- บรรทัดที่ i สำหรับ $i = 1, 2, \ldots, n$ จะมีจำนวนสองจำนวน ซึ่งก็คือ x_i^* และ y_i^* ซึ่งระบุพิกัดของ Radio Tower ที่ใช้พลังงานรวมต่อวินาทีในการสื่อสารกับ Sensor ตัวที่ 1 ถึง i น้อยที่สุด ค่าพิกัด x_i^* และ y_i^* จะต้องเป็นจำนวนเต็มหรือจำนวนจริงที่มีตัวเลขหลังจุดทศนิยมไม่เกิน 2 ตำแหน่ง

```
1     x_1*  y_1*
2     x_2*  y_2*
...
n     x_n*  y_n*
```

Example Input	Example Output
4 1 2 1 5 5 7 7 1	1 2 1 2 2 4 2 4

Scoring

โปรแกรมของคุณจะถูกทดสอบกับ Test Cases ที่มีเงื่อนไขต่าง ๆ ดังนี้

SMALL (คะแนน 20%)

รับประกันว่าจำนวน Sensor ทั้งหมดจะสอดคล้องกับเงื่อนไข $1 \le n \le 2{,}000$ และค่าพิกัด (x, y) ของ Sensor ที่จะติดตั้งทุกตัวจะสอดคล้องกับเงื่อนไข $-100 \le x, y \le 100$

MEDIUM (คะแนน 35%)

รับประกันว่าจำนวน Sensor ทั้งหมดจะสอดคล้องกับเงื่อนไข $1 \le n \le 2{,}000$

LARGE (คะแนน 45%)

ไม่มีเงื่อนไขเพิ่มเติม

Limitations

โปรแกรมจะถูกจำกัดเวลาอยู่ที่ 0.8 วินาทีต่อ Test Case (baseline) และถูกจำกัดหน่วยความจำอยู่ที่ 512 MB

- สำหรับโปรแกรมที่เขียนด้วยภาษา C หรือ C++ จะถูกจำกัดเวลาเท่ากับค่า baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา Go หรือ Java จะถูกจำกัดเวลาอยู่ที่ 1.5 เท่าของ baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา JavaScript หรือ Python จะถูกจำกัดเวลาอยู่ที่ 2.5 เท่าของ baseline ข้างต้น

3.3 TechJam–CRC

Problem Statement

ในการเตรียมจัดงานแข่งขัน TechJam 2018 นั้นจำเป็นต้องถ่ายเกรับส่งข้อความจำนวนมาก ซึ่งข้อความดังกล่าวสามารถมองเป็น Bit String (หรือสตริงที่ประกอบไปด้วยเลขโดด 0 หรือ 1) จำนวนมหาศาลได้ ทีมผู้จัดงานจึงได้ออกแบบระบบ Checksum แบบใหม่ขึ้นมาใช้งานเองเป็นกาลเฉพาะ มีชื่อเรียกว่า “TechJam–CRC”

การคำนวณหา TechJam–CRC ของข้อความหนึ่ง ๆ จะต้องเป็นข้อมูลนำเข้าในรูปของ Bit String เท่านั้น และจะต้องมี Key (กุญแจ) ซึ่งเป็น Bit String อีกตัวหนึ่งเอาไว้ประกอบการคำนวณ TechJam–CRC ของข้อความดังกล่าว โดยที่ความยาวของ Key จะต้องสั้นกว่าหรือเท่ากับความยาวของข้อความตั้งต้น

การคำนวณหา TechJam–CRC ของข้อความหนึ่ง ๆ ประกอบด้วยขั้นตอนดังต่อไปนี้

```
function TechJam_CRC(M, K):
    # M is a bit string to compute the CRC
    # K is the key bit string such that length(K) ≤ length(M)
    if length(M) == length(K) then:
        return M
    end
    c := Last digit of M
    Remove last digit from bit string M
    if c == 1 then:
        M := M xor K
    end
    return TechJam_CRC(M, K)
end
```

สังเกตว่า Recursive Call ของ `TechJam_CRC` แต่ละครั้ง ความยาวของ Bit String `M` จะลดลงทีละหนึ่งจนกว่าสตริงดังกล่าวจะยาวเท่ากับ Key `K` นอกจากนั้น ค่า CRC ผลลัพธ์จะเป็น Bit String ที่มีความยาวเท่ากับ Key `K` ดังกล่าวเสมอ

ข้อควรทราบ: **`xor`** หรือ Exclusive OR ระหว่าง Bit String สองสตริง ให้คำนวณแบบ Bitwise Exclusive OR ในกรณีที่สตริงทั้งสองอันยาวไม่เท่ากัน ให้ทำการ Pad สตริงที่สั้นกว่าด้วย Leading Zero จนกว่าสตริงทั้งสองจะยาวเท่ากัน เช่น การหา `101100` **`xor`** `1001` จะได้ผลลัพธ์เป็น `100101`

```
1 0 1 1 0 0
    1 0 0 1 xor
-----------
1 0 0 1 0 1
```

ตัวอย่างการหา TechJam–CRC

สมมติว่าเราต้องการคำนวณ `TechJam_CRC(M=0110010, K=110)` ฟังก์ชันจะมีกระบวนการทำงานดังนี้

รอบที่	สิ่งที่เกิดขึ้น
	ให้ `M = 0110010` และ `K = 110`
1	`M = 0110010`
	`c = 0` ดังนั้นแล้วค่าใหม่คือ `M = 011001`
2	`M = 011001`
	`c = 1` ดังนั้นแล้วค่าใหม่คือ `M = 01100 xor 110 = 01010`
3	`M = 01010`
	`c = 0` ดังนั้นแล้วค่าใหม่คือ `M = 0101`
4	`M = 0101`
	`c = 1` ดังนั้นแล้วค่าใหม่คือ `M = 010 xor 110 = 100`

ดังนั้นแล้วผลลัพธ์ของ `TechJam_CRC(M=0110010, K=110)` คือ `100`

ส่วนของข้อความที่หายไป

ในการใช้งานจริงนั้น ก่อนที่เราจะส่งข้อความ Bit String `M` ผ่านระบบเครือข่ายนั้น เราจะคำนวณค่า Checksum ของ `M` ก่อน ซึ่งก็คือ `H = TechJam_CRC(M, K)` แล้วจึงส่งทั้งสตริง `M` และ `H` ไปพร้อมกัน (**หมายเหตุ:** ค่า Key `K` เป็นค่า Bit String ที่ตกลงกันไว้ล่วงหน้าแล้ว)

ในบางครั้ง เมื่อเราส่งข้อความ `M` ผ่านระบบเครือข่ายที่มีคลื่นรบกวนแล้วนั้น ผู้รับสารไม่สามารถอ่านค่าบิตบางตัวของ `M` ได้เพราะมีคลื่นรบกวนมากเกินไป เราจะเรียกข้อความที่ผู้รับสารมองเห็นดังกล่าวว่า `M*` ซึ่งเกิดจากข้อความ `M` เดิมที่ค่าบิตบางตัวอ่านไม่ออก (ซึ่งอาจจะเขียนด้วยเครื่องหมายปรัศนี `'?'` แทนเลขโดด `'0'` และ `'1'`) แต่เมื่อเราทราบค่า Checksum `H` ของข้อความ `M` เดิมและค่า Key `K` แล้ว เราอาจจะ Recover บิตบางตัวที่เดิมทีอ่านไม่ออกในข้อความ `M*` ได้

Objectives

จงเขียนโปรแกรมเพื่อรับ Input Data ดังต่อไปนี้

- ค่าสตริง `M*` ที่มีคลื่นรบกวน ซึ่งมีความยาว `n` (โดยที่ `1 ≤ n ≤ 100,000`)
  รับประกันว่าในสตริงนี้จะประกอบไปด้วยคลื่นรบกวน `'?'` ไม่เกิน 1000 ตัวอักษร
- ค่า Key `K` ซึ่งมีความยาว `p` (โดยที่ `1 ≤ p ≤ 100` และ `p ≤ n`)
- และค่า Checksum `H` ความยาว `p` ของข้อความ `M` เดิม

แล้วจึงคำนวณหาข้อความดั้งเดิม `M` จาก Input Data ข้างต้น แล้วจึงคืนคำตอบดังกล่าวออกมาเป็น Output Data

หากมีหลายคำตอบที่เป็นไปได้ ให้แสดงคำตอบที่เมื่อตีความเป็นจำนวนเต็มแบบ Unsigned Integer แล้วจะมีค่าน้อยที่สุดที่เป็นไปได้ แต่ถ้าไม่มีคำตอบที่เป็นไปได้เลยจาก Input Data ที่โจทย์กำหนดให้ ให้แสดงคำว่า `"IMPOSSIBLE"`

Interfaces and Data Format

โปรแกรมที่เขียนขึ้นจะต้องรับ Input Data ผ่าน Standard Input ซึ่งมีรูปแบบดังต่อไปนี้

- บรรทัดแรกมีจำนวนเต็มสองจำนวน `n` และ `p` ซึ่งระบุถึงความยาวของข้อความ `M` และ Key `K` ตามลำดับ
- บรรทัดที่สองระบุสตริงความยาว `n` ซึ่งเป็นข้อความ `M*` ที่อาจมีคลื่นรบกวน
- บรรทัดที่สามระบุสตริงยาว `p` ซึ่งเป็นค่า Key `K`
- บรรทัดที่สี่ระบุสตริงยาว `p` ซึ่งเป็นค่า Checksum `H = TechJam_CRC(M, K)`

```
n p
M*
K
H
```

โปรแกรมที่เขียนขึ้นจะต้องคืน Output Data ผ่าน Standard Output เป็นข้อความ Bit String ดั้งเดิม `M` หรือตอบคำว่า `"IMPOSSIBLE"` ในกรณีที่ไม่มีคำตอบ ตามที่ระบุไว้ในหัวข้อ Objectives ข้างต้น

Example Input	Example Output
7 3 0110??? 110 100	0110010
6 4 1????? 1011 0010	100110
6 4 1?1??? 1011 0010	IMPOSSIBLE
2 2 11 11 11	11
2 2 10 11 11	IMPOSSIBLE

หมายเหตุ: ในตัวอย่างที่สอง มี 2 คำตอบคือ `"100110"` และ `"110001"` แต่เนื่องจาก `"100110"` มีค่าน้อยกว่าจึงตอบสตริงดังกล่าว

Scoring

โปรแกรมของคุณจะถูกทดสอบกับ Test Cases ที่มีเงื่อนไขต่าง ๆ ดังนี้

SMALL (คะแนน 20%)

รับประกันว่าความยาวของข้อความจะสอดคล้องกับเงื่อนไข $1 \le n \le 10$

MEDIUM (คะแนน 35%)

รับประกันว่าความยาวของข้อความจะสอดคล้องกับเงื่อนไข $1 \le n \le 100$ และในข้อความที่ผู้รับสารเห็นจะมีคลื่นรบกวน `'?'` ไม่เกิน 30 ตัว

LARGE (คะแนน 45%)

ไม่มีเงื่อนไขเพิ่มเติม

Limitations

โปรแกรมจะถูกจำกัดเวลาอยู่ที่ 0.6 วินาทีต่อ Test Case (baseline) และถูกจำกัดหน่วยความจำอยู่ที่ 512 MB

- สำหรับโปรแกรมที่เขียนด้วยภาษา C หรือ C++ จะถูกจำกัดเวลาเท่ากับค่า baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา Go หรือ Java จะถูกจำกัดเวลาอยู่ที่ 1.5 เท่าของ baseline ข้างต้น
- สำหรับโปรแกรมที่เขียนด้วยภาษา JavaScript หรือ Python จะถูกจำกัดเวลาอยู่ที่ 2.5 เท่าของ baseline ข้างต้น

บทที่ 4
โจทย์ปริศนา Final Round

4.1 Question 1

มีของเล่นทั้งสิ้น 2019 ชนิด ชนิดละหลายชิ้น ต้องการนำของเล่นดังกล่าวไปแจกให้เด็กเป็นจำนวนคนมากที่สุด โดยที่สอดคล้องกับเงื่อนไขว่า

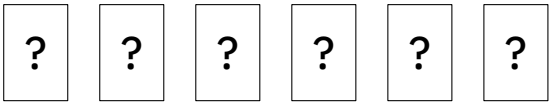
- เด็กแต่ละคนจะต้องได้ของขวัญอย่างน้อย 1 ชิ้น และชนิดละไม่เกิน 1 ชิ้น
- สำหรับเด็ก x และ y สองคนใด ๆ จะต้องพบว่าเงื่อนไขย่อยต่อไปนี้จะเป็นจริง 1 ข้อ
  - x มีของเล่นทุกชนิดที่ y มี แต่มีของเล่นของ x บางชนิดที่ y ไม่มี
  - y มีของเล่นทุกชนิดที่ x มี แต่มีของเล่นของ y บางชนิดที่ x ไม่มี
  - x และ y ไม่ครอบครองของเล่นชนิดใดร่วมกันเลย

อยากทราบว่าเราจะต้องเตรียมของเล่นอย่างน้อยที่สุดกี่ชิ้น และอย่างมากที่สุดกี่ชิ้น?

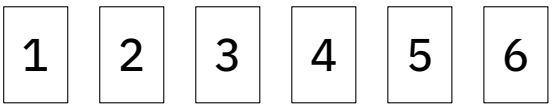
4.2 Question 2

มีการ์ดไพ่ทั้งสิ้น 6 ใบ แต่ละใบเขียนหมายเลข 1 ถึง 6 หมายเลขละหนึ่งใบพอดี ไพ่ทั้ง 6 ใบนี้ถูกนำมาวางเรียงกันเป็นแถวอยู่แถวหนึ่ง เราจะกล่าวว่าไพ่ใบหนึ่ง (ซึ่งเขียนหมายเลข i) อยู่ในตำแหน่งที่ถูกต้องก็ต่อเมื่อไพ่ใบดังกล่าวอยู่ในตำแหน่งที่ i โดยเริ่มนับจากใบซ้ายสุดเป็นไพ่ใบแรก

- ในตอนแรกเริ่ม ไม่มีไพ่ใบใดที่อยู่ในตำแหน่งที่ถูกต้องเลย



- จากนั้นเราเลือกสลับไพ่คู่หนึ่งที่อยู่ติดกัน พบว่ามีไพ่ 1 ใบอยู่ในตำแหน่งที่ถูกต้อง
- จากนั้นสลับไพ่อีกคู่หนึ่งที่อยู่ติดกัน (ซึ่งเป็นคนละคู่กับรอบที่แล้ว) พบว่ามีไพ่ 2 ใบอยู่ในตำแหน่งที่ถูกต้อง
- จากนั้นสลับไพ่อีกคู่หนึ่งที่อยู่ติดกัน (ซึ่งเป็นคนละคู่กับรอบที่แล้ว) พบว่ามีไพ่ 1 ใบอยู่ในตำแหน่งที่ถูกต้อง
- จากนั้นสลับไพ่อีกคู่หนึ่งที่อยู่ติดกัน (ซึ่งเป็นคนละคู่กับรอบที่แล้ว) พบว่ามีไพ่ 1 ใบอยู่ในตำแหน่งที่ถูกต้อง
- จากนั้นสลับไพ่อีกคู่หนึ่งที่อยู่ติดกัน (ซึ่งเป็นคนละคู่กับรอบที่แล้ว) พบว่ามีไพ่ 2 ใบอยู่ในตำแหน่งที่ถูกต้อง
- จากนั้นสลับไพ่อีกคู่หนึ่งที่อยู่ติดกัน (ซึ่งเป็นคนละคู่กับรอบที่แล้ว) พบว่ามีไพ่ 3 ใบอยู่ในตำแหน่งที่ถูกต้อง
- จากนั้นสลับไพ่อีกคู่หนึ่งที่อยู่ติดกัน (ซึ่งเป็นคนละคู่กับรอบที่แล้ว) พบว่ามีไพ่ 4 ใบอยู่ในตำแหน่งที่ถูกต้อง
- จากนั้นสลับไพ่อีกคู่หนึ่งที่อยู่ติดกัน (ซึ่งเป็นคนละคู่กับรอบที่แล้ว) พบว่าไพ่ทุกใบอยู่ในตำแหน่งที่ถูกต้อง



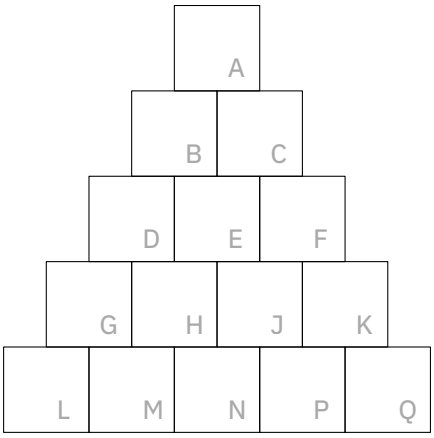
อยากทราบว่าไพ่ทั้ง 6 ใบในตอนแรกเริ่มคือหมายเลขใดบ้างตามลำดับ?

4.3 Question 3

จงเติมจำนวนเต็ม 1 ถึง 15 ลงในพิระมิดต่อไปนี้ ช่องละ 1 จำนวนโดยไม่ใช้จำนวนซ้ำกัน โดยที่สอดคล้องกับเงื่อนไขดังต่อไปนี้

- สำหรับแต่ละพิระมิดย่อย 

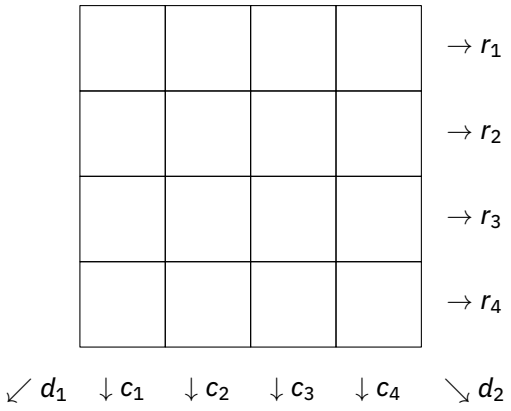
ที่ปรากฏในพิระมิดใหญ่ด้านล่าง จะพบว่าจำนวนในช่อง ก จะเป็นผลรวมหรือผลต่างของจำนวนในช่อง ข และ ค
- จำนวนในช่องต่าง ๆ จะสอดคล้องกับอสมการ A < E < N, L < H < F, และ Q < J < D



4.4 Question 4

จงเติมจำนวนเต็ม 1 ถึง 16 ลงในตารางขนาด 4×4 ดังต่อไปนี้ ช่องละ 1 จำนวนโดยไม่ใช้จำนวนซ้ำกัน โดยกำหนดนิยามผลรวมในแนวเส้นตรงดังนี้

- กำหนดให้ c_1, c_2, c_3, c_4 คือผลรวมในแนวตั้งของคอลัมน์ที่ 1, 2, 3, 4 ตามลำดับ
- กำหนดให้ r_1, r_2, r_3, r_4 คือผลรวมในแนวนอนของแถวที่ 1, 2, 3, 4 ตามลำดับ
- กำหนดให้ d_1 คือผลรวมแนวทแยงจากบนขวาไปล่างซ้าย
- กำหนดให้ d_2 คือผลรวมแนวทแยงจากบนซ้ายไปล่างขวา



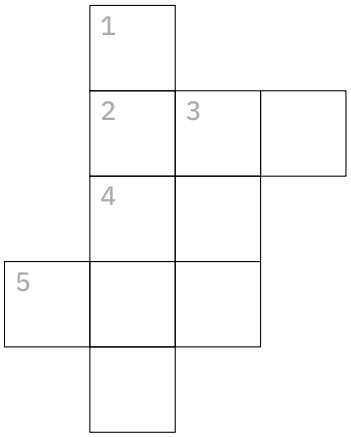
และการเติมจำนวนต้องสอดคล้องกับเงื่อนไขดังต่อไปนี้

- จำนวนเต็มสองจำนวนที่อยู่ติดกัน จะต้องไม่อยู่ในช่องติดกันในตาราง ในแนวตั้งและแนวนอน
- ผลรวมเป็นไปตามอสมการ $c_1 < c_2 < c_3 < c_4$ และ $r_1 < r_2 < r_3 < r_4$
- และเซต $\{c_1, c_2, c_3, c_4, r_1, r_2, r_3, r_4, d_1, d_2\} = \{23, 27, 30, 31, 36, 37, 39, 40, 43, 48\}$

4.5 Question 5

จงเติมเลขโดด 0 ถึง 9 ลงในตาราง Crossword ดังที่แสดงอยู่นี้ ช่องละ 1 ตัวโดยไม่ใช้เลขโดดซ้ำกัน ทำให้เกิดจำนวน 5 จำนวนเมื่ออ่านตามแนว 1–Down, 2–Across, 3–Down, 4–Across, และ 5–Across ที่สอดคล้องกับเงื่อนไขดังต่อไปนี้

- จำนวนทั้ง 5 จำนวน ไม่ขึ้นต้นด้วยเลขโดด 0
- จำนวนทั้ง 5 จำนวน เป็นจำนวนเฉพาะ
- ผลรวมของจำนวนทั้ง 5 จำนวนข้างต้น ก็เป็นจำนวนเฉพาะเช่นกัน



บทที่ 5
Containerized Application [Survey Robots]

5.1 Task Description

สำหรับโจทย์ในข้อนี้ ผู้เข้าแข่งขันจะต้องเขียน Web Service Application ขึ้นมา 1 ตัวเพื่ออำนวยความสะดวกแก่ Robot หลายตัวที่ถูกปล่อยให้ไปสำรวจพื้นที่สองมิติในบริเวณที่เพิ่งมีสะเก็ดดาวตกพุ่งลงสู่ผิวโลก กล่าวคือ Robot แต่ละตัว (ซึ่งอยู่เหนือการควบคุมของ Web Service กลาง) จะคอยส่งข้อมูลเกี่ยวกับตนเองหรือสิ่งที่ตนเองสำรวจพบมาอัปเดตแก่ Web Service นี้ หรืออาจจะส่ง Query บางอย่างมาให้ Web Service กลางช่วยคำนวณให้ โดยการสื่อสารจะเป็นไปตาม API Specification ที่ถูกกำหนดให้

Application ของผู้เข้าแข่งขันจะต้องถูกบรรจุเป็น Docker Image เพื่อความสะดวกในการทดสอบความถูกต้องและนำไป Deploy ได้อย่างสะดวกรวดเร็วต่อไปนี้

ผู้เข้าแข่งขันสามารถอ่าน Requirement ของ Web Service Application ได้จากเอกสารฉบับนี้ และสามารถอ่าน API Specification เพิ่มเติมได้จากไฟล์ `robots_spec.html` ที่แนบมากับเอกสารนี้

Real World Simulation

เพื่อจำลองสถานการณ์จริงในการทำงานในโลกจริง ในระหว่างการแข่งขันนี้อาจมีการขอแก้ไข Requirement ของโจทย์เพิ่มเติม ซึ่งส่วนมากจะมาในรูปแบบของ Feature Request เพิ่มเติมจากของเก่า ผู้เข้าแข่งขันที่สามารถ Implement Feature เหล่านี้ได้และทำงานได้อย่างถูกต้อง ก็จะได้คะแนนเพิ่มเติมในส่วนนั้นไป

การขอแก้ไข Requirement อาจเกิดขึ้นได้มากกว่า 1 ครั้งตลอดการแข่งขัน และในแต่ละครั้ง เอกสารฉบับนี้ก็จะถูกเพิ่มเติมแก้ไขพร้อม ๆ กับไฟล์ API Specification ผู้เข้าแข่งขันสามารถเลือกได้จะว่า Implement หรือไม่ Implement Feature ใหม่ ๆ ที่เกิดจากการขอแก้ไข Requirement ในครั้งต่าง ๆ ได้

ผู้เข้าแข่งขันที่เขียนโปรแกรมให้ทำงานได้อย่างถูกต้อง และมีทักษะในการพัฒนาซอฟต์แวร์ที่ดีเท่านั้น (ซึ่งรวมไปถึงการเขียนโค้ดที่ Maintainable และ Extensible ได้) จะสามารถอยู่รอดในการจำลองสถานการณ์จริงได้

5.2 Contest Rules and Restrictions

Source Code

- Source Code และ Scripts ทั้งหมดที่ใช้งานจะต้องเป็นผลงานที่ผู้เข้าแข่งขันเขียนขึ้นเอง หรือเป็นผลงานที่เป็น Free and Open-Sourced Software และไม่มีข้อต้องห้ามหรืออุปสรรคที่ทำให้บุคคลสาธารณะไม่สามารถเข้าถึงเทคโนโลยีดังกล่าว เท่านั้น
- ในการพัฒนา Application นี้ ผู้เข้าแข่งขันควรมีทักษะในการใช้ Git เพื่อทำ Source Code Versioning

  ก่อนหมดเวลาแข่งขัน ผู้เข้าแข่งขันจะต้องนำ Source Code ทั้งหมดที่จำเป็นในการสร้าง Docker Image ขึ้นสู่ GitHub, BitBucket, หรือ GitLab Account ของตนเอง ผู้เข้าแข่งขันสามารถเลือก Commit ที่ถูกนำขึ้น Platform ข้างต้นก่อนหมดเวลาการแข่งขันลง Commit ใดก็ได้ ให้คณะกรรมการพิจารณาตรวจให้คะแนน

- **คำแนะนำ:** ผู้เข้าแข่งขันอาจจะสร้าง Private Repository ในตอนต้นและใช้งานในลักษณะนี้ระหว่างการแข่งขัน และก่อนหมดเวลาจึงแก้ไข Permission ให้เป็น Public Repository
- **คะแนนโบนัส:** หากผู้เข้าแข่งขันเลือกปล่อยผลงาน Application นี้ด้วย Open Source Software License ที่เป็น GPL-Compatible License จะได้คะแนนเพิ่ม 10 ปิ๊บแต้ม

*ดูเพิ่มเติมที่ `https://www.gnu.org/licenses/license-list.en.html#GPLCompatibleLicenses`

Building and Running Docker

- Source Code ทั้งหมดที่จำเป็นในการสร้าง Docker Image จะต้องถูกทำ Commit เข้า Git Repository ส่วนไฟล์ `Dockerfile` จะต้องอยู่ใน Root Directory ของ Git Repository เท่านั้น
- Source Code ทั้งหมดจะต้องสามารถนำมา Build เป็น Docker Image ได้โดยไม่จำเป็นต้องกำหนด Build Options ใด ๆ เพิ่มเติม
- Base Docker Image ที่อนุญาตให้ใช้ในการแข่งขัน มีดังต่อไปนี้
  - `gcc:8.3`, `gcc:9.2`
  - `golang:1.12-buster`, `golang:1.13-buster`
  - `node:8.16-buster`, `node:10.17-buster`, `node:12.13-buster`
  - `openjdk:8-stretch`, `openjdk:11-stretch`, `openjdk:14-buster`
  - `python:3.7-buster`, `python:3.8-buster`
  - `kernelci/build-clang-9:latest`
  - `ubuntu:18.04`
- ในระหว่างการ Build Docker Image อนุญาตให้ดาวน์โหลดและติดตั้ง External Package ต่าง ๆ จาก Repository มาตรฐานของภาษานั้น ๆ โดยใช้ Packaging Manager ได้ แต่ไม่อนุญาตให้ดาวน์โหลดหรือติดตั้งสิ่งอื่นใดนอกเหนือจากนั้น
  - สำหรับภาษา Go อนุญาตให้ติดตั้ง Package ที่มีข้อมูลปรากฏในเว็บไซต์ `https://pkg.go.dev/` ได้
  - สำหรับภาษา JavaScript อนุญาตให้ติดตั้ง Package จาก Public NPM Registry ได้ (ตามที่ปรากฏใน `https://npmjs.com`)
  - สำหรับภาษา Python อนุญาตให้ติดตั้ง Package จาก The Python Package Index ได้ (ตามที่ปรากฏใน `https://pypi.org`)
  - หากต้องการใช้ Package สำหรับภาษาอื่น ๆ กรุณาสอบถามคณะกรรมการเป็นรายกรณี

  Package ที่ถูกติดตั้งข้างต้นจะต้องเป็น Free and Open-Sourced Software และไม่มีข้อต้องห้ามหรืออุปสรรคที่ทำให้บุคคลสาธารณะไม่สามารถเข้าถึงเทคโนโลยีดังกล่าว เท่านั้น
- เมื่อ Docker Image ถูกนำมาสร้าง Container จะมีการกำหนด Environment Variables ตามที่ระบุไว้ใน Feature ต่าง ๆ ที่อยู่ในเอกสารฉบับนี้ นอกเหนือจากนี้จะไม่มีการกำหนด Commands, Entrypoints, Network Settings, Restart Policies, Mounted Volumes และ Working Directory ใด ๆ เพิ่มเติม

- **คำแนะนำ:** ผู้แข่งขันควรกำหนดค่า Default ของ `CMD`, `ENTRYPOINT`, `EXPOSE`, `USER`, `WORKDIR` ของ `Dockerfile` ด้วยตนเอง
- **หมายเหตุ:** เนื่องจาก Docker Container จะถูกสร้างจาก Docker Image โดยไม่มีการกำหนด Command ดังนั้นแล้ว Docker Container จะรัน Command ตามค่า Default ที่ถูกกำหนดโดย `CMD` ดังที่กล่าวไว้ข้างต้น

Web Service Application

- Web Service Application ที่เขียนขึ้นจะต้องเปิดฟัง Port หมายเลข `8000` (TCP) เพื่อรอรับและตอบสนองต่อ Request ที่ถูกส่งมาจากฝั่ง Client (ผู้แข่งขันสามารถกำหนดให้ Port `8000` เป็นค่าหนึ่งของ `EXPOSE` ใน `Dockerfile` ได้)
- Web Service Application ที่เขียนขึ้นควรจะสามารถ Handle ในกรณีที่ Request มีรูปแบบของข้อมูลที่ไม่ถูกต้องได้ โดยสามารถรีเทิร์น Response ด้วยสเตตัส 400 Bad Request
- เพื่ออำนวยความสะดวกแก่ผู้เข้าแข่งขัน เราจะรับประกันดังต่อไปนี้
  - ทีมงานจะไม่ Monitor ผลลัพธ์ที่ถูกปล่อยออกมาผ่าน Standard Output หรือ Standard Error ของ Docker Container แต่อย่างใด ซึ่งผู้เข้าแข่งขันสามารถใช้ช่องทางดังกล่าวสำหรับ Debugging ได้
  - ทีมงานรับประกันว่าในระหว่างการแข่งขันจะไม่มีการประกาศ Endpoint ใดใน API Specification ที่มี Path ขึ้นต้นด้วย **`/private`** ซึ่งผู้เข้าแข่งขันสามารถสร้าง Endpoint ภายใต้ Path นี้เพิ่มเติมเพื่อใช้สำหรับ Debugging ได้เช่นกัน (ยกตัวอย่างเช่น ผู้แข่งขันอาจกำหนด Endpoint **`GET /private/ping`** สำหรับทดสอบ Connection ก็ได้)
  - ทีมงานรับประกันว่าในระหว่างการแข่งขันจะไม่มีการประกาศ Environment Variable ใดที่ขึ้นด้วยด้วยคำว่า **`PRIVATE_`** ซึ่งผู้เข้าแข่งขันสามารถใช้ประกาศ Environment Variable เพิ่มเติมเพื่อใช้สำหรับทำ Testing ได้ แต่พึงระวังว่าในขั้นตอนการตรวจสอบและให้คะแนนโปรแกรม จะไม่มีการป้อนค่าใด ๆ ให้แก่ Environment Variable ในกลุ่มนี้เช่นกัน

5.3 Baseline Features

Web Service Application ที่เขียนขึ้นจะต้องอำนวยความสะดวกให้แก่ Robot ที่ถูกปล่อยให้ไปสำรวจพื้นที่สองมิติ อันประกอบไปด้วย Feature พื้นฐานดังนี้

- Robot แต่ละตัวสามารถ Query เพื่อสอบถามระยะห่างระหว่างจุด 2 จุดกำหนดให้ได้
- ข้อมูลตำแหน่งของจุดพิกัด $(x, y) \in \mathbb{Z}^2$ ที่จะปรากฏใน Input Data หรือ Output Data จะสามารถเขียนในรูปของ Object ที่มี 2 Subfield ซึ่งได้แก่ `"x"` และ `"y"`

API Endpoints

`POST /distance`

Query เพื่อคำนวณระยะห่างระหว่างจุด 2 จุดบนระนาบสองมิติ

โดย Request Object จะประกอบไปด้วย JSON Field `"first_pos"` และ `"second_pos"` ซึ่งระบุถึงตำแหน่งของจุดทั้งสองจุดของ Input Data จุดแต่ละจุดของ Input Data จะอยู่ในรูปของพิกัดจำนวนเต็ม $(x, y) \in \mathbb{Z}^2$

ส่วน Response Object จะประกอบไปด้วย JSON Field เดียวคือ `"distance"` ซึ่งเป็น Euclidean Distance ระหว่างจุดสองจุดดังกล่าว โดยคำตอบจะต้องคาดเคลื่อนไม่เกิน 0.001 จากคำตอบจริง

รายละเอียดเพิ่มเติมสามารถหาอ่านได้จากไฟล์ API Specification ที่แนบมากับเอกสารนี้

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
   "first_pos": {"x": 3, "y": -2},
   "second_pos": {"x": -1, "y": 1}
}
```

Response #1:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 5
}
```

- **Request #2:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
   "first_pos": {"x": 0, "y": 0},
   "second_pos": {"x": 1, "y": 2}
}
```

Response #2:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 2.236
}
```

5.4 Feature Extension: A

เราจะเพิ่ม Feature ให้แก่ Web Service Application ดังนี้

- Robot แต่ละตัวสามารถส่งข้อมูลตำแหน่งปัจจุบันของตนเองมาอัปเดตให้กับ Web Service กลางได้ (Robot แต่ละตัวจะระบุตัวเองด้วย ID ซึ่งเป็นจำนวนเต็มในช่วง 1 ถึง 999,999)
- ใครก็ได้สามารถ Query สอบถามตำแหน่งที่ทราบล่าสุดของ Robot ตัวหนึ่งที่กำหนดให้ได้
- เดิมทีนั้นเราจะอ้างอิงถึงจุดบนระนาบสองมิติด้วยระบบพิกัด (x, y) แต่สำหรับ Feature นี้เราจะซัพพอร์ตวิธีการอ้างอิงถึงจุดบนระนาบสองมิติด้วย Robot ID กล่าวคือเราจะใช้ ID ของ Robot เพื่อระบุถึงตำแหน่งพิกัดล่าสุดที่ Robot ดังกล่าวรายงานอัปเดตเข้ามาในระบบ Service กลาง (ผ่าน Feature ใหม่ที่ระบุในข้อแรก)

API Endpoints

สำหรับ Feature Extension นี้จะต้อง Implement HTTP Endpoint เพิ่มเติมอีก 2 จุดคือ

`PUT /robot/{robot_id}/position`
: Update ข้อมูลตำแหน่งปัจจุบันของ Robot ที่กำหนดให้ โดย Path Parameter `robot_id` จะระบุ Robot ID ของ Robot ดังกล่าว และข้อมูลจาก JSON Field `"position"` จะระบุตำแหน่งปัจจุบันของ Robot ดังกล่าว (โดยให้นำข้อมูลนี้ไปแทนที่ข้อมูลเก่าของ Robot ข้างต้นหากมีข้อมูลเก่าอยู่แล้ว)

`GET /robot/{robot_id}/position`
: Query เพื่อเรียกดูข้อมูลตำแหน่งที่อัปเดตล่าสุดของ Robot ที่กำหนดให้ โดย Path Parameter `robot_id` จะระบุ Robot ID ของ Robot

รายละเอียดเพิ่มเติมสามารถหาอ่านได้จากไฟล์ API Specification ที่แนบมากับเอกสารนี้

Modified Changes

- ดังที่ได้เกริ่นไว้ข้างต้นแล้ว ผู้เข้าแข่งขันจะต้องแก้ไข Endpoint **`POST /distance`** เพื่อให้ JSON Field `"first_pos"` และ `"second_pos"` สามารถรองรับการระบุตำแหน่งบนพื้นที่สองมิติด้วย Robot ID ได้ โดยข้อมูล Robot ID จะมาในรูปแบบของสตริงที่สอดคล้องกับ Regular Expression ดังต่อไปนี้

  ```
  "^robot#([1-9][0-9]*)$"
  ```

  และตำแหน่งพิกัดของจุดดังกล่าวจะเป็นค่าตำแหน่งของ Robot ข้างต้นนี้ที่ถูกอัปเดตล่าสุดผ่าน Endpoint **`PUT /robot/{robot_id}/position`**

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 3, "y": 4}
}
```

Response #1:

```
HTTP/1.1 204 No Content
```

- **Request #2:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
   "first_pos": "robot#1",
   "second_pos": {"x": 0, "y": 8}
}
```

Response #2:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 5
}
```

- **Request #3:**

```
GET /robot/1/position HTTP/1.1
```

Response #3:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "position": {"x": 3, "y": 4}
}
```

- **Request #4:**

```
GET /robot/2/position HTTP/1.1
```

Response #4:

```
HTTP/1.1 404 Not Found
```

- **Request #5:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 0, "y": 0}
}
```

Response #5:

```
HTTP/1.1 204 No Content
```

- **Request #6:**

```
PUT /robot/2/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 1, "y": 1}
}
```

Response #6:

```
HTTP/1.1 204 No Content
```

- **Request #7:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
   "first_pos": "robot#2",
   "second_pos": "robot#1"
}
```

Response #7:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 1.414
}
```

5.5 Feature Extension: B

เราจะเพิ่ม Feature ให้แก่ Web Service Application ดังนี้

- สามารถกำหนดได้ว่าจะใช้ Distance Metric ใดในการวัดระยะห่างระหว่างจุดสองจุดของ Endpoint **POST `/distance`** ได้

Modified Changes

- ผู้เข้าแข่งขันจะต้องแก้ไข Endpoint **POST `/distance`** เพื่อให้รองรับ JSON Field ใหม่ชื่อว่า `"metric"` ซึ่งจะเป็นสตริงได้เพียง 2 แบบคือ `"euclidean"` (default) หรือ `"manhattan"`

  ในกรณีที่เป็นค่าสตริง `"manhattan"` ค่าระยะห่างระหว่างจุดสองจุดใน Input Data จะต้องถูกคำนวณเป็น Manhattan Distance แทนที่จะเป็น Euclidean Distance

รายละเอียดเพิ่มเติมสามารถหาอ่านได้จากไฟล์ API Specification ที่แนบมากับเอกสารนี้

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
   "first_pos": {"x": 3, "y": -2},
   "second_pos": {"x": -1, "y": 1}
}
```

Response #1:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 5.000
}
```

- **Request #2:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
   "first_pos": {"x": 3, "y": -2},
   "second_pos": {"x": -1, "y": 1},
   "metric": "manhattan"
}
```

Response #2:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 7.000
}
```

- **Request #3:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
   "first_pos": {"x": 3, "y": -2},
   "second_pos": {"x": -1, "y": 1},
   "metric": "euclidean"
}
```

Response #3:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 5.000
}
```

5.6 Feature Extension: C

> **หมายเหตุ:** Web Service Application ของผู้เข้าแข่งขันจำเป็นต้อง Implement Feature Extension A ก่อนจึงจะ Implement Feature ใหม่ต่อไปนี้ได้

เราจะเพิ่ม Feature ให้แก่ Web Service Application ดังนี้

- Operator ที่เป็นมนุษย์สามารถ Query สอบถามได้ว่าจากจุดที่กำหนดให้ Robot ตัวใดอยู่ใกล้เคียงจุดดังกล่าวมากที่สุด (ภายใต้ Euclidean Distance Metric) โดยอ้างอิงจากข้อมูลตำแหน่งล่าสุดที่ Robot แต่ละตัวอัปเดตเข้ามากับ Service กลาง หากมี Robot หลายตัวที่มีระยะห่างจากจุดหนึ่งเท่ากัน ให้พิจารณา Robot ที่มีค่า ID น้อยกว่าก่อนเป็นอันดับแรก

API Endpoints

สำหรับ Feature Extension นี้จะต้อง Implement HTTP Endpoint เพิ่มเติมอีก 1 จุดคือ

`POST /nearest`

Query เพื่อเรียกดู Robot ตัวที่อยู่ใกล้กับจุดที่กำหนดให้มากที่สุด

โดย Request Object จะมี JSON Field `"ref_position"` ซึ่งเป็นข้อมูลของจุดอ้างอิงที่กำหนดให้ ในรูปของพิกัด $(x, y) \in \mathbb{Z}^2$ ที่เป็น Object ที่มี 2 Subfield ซึ่งได้แก่ `"x"` และ `"y"`

ส่วน Response Object จะประกอบไปด้วย JSON Field `"robot_ids"` ซึ่งเป็นอาร์เรย์ของ Robot ID ที่ค้นพบจำนวน 0 หรือ 1 ตัว (อาร์เรย์อาจจะว่างเปล่าได้ หากไม่เคยมี Robot ตัวใดที่อัปเดตข้อมูลตำแหน่งของตนเองเลย)

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
POST /nearest HTTP/1.1
Content-Type: application/json

{
  "ref_position": {"x": -1, "y": 1}
}
```

Response #1:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
  "robot_ids": []
}
```

- **Request #2:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 1, "y": 1}
}
```

Response #2:

```
HTTP/1.1 204 No Content
```

- **Request #3:**

```
PUT /robot/2/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 0, "y": 0}
}
```

Response #3:

```
HTTP/1.1 204 No Content
```

- **Request #4:**

```
POST /nearest HTTP/1.1
Content-Type: application/json

{
   "ref_position": {"x": -1, "y": 1}
}
```

Response #4:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "robot_ids": [2]
}
```

- **Request #5:**

```
POST /nearest HTTP/1.1
Content-Type: application/json

{
   "ref_position": {"x": 1, "y": 0}
}
```

Response #5:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "robot_ids": [1]
}
```

5.7 Feature Extension: D

> **หมายเหตุ:** Web Service Application ของผู้เข้าแข่งขันจำเป็นต้อง Implement Feature Extension A ก่อนจึงจะ Implement Feature ใหม่ต่อไปนี้ได้

เราจะเพิ่ม Feature ให้แก่ Web Service Application ดังนี้

- เมื่อ Robot ใช้คลื่นเรดาร์ตรวจพบวัตถุต่างดาวพบ (เช่น เศษของสะเก็ดดาวตก) แล้ว Robot ดังกล่าวจะสามารถรายงานข้อมูลดังกล่าวมายัง Web Service กลางได้ ข้อมูลดังกล่าวจะประกอบไปด้วย Object DNA (ซึ่งเป็นสตริงที่ประกอบไปด้วย `'a'`–`'z'` ความยาว 16 อักษร และสามารถ Uniquely Identify วัตถุที่เป็น Alien Object ได้) และข้อมูลระยะห่างระหว่าง Robot ที่ทำการสำรวจและวัตถุดังกล่าว (ภายใต้ Euclidean Distance Metric)
- Operator ที่เป็นมนุษย์สามารถ Query เพื่อสอบถามว่าวัตถุต่างดาวหนึ่ง ๆ น่าจะอยู่ในตำแหน่งใดพื้นที่ โดยอาศัยข้อมูลเรดาร์ที่ Robot รายงานเข้ามา โดยให้ตอบค่าพิกัดเป็นเศษเป็นจำนวนเต็ม $(x, y) \in \mathbb{Z}^2$

API Endpoints

สำหรับ Feature Extension นี้จะต้อง Implement HTTP Endpoint เพิ่มเติมอีก 2 จุดคือ

`POST /alien/{object_dna}/report`

Update ข้อมูลการใช้คลื่นเรดาร์กับวัตถุต่างดาว โดย Path Parameter `object_dna` จะระบุ Object DNA ของวัตถุต่างดาวดังกล่าว และข้อมูลจาก JSON Field `"robot_id"` จะระบุ ID ของ Robot ที่กำลังรายงานผล และ JSON Field `"distance"` จะระบุระยะห่างระหว่าง Robot และวัตถุต่างดาวดังกล่าว

`GET /alien/{object_dna}/position`

Query เพื่อให้ Web Service กลางช่วยคำนวณหาตำแหน่งที่ถูกต้องแน่นอนของวัตถุต่างดาวที่กำหนดให้ โดย Path Parameter `object_dna` จะระบุ Object DNA ของวัตถุต่างดาวข้างต้น

หากสามารถคำนวณตำแหน่งของวัตถุต่างดาวได้อย่างเจาะจง จะต้องคืน Status 200 OK พร้อมกับ Response Object ที่มี JSON Field `"position"` ที่ระบุตำแหน่งของวัตถุต่างดาวที่คำนวณได้ หากข้อมูลที่มีอยู่ไม่เพียงพอที่จะคำนวณตำแหน่งที่แน่นอนได้ในเชิงเรขาคณิต จะต้องคืน Status 424 Failed Dependency แทน

รายละเอียดเพิ่มเติมสามารถหาอ่านได้จากไฟล์ API Specification ที่แนบมากับเอกสารนี้

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 0, "y": 0}
}
```

Response #1:

```
HTTP/1.1 204 No Content
```

- **Request #2:**

```
POST /alien/abcdefghijklmnop/report HTTP/1.1
Content-Type: application/json

{
   "robot_id": 1,
   "distance": 3.162
}
```

Response #2:

```
HTTP/1.1 200 OK
```

- **Request #3:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 5, "y": -2}
}
```

Response #3:

```
HTTP/1.1 204 No Content
```

- **Request #4:**

```
POST /alien/abcdefghijklmnop/report HTTP/1.1
Content-Type: application/json

{
   "robot_id": 1,
   "distance": 2.236
}
```

Response #4:

```
HTTP/1.1 200 OK
```

- **Request #5:**

```
GET /alien/abcdefghijklmnop/position HTTP/1.1
```

Response #5:

```
HTTP/1.1 424 Failed Dependency
```

- **Request #6:**

```
PUT /robot/2/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 3, "y": -3}
}
```

Response #6:

```
HTTP/1.1 204 No Content
```

- **Request #7:**

```
POST /alien/abcdefghijklmnop/report HTTP/1.1
Content-Type: application/json

{
   "robot_id": 2,
   "distance": 2.000
}
```

Response #7:

```
HTTP/1.1 200 OK
```

- **Request #8:**

```
GET /alien/abcdefghijklmnop/position HTTP/1.1
```

Response #8:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "position": {"x": 3, "y": -1}
}
```

5.8 Feature Extension: E

> **หมายเหตุ:** Web Service Application ของผู้เข้าแข่งขันจำเป็นต้อง Implement Feature Extension A ก่อนจึงจะ Implement Feature ใหม่ต่อไปนี้ได้

เราจะเพิ่ม Feature ให้แก่ Web Service Application ดังนี้

- Operator ที่เป็นมนุษย์สามารถ Query เพื่อสอบถามว่า Robot สองตัวที่อยู่ใกล้กันมากที่สุด ณ เวลาปัจจุบันนั้นมีระยะห่างกันเท่าใด (ภายใต้ Euclidean Distance Metric)

API Endpoints

สำหรับ Feature Extension นี้จะต้อง Implement HTTP Endpoint เพิ่มเติมอีก 1 จุดคือ

`GET /closestpair`

Query เพื่อให้ Web Service ค้นหาระยะห่างระหว่าง Robot คู่ใด ๆ ที่มีค่าน้อยที่สุด โดยอ้างอิงข้อมูลตำแหน่งของ Robot ที่อัปเดตล่าสุด

Response Object จะมี JSON Field เดียวคือ `"distance"` ซึ่งระบุค่าระยะห่างที่น้อยที่สุดระหว่าง Robot คู่ใด ๆ

รายละเอียดเพิ่มเติมสามารถหาอ่านได้จากไฟล์ API Specification ที่แนบมากับเอกสารนี้

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": -1, "y": 1}
}
```

Response #1:

```
HTTP/1.1 204 No Content
```

- **Request #2:**

```
GET /closestpair HTTP/1.1
```

Response #2:

```
HTTP/1.1 424 Failed Dependency
```

- **Request #3:**

```
PUT /robot/2/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 3, "y": -3}
}
```

Response #3:

```
HTTP/1.1 204 No Content
```

- **Request #4:**

```
PUT /robot/4/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 5, "y": -2}
}
```

Response #4:

```
HTTP/1.1 204 No Content
```

- **Request #5:**

```
GET /closestpair HTTP/1.1
```

Response #5:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 2.236
}
```

5.9 Feature Extension: F

> **หมายเหตุ:** Web Service Application ของผู้เข้าแข่งขันจำเป็นต้อง Implement Feature Extension C ก่อนจึงจะ Implement Feature ใหม่ต่อไปนี้ได้

เราจะเพิ่ม Feature ให้แก่ Web Service Application ดังนี้

- ใน Feature Extension C เดิมทีนั้น Endpoint **`POST /nearest`** จะสามารถ Query เพื่อค้นหา Robot ที่อยู่ใกล้กับจุดที่กำหนดให้มากที่สุดเพียง 1 ตัว ในส่วนนี้ เราจะรับ Input Parameter `k` เพิ่มเติม เพื่อระบุว่าเราต้องการค้นหา Robot ที่อยู่ใกล้กับจุดที่กำหนดให้มากที่สุดทั้งสิ้น `k` ตัว

Modified Changes

- ผู้เข้าแข่งขันจะต้องแก้ไข Endpoint **`POST /nearest`** เพื่อให้รองรับ JSON Field ใหม่ชื่อว่า `"k"` ซึ่งจะเป็นจำนวนเต็มที่ระบุจำนวน Robot ที่อยู่ใกล้จุด `"ref_position"` มากที่สุดที่ต้องการค้นหา

  Response Object จะมีอาร์เรย์ `"robot_ids"` ความยาวไม่เกิน `k` ซึ่งระบุ Robot ID เรียงตามลำดับระยะห่างจากจุดที่กำหนดให้ จากน้อยที่สุดไปหามากที่สุด หากมี Robot หลายตัวที่มีระยะห่างจากจุดดังกล่าวเท่านั้น ให้พิจารณา Robot ที่มีค่า ID น้อยกว่าก่อนเป็นอันดับแรก

รายละเอียดเพิ่มเติมสามารถหาอ่านได้จากไฟล์ API Specification ที่แนบมากับเอกสารนี้

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
  "position": {"x": 1, "y": 1}
}
```

  Response #1:

```
HTTP/1.1 204 No Content
```

- **Request #2:**

```
POST /nearest HTTP/1.1
Content-Type: application/json

{
  "ref_position": {"x": -1, "y": 1},
  "k": 2
}
```

Response #2:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "robot_ids": [1]
}
```

- **Request #3:**

```
PUT /robot/2/position HTTP/1.1
Content-Type: application/json

{
   "position": {"x": 0, "y": 0}
}
```

Response #3:

```
HTTP/1.1 204 No Content
```

- **Request #4:**

```
POST /nearest HTTP/1.1
Content-Type: application/json

{
   "ref_position": {"x": -1, "y": 1}
}
```

Response #4:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "robot_ids": [2]
}
```

- **Request #5**

```
POST /nearest HTTP/1.1
Content-Type: application/json

{
   "ref_position": {"x": -1, "y": 1},
   "k": 2
}
```

Response #5:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "robot_ids": [2, 1]
}
```

- **Request #6:**

```
POST /nearest HTTP/1.1
Content-Type: application/json

{
   "ref_position": {"x": 1, "y": 0},
   "k": 2
   }
```

Response #6:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "robot_ids": [1, 2]
}
```

5.10 Feature Extension: G

> **หมายเหตุ:** Web Service Application ของผู้เข้าแข่งขันจำเป็นต้อง Implement Feature Extension A ก่อนจึงจะ Implement Feature ใหม่ต่อไปนี้ได้

เราจะเพิ่ม Feature ให้แก่ Web Service Application ดังนี้

- เนื่องจาก Robot บางตัวเป็นรุ่นเก่า ซึ่งทำงานตาม Legacy API Specification เราจึงจำเป็นต้องแก้ไข Endpoint 2 อัน (ได้แก่ **`POST /distance`** และ **`PUT /robot/{robot_id}/position`**) ให้รองรับค่าพิกัด $(x, y) \in \mathbb{Z}^2$ ที่เป็น “Legacy Format X” ได้อีกด้วย

  กล่าวคือจากเดิมที่ค่าพิกัดจะระบุด้วย Object ที่ประกอบไปด้วย 2 Subfield `"x"` และ `"y"` ใน Legacy Format X นี้จะใช้ Subfield `"east"` หรือ `"west"` แทนค่าพิกัด `"x"` เดิม (ในทิศบวกและลบตามลำดับ) และจะใช้ Subfield `"north"` ละ `"south"` แทนค่าพิกัด `"y"` เดิม (ในทิศบวกและลบตามลำดับ)

  ยกตัวอย่างเช่น `{"x": 0, "y": -1}` สามารถเขียนเป็น `{"south": 1, "east": 0}` แทนได้ หรือ `{"x": -2, "y": 3}` สามารถเขียนเป็น `{"north": 3, "west": 2}` แทนได้ เป็นต้น

Modified Changes

- ผู้เข้าแข่งขันจะต้องแก้ไข Endpoint **`POST /distance`** เพื่อให้ JSON Field `"first_pos"` และ `"second_pos"` ของ Request Object สามารถรองรับพิกัดแบบ Legacy Format X ที่ระบุข้างต้นได้
- ผู้เข้าแข่งขันจะต้องแก้ไข Endpoint **`PUT /robot/{robot_id}/position`** เพื่อให้ JSON Field `"position"` ของ Request Object สามารถรองรับพิกัดแบบ Legacy Format X ที่ระบุข้างต้นได้

รายละเอียดเพิ่มเติมสามารถหาอ่านได้จากไฟล์ API Specification ที่แนบมากับเอกสารนี้

Example Requests and Responses

พิจารณาลำดับของการเรียก API Endpoints ต่อไปนี้

- **Request #1:**

```
POST /distance HTTP/1.1
Content-Type: application/json

{
  "first_pos": {"south": 2, "east": 3},
  "second_pos": {"north": 1, "west": 1}
}
```

Response #1:

```
HTTP/1.1 200 OK
Content-Type: application/json

{
   "distance": 5
}
```

- **Request #2:**

```
PUT /robot/1/position HTTP/1.1
Content-Type: application/json

{
   "position": {"east": 3, "south": 4}
}
```

Response #2:

```
HTTP/1.1 204 No Content
```

5.11 Testing Scenarios

Web Service Application จะถูกนำมาทดสอบกับสถานการณ์ต่าง ๆ และหากทำงานได้อย่างถูกต้องภายใต้สถานการณ์ดังกล่าว จะได้คะแนนดังต่อไปนี้

ปืบแต้ม	สถานการณ์ที่ทดสอบ Application	
+100	**Implement:** Baseline Feature	ไม่เกิน 500 Request
+200	**Implement:** Baseline Feature + A	ไม่เกิน 500 Request
+50	**Implement:** Baseline Feature + B	ไม่เกิน 500 Request
+90	**Implement:** Baseline Feature + A + C	ไม่เกิน 3,000 Request (Level 1)
+60	**Implement:** Baseline Feature + A + C	ไม่เกิน 150,000 Request (Level 2)
+90	**Implement:** Baseline Feature + A + D	ไม่เกิน 15,000 Request
+90	**Implement:** Baseline Feature + A + E	ไม่เกิน 100 Request (Level 1)
+40	**Implement:** Baseline Feature + A + E	ไม่เกิน 3,000 Request (Level 2)
+40	**Implement:** Baseline Feature + A + E	ไม่เกิน 150,000 Request (Level 3)
+80	**Implement:** Baseline Feature + A + C + F	ไม่เกิน 3,000 Request
+50	**Implement:** Baseline Feature + A + G	ไม่เกิน 500 Request
+10	ผู้เข้าแข่งขันเลือกที่จะปล่อย Source Code ของ Application นี้เป็น GPL-Compatible Open Source License จะได้คะแนนพิเศษส่วนนี้ไป	

หมายเหตุ: สำหรับแต่ละ Test Scenario ข้างต้น จะต้องรองรับ Request ทั้งหมดภายในเวลา 5 นาที และจะมี Resource ให้เป็น RAM 2 GB และ CPU จำนวน 2 cores

Final Scoring

คะแนนที่ผู้เข้าแข่งขันได้จากสถานการณ์ที่ทดสอบข้างต้นนี้ จะถูกนำมา Normalized ให้เป็นคะแนนเต็มสำหรับโจทย์ Containerized Application นี้ (จากคะแนนเต็ม 900 ปืบแต้มเหลือคะแนนเต็ม 80 คะแนน)

5.12 Contest Post-mortem

เวลาประกาศ Feature ใหม่

ตลอดเวลา 4 ชั่วโมงของการแข่งขันโจทย์ Containerized Web Service Application มีการประกาศ Feature ใหม่ระหว่างการแข่งขัน ณ เวลาต่าง ๆ ดังต่อไปนี้ (เวลาที่ปรากฏเป็นเวลาที่เริ่มนับจากเริ่มต้นจับเวลา)

- [0:00] ประกาศ **Baseline Feature**
- [0:25] ประกาศ **Feature A**
- [0:35] ประกาศ **Feature B**
- [0:55] ประกาศ **Feature C**
- [1:20] ประกาศ **Feature D**
- [2:00] ประกาศ **Feature E**
- [2:35] ประกาศ **Feature F**
- [3:30] ประกาศ **Feature G**

Implementation ของผู้เข้าแข่งขัน

ผู้เข้าแข่งขัน Final Round ได้ใช้ความสามารถของตนเองเขียน Web Service Application ออกมา โดยสามารถดูเพิ่มเติมได้จาก Repository URL ดังต่อไปนี้

- [0010] **12eminiscenc3**
  `https://gitlab.com/disneyp/techjam`
- [0028] **Boring**
  `https://github.com/Heybabii/Boring`
- [0105] **PalmPTSJ**
  `https://github.com/op01/techjam2019-deepcode`
- [0143] **RMB**
  `https://github.com/Lucina101/ROBOTQUERY`
- [0160] **FM and POL**
  `https://github.com/polsurakit/techjam2019`
- [0166] **มาหาเงินจ่ายค่าเซิฟ**
  `https://github.com/karnjj/Techjam_Final`
- [0293] **FIRST VS THE WORLD**
  `https://github.com/nisaruj/techjam2019`
- [0362] **stang**
  `https://github.com/Kritsatorn/Robot2`
- [0489] **Tech Jammer**
  `https://github.com/phirasit/techjam_robot`

- [0612] **ชาเขียวแสนอร่อย**
  https://github.com/pkktino/techjam-final-matcha
- [0655] **InwJuneสั่งมา**
  https://github.com/nick41746/Techjam-Application
- [0747] **↓ everyone ↓**
  https://github.com/vzsky/techjam_everyone
- [0833] **ชื่ออะไรดี**
  https://github.com/miello/Robot
- [0970] **- Shadow -**
  https://github.com/kritsr/tj_ws
- [1017] **กะเพราหมูกรอบอร่อย:)**
  https://github.com/itzmeowww/Techjam2019
- [1018] **Beginner Stand**
  https://gitlab.com/beginner-stand/robots
- [1149] **O analyzed**
  https://github.com/Chomtana/techjam-2019-real
- [1183] **Wiggle Toes**
  https://github.com/blutarche/tj2019-app
- [1211] **Special Unit**
  https://github.com/Phon1209/Techjam
- [1252] **SIRCOSTUMEPREMIER**
  https://github.com/szawinis/techjam
- [1254] **yee**
  https://github.com/matikant/Techjam2019-Containerized-Application-Task
- [1289] **ALONE**
  https://github.com/RogerKSI/techjam2019-1289
- [1300] **Runtime Terror**
  https://github.com/nattichai/RuntimeTerror