International Olympiad in Informatics 2012
23-30 September 2012
Sirmione - Montichiari, Italy
Competition tasks, day 1: Leonardo's inventions and projects

odometer
ไทย — 1.2

มาตรวัดระยะทางด้วยการหย่อนลูกแก้ว

ลีโอนาร์โดประดิษฐ์ต้นแบบของ มาตรวัดระยะทาง มาตรวัดระยะทางเป็นรถเข็นที่วัดระยะทางโดยการหย่อนลูกแก้วในขณะที่ล้อของรถเข็นนั้นหมุนไปเรื่อย ๆ จำนวนของลูกแก้วที่หย่อนลงก็คือจำนวนรอบที่ล้อหมุน ซึ่งจะทำให้ผู้ใช้สามารถคำนวณระยะทางที่มาตรวัดระยะทางเคลื่อนที่ไปได้ ในฐานะที่คุณเป็นนักคอมพิวเตอร์ เราได้เพิ่มระบบซอฟต์แวร์ควบคุมเพื่อเพิ่มความสามารถให้กับมาตรวัดนี้ หน้าที่ของคุณคือการเขียนโปรแกรมมาตรวัดระยะทางตามกฎที่กำหนดให้ดังต่อไปนี้

กริดงาน

มาตรวัดระยะทางเคลื่อนที่อยู่บนกริดสี่เหลี่ยมจัตุรัสในจินตนาการขนาด 256×256 ช่อง แต่ละช่องจะมีลูกแก้วได้ไม่เกิน 15 ลูกและแต่ละช่องจะถูกอ้างอิงด้วยคู่ลำดับ (แถว, คอลัมน์) โดยค่าของคู่ลำดับอยู่ในช่วง $0, \ldots, 255$. กำหนดให้ ช่อง (i, j) จะเป็นช่องที่ติดกับช่อง $(i - 1, j)$, $(i + 1, j)$, $(i, j - 1)$ และ $(i, j + 1)$ (ถ้ามีช่องเหล่านั้นอยู่) ช่องใด ๆ ที่อยู่ในแถวแรกหรือแถวสุดท้าย หรือ คอลัมน์แรกหรือคอลัมน์สุดท้าย จะเรียกว่าขอบ มาตรวัดระยะทางจะเริ่มต้นที่ช่อง $(0, 0)$ (ช่องมุมเหนือสุดตะวันตกสุด) และหันหน้าไปทิศเหนือเสมอ

คำสั่งพื้นฐาน

เราสามารถโปรแกรม มาตรวัดระยะทาง ด้วยคำสั่งต่อไปนี้

- `left` — หันไปทางซ้าย 90 องศา (ทวนเข็มนาฬิกา) และยังอยู่ในช่องเดิม (เช่น ถ้ากำลังหันหน้าไปทางทิศใต้ ก็จะหันไปทางทิศตะวันออกหลังจากทำตามคำสั่ง)
- `right` — หันไปทางขวา 90 องศา (ตามเข็มนาฬิกา) และยังอยู่ในช่องเดิม (เช่น ถ้ากำลังหันหน้าไปทางทิศตะวันตก ก็จะหันไปทางทิศเหนือหลังจากทำตามคำสั่ง)
- `move` — เดินไปข้างหน้าหนึ่งช่อง (ตามทิศทางปัจจุบันของมาตรวัดระยะทาง) ไปยังช่องข้างเคียง ถ้าไม่มีช่องให้ไปแล้ว (เช่น อยู่ที่ขอบของทิศทางนั้นแล้ว) คำสั่งนั้นก็จะไม่มีผลใด ๆ
- `get` — เอาลูกแก้วหนึ่งลูกออกจากช่องปัจจุบัน ถ้าไม่มีลูกแก้วอยู่ คำสั่งนี้ก็จะไม่มีผลใดๆ
- `put` — หย่อนลูกแก้วหนึ่งลูกในช่องปัจจุบัน ถ้าช่องนั้นมีลูกแก้ว 15 ลูกแล้วคำสั่งนี้จะไม่มีผลใดๆ อย่างไรก็ตามมาตรวัดระยะทางมีลูกแก้วไม่จำกัด
- `halt` — จบการทำงาน

มาตรวัดระยะทางจะทำงานคำสั่งต่าง ๆ ตามลำดับที่ได้รับมาในโปรแกรม โปรแกรมจะต้องมีคำสั่งไม่เกินหนึ่งคำสั่งต่อหนึ่งบรรทัด บรรทัดว่างจะถูกข้าม สัญลักษณ์ # ระบุถึง comment เราไม่สนใจข้อความใด ๆ ที่ตามหลังสัญลักษณ์ดังกล่าวจนถึงท้ายบรรทัด ถ้ามาตรวัดระยะทางทำงานถึงจุดจบของโปรแกรม การทำงานจะหยุดลง

ตัวอย่างที่ 1

ให้พิจารณาโปรแกรมสำหรับมาตรวัดระยะทางต่อไปนี้ โปรแกรมนี้จะทำให้มาตรวัดระยะทางไปยังช่อง $(0,2)$ หันหน้าไปทางตะวันออก (ให้สังเกตว่าคำสั่ง `move` คำสั่งแรกนั้นถูกเพิกเฉยเพราะว่ามาตรวัดระยะทางนั้น

อยู่ที่ช่องมุมเหนือสุดตะวันตกสุดและหันหน้าไปทางทิศเหนือ)

```
move  # ไม่มีผล
right
# ขณะนี้มาตรวัดระยะทางมุ่งหน้าทิศตะวันออก
move
move
```

ป้าย (label), ขอบ และ ลูกแก้ว

เพื่อที่จะเปลี่ยนทิศทางการทำงานตามสถานะปัจจุบันของโปรแกรม คุณสามารถใช้ ป้าย (label) ซึ่งเป็นสายอักขระที่ขึ้นกับตัวพิมพ์เล็ก/พิมพ์ใหญ่ ยาวไม่เกิน 128 ตัวอักษร ซึ่งอาจประกอบด้วย a, ..., z, A, ..., Z, 0, ..., 9 คำสั่งใหม่ที่เกี่ยวข้องกับ ป้าย แสดงตามรายการด้านล่าง ซึ่ง L หมายถึงป้ายที่ถูกต้อง

- L: (เช่น L ตามด้วย colon ':') — ประกาศตำแหน่งภายในโปรแกรมของป้าย L การประกาศป้ายทั้งหมดจะต้องไม่ซ้ำกัน การประกาศป้ายไม่มีผลใดต่อมาตรวัดระยะทาง
- `jump` L — ประมวลผลต่อไปโดยกระโดดไปทำคำสั่งในบรรทัดที่มีป้าย L โดยไม่มีเงื่อนไข
- `border` L — ประมวลผลโปรแกรมโดยกระโดดไปยังบรรทัดที่มีป้ายL, ถ้ามาตรวัดระยะทางอยู่บนขอบหันหน้าเข้าหาขอบของกริด (กล่าวคือ คำสั่ง `move` จะไม่มีผลใดๆ) มิฉะนั้น การประมวลผลจะทำต่อเนื่องตามปรกติ และคำสั่งนี้ก็จะไม่มีผลใดๆ
- `pebble` L — ประมวลผลโปรแกรมโดยกระโดดไปยังบรรทัดที่มีป้ายL ถ้าช่องปัจจุบันมีลูกแก้วอย่างน้อยหนึ่งลูก มิฉะนั้นการประมวลผลจะทำต่อเนื้องตามปรกติ และคำสั่งนี้ก็จะไม่มีผลใดๆ

ตัวอย่างที่ 2

โปรแกรมต่อไปนี้หาลูกแก้วลูกที่อยู่ตะวันตกสุดของแถวศูนย์ และหยุด ณ ตำแหน่งนั้น ถ้าไม่มีลูกแก้วที่แถวศูนย์มันจะหยุดที่ขอบของแถวดังกล่าว โปรแกรมนี้ใช้ label สองตัวคือ `leonardo` และ `davinci`

```
right
leonardo:
pebble davinci  # พบลูกแก้ว
border davinci  # สุดแถว
move
jump leonardo
davinci:
halt
```

มาตรวัดระยะทางเริ่มต้นโดยหันหน้าไปทางขวา ลูปเริ่มด้วยการประกาศ label `leonardo:` และจบด้วยคำสั่ง `jump` ในลูปนั้น มาตรวัดระยะทางตรวจสอบว่ามีลูกแก้วหรือว่าอยู่ที่ขอบของแถวหรือไม่ ถ้าไม่ใช่ มาตรวัดระยะทางจะทำงานคำสั่ง `move` เพื่อเคลื่อนที่จากช่อง $(0,j)$ ไปยังช่อง $(0,j+1)$ ที่อยู่ติดกัน (คำสั่ง `halt` นั้นจริง ๆ แล้วจะมีหรือไม่มีก็ไม่ต่างกันเพราะว่าโปรแกรมจะหยุดทำงานอยู่ดี)

ปัญหา

คุณควรส่งโปรแกรมในภาษาของมาตรวัดระยะทางตามที่ระบุไว้ข้างต้นเพื่อจะให้มาตรวัดระยะทางทำงานตามที่คาดไว้ งานย่อยแต่ละงาน (ดูด้านล่าง) ระบุพฤติกรรมที่มาตรวัดระยะทางจะต้องทำและข้อกำหนดที่คำตอบที่ส่งจะต้องเป็นไปตามข้อกำหนดนั้นจะครอบคลุมถึงสองเรื่องต่อไปนี้

- ขนาดโปรแกรม — ความยาวของโปรแกรมควรจะต้องสั้น ขนาดโปรแกรมวัดได้จากจำนวนคำสั่งที่ใช้ ซึ่งในการวัดขนาดโปรแกรม เรา ไม่นับ การประกาศใช้ป้าย (label), คำอธิบายอื่นๆ และบรรทัดว่าง

- ระยะเวลาในการประมวลผล — โปรแกรมจะต้องหยุดทำงานเร็วพอ ระยะเวลาในการประมวลผลคือจำนวนของ step ที่ทำงาน คำสั่งทั้งหมดที่ทำงานจะถูกนับเป็น step ไม่ว่าคำสั่งนั้นจะมีผลหรือไม่ก็ตาม การประกาศใช้ป้าย (label), คำอธิบายอื่นๆ (comment) และบรรทัดว่างไม่นับเป็น step

ในตัวอย่าง 1 ขนาดโปรแกรมเป็น 4 และระยะเวลาในการประมวลผลเป็น 4 ในตัวอย่าง 2 ขนาดโปรแกรมเป็น 6 และเมื่อทำงานบนกริดที่มีลูกแก้วในช่อง (0,10) ระยะเวลาในการประมวลผลจะเป็น 43 step ดังนี้ `right`, วนลูป 10 รอบ แต่ละรอบใช้ 4 step (`pebble davinci`; `border davinci`; `move`; `jump leonardo`) และท้ายสุด `pebble davinci` และ `halt`

งานย่อยที่ 1 [9 คะแนน]

เริ่มต้นมีลูกแก้ว x ลูกในช่อง (0,0) และ y ลูกในช่อง (0,1) โดยที่ช่องอื่น ๆ นั้นไม่มีลูกแก้ว พึงระลึกว่ามันจะมีลูกแก้วไม่เกิน 15 ลูกในแต่ละช่อง จงเขียนโปรแกรมที่หยุดการทำงานโดยที่มาตรวัดระยะทางอยู่ในช่อง (0,0) ถ้า $x \leq y$ และหยุดในช่อง (0,1) ในกรณีอื่น ๆ (เราไม่สนใจทิศทางที่มาตรวัดระยะทางหันไปเมื่อจบการทำงาน เราไม่สนใจด้วยเช่นกันว่ามีลูกแก้วอยู่กี่ลูกอยู่ที่ไหนบ้างเมื่อจบการทำงาน)

ข้อจำกัด: ขนาดโปรแกรม ≤ 100, ระยะเวลาในการประมวลผล $\leq 1{,}000$.

งานย่อยที่ 2 [12 คะแนน]

เหมือนงานข้างต้น แต่เมื่อโปรแกรมสิ้นสุดการทำงาน ช่อง (0, 0) จะต้องมีลูกแก้ว x ลูก และ ช่อง (0, 1) จะต้องมีลูกแก้ว y ลูก

ข้อจำกัด: ขนาดโปรแกรม ≤ 200, ระยะเวลาในการประมวลผล $\leq 2{,}000$.

งานย่อยที่ 3 [19 คะแนน]

มีลูกแก้วสองลูกเท่านั้นอยู่สักแห่งในแถวที่ 0: ลูกที่หนึ่งอยู่ในช่อง (0, x) และอีกลูกอยู่ในช่อง (0, y) โดยที่ x และ y มีค่าต่างกัน และ x + y เป็นเลขคู่ จงเขียนโปรแกรมเพื่อปล่อยมาตรวัดระยะทางไว้ที่ช่อง (0, (x + y) / 2), เช่น, ตำแหน่งตรงกลางพอดีระหว่างสองช่องมีลูกแก้วอยู่ สถานะสุดท้ายของกริดไม่มีผลเกี่ยวข้องใดๆ

ข้อจำกัด: ขนาดโปรแกรม ≤ 100, ระยะเวลาในการประมวลผล $\leq 200{,}000$.

งานย่อยที่ 4 [ได้ไม่เกิน 32 คะแนน]

มีลูกแก้วอย่างมาก 15 ลูกในกริด ไม่มีสองลูกใด ๆ อยู่ในช่องเดียวกัน เขียนโปรแกรมที่รวบรวมลูกแก้วทั้งหมดไว้ที่ช่องมุมเหนือสุดตะวันตกสุด กล่าวโดยละเอียดคือ ถ้ามีลูกแก้ว x ลูกในกริด ณ ตอนเริ่มต้น ในตอนจบนั้นจะต้องมีลูกแก้ว x ลูกพอดีในช่อง (0,0) และไม่มีลูกแก้วเหลืออยู่ในช่องอื่นใด

คะแนนสำหรับงานย่อยนี้ขึ้นอยู่กับระยะเวลาในการประมวลผลของโปรแกรมที่ส่งเข้ามา กล่าวโดยละเอียดคือ ถ้าให้ L เป็นระยะเวลาในการประมวลผลมากสุดจากกรณีทดสอบต่าง ๆ คะแนนของคุณจะเป็น:

- 32 คะแนน ถ้า $L \leq 200{,}000$;
- $32 - 32 \log_{10}(L / 200{,}000)$ คะแนน ถ้า $200{,}000 < L < 2{,}000{,}000$;
- 0 คะแนน ถ้า $L \geq 2{,}000{,}000$.

ข้อจำกัด: ขนาดของโปรแกรม ≤ 200.

งานย่อยที่ 5 [ได้ไม่เกิน 28 คะแนน]

ในแต่ละช่องของกริดจะมีลูกแก้วอยู่กี่ลูกก็ได้ (ระหว่าง 0 ถึง 15 ลูก) จงเขียนโปรแกรมที่จะหาช่องที่มีลูกแก้วน้อยที่สุด กล่าวคือ เขียนโปรแกรมที่ทำให้มาตรวัดระยะทางหยุดการทำงาน ณ ช่อง (i, j) โดยที่ช่องอื่น ๆ มีลูกแก้วอย่างน้อยเท่ากับช่อง (i, j) หลังจากที่โปรแกรมนี้ทำงานเสร็จแล้ว จำนวนของลูกแก้วในแต่ละช่องจะต้องเท่ากับจำนวนลูกแก้วก่อนที่โปรแกรมจะเริ่มทำงาน

คะแนนของงานย่อยนี้ขึ้นอยู่กับขนาดของโปรแกรม P ที่ได้จากโปรแกรมที่คุณส่งมา โดยรายละเอียดของการให้คะแนนมีดังนี้

- 28 คะแนน ถ้า $P \leq 444$;
- $28 - 28 \log_{10}(P / 444)$ คะแนน ถ้า $444 < P < 4,440$;
- 0 คะแนน ถ้า $P \geq 4,440$.

ข้อจำกัด: ระยะเวลาในการประมวลผล ≤ 44,400,000.

รายละเอียดสำหรับการเขียนโปรแกรม

คุณจะต้องส่งไฟล์หนึ่งไฟล์ต่องานย่อยเท่านั้น โดยให้เขียนตามกฎไวยากรณ์ที่กำหนดข้างต้น ไฟล์ที่ส่งแต่ละไฟล์จะมีขนาดได้ไม่เกิน 5 MiB สำหรับแต่ละงานย่อย โปรแกรมมาตรวัดระยะทางจะถูกทดสอบด้วยกรณีทดสอบเพียงสองถึงสามอัน และคุณจะได้รับผลตอบกลับตามทรัพยากรที่คุณใช้ในโปรแกรมของคุณ ในกรณีที่โปรแกรมไม่ถูกต้องตามหลักไวยากรณ์ซึ่งจะทำให้ทดสอบไม่ได้ คุณจะได้รับข้อมูลของ syntax error นั้น

คุณไม่จำเป็นต้องส่งโปรแกรมมาตรวัดระยะทาง สำหรับทุก ๆ งานย่อย ถ้าในการส่งครั้งนั้นไม่มีโปรแกรมมาตรวัดระยะทางสำหรับงานย่อย X การส่งที่มีงานย่อย X ครั้งล่าสุดจะถูกรวมไว้โดยอัตโนมัติ ถ้าไม่มีโปรแกรมนั้นเลย งานย่อยข้อนั้นจะได้ศูนย์คะแนนในการส่งครั้งนั้น

คะแนนจากการส่งคือผลรวมของคะแนนจากงานย่อย และคะแนนสุดท้ายคือ คะแนนสูงสุดที่ได้จากการส่งแบบ release ทั้งหมดและการส่งครั้งสุดท้าย

ตัวจำลองสถานการณ์

คุณจะได้รับระบบจำลองสถานการณ์ของมาตรวัดระยะทางสำหรับการทดสอบ คุณสามารถป้อนโปรแกรมของคุณและกริดเข้าไปทดสอบในระบบจำลองสถานการณ์ได้ โปรแกรมของมาตรวัดระยะทางนี้เขียนด้วยรูปแบบเดียวกับที่ใช้ในการส่งงาน (ดังเช่นที่อธิบายไว้ข้างต้น)

เราจะอธิบายลักษณะของกริดโดยใช้รูปแบบต่อไปนี้ : แต่ละบรรทัดในไฟล์จะต้องประกอบด้วยตัวเลข 3 ตัว คือ R, C และ P ซึ่งหมายถึง ในช่องที่แถว R และคอลัมน์ C มีลูกแก้ว P ลูก ช่องทุกช่องที่ไม่ถูกระบุค่า จะถือว่าไม่มีลูกแก้วอยู่ ตัวอย่างเช่น พิจารณาไฟล์:

```
0 10 3
4 5 12
```

ไฟล์ข้างต้นนี้ระบุถึงกริดที่มีลูกแก้ว 15 ลูก: 3 ลูกในช่อง (0, 10) และ 12 ลูกในช่อง (4, 5).

คุณสามารถเรียกใช้ตัวทดสอบโดยเรียกจากโปรแกรม `simulator.py` ซึ่งอยู่ในไดเร็กทอรี่ของโจทย์ โดยใส่อาร์กิวเมนต์เป็นชื่อไฟล์โปรแกรม ระบบจำลองสถานการณ์สามารถรับคำสั่ง command line option ดังต่อไปนี้:

- `-h` จะแสดงข้อมูลเบื้องต้นเกี่ยวกับ option ที่มีให้
- `-g GRID_FILE` อ่านค่าคำอธิบายกริดจากไฟล์ `GRID_FILE` (ถ้าไม่กำหนดค่า: กริดว่าง);
- `-s GRID_SIDE` กำหนดขนาดของกริดเป็น `GRID_SIDE x GRID_SIDE` (ถ้าไม่กำหนด: 256, ซึ่งใช้ตามข้อกำหนดของโจทย์); การใช้กริดขนาดเล็กจะมีประโยชน์สำหรับการดีบั๊กโปรแกรม
- `-m STEPS` จำกัดจำนวนขั้นตอนการประมวลผล (step) ในตัวจำลองสถานการณ์สูงสุดไม่เกิน `STEPS` ขั้น
- `-c` เข้าสู่โหมดการคอมไพล์ โดยในโหมดการคอมไพล์นี้ ตัวจำลองสถานการณ์จะคืนค่าแบบเดียวกับเอาต์พุต แต่แทนที่จะจำลองด้วย Python มันจะสร้างพร้อมคอมไพล์โปรแกรมภาษา C เล็ก ๆ อันหนึ่ง ซึ่งจะทำให้เสีย overhead ในตอนเริ่มต้น แต่จะให้ผลลัพธ์ที่เร็วกว่า ซึ่งคุณควรจะใช้มันเมื่อโปรแกรมของคุณต้องทำงานมากกว่า 10,000,000 ขั้น

จำนวนครั้งการส่งโปรแกรม

คุณไม่สามารถส่งโปรแกรมสำหรับโจทย์ข้อนี้ได้เกิน 128 ครั้ง