Хайрган хэмжигч

Ленорда Давинчи "Одометр"-ын анхны хувилбарыг зохион бүтээжээ. Энэ нь зайг хэмжихэд зориулагдсан хөнгөн тэрэг байсан бөгөөд тэргэний дугуй эргэх тоолонд хайрга унагаан түүний тусламжтайгаар зайг хэмждэг байсан. Унагаасан хайрганы тоогоор дугуйны эргэлтийг тооцолж "Одометр"-ын туулсан зайг тооцоолдог байжээ. Бид програм зохиогчдын хувьд Одометрийн гүйцэтгэх үйлдлийг баяжуулж Одометрт удирдах програмыг суулгасан. Таны даалгавар бол Одометрыг доор дурьдсан үйлдлүүдийн дагуу програмчлах.

Үйлдлийн хүснэгт

Одометр 256x256 хэмжээтэй квадрат хүснэгт дээр явж байгаа гэж төсөөл. Нүд бүр хамгийн ихдээ 15 хайрга агуулна. Нүдний координат нь (мөр, багана) гэсэн хослолоор өгөгдөнө. Кординат бүр нь 0..255 хүртэлх тоон утгатай байна. (i, j) гэсэн нүд өгөгдсөн гэж үзвэл (i - 1, j), (i + 1, j), (i, j - 1) болон (i, j + 1) нүднүүд нь түүнтэй хөрш нүднүүд байна. Хамгийн эхний эсвэл сүүлийн мөр мөн хамгийн эхний эсвэл сүүлийн багана дахь нүднүүдийг *хүрээ* гэнэ. Одометр үргэлж (0, 0) (баруун-хойд булан) нүднээс явж эхлэхдээ хойд зүгт харж зогссон байна.

Үндсэн командууд

Одометрыг програмчлахдаа дараах командуудыг ашиглана.

- `left` — зүүн тийш 90 градус эргэх ба (цагийн зүүний эсрэг) зогсож буй нүдэндээ үлдэнэ (жишээ нь урд зүгт харж байсан бол уг командын дараа зүүн тийш харсан байна).
- `right` — баруун тийш 90 градус эргэх ба (цагийн зүүний дагуу) зогсож буй нүдэндээ үлдэнэ (жишээ нь баруун зүгт харж байсан бол уг командын дараа хойд зүгт харсан байна).
- `move` - харж байгаа зүгрүүгээ нэг алхамаар урагшлах буюу тухайн зүгт байгаа хөрш нүдрүүгээ шилжинэ. (Одометрийн харж байгаа зүгт) Хэрвээ харж байгаа зүгт нь хөрш нүд байхгүй бол өөрөөр хэлбэл тухайн чиглэл дахь хүрээнд зогсож байвал энэ команд ямар ч нөлөөгүй.(Одометр хөдөлгөөн хийхгүй)
- `get` — зогсож байгаа нүднээсээ нэг хайрга авна. Хэрвээ одоогийн нүдэнд хайрга байхгүй бол энэ команд нөлөөгүй.
- `put` — зогсож байгаа нүдэндээ нэг хайрга тавина. Хэрвээ тухайн нүдэнд аль хэдийн 15 хайрга байвал энэ команд нөлөөлөхгүй.

- `halt` — үйл ажиллагааг дуусгана.

Одометр нь програмд өгөгдсөн командуудыг дарааллынх нь дагуу мөр мөрөөр нь биелүүлнэ. Хоосон мөрийг тоохгүй орхино.`#` тэмдэгтийн дараахийг тайлбар гэж үзэх бөгөөд мөрийн төгсгөл хүртэл тоохгүй орхино. Хэрвээ Одометр програмын төгсгөлд хүрвэл биелүүлэлт төгсөнө.

Жишээ 1

Дараахи програм одометрт өгөгдсөн гэж үзье.Үүний үр дүнд одометр (0, 2) нүдэнд зүүн тийш харж зогсоно. Эхний `move` командыг тоохгүй орхино гэдгийг анхаар. Учир нь одометр анх баруун-хойд зүгт хойд зүгт харж хүрээн дээр зогсож байгаа.

```
move  # нөлөөгүй үйлдэл
right
# одоо одометр зүүн зүгт харж зогсоно.
move
move
```

Хаяг, Хүрээ and Хайрга

Програмын ажиллах дарааллыг өөрчлөхийн тулд label буюу хаяглалт хэрэглэж болно.Хаяг нь 128 хүртэлхи урттай том жижиг үсэг `a`, …, `z`, `A`, …, `Z`, `0`, …, `9` тэмдэгтүүдээс бүрдсэн тэмдэгт мөр байна. label-тэй холбоотой шинэ командуудыг доор жагсаавал. Доорхи тодорхойлолтонд *`L`* үсгээр хүчин төгөлдөр хаяглалтыг тэмдэглэлээ.

- *`L`*`:` (i.e. *`L`* тодорхойлох цэгтэй ‘`:`’)— програм дахь L хаягны байрлалыг тодорхойлно.Тодорхойлсон бүх хаяглалтууд нь цор ганц байх ёстой. Хаяг тодорхойлох нь одометрт ямар нэг байдлаар нөлөөлөхгүй.

- `jump` *`L`* — үйлдлийг аливаа нөхцөлгүйгээр L хаяг бүхий командруу шилжүүлнэ.

- `border` *`L`* —Хэрвээ одометр хүснэгтийн хүрээн дээр гадагшаа харсан байрлалтай зогсож байвал *`L`* хаяг бүхий мөрлүү шилжиж үйлдлийг үргэлжлүүлнэ. Бусад тохиолдолд уг команд нөлөө үзүүлэхгүй Одометр дараагийн командыг хэрэгжүүлнэ.

- `pebble` *`L`* — одоогийн зогсож буй нүдэнд нь дор хаяж нэг хайрга байвал L хаяг бүхий мөрлүү шилжиж цааш ажиллана. Бусад тохиолдолд энэ команд нөлөөлөхгүй.

Жишээ 2

Дараахи програм нь 0 дугаар мөрөн дэх хайргатай эхний нүд таартал яваад тэндээ зогсож үйлдлээ дуусгана. Хэрвээ 0 дугаар мөрөнд нэгч хайргатай нүд байхгүй бол одометр хүснэгтийн зах мөрийн төгсгөлд хүрээд үйлдлээ дуусгана. Энд `leonardo`болон `davinci`гэсэн 2 хаягыг хэрэглэж байна.

```
right
leonardo:
pebble davinci  # хайрга олдвол ажиллана
border davinci  # мөрийн төгсгөлд хүрвэл ажиллана
move
jump leonardo
davinci:
halt
```

Одометр баруун тийш эргэж үйлдлээ эхлэнэ. `leonardo:` гэсэн хаяглалтыг зарлахаас эхлээд `jump leonardo` гэсэн команд хүртэл давталт хийгдэнэ. Давталтан дотор Одометр хайрга байгаа эсэхийг болон хүрээнд очсон эсэхийг шалгана. Хэрвээ үгүй бол Одометр одоогийн нүд (0, j)-ээс зэргэлдээх нүд (0, j + 1)-рүү `move` явна. (Энэ тохиолдолд програм тэртээ тэргүй төгсөж байгаа тул `halt` комманд нь зайлшгүй хэрэгцээтэй биш юм.)

Даалгавар

Дээр заасны дагуу одометрийн ойлгох хэлээр одометрийн хүлээгдэж байгаагын дугуу ажиллуулах програмыг илгээх хэрэгтэй. Дэд даалгавар бүрт одорметрийн гүйцэтгэх ёстой даалгавар болон програмд тавигдах шаардлагууд өгөгдөнө.Шаадлагууд нь дараахи атрибутуудын дагуу тавигдана.

- *Program size* — Програмын хэмжээ: програм нь хангалттай богино байх ёстой. Програмын хэмжээ гэдэг нь үүнд байгаа командуудын тоогоор илэрхийлэгдэнэ. Хаяг тодорхойлох, тайлбар бичих болон хоосон зай програмын хэмжээнд *тоологдохгүй* .

- *Execution length* — Гүйцэтгэлийн урт: Програм нь аль болох хурдан биелэгдэх ёстой. Энэ нь биелэгдсэн алхамуудын тоогоор илэрхийлэгдэнэ. Команд Одометртын төлвийг өөрчилсөн эсэхээс үл хамааран биелэлт бүрийг 1 алхам гэж тоолно. Тэмдэгтийн зарлагаа, тайлбар болон хоосон мөрийг алхамд тооцохгүй.

1-р жишээнд програмын хэмжээ нь 4, гүйцэтгэлийн урт нь 4. Харин 2-р жишээнд програмын хэмжээ нь 6, зөвхөн (0, 10) гэсэн нүдэнд 1 хайрга бүхий хүснэгт дээр ажилласан гэж үзвэл гүйцэтгэлийн урт нь 43. Эдгээр алхамууд нь `right`, 10 удаагийн давталт, давталт бүр 4 алхамтай (`pebble davinci`; `border davinci`; `move`; `jump leonardo`), тэгээд эцэст нь `pebble davinci` болон `halt` команд хийгдэнэ.

Дэд даалгавар1 [9 оноо]

Эхлэх үед (0, 0) нүдэнд x ширхэг, (0, 1) нүдэнд y ширхэг хайрга байна. Бусад нүд нь хоосон байна. Хамгийн ихдээ 15 хайрга байна гэдгийг сана. Одометрийн $x \leq y$ үед (0, 0) нүдэнд эсрэг тохиолдолд (0, 1) нүдэнд очсон байхаар үйлдэлийг гүйцэтгэх програм бич. (Одометр командуудыг биелүүлж дууссаны дараа аль зүгт харж зогссон байх, хүснэгтэнд хэдэн хайрга үлдэх болон хаана байрласан байх нь бидэнд хамаагүй)

Хязгаарлалт : программын хэмжээ ≤ 100, гүйцэтгэллийн урт ≤ 1 000.

Дэд даалгавар 2 [12 оноо]

Дээрхитэй ижил даалгавар гэхдээ програм төгсөх үед (0, 0) нүдэнд яг x хайрга, (0, 1) нүдэнд яг y хайрга байх ёстой.

Хязгаарлалт : программын хэмжээ ≤ 200, гүйцэтгэллийн урт ≤ 2 000.

Дэд даалгавар 3 [19 оноо]

0 дугаар мөрөнд яг 2 ширхэг хайрга байрласан байгаа. Нэг нь (0, x) нүдэнд нөгөөх нь (0, y) нүдэнд, x, y хоёрын утга ялгаатай байх бөгөөд нийлбэр нь тэгш байна. одометрийг яг (0, (x + y) / 2) нүдэнд үлдээх програм бич.Өөрөөр хэлбэл яг дунд талын нүдэнд аваачина. Хүснэгтийн сүүлчийн төлөв нь бидэнд хамааралгүй.

Хязгаарлалт : *программын хэмжээ ≤ 100, гүйцэтгэллийн урт ≤ 200 000.*

Дэд даалгавар 4 [32 хүртэлхи оноо]

Хүснэгтэнд хамгийн ихдээ 15 хайрга байгаа. Аль ч хоёр хайрга нь нэг нүдэнд байрлаагүй. Бүх хайргуудыг баруун-хойд буланд авчрах програм бич. Өөрөөр хэлбэл хэрвээ x хайрга хүснэгтэнд өгөгдсөн бол яг x хайрга (0, 0) нүдэнд цугларч бусад бүх нүд хоосон байх ёстой.

Энэ дэд даалгаврын оноо нь илгээсэн програмын гүйцэтгэлийн уртаас хамаарна. Өөрөөр хэлбэл хэрвээ L нь чиний програмын янз бүрийн тестэнд ажилласан гүйцэтгэлийн уртын хамгийн их нь L гэвэл, чиний оноо:

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

'Хязгаарлалт : *программын хэмжээ ≤ 200*

Дэд даалгавар 5 [28 хүртэлхи оноо]

Хүснэгтийн нүд бүрт хэдэн ч ширхэг хайрга байж болно (0-оос 15-ийн хооронд). Хамгийн багыг олох програм бич. Одометр аль нэг нүдний хувьд (i, j) нүдэн дахь хайрганы тоо нь бусад бүх нүд нь дор хаяж (i, j) хайрга агуулна. Програм ажиллаж дууссаны дараа хүснэгтэн дахь хайрганы тоо нь анхны тоотойгоо ижил байх ёстой.

Энэ дэд даалгаврын оноо нь програмын хэмжээ P-гээс хамаарна. Тодорхой хэлбэл

- 28 оноо хэрвээ $P \leq 444$;
- $28 - 28 \log_{10}(P / 444)$ оноо хэрвээ $444 < P < 4\,440$;

- 0 оноо хэрвээ $P \geq 4\,440$.

Хязгаарлалт : *гүйцэтгллийн урт ≤44 400 000.*

Хэрэгжүүлэлтийн деталь

Дэд даалгавар бүрийн хувьд дээр заасан команд бүхий 1 файл илгээх эрхтэй. Илгээсэн файл бүр хамгийн идхээ 5мб хэмжээтэй байж болно. Дэд даалгавар бүрийн хувьд Одометр кодыг хэдэн тестэн дээр шалгаж үзэх бөгөөд тухайн програм хир их нөөц хэрэглэсэнийг харуулах хариу авна. Хэрвээ код бичлэгийн хувьд /syntax error/ алдаатай байвал түүнийг ажиллуулах боломжгүй бөгөөд бичлэгийн алдаа өгнө.

Бүх дэд даалгаврын хувьд өөр Одометр програм илгээх албагүй бөгөөд хэрвээ Х дэд даалгаварын хувьд илгээсэн програм байхгүй бол хамгийн сүүлийн илгээсэн програмыг автоматаар хариу болгон авна.Хэрвээ тийм програм байхгүй бол тухайн дэд даалгаварт 0 оноо авна.

Илгээсэн бодолтуудыг оноо нь дэд даалгавар бүрээс авсан оноонуудын нийлбэр байх бөгөөд эцсийн оноо нь хамгийн сүүлийн илгээлт болон түлхүүр хийн илгээсэн бодолтын хамгийн өндөр оноогоор тогтооогдоно.

Simulator / Загвар

Бодолтоо шалгах зорилгоор та бүхэнд Одометрийн загвар өгөгдсөн байгаа. Бөгөөд уг загварт өөрийн програм болон хүснэгтийн хэмжээг өгч таны бичсэн зааврын дагуу | Одеметр хэрхэн ажиллаж байгааг харах боломжтой. Одометр програм дээр дурьдсан илгээх файлтай ижил форматаар бичигдэнэ.

Хүснэгтийн дүрслэл нь дараахь хэлбэрээр өгөгдөнө. Файлын мөр бүрт R, C and P 3 тоо байна энэ нь R нь мөрний C баганад P хайрга байна гэсэн утгыг илэрхийлнэ. Ингэж өгөгдөөгүй бусад нүднүүд нь хайрга агуулаагүй гэсэн үг.Жишээ нь дараахь файл нь:

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

Энэ файлаар дүрслэгдсэн хүснэгтэнд нийт 15 хайрга байх бөгөөд 3 нь (0, 10) нүдэнд 12 нь (4, 5) нүдэнд байрласан байна.

Уг загварчлалыг ашиглан програмаа шалгахын тулд бодлогын даалгавар бүхий хавтсан б у й `simulator.py` гэсэн файлыг дуудаж ажиллуулна.Дуудахдаа өөрийн шалгах програм бүхий файлаа аргумент болгон дамжуулна. Загварчлагч нь дараахи командуудыг хүлээн авах боломжой. Үүнд:

- `-h` нь боломжит сонголтуудыг харуулна
- `-g GRID_FILE` нь хүснэгтийн тодорхойлолтыг GRID_FILE файлаас уншина (default: empty grid);

- `-s GRID_SIDE` нь хүснэгтийн хэмжээг GRID_SIDE x GRID_SIDE гэж тогтооно (default: 256,бодлогын даалгаварт заасанчлан); жижиг хэмжээтэй хүснэгт ашиглан хариугаа шалгах боломжтой.

- `-m STEPS` гэдэг нь гүйцэтгэгдэх алхмуудын тоог хамгийн ихдээ STEPS байна гэж хязгаарлана;

- `-c` compiledах хэлбэр нь адилхан хэдий ч ажиллах файл нь C байх бөгөөд ажиллуулах хурд нь илүү их байна. Чиний програм 10 000 000-аас олон алхам хийх үед энэ командыг хэрэглэвэл илүү хурдан байх болно.

Илгээлтийн тоо

Энэхүү бодлогыг хамгийн ихдээ 128 удаа илгээж /submit/ хийж болно.