Előzmények

Frissítés

Commit dátum	Commit msg	Level 1	Level 2	Level 3	Level 4	Level 5
11-01 13:35:33	3 decimals	32195	32196	32197	32198	32199
11-01 13:30:58	SUCCESS	32186	32187	32188	32189	32190
10-31 16:00:23	talan jo	30557	30558	30559	30560	30561
10-29 18:14:46	not good	23882	23884	23888	23891	23894
10-29 01:43:23	test	12142	12143	12145	12147	12149

Ericsson Programozó Bajnokság 2020

I. forduló

v1.1.0

Bevezető - Vírusirtók

GAIA világában sajnos egy makacs vírusos betegség ütötte fel a fejét és aktív terjedéssel fertőzi a lakosságot már 1 éve. Az országok összefogtak és közösen, mesterséges intelligenciát is felhasználva rekord idő alatt fejlesztették ki a hatékony vakcinát. A vakcina összetétele nyilvános mindenkinek és az országok el is kezdték gyártani. Azonban a gyártásnak vannak kapacitáskorlátai, a fertőzött kerületek (nagyobb területegységek) nem gyárthatnak vakcinát a karantén miatt. A vírus gyorsan terjed és országod vezetői a Te csapatodat bízták meg, hogy fejlesszetek olyan MI alapú megoldást, mely a korlátozottan rendelkezésre álló vakcinát a leghatékonyabban alkalmazza, minél több kerület felszabadítása érdekében, lehetőleg a többi országnál gyorsabban. A vírusmentesített kerületekben újabb gyártókapacitásokat nyerünk, a végső cél pedig, hogy a többi országgal együttműködve vírusmentessé tegyétek a világunkat!
Az általatok kifejlesztendő mesterséges intelligenciának a vírus jövőbeli terjedésének megjósolásával és a kapacitásának helyes csoportosításával meg kell tisztítania GAIA világát. A Ti algoritmusotoknak milyen gyorsan, és milyen hatékonysággal sikerül ezt megtenni?

A tudósok rájöttek a vírus terjedésének véletlenszerűségi faktoraira, ami segítségével előre megjósolható az, hogy a jelenlegi fertőzöttségi adatok mellett mikor, és hogyan fog továbbterjedni a vírus. **Az első fordulóban a programotoknak ezen szimulációt kell megvalósítania.**

Programlogika

A szimuláció egy négyzetrácsos pályán játszódik, ahol a négyzetrácsok kerületekbe csoportosulnak. Minden négyzetrács egy-egy logikai területet jelöl, ahol mérve van a vírus mennyisége. Ezen területeket kell vírusmentessé tenni a csapat vakcináival, hogy a teljesen megtisztított kerületeken újabb vakcinagyártási lehetőséget, és biztonságot nyerjetek.
A koordináták (y,x) bal felső saroktól, (0,0)-val kezdődnek.
A vírus kezdetben minden kerületben megjelenik. Az élszomszédos, még nem teljesen megtisztított kerületű területekre terjedhet, az adott hely vírusmennyiségétől és determinisztikus továbbterjedési véletlenszerűségétől függően. A fertőzöttséget egy 0-100 -as skálán jelöljük.

A szimuláció menete

A visszacsatolás körökre bontott, minden körben a következő lépések történnek:

1. Az országok a vakcináikat elhelyezik/átcsoportosítják az általuk meghatározott területekre
2. A vakcinák beadása a fertőzötteknek, és azok kontaktjainak. Majd ugyanitt **a fertőzöttek esetleges gyógyulása.**

3. A teljesen felszabadított kerületek biztonsága aktiválódik, a vakcina gyártása megkezdődik. A kerületben lévő területi maradék vakcinák visszakerülnek az ország tartalékai közé
4. **A vírus terjed**
5. A legyártott vakcina kapacitások, levonva a kerületi védekezési vakcinákat, hozzáadódnak az országok tartalékaihoz

Az első fordulóban, a szimulációnál csak a 2-es pont vége, és a 4-es pont játszódik le. A többi pont a II. fordulón részletesebben lesz kifejtve. Érdemes már ez szerint a kódot fejleszteni, mert későbbiekben a többi akciót is nektek kell szimulálni.

Az elemek

- egy ország rendelkezik:
  - azonosítóval (ID), egész
  - össz gyártókapacitással (amiből le van vonva a kerületi védekezési vakcinaszám) (TPC), egész
  - tartalék vakcinákkal (RV), egész
  - elfoglalt kerületi azonosító listával (ASID), egész számok
- egy terület rendelkezik:
  - kerület azonosítóval, egész
  - fertőzöttség mutatóval (`0-100`), egész
  - beoltottság/gyógyultság mutatóval (`0-100`), egész
  - lakottsági mutatóval (1-5), egész
    azaz egy egységnyi vakcina mekkora fertőzöttséget képes kezelni ezen a területen (1 = nagyváros, 5 = tanya)
  - országonként tárolt vakcina darabszám, számpárok listája (országazonosító + darabszám (pozitív egész))

Véletlen faktorok

A kezdeti üzenetben küldeni fog a szerver 4 véletlen faktor seed-et, ami a faktorok első véletlen száma is lesz.

Bármelyik faktor véletlenszám-generáláshoz használva van, a következő algoritmust kell rá utána lefuttatni, így folyamatosan változik!: `factor = factor * 48271 % 0x7fffffff`, ahol a műveletek előjeletlen, 64 bites számábrázolásban értendőek.

A fertőzöttek gyógyulása

Néha, ha egy területen régebb óta van fertőzés, akkor van esély arra, hogy a fertőzöttségi mutató csökken, míg a gyógyultság mutató nő egy területen. Ennek kiszámolásához a következő képletet kell alkalmazni oszlop és sorfolytonosan ((`0`, `0`), (`0`, 1), (`0`, 2) ... (1, `0`), (1, 1), (1, 2) ...) :

```
healing(curr_tick, coord) => width + height < curr_tick
? floor(min(i = [1 .. width + height],
tick_info[curr_tick - i, coord].infection_rate) *
(factor1() % 10) / 20.0) : 0
```

szavakkal: Az előző tickek (pályaméret width + height darabszámú) fertőzöttségi mutatóinak minimuma szorozva az első véletlen faktor 10-zel való osztási maradékával (`0`-9). Az eredmény osztva 20-al, és ennek az alsó egészrésze. Ha még nem értük el a width + height -edik kört, akkor `0`.

Ha egy területen az infection_rate == `0`, akkor a healing() függvényt azon a területen nem kell kiszámítani!

A vírus-terjedési szabályok

Vírus csak már fertőzött területen, illetve annak közvetlen élszomszédos területén jelenhet meg. Ez hozzáadódik az adott terület fertőzöttségi mutatójához. A kiszámolásának képlete oszlop és sorfolytonosan ((0, 0), (0, 1), (0, 2) ... (1, 0), (1, 1), (1, 2) ...) :

```
infection(curr_tick, coord) => ceil((avg(i = [1 ..
mini(factor2() % 10 + 10, curr_tick)], infection(curr_tick - i,
coord)) +
    sum(c = {coord, neighbours(coord)}; t = factor3() % 7 + 3,
tick_info[curr_tick-1, c].infection_rate >
        (start_info[coord].district != start_info[c].district ?
2 :
            coord != c ? 1 : 0) * t ?
       clamp(start_info[coord].population -
start_info[c].population, 0, 2) + 1 : 0)) * (factor4() % 25 +
50) / 100.0)
```

a "rekurzív" függvényhívás miatt szükség van az első elemre, ami a következőképpen néz ki: `infection(0, coord) => tick_info[0, coord].infection_rate > 0 ? 1 : 0` Maguk az értékek nem változnak körről körre, azok kimenetét el kell menteni. Azaz nem kell minden egyes körben újra kiszámolni az `infection(1, ...)` -t, azt elegendő csak az adott tick-ben.

szavakkal: A második véletlen faktor 10-zel való osztási maradéka + 10 darab előző vírusterjedés átlaga az adott cellán. Ehhez hozzáadva az adott cella és a szomszédjai által meghatározandó átfertőződési mutató összeget. Az átfertőződési mutató kiszámolása előtt a `t` átfertőződési hajlandóságot generáljuk a harmadik véletlen faktor 7-tel való osztási maradéka + 3 -mal. Az átfertőződési mutató egy összeg, aminek kiszámolásához adjuk össze a hely populációs különbségéből adódó, 1-3 érték közé beszorított fertőzési lehetőséget, ha az előző körös fertőzöttség nagyobb, mint a hajlandóság és a távolság szorzata. A távolság helyben 0, megegyező kerületben 1, egyébként 2.

Az így eddig kiszámolt összeget megszorozzuk a negyedik véletlen faktor 25-tel való osztási maradéka + 50-nel, és az egészet leosztjuk 100-al, majd vesszük a felső egészrészét.

Ez hozzáadódik a fertőzöttségi mutatóhoz, de **a fertőzöttségi, illetve gyógyultsági mutató összege maximálisan 100 lehet**, így az `infection` függvény visszatérési értéke maximalizálva van az adott körre, 100 - fertőzöttségi mutató - gyógyultsági mutató.

Ha egy területre nem terjedhet vírus a kerülete tisztasága miatt, akkor az ő infection() függvényét nem kell kiszámítani!

Protokoll

A GIT-be feltöltött programkódnak a standard inputján és a standard outputján keresztül kell kommunikálnia, a következő protokoll szerint:

Minden üzenetet egy pontot, és csak azt tartalmazó sor követ.

Kezdeti üzenet

A kezdeti üzenetben a webes felületen található tokent, illetve opcionálisan az általatok gondolt kezdeti konfiguráció seed-jét lehet megadni, ami egy egész szám (ez felel a determinisztikus visszajátszhatóságért). Ez az üzenet kezdetben már be van égetve a kódotokba.

```
START <team-token:string> <seed:int>
.
```

Sikeres csatlakozás esetén

Ha sikeresen elindul egy játék, akkor ezen játék kezdeti konfigurációját és paramétereit kapjátok meg. Erre a programotoknak nem kell válaszolnia. Az üzenet elküldése után a játékszerver vár 2 másodpercet, majd utána küldi az első kör információjának adatát.

```
START <game-id> <max-tick-id> <countries-count>
FACTORS <factor-1> <factor-2> <factor-3> <factor-4>
FIELDS <rows> <columns>
FD 0 0 <district> <inf_rate> <popul>
FD 0 1 <district> <inf_rate> <popul>
FD 0 2 <district> <inf_rate> <popul>
<...>
FD 1 0 <district> <inf_rate> <popul>
FD 1 1 <district> <inf_rate> <popul>
FD 1 2 <district> <inf_rate> <popul>
<...>
FD 2 0 <district> <inf_rate> <popul>
FD 2 1 <district> <inf_rate> <popul>
FD 2 2 <district> <inf_rate> <popul>
<...>
.
```

ahol

- `<game-id>` a játék egyedi azonosítója, egész
- `<max-tick-id>` ennyi kört fogunk leszimulálni, egész
- `<countries-count>` ennyi ország vesz részt a vakcinaosztásban. Ez jelenleg `0`.
- `FACTORS` utáni paraméterek, seed alapján kigenerált 4 véletlen szám, `0` és 2147483647 között.
- `FIELDS` utáni paraméterek a négyzetrács nagyságát jelölő számok (sorok, oszlopok száma), egészek. Ez után a területeket leíró számhármasok következnek, `<rows>*<columns>` darab
- `district` az adott mező melyik kerülethez tartozik, egész
- `inf_rate` az adott mező kezdeti fertőzöttségi mutatója
- `popul` az adott mező lakottsági mutatója

Minden kör elején

Egy lekérés megy felétek, hogy hanyadik kör információját kell visszaszolgáltatnotok a szervernek.

```
REQ <game-id> <tick-id> <country-id>
.
```

Itt a `<country-id>` jelenleg `0`.

A kérések tick szerint szigorúan monoton növekvő sorrendben fognak érkezni (`0`-val kezdve). Lehetnek olyan esetek, amikor kimaradhat 1-1 tick lekérdezés.

Válaszlehetőségek

Minden körben pontosan 1 üzenetet küldhet el a programotok. A `RES` üzenet után a `REQ` utáni 3 számnak kell szerepelnie.

Utána táblázatszerűen fel kell sorolnotok az összes terület fertőzöttségi mutatóját. A `0`. körben nincs más teendőtök mint a kezdeti üzenetben felsorolt mezők fertőzöttségi mutatóit visszaküldítek.

```
RES <game-id> <tick-id> <country-id>
<0-0-infection-rate> <0-1-infection-rate> <0-2-infection-rate>
<...>
<1-0-infection-rate> <1-1-infection-rate> <1-2-infection-rate>
<...>
<2-0-infection-rate> <2-1-infection-rate> <2-2-infection-rate>
<...>
<...>
.
```

Hiba, időtúllépés

Ha bármilyen hibás kérés érkezik a szerver felé (sikertelen csatlakozást is beleértve), vagy időtúllépés történik (**2 másodperc**nél tovább gondolkozik a programotok **egy körön belül**), akkor a szerver a következő üzenet után bontja a kapcsolatot (azaz bezárja a program standard inputját).

```
WRONG <reason:string>
.
```

Van egy **globális timeout** is, ami **3 perc**. Ha az alatt nem fejeződik be a szimuláció, akkor automatikusan bontja a kapcsolatot a szerver, és az addig lejátszott szimuláció sikertelen lesz!

A szimuláció vége

A szimuláció többféleképpen végződhet:

- a fent már kifejtett `WRONG` üzenettel.
- a sikeres szimulációt követő `SUCCESS` üzenettel.
- a sikertelen szimulációt követő `FAILED` üzenettel.

Tesztelés

A felület "Információk" fülön leírt 3 módszerrel lehet tesztelni a megoldásotokat.

Ha legalább egyszer sikerül megoldani egy GIT-re feltöltött, (és fordult) kóddal az 5 általunk választott tesztesetre a szimulálást, akkor továbbjutottatok. Ez után is lehet nyugodtan továbbra is tesztelni/fejleszteni a kódotokat.

Példa szimuláció

Beletettük a faktorokat minden kör előtt, segítségként.

```
START 1 44 1
FACTORS 1569741360 1785505948 516548029 1302116447
FIELDS 6 4
FD 0 0 1 0 5
FD 0 1 1 1 1
FD 0 2 1 0 5
FD 0 3 3 0 5
FD 1 0 0 0 5
FD 1 1 1 0 4
FD 1 2 3 2 1
FD 1 3 3 0 2
FD 2 0 0 1 1
FD 2 1 2 0 5
FD 2 2 3 0 5
FD 2 3 5 0 5
FD 3 0 0 0 2
FD 3 1 2 1 1
FD 3 2 2 0 4
FD 3 3 5 2 1
FD 4 0 0 0 5
FD 4 1 2 0 5
FD 4 2 5 0 4
FD 4 3 5 0 3
FD 5 0 4 0 5
FD 5 1 4 1 1
FD 5 2 4 0 2
FD 5 3 4 0 5
.
REQ 1 0 0
.
<FACTORS BEFORE GEN: 1569741360, 1785505948, 516548029,
1302116447>
RES 1 0 0
0 1 0 0
0 0 2 0
1 0 0 0
0 1 0 2
0 0 0 0
0 1 0 0
.
REQ 1 1 0
.
<FACTORS BEFORE GEN: 1569741360, 1022357306, 413011888,
1815589382>
RES 1 1 0
0 3 0 0
0 0 4 0
3 0 0 0
0 3 0 4
0 0 0 0
0 3 0 0
.
REQ 1 2 0
.
<FACTORS BEFORE GEN: 1569741360, 6014329, 2054533564,
1227144796>
RES 1 2 0
0 5 0 0
0 0 6 0
5 0 0 0
0 5 0 6
0 0 0 3
0 5 0 0
.
REQ 1 3 0
.
<FACTORS BEFORE GEN: 1569741360, 383700820, 340745821,
445273369>
RES 1 3 0
0 7 0 0
0 0 8 2
```

```
7 0 0 2
0 7 0 8
0 2 0 6
3 7 0 0
.
REQ 1 4 0
.
<FACTORS BEFORE GEN: 1569741360, 1880906878, 484187638,
2082349629>
RES 1 4 0
3 9 2 0
3 5 11 4
9 0 3 4
2 9 0 10
0 4 2 9
5 9 2 0
.
REQ 1 5 0
.
<FACTORS BEFORE GEN: 1569741360, 617664374, 161743089,
395524291>
RES 1 5 0
6 11 7 0
6 11 14 7
11 2 6 8
6 11 3 12
0 9 5 13
8 11 5 0
.
REQ 1 6 0
.
<FACTORS BEFORE GEN: 1569741360, 1548699835, 347865686,
823536191>
RES 1 6 0
10 14 14 2
10 15 16 10
14 5 10 12
10 14 6 15
0 14 8 18
12 14 8 0
.
REQ 1 7 0
.
<FACTORS BEFORE GEN: 1569741360, 479599705, 9473732, 163190269>
RES 1 7 0
15 18 19 6
16 22 22 13
17 12 14 16
14 17 12 19
3 18 11 23
16 17 11 4
.
REQ 1 8 0
.
<FACTORS BEFORE GEN: 1569741360, 1435262923, 2002862586,
1845173464>
RES 1 8 0
20 23 25 10
21 29 27 17
21 19 18 22
19 21 17 23
7 23 15 28
20 22 15 9
.
REQ 1 9 0
.
<FACTORS BEFORE GEN: 1569741360, 283118234, 686628436,
959159834>
RES 1 9 0
25 27 33 15
28 36 32 21
```

```
26 28 24 29
24 26 25 27
12 30 20 34
26 27 19 15
.
REQ 1 10 0
.
<FACTORS BEFORE GEN: 1569741360, 632131996, 841847314,
556384508>
RES 1 10 0
30 31 41 19
35 43 37 26
31 38 31 36
30 32 33 31
16 39 26 40
31 31 23 21
.
REQ 1 11 0
.
<FACTORS BEFORE GEN: 547987797, 998305616, 168760390,
544770571>
RES 1 11 0
35 36 48 23
42 53 43 31
36 48 40 45
35 37 40 35
22 46 34 47
37 36 28 26
.
REQ 1 12 0
.
<FACTORS BEFORE GEN: 1422926888, 126569080, 1674030537,
389222294>
RES 1 12 0
41 41 57 28
48 60 49 37
42 59 48 53
41 41 47 38
27 55 41 53
42 41 32 32
.
REQ 1 13 0
.
<FACTORS BEFORE GEN: 2115025341, 158253104, 85701296,
788374264>
RES 1 13 0
48 46 66 33
56 70 55 42
45 68 57 61
46 48 55 43
32 63 49 58
48 45 37 38
.
REQ 1 14 0
.
<FACTORS BEFORE GEN: 1100958129, 861029523, 344770213,
1113065638>
RES 1 14 0
54 49 75 38
63 79 56 48
49 77 63 69
52 51 64 47
38 70 57 64
53 48 42 44
.
REQ 1 15 0
.
<FACTORS BEFORE GEN: 720931591, 279139747, 165402561,
634913238>
RES 1 15 0
58 51 83 45
```

```
70 84 59 53
52 89 71 76
58 53 71 48
43 79 65 66
60 53 48 50
.
REQ 1 16 0
.
<FACTORS BEFORE GEN: 1991168147, 750228336, 1542347140,
2038793561>
RES 1 16 0
64 53 90 50
74 96 62 57
52 99 79 81
61 60 79 52
49 88 72 69
66 53 50 57
.
REQ 1 17 0
.
<FACTORS BEFORE GEN: 1167823981, 230423476, 1025452634,
520016254>
RES 1 17 0
65 56 95 56
76 88 67 60
58 95 81 89
62 65 86 52
56 91 75 73
73 54 53 65
.
REQ 1 18 0
.
<FACTORS BEFORE GEN: 1177475263, 899903720, 526939350,
1505538801>
RES 1 18 0
70 57 89 59
85 86 67 61
62 91 88 90
64 68 87 57
63 87 77 75
75 54 53 68
.
REQ 1 19 0
.
<FACTORS BEFORE GEN: 869467872, 581233639, 1231733022,
2105150094>
RES 1 19 0
73 53 81 59
85 77 63 62
60 80 81 77
60 74 84 50
65 87 84 72
76 59 50 73
.
REQ 1 20 0
.
<FACTORS BEFORE GEN: 605679653, 333122171, 2024958096,
1710010668>
RES 1 20 0
77 57 65 67
84 75 64 62
61 65 77 74
65 76 73 46
73 70 87 56
77 56 48 75
.
REQ 1 21 0
.
<FACTORS BEFORE GEN: 362652637, 1037269486, 460621826,
1648688280>
RES 1 21 0
```

```
80 56 61 73
74 57 49 65
59 63 69 59
57 66 71 46
74 66 87 42
78 45 44 76
.
REQ 1 22 0
.
<FACTORS BEFORE GEN: 383321049, 1190622791, 1818412007,
1048797851>
RES 1 22 0
67 51 50 76
74 38 30 71
62 40 55 54
48 48 60 51
71 66 85 36
62 33 50 69
.
REQ 1 23 0
.
<FACTORS BEFORE GEN: 1925102331, 961036415, 219588534,
472065370>
RES 1 23 0
46 46 30 81
55 23 24 62
47 38 42 33
32 48 49 48
74 51 68 27
53 28 39 62
.
REQ 1 24 0
.
<FACTORS BEFORE GEN: 1761994407, 563819020, 192601107,
1089131195>
RES 1 24 0
44 33 18 82
36 14 24 48
43 38 38 19
29 39 37 42
60 51 60 17
40 27 38 47
.
REQ 1 25 0
.
<FACTORS BEFORE GEN: 303682121, 264071746, 1337958090,
1059774864>
RES 1 25 0
29 30 13 62
33 14 23 48
33 31 23 19
25 28 36 30
43 51 36 17
28 22 21 31
.
REQ 1 26 0
.
<FACTORS BEFORE GEN: 688061941, 1896528991, 183857767,
1570412676>
RES 1 26 0
28 17 12 57
19 14 22 48
24 19 20 11
15 23 20 18
31 39 22 10
27 13 16 19
.
REQ 1 27 0
.
<FACTORS BEFORE GEN: 712249430, 395690865, 1581272840,
1607088193>
```

```
RES 1 27 0
27 10 12 32
11 10 16 29
24 13 15 9
11 13 13 13
28 38 17 9
17 9 16 12
.
REQ 1 28 0
.
<FACTORS BEFORE GEN: 1824523820, 2133272645, 24923383,
1393102470>
RES 1 28 0
27 9 12 23
8 10 16 16
14 13 15 7
11 8 13 13
16 25 16 6
11 9 16 12
.
REQ 1 29 0
.
<FACTORS BEFORE GEN: 448389383, 1695505075, 779311082,
16549957>
RES 1 29 0
27 9 9 23
7 10 12 16
12 12 11 5
8 7 12 11
9 20 14 4
10 8 12 11
.
REQ 1 30 0
.
<FACTORS BEFORE GEN: 309031009, 20138740, 457163966,
1819280559>
RES 1 30 0
23 7 8 23
5 6 12 15
11 12 10 5
6 7 9 11
9 20 13 3
10 8 7 10
.
REQ 1 31 0
.
<FACTORS BEFORE GEN: 801903598, 1237817943, 1438423953,
971277097>
RES 1 31 0
13 7 6 13
4 6 12 9
9 10 6 4
6 7 8 9
5 12 8 3
10 8 5 8
.
REQ 1 32 0
.
<FACTORS BEFORE GEN: 843488910, 1305092661, 1496260088,
1221083010>
RES 1 32 0
8 6 6 8
4 5 10 9
9 8 4 3
4 5 6 9
3 8 7 3
6 8 5 8
.
REQ 1 33 0
.
<FACTORS BEFORE GEN: 378525122, 1713042792, 443035652,
```

```
1434736320>
RES 1 33 0
8 5 4 8
3 3 10 5
9 7 3 3
3 5 6 8
3 8 4 3
4 8 5 5
.
REQ 1 34 0
.
<FACTORS BEFORE GEN: 1443802580, 1887177537, 1327271472,
93169310>
RES 1 34 0
8 3 3 6
3 3 9 5
9 4 3 3
3 4 5 6
3 6 3 3
4 8 5 4
.
REQ 1 35 0
.
<FACTORS BEFORE GEN: 598920298, 1352700103, 299325157,
1703332290>
RES 1 35 0
8 3 3 6
2 2 9 4
6 4 3 2
2 3 5 5
3 5 2 3
3 8 4 3
.
REQ 1 36 0
.
<FACTORS BEFORE GEN: 2063463349, 1724264693, 808160597,
939782278>
RES 1 36 0
5 3 3 6
2 2 5 3
4 3 3 2
2 3 3 5
2 4 2 3
2 6 4 3
.
REQ 1 37 0
.
<FACTORS BEFORE GEN: 843876830, 2023823035, 1658827459,
1858500576>
RES 1 37 0
3 3 2 5
2 2 4 3
4 2 3 2
2 2 2 4
2 3 2 3
2 4 4 2
.
REQ 1 38 0
.
<FACTORS BEFORE GEN: 949848945, 1775582037, 997925887,
1315571789>
RES 1 38 0
3 2 2 3
2 2 4 3
3 2 3 2
2 2 2 3
2 3 2 3
2 4 4 2
.
REQ 1 39 0
.
```

```
<FACTORS BEFORE GEN: 1284357518, 493170796, 1085477718,
81895802>
RES 1 39 0
3 2 2 3
2 2 3 3
3 2 2 2
2 2 2 3
2 3 2 3
2 3 4 2
.
REQ 1 40 0
.
<FACTORS BEFORE GEN: 1764633585, 1740844305, 1427300617,
911089236>
RES 1 40 0
2 2 2 3
2 2 2 3
2 2 2 2
2 2 2 2
2 3 2 2
2 3 3 2
.
REQ 1 41 0
.
<FACTORS BEFORE GEN: 31835747, 1002495777, 971977884,
2095126362>
RES 1 41 0
2 2 2 2
2 2 2 3
2 2 2 2
2 2 2 2
2 3 2 2
2 2 3 2
.
REQ 1 42 0
.
<FACTORS BEFORE GEN: 543441795, 1839128744, 1261144868,
1211118132>
RES 1 42 0
2 2 2 2
2 2 2 3
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
.
REQ 1 43 0
.
<FACTORS BEFORE GEN: 311896565, 2112409359, 1996022692,
49058578>
RES 1 43 0
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
2 2 2 2
.
SUCCESS
.
```