



## International Olympiad in Informatics 2013

6-13 July 2013

Brisbane, Australia

Day 2 tasks

**game**

magyar — 1.1

Bazza és Shazza egy  $R \times C$  méretű játéktáblán játszik. A sorok sorszámai fentről lefelé:  $0, \dots, R - 1$ , az oszlopok sorszáma balról jobbra:  $0, \dots, C - 1$ . A  $(p, q)$  számpár jelöli a tábla  $p$ . sorában és  $q$ . oszlopában levő mezőt. Minden mezőben egy nemnegatív egész szám van, kezdetben mindegyik 0.

Bazza kétféle lépést tehet:

- módosítja a  $(p, q)$  mező értékét;
- megkéri Shazzát, hogy számítsa ki a  $(p, q)$  bal felső,  $(u, v)$  jobb alsó mezőkkel megadott blokkban levő számok legnagyobb közös osztóját.

Bazza legfeljebb  $N_U + N_Q$  lépést tesz ( $N_U$  módosítás és  $N_Q$  kérdés) a játék során.

Feladatod megadni a helyes válaszokat a kérdésekre.

## Példák

Legyen  $R = 2$  és  $C = 3$ , és Bazza módosításokkal kezd:

- A  $(0, 0)$  mező módosítása 20-ra;
- A  $(0, 2)$  mező módosítása 15-re;
- A  $(1, 1)$  mező módosítása 12-re.

20	0	15
0	12	0

A tábla a képek megfelelő. Bazza két kérdést tesz fel:

- A  $(0, 0)$  és  $(0, 2)$  ellentétes sarkú blokkra: A blokkban 3 szám van: 20, 0 és 15, legnagyobb közös osztójuk 5.
- A  $(0, 0)$  és  $(1, 1)$ : ellentétes sarkú blokkra: A blokkban 4 szám van: 20, 0, 0 és 12, legnagyobb közös osztójuk 4.

Most Bazza módosít két mezőt:

- A  $(0, 1)$  mező módosítása 6-ra;
- A  $(1, 1)$  mező módosítása 14-re.

20	6	15
0	14	0

Az új tábla látható a képen. Bazza megint két kérdést tesz fel:

- A  $(0, 0)$  és  $(0, 2)$  ellentétes sarkú blokkra: A blokkban 3 szám van: 20, 6 és 15, legnagyobb közös osztójuk 1.
- A  $(0, 0)$  and  $(1, 1)$  ellentétes sarkú blokkra: A blokkban 4 szám van: 20, 6, 0 és 14, legnagyobb közös osztójuk 2.

Azaz Bazza összesen  $N = 9$  lépést tett a játékban.

## Megvalósítás

Az `init()` és `update()` eljárások és a `calculate()` függvény megvalósítását tartalmazó fájlt kell beküldened.

A gépeden található (`game.c`, `game.cpp` és `game.pas`) tartalmaz egy `gcd2(X, Y)` függvényt, amely  $X$  és  $Y$  legnagyobb közös osztóját számítja ki. Ha  $X = Y = 0$ , akkor `gcd2(X, Y)` eredménye `0` lesz.

A függvény elég gyors ahhoz, hogy maximális pontszámot kapjál, futási ideje arányos  $\log(X + Y)$ -nal.

**A eljárásod: `init()`**

C/C++ `void init(int R, int C);`

Pascal `procedure init(R, C : LongInt);`

**Leírás**

A beküldésed tartalmazza ezt az eljárást!

Az eljárás megkapja a tábla méretét, ebben inicializálhatod a globális változókat és adatszerkezeteket. Egyszer hívják meg, az `update()` és a `calculate()` hívások előtt.

**Paraméterek**

- `R` : a sorok száma.
- `C` : az oszlopok száma.

**Az eljárásod: `update()`**

C/C++ `void update(int P, int Q, long long K);`

Pascal `procedure update(P, Q : LongInt; K : Int64);`

**Leírás**

A beküldésed tartalmazza ezt az eljárást!

Ezt az eljárást hívják, amikor Bazza módosítja egy mező értékét.

**Paraméterek**

- `P` : a mező sorindexe ( $0 \leq P \leq R - 1$ ).
- `Q` : a mező oszlopindexe ( $0 \leq Q \leq C - 1$ ).
- `K` : a mező új értéke ( $0 \leq K \leq 10^{18}$ ).

## A függvényed: `calculate()`

C/C++ `long long calculate(int P, int Q, int U, int V);`

Pascal `function calculate(P, Q, U, V : LongInt) : Int64;`

### Leírás

A beküldésed tartalmazza ezt az eljárást!

A függvény számítja ki a  $(P, Q)$  és  $(U, V)$  ellentétes sarkú blokkban levő számok legnagyobb közös osztóját. A blokkban a sarokpontok is benne vannak.

Ha minden mező értéke 0, akkor 0 értéket adjon vissza!

### Paraméterek

- $P$ : a blokk bal felső cellája sorindexe ( $0 \leq P \leq R - 1$ ).
- $Q$ : a blokk bal felső cellája oszlopindexe ( $0 \leq Q \leq C - 1$ ).
- $U$ : a blokk jobb alsó cellája sorindexe ( $P \leq U \leq R - 1$ ).
- $V$ : a blokk jobb alsó cellája oszlopindexe ( $Q \leq V \leq C - 1$ ).
- *Visszatérési érték*: a blokkban levő számok legnagyobb közös osztója, vagy 0, ha a blokkban minden szám 0.

## Mintapélda

A példa leírása:

Függvényhívás	Visszatérési érték
<code>init(2, 3)</code>	
<code>update(0, 0, 20)</code>	
<code>update(0, 2, 15)</code>	
<code>update(1, 1, 12)</code>	
<code>calculate(0, 0, 0, 2)</code>	5
<code>calculate(0, 0, 1, 1)</code>	4
<code>update(0, 1, 6)</code>	
<code>update(1, 1, 14)</code>	
<code>calculate(0, 0, 0, 2)</code>	1
<code>calculate(0, 0, 1, 1)</code>	2

## Korlátok

- Időlimit: Lásd a részfeladatoknál!
- Memória limit: Lásd a részfeladatoknál!
- $1 \leq R, C \leq 10^9$
- $0 \leq N \leq 10,000$
- $0 \leq K \leq 10^{18}$ , ahol  $K$  a módosítás harmadik paramétere.

## Részfeladatok

A részfeladat paramétereiket nézd meg az angol leírásban!

Részfeladat	Pontszám	R	C	$N_u$	$N_q$	Időlimit	Memória limit

## Gyakorlás

A mintaértékelő a `game.in` fájlból olvassa a bemenetet, a következő formában:

- 1. sor: `R C N`
- következő `N` sor mindegyike egy művelet leírását tartalmazza, a végrehajtásuk sorrendjében:

A művelet leírása:

- `update(P, Q, K)` esetén: `1 P Q K`
- `calculate(P, Q, U, V)` esetén: `2 P Q U V`

A példának az alábbi fájl felel meg:

```
2 3 9
1 0 0 20
1 0 2 15
1 1 1 12
2 0 0 0 2
2 0 0 1 1
1 0 1 6
1 1 1 14
2 0 0 0 2
2 0 0 1 1
```

## Nyelvi előírások

**C/C++** Importáld a `#include "game.h"`-t.

**Pascal** Definiáld a `unit Game`-t. A tömbök indexelése `0`-tól kezdődik (nem `1`-től).

A mezőkben levő számok nagyon nagyok lehetnek, ezért C/C++ esetén a `long long` típust, Pascal esetén a `Int64` típust kell használni.