International Olympiad in Informatics
2013
6-13 July 2013
Brisbane, Australia
Day 2 tasks

game

Japanese — 1.1

Bazza と Shazza がゲームで遊んでいる．$0, \cdots, R-1$ の番号がつけられた R 行と $0, \cdots, C-1$ の番号がつけられた C 列からなるグリッド状のマス目があり，P 行 Q 列のマスを (P, Q) と表す．各マスには非負整数が 1 個書かれており，それらはゲーム開始時にはすべて 0 である．

ゲームは次のように進む．Bazza は任意のタイミングで，以下のいずれかの行動をとる：

- マス (P, Q) に書かれている整数を任意の値に更新する．
- Shazza に，(P, Q) と (U, V) を向かい合う隅としてもつ長方形内にある整数すべての最大公約数 (GCD) を計算するよう質問する．この長方形は (P, Q) と (U, V) を含む．

Bazza は $N_U + N_Q$ 回の行動 (N_U 回の更新と N_Q 回の質問) をとる．その後はこのゲームに飽きてしまい，クリケットをしに外へ行ってしまうだろう．

あなたの課題は，質問に対する正しい答えを求めることである．

例 (Example)

$R = 2$，$C = 3$ とする．まず，Bazza が以下の更新を行う：

- マス $(0, 0)$ に書かれた整数を 20 に更新する．
- マス $(0, 2)$ に書かれた整数を 15 に更新する．
- マス $(1, 1)$ に書かれた整数を 12 に更新する．

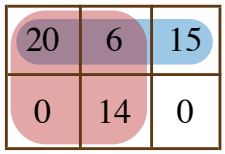
20	0	15
0	12	0

このときマス目は上の図のようになる．そして，Bazza は次のような長方形における GCD の計算を要求する：

- 向かい合う隅が $(0, 0)$ と $(0, 2)$：この長方形内には 3 個の整数 20, 0, 15 があり，それらの GCD は 5 である．
- 向かい合う隅が $(0, 0)$ と $(1, 1)$：この長方形内には 4 個の整数 20, 0, 0, 12 があり，それらの GCD は 4 である．

次に，Bazza が以下の更新を行う：

- マス $(0, 1)$ に書かれた整数を 6 に更新する．
- マス $(1, 1)$ に書かれた整数を 14 に更新する．



今度はマス目は上の図のようになる．Bazza は次のような長方形における GCD の計算をもう一度要求する：

- 向かい合う隅が $(0, 0)$ と $(0, 2)$：この長方形内には 3 個の整数 20, 6, 15 があり，これらの GCD は 1 である．
- 向かい合う隅が $(0, 0)$ と $(1, 1)$：この長方形内には 4 個の整数 20, 6, 0, 14 があり，これらの GCD は 2 である．

この例では，Bazza は $N_U = 5$ 回の更新と $N_Q = 4$ 回の質問を行った．

実装 (Implementation)

あなたは，プロシージャー `init()`，`update()` と関数 `calculate()` を実装し，それらを含む 1 つのファイルを提出しなければならない．これらの関数およびプロシージャーの仕様は以下のとおりである．

なお，あなたを補助するために，あなたの計算機上に与えられている解法テンプレート (`game.c`，`game.cpp`，`game.pas`) にはそれぞれ関数 `gcd2(X, Y)` が含まれている．この関数は与えられた2つの非負整数 `X` と `Y` の GCD を計算する．もし $X = Y = 0$ ならば，`gcd2(X, Y)` の戻り値も `0` となる．

この関数は $\log(X + Y)$ に比例する時間以下で実行でき，満点を取るのに十分なほど速い．

あなたのプロシージャー (Your Procedure): `init()`

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

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

説明 (Description)

あなたの提出は，このプロシージャーを実装していること．

このプロシージャーにはグリッドのサイズが与えられ，あなたは任意のグローバル変数とデータ構造を初期化することができる．このプロシージャーは，`update()`，`calculate()` のどの呼び出しよりも前に1回だけ呼び出される．

引数 (Parameters)

- `R`：行の数．
- `C`：列の数．

あなたのプロシージャー (Your Procedure): `update()`

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

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

説明 (Description)

あなたの提出は，このプロシージャーを実装していること．

このプロシージャーは，Bazza がマスに書かれている整数を更新するときに呼び出される．

引数 (Parameters)

- `P`：マスの行番号 ($0 \le P \le R-1$)．
- `Q`：マスの列番号 ($0 \le Q \le C-1$)．
- `K`：そのマスに書かれる新しい整数 ($0 \le K \le 10^{18}$)．更新前の値と同じかもしれない．

あなたの関数 (Your Function): `calculate()`

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

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

説明 (Description)

あなたの提出は，このプロシージャーを実装していること．

この関数は `(P, Q)` と `(U, V)` を向かい合う隅としてもつ長方形の中にあるすべての整数の GCD を計算しなければならない． この長方形はマス `(P, Q)` と `(U, V)` を含む．

もしこの長方形に含まれるすべての整数が `0` であるなら，この関数の戻り値は `0` である．

引数と戻り値 (Parameters)

- `P`: 長方形の左上の隅のマスの行番号 (`0 ≤ P ≤ R - 1`).
- `Q`: 長方形の左上の隅のマスの列番号 (`0 ≤ Q ≤ C - 1`).
- `U`: 長方形の右下の隅のマスの行番号 (`P ≤ U ≤ R - 1`).
- `V`: 長方形の右下の隅のマスの列番号 (`Q ≤ V ≤ C - 1`).
- **Returns** (戻り値): 長方形の中にあるマス目に書かれたすべての整数の GCD．ただし，これらの整数がすべて `0` ならば，`0` を返すこと．

やりとりの例 (Sample Session)

次のやりとりは，上に述べた例を表す．

関数呼び出し	戻り値
`init(2, 3)`	
`update(0, 0, 20)`	
`update(0, 2, 15)`	
`update(1, 1, 12)`	
`calculate(0, 0, 0, 2)`	`5`
`calculate(0, 0, 1, 1)`	`4`
`update(0, 1, 6)`	
`update(1, 1, 14)`	
`calculate(0, 0, 0, 2)`	`1`
`calculate(0, 0, 1, 1)`	`2`

制限 (Constraints)

- 時間制限: "小課題"の項目を参照せよ．
- メモリ制限: "小課題"の項目を参照せよ．
- $1 \leq R, C \leq 10^9$
- $0 \leq K \leq 10^{18}$．ここで K は Bazza がマスに書き込む任意の整数である．

小課題 (Subtasks)

小課題のパラメーターについては，英語版の問題文を参照せよ．

小課題	得点	R	C	N_U	N_Q	時間制限	メモリ制限

試行 (Experimentation)

与えられる採点プログラムのサンプルは，ファイル `game.in` から入力を読み込む．このファイルは次のフォーマットで書かれていなければならない：

- 1 行目： `R C N`
- 続く `N` 行: 1 行ごとに 1 つの行動．ただし，Bazza が行動する順番どおりに記述すること．

それぞれの行動を表す行は次のうちのいずれかのフォーマットで書かれていなければならない：

- `update(P, Q, K)` を表すには，`1 P Q K` とする．
- `calculate(P, Q, U, V)` を表すには，`2 P Q U V` とする．

例えば，上の例は次のフォーマットで与えられる：

```
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
```

言語に関する注意 (Language Notes)

C/C++ `#include "game.h"` を行うこと．

Pascal `unit Game` を定義すること．すべての配列は `0` から始まる (`1` からではない)．

各マスに書かれる整数は非常に大きくなりうる．C/C++ を使う参加者は `long long` 型を使うことが推奨される．Pascal を使う参加者は `Int64` 型を使うことが推奨される．