Problema: TRI
Triangles

italian

CEOI 2018, giorno 2. Memoria disponibile: 256 MB. *16.08.2018*

Byteland è un bel paese con n ($n \geq 3$) città, rappresentate da n punti distinti nel piano 2D. Le città sono numerate da 1 a n. Visto che sei un turista non conosci l'esatta posizione delle città in Byteland. Da una guida turistica hai imparato che non ci sono 3 città allineate.

L'inviluppo convesso di un insieme di n punti è un poligono convesso con la minore area possibile tale che tutti gli n punti sono dentro o sul bordo di tale poligono.

Un poligono convesso ha tutti gli angoli minori di 180 gradi e non può auto-intersecarsi.

Il tuo compito è di trovare il numero di vertici nel bordo dell'inviluppo convesso dell'insieme delle città di Byteland. Puoi solo effettuare richieste su triple di numeri di città **distinti** i, j, k ($1 \leq i, j, k \leq n$). Tali domande riguardano un triangolo con vertici nelle città i, j, k. La risposta alla domanda indica se l'attraversamento delle città del triangolo in ordine i, j, k è orario o antiorario.

Libreria per la comunicazione

Il tuo programma deve usare una libreria che permette di effettuare domande e comunicare la risposta finale.

La libreria (`trilib.h` per C e C++) fornisce le seguenti funzioni:

- `int get_n();`
  Ritorna il numero di città.
- `bool is_clockwise(int i, int j, int j);`
  Ritorna `true` se i vertici del triangolo i, j, k ($1 \leq i, j, k \leq n, i \neq j \neq k \neq i$) sono in ordine orario, `false` se sono in ordine antiorario.
- `void give_answer(int s);`

Per Java, la classe `trilib` fornisce i seguenti metodo:

- `static public int get_n();`
- `static public boolean is_clockwise(int a, int b, int c);`
- `static public void give_answer(int s);`

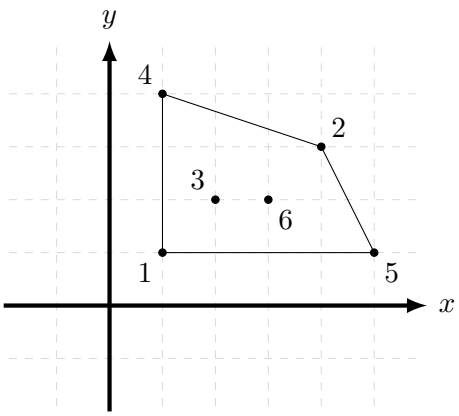
Dopo che il tuo programma chiama `give_answer` non può più effettuare altre richieste. Il programma deve chiamare `give_answer` esattamente una volta.

In questo problema non è consentito leggere da standard input nè scrivere nello standard output. Dopo aver chiamato `give_answer` il tuo programma deve terminare immediatamente.

Puoi assumere che la posizione delle città è stabilita in anticipo e non cambierà durante l'esecuzione del programma (cioè la libreria si comporta in modo completamente deterministico). Per esempio, nell'input d'esempio (vedi sotto) chiamare `give_answer(4)` e immediatamente uscire passerebbe il test case. Al tuo programma è concesso indovinare la soluzione senza essere sicuro.

Interazione d'esempio

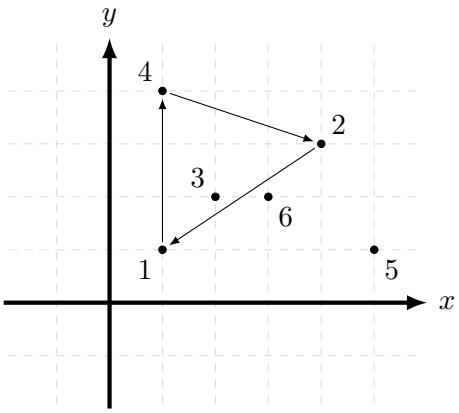
Considera $n = 6$ città disposte in $(1, 1)$, $(4, 3)$, $(2, 2)$, $(1, 4)$, $(5, 1)$, $(3, 2)$ come mostrato in figura. L'inviluppo complesso è segnato da delle linee. Contiene 4 vertici nel suo bordo, quindi la risposta è 4.



La seguente tabella mostra un esempio di interazione con la libreria che corrisponde a questo esempio.

Chiamata	Valore di ritorno
`get_n()`	6
`is_clockwise(1, 4, 2)`	`true`
`is_clockwise(4, 2, 1)`	`true`
`is_clockwise(1, 2, 4)`	`false`
`is_clockwise(3, 6, 5)`	`true`
`give_answer(4)`	–

La figura sottostante mostra il triangolo della prima richiesta. Le città 1, 4, 2 sono in ordine orario, quindi il valore ritornato è `true`.



Valutazione

L'insieme dei test è diviso nei seguenti subtask con limitazioni aggiuntive. I test in ogni subtask consistono in uno o più gruppi di test. Ogni gruppo di test contiene uno o più test case.

In tutti i test $3 \leq n \leq 40\,000$. Puoi chiamare `is_clockwise` al più 1 000 000 volte per test case.

Subtask	Limitazioni	Punti
1	$n \leq 50$	15
2	$n \leq 500$	20
3	$n \leq 15\,000$	20
4	al più un punto non è nel bordo dell'inviluppo convesso	20
5	nessuna limitazione aggiuntiva	25

Esperimenti

Nella cartella `public` è presente un esempio di libreria che ti permette di provare la correttezza della tua soluzione. La libreria legge la descrizione di Byteland dallo standard input secondo il seguente formato:

- nella prima riga un intero n, il numero di città,
- nelle successive n righe: due interi ciascuna, le coordinate della i-esima città.

La libreria fornita **non** controlla la correttezza della soluzione. In più non controlla nemmeno la correttezza dell'input. Non sarà la stessa del sistema di valutazione nel server.

Un input d'esempio per la libreria è fornito nel file `tri0.in`.

Dopo che `give_answer` viene chiamata la libreria stampa la soluzione e il numero di chiamate a `is_clockwise` nello standard output.

Per compilare la soluzione con la libreria d'esempio puoi usare i seguenti comandi:

- **C**: `gcc -O2 -static trilib.c tri.c -lm -std=gnu99`
- **C++**: `g++ -O2 -static trilib.c tri.cpp -lm -std=c++11`

Per Java non è necessario usare comandi speciali per compilare la soluzione con la libreria.

I file con la soluzione e la libreria devono essere nella stessa cartella.