Zadanie: TRI
Trójkąty

polish

CEOI 2018 Warsaw POLAND

CEOI 2018, dzień 2. Dostępna pamięć: 256 MB. *16.08.2018*

Bajtocja jest przepięknym krajem z n ($n \geq 3$) miastami, reprezentowanymi jako n różnych punktów na płaszczyźnie. Miasta sa ponumerowane od 1 do n. Jako turysta, nie znasz dokładnego położenia miast w Bajtocji. Z turystycznej broszury dowiedziałeś się, że żadne trzy miasta nie są współliniowe.

Otoczką wypukłą zbioru n punktów nazywamy wielokąt wypukły o najmniejszym możliwym polu, który zawiera wszytkie n punktów wewnątrz, bądź na brzegu tego wielokąta. Wielokąt wypukły ma wszystkie kąty mniejsze niż 180 stopni oraz nie może zawierać samoprzecięć.

Twoim zadaniem jest znalezienie liczby punktów na brzegu otoczki wypukłej zbioru miast Bajtocji. Możesz jedynie zadawać pytania o trójki **różnych** numerów miast i, j, k ($1 \leq i, j, k \leq n$). Takie pytanie rozważa trójkąt o wierzchołkach w miastach i, j, k. odpowiedź na takie pytanie wskazuje, czy przechodzenie wierzchołków trójkąta w kolejności i, j, k następuje w kierunku zgodnym, czy przeciwnym do ruchu wskazówek zegara.

Komunikacja

Twój program powinien używać biblioteki, która umożliwia zadawanie pytań oraz udzielanie odpowiedzi.

Biblioteka (`trilib.h` dla C i C++) udostępnia następujące funkcje:

- `int get_n();`
  Zwraca liczbę miast.
- `bool is_clockwise(int a, int b, int c);`
  Zwraca `true`, jeżeli wierzchołki trójkąta a, b, c ($1 \leq a, b, c \leq n, a \neq b \neq c \neq a$) dane są w kierunku zgodnym z ruchem wskazówek zegara oraz `false`, jeżeli są dane przeciwnie do ruchu wskazówek zegara.
- `void give_answer(int s);`

Dla Javy, klasa `trilib` udostępnia następujące metody:

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

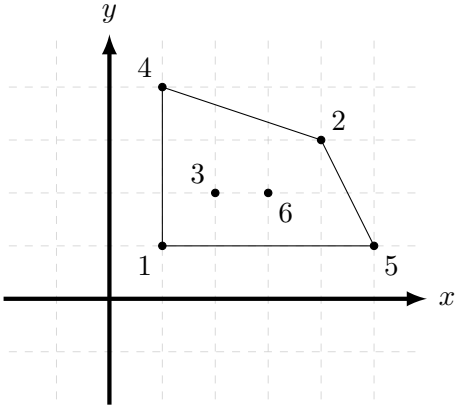
Po tym, jak Twój program wywoła `give_answer`, nie może już zadać więcej pytań. Funkcja `give_answer` powinna zostać wywołana dokładnie raz.

W tym problemie, nie możesz czytać ze standardowego wejścia, ani wypisywać nic na standardowe wyjście. Po wywołaniu `give_answer`, Twój program powinien niezwłocznie się zakończyć.

Możesz założyć, że lokalizacje punktów są ustalone z góry i nie będą zmieniać się podczas wywołania programu (to jest, biblioteka działa w całkowicie deterministyczny sposób). Dla przykładu, w teście przykładowym (patrz niżej), wywołanie `give_answer(4)` i niezwłoczne zakończenie programu, powinno przejść ten test. Twój program może spróbować zgadnąć odpowiedź, nie będąc jej pewien.

Przykładowa komunikacja

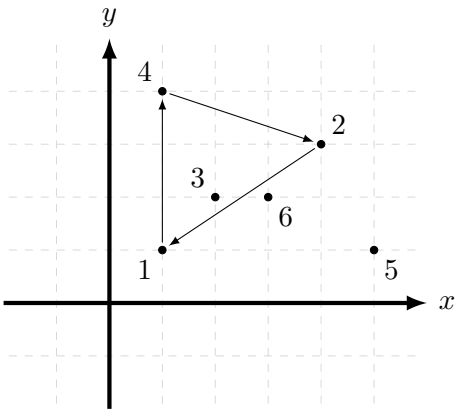
Rozważmy $n = 6$ miast w punktach $(1, 1)$, $(4, 3)$, $(2, 2)$, $(1, 4)$, $(5, 1)$, $(3, 2)$, tak jak na rysunku poniżej. Otoczka wypukła jest zaznaczona odcinkami. Zawiera ona cztery punkty na brzegu, stąd odpowiedź to 4.



Poniższa tabela pokazuje przykładową interakcję z biblioteką, która odpowiada temu przykładowi.

Wywołanie	Zwrócona wartość
`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)`	–

Rysunek poniżej pokazuje trójkąt z pierwszego zapytania. Miasta 1, 4, 2 są w kolejności zgodnej z ruchem wskazówek zegara, stąd zwrócona wartość to `true`.



Ocenianie

Zestaw testów dzieli się na następujące podzadania z dodatkowymi ograniczeniami. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów. Każda grupa testów może zawierać jeden lub wiele testów.

We wszystkich testach $3 \leq n \leq 40\,000$. W każdym przypadku testowym funkcję `is_clockwise` możesz wywołać co najwyżej 1 000 000 razy.

Podzadanie	Ograniczenia	Punkty
1	$n \leq 50$	15
2	$n \leq 500$	20
3	$n \leq 15\,000$	20
4	co najwyżej jeden punkt nie leży na brzegu otoczki wypukłej	20
5	brak dodatkowych ograniczeń	25

Eksperymenty

W folderze `public` znajduje się przykładowa biblioteka pozwalająca na testowanie formalnej poprawności Twojego rozwiązania. Biblioteka czyta opis Bajtocji ze standardowego wejścia w następującym formacie:

- w pierwszym wierszu liczba całkowita n, liczba miast,
- w kolejnych n wierszach: każdy po dwie liczby całkowite, współrzędne i-tego miasta.

Udostępniona biblioteka **nie sprawdza w żaden sposób** poprawności Twojego rozwiązania. Nie sprawdza także poprawności wejścia. Nie jest taka sama, jak (sekretna) biblioteka na serwerze.

Przykładowe wejście dla biblioteki jest dane w pliku `tri0.in`.

Po wywołaniu `give_answer`, biblioteka wypisuje daną odpowiedź oraz liczbę wywołań `is_clockwise` na standardowe wyjście.

Aby skompilować rozwiązanie z biblioteką przykładową, możesz użyć następujących komend:

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

Dla Javy nie potrzebujesz specjalnej komendy, aby skompilować rozwiązanie z biblioteką.

Pliki z rozwiązaniem i biblioteką powinny być w tym samym folderze.