Piotr Chrząstowski
Treść zadania, Opracowanie

Rafał Rusin
Program

Wyspy

Bajtocja jest oblana oceanem. Na jej terenie znajdują się jeziora. Na tych jeziorach wyspy, na tych wyspach zdarzają się dalsze jeziorka, a na nich wysepki i tak dalej. Ocean ma stopień zero. Bajtocja, jako wyspa ma stopień 1. Jeziora na wyspach Bajtocji stopień 2, itd., czyli jezioro ma stopień $w+1$, jeśli znajduje się na wyspie stopnia w, a wyspa ma stopień $j+1$, jeśli znajduje się na jeziorze stopnia j. Wynika stąd oczywiście, że wszystkie stopnie wysp są nieparzyste, a jezior i oceanu parzyste.

Wszystkie jeziora i wyspy mają linie brzegowe w kształcie wielokątów o prostopadłych kolejnych bokach (równoległych do osi układu współrzędnych), a ich wierzchołki mają współrzędne całkowite. Żadne dwie linie brzegowe nie przecinają się, ani nie stykają się.

Mając dane kontury linii brzegowych, wyznacz maksymalny stopień wyspy/jeziora w Bajtocji.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia opisy linii brzegowych wysp i jezior,*
- *obliczy maksymalny stopień jeziora/wyspy,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu wejścia zapisana jest jedna dodatnia liczba całkowita n, liczba linii brzegowych, $1 \leqslant n \leqslant 40000$. Linie brzegowe są opisane w kolejnych wierszach, po jednej w wierszu. Każdy z tych wierszy zawiera nieujemne liczby całkowite pooddzielane pojedynczymi odstępami. Pierwsza liczba w wierszu to k, parzysta liczba punktów tworzących linię brzegową, $4 \leqslant k \leqslant 10000$. Dalej w wierszu znajduje się k liczb: $x_1, x_2, \ldots, x_k$, $0 \leqslant x_i \leqslant 10^8$. Kolejne punkty tworzące linię brzegową to: (x_1, x_2), (x_3, x_2), (x_3, x_4), (x_5, x_4), $\ldots$ (x_{k-1}, x_k), (x_1, x_k). Są podane we współrzędnych kartezjańskich oraz opisują linię brzegową lewoskrętnie (czyli idąc z punktu i do $i+1$, wnętrze mamy po lewej stronie). Linie brzegowe są podane w takiej kolejności, że:

- *linia brzegowa każdego jeziora jest podana zawsze po linii brzegowej wyspy, na której się znajduje,*
- *linia brzegowa każdej wyspy jest podana zawsze po linii brzegowej jeziora, na którym się znajduje.*

Do opisania całej mapy użyto nie więcej niż 200000 punktów.

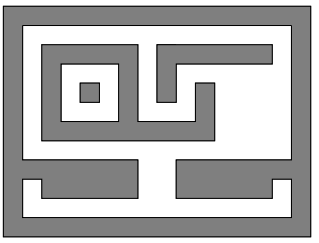
Wyjście

Twój program powinien wypisać w pierwszym i jedynym wierszu wyjścia jedną liczbę całkowitą: maksymalny stopień jeziora/wyspy.

Przykład

Dla danych wejściowych:

```
6
4 1 0 17 12
16 10 4 16 11 2 4 8 2 3 3 2 1 16 3 15
2
8 8 10 3 5 12 8 11 6
4 4 6 7 9
4 6 8 5 7
6 10 9 15 10 9 7
```



poprawnym wynikiem jest:

```
5
```

Rozwiązanie

Zróbmy kilka początkowych spostrzeżeń. Po pierwsze możemy się skupić jedynie na odcinkach poziomych. Ponieważ każdy koniec odcinka pionowego jest jednocześnie końcem pewnego odcinka poziomego to skupiając się tylko na odcinkach poziomych nie przeoczymy żadnego fragmentu brzegu. Patrząc tylko na odcinki poziome, ponumerowane w kolejności ich występowania na brzegu, możemy sobie w jednoznaczny sposób odtworzyć położenie brakujących odcinków pionowych.

Kolejnym spostrzeżeniem jest to, że jeśli z lewego końca pewnego odcinka spojrzymy w górę, to albo nie zobaczymy już żadnej linii brzegowej i wtedy ten odcinek jest brzegiem Bajtocji i ma stopień 1, albo zobaczymy brzeg bezpośrednio nad nami. Są wtedy trzy przypadki:

- Odcinek, który zobaczyliśmy bezpośrednio nad nami należy do tego samego brzegu, co nasz odcinek i wtedy oczywiście ograniczamy ten sam obszar.
- Odcinek ten należy do innego brzegu i jest skierowany na prawo. Wtedy oznacza to, że jego wnętrze jest po jego drugiej stronie i tamten odcinek należy do brzegu innego obszaru, ale mającego ten sam stopień, co nasz obszar.
- Odcinek ten należy do innego brzegu i jest skierowany na lewo. Wtedy oznacza to, że jesteśmy podobszarem obszaru otoczonego tamtym brzegiem, zatem nasz stopień będzie o 1 większy.

W rzeczywistości można nawet nieco uprościć rozumowanie: to, czy odcinek sąsiedni z góry jest z tego samego brzegu nie ma znaczenia, liczy się tylko jego zwrot: jeśli jest w prawo, to nie zmieniamy stopnia, jeśli jest w lewo, to nasz stopień będzie o 1 większy.

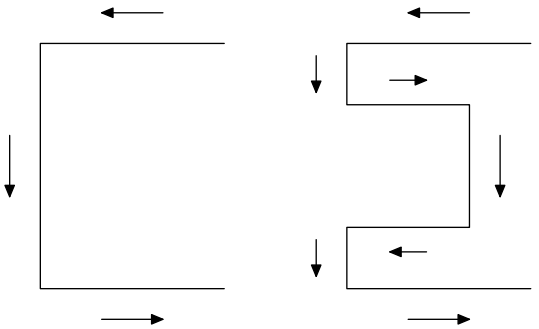
Formalny dowód tych spostrzeżeń wymaga dość głębokiej wiedzy z dziedziny topologii i wykracza poza program szkolny; na szczęście są to fakty na tyle intuicyjne, że można je w miarę sensownie uzasadnić.

Zrobimy to metodą indukcyjną. Każdą linię brzegową można przedstawić jako kształt powstały z prostokąta przez wciskanie albo wyciskanie z niego kawałków brzegu. Przy każdym takim wciśnięciu lub wyciśnięciu dodajemy prostopadłe krawędzie łączące wciśnięty odcinek z resztą brzegu (lub w razie potrzeby usuwamy kawałki brzegu, jeśli nie ma z czym łączyć). Dowód teraz polega na tym, żeby pokazać, że dla linii brzegowych będących prostokątami zachodzi teza (baza indukcji), a następnie, że pojedyncze wciśnięcie odcinka nie zmienia prawdziwości tezy. Indukcja będzie zatem względem liczby wciśnięć potrzebnych do uzyskania zadanego kształtu linii brzegowych.

To, że dla linii brzegowych będących prostokątami teza zachodzi, powinno być dość oczywiste. Prosta indukcja względem liczby prostokątów powinna nas o tym przekonać i tę część uzasadnienia pominiemy.

Wracając do uzasadnienia naszej tezy zauważmy, że wciśnięcie odcinka poziomego nie zmienia jej prawdziwości: nie może się on wszakże przeciąć z żadnym innym odcinkiem, więc jego sąsiedztwo z góry się nie zmieni. Pozostanie on też sąsiadem dla swoich „kolegów" z dołu. Nikt się między nich „nie wetnie" i nie rozerwie tej zależności między stopniami, którą ustaliliśmy na podstawie założenia indukcyjnego.

W przypadku odcinków pionowych dodane nowe poziome krawędzie mogą przysłonić sąsiedztwo pionowe dla odcinków, które znajdują się na dole. Ze względu na symetrię sytuacji ustalmy, że „wciskamy" w stronę prawą odcinek znajdujący się na lewej krawędzi konturu, czyli krawędzi skierowanej do dołu — patrz rysunek poniżej. Dla pozostałych 3 przypadków, kiedy lewą krawędź konturu wciskamy w lewą stronę oraz gdy prawą krawędź konturu wciskamy w stronę prawą bądź lewą, rozumowanie przebiega analogicznie.



Kluczową obserwacją jest tu spostrzeżenie, że wciśnięcie takie zawsze prowadzi do sytuacji, w której nad odcinkiem skierowanym w prawo będzie odcinek skierowany w lewo, tak jak przed wciśnięciem, i nad każdym odcinkiem skierowanym w lewo będzie odcinek skierowany w prawo. Zatem w ramach jednego obszaru zawsze zwroty sąsiednich w pionie odcinków będą różne. Jeśli Czytelnik został tu przekonany, to dobrze, choć nie jest to ścisły dowód.

Natomiast przypadek, kiedy jeden obszar jest podobszarem drugiego rozpoznajemy po tym, że każdy odcinek podobszaru, jest jednakowo skierowany, jak dowolny jego sąsiad z

obszaru go otaczającego. Sąsiedztwo rozumiemy tu jako najbliższy odcinek przecięty przez prostą prostopadłą do badanego odcinka.

Aby zrealizować sprawdzenie przedstawionych warunków, zastosujemy narzucającą się tu technikę *zamiatania*, opisywaną już w materiałach z poprzednich olimipad[1]. Technika ta, bardzo typowa dla wielu zadań geometrii obliczeniowej, wprowadza porządek przy przeglądaniu badanych obiektów na płaszczyźnie i pozwala nie uronić żadnego z nich, organizując ich przetwarzanie w efektywny sposób. Interesujące nas poziome odcinki tworzące linie brzegowe posortujemy od lewej do prawej względem ich współrzędnych x-owych, a w przypadku równych współrzędnych x-owych, względem współrzędnych y-kowych od góry do dołu. Każdy odcinek będzie zatem występował dwukrotnie: raz reprezentowany przez lewą współrzędną, a raz przez prawą. Następnie pionową miotłą będziemy zamiatali płaszczyznę od lewej do prawej, pobierając (i w miarę potrzeby usuwając) odcinki po kolei, zgodnie z naszym porządkiem. W każdym momencie w pionowej miotle będą się znajdowały te odcinki, które ją aktualnie przecinają, a punktami zatrzymania się miotły w marszu od lewej do prawej będą x-owe współrzędne odcinków. Zauważmy, że zgodnie z warunkami zadania wszystkie współrzędne y znajdujące się w miotle muszą być różne.

Pierwszy odcinek, który napotkamy będzie oczywiście częścią linii brzegowej Bajtocji — najbardziej na zachód wysuniętym cyplem. Może zresztą być kilka równie daleko wysuniętych odcinków, co nam w niczym nie przeszkodzi; będziemy tylko pamiętać, żeby przetwarzać takie odcinki o równych lewych współrzędnych z góry na dół. Pierwszemu odcinkowi nadamy stopień $d = 1$ i włożymy do pustej miotły. Dalsze odcinki poziome będziemy przetwarzali zgodnie z następującą zasadą. Jeżeli napotkamy prawy koniec odcinka, to odnajdujemy ten odcinek w miotle i usuwamy. Jeżeli natomiast napotkamy lewy koniec odcinka, to patrzymy w górę. Niech s będzie najbliższym od góry sąsiadem badanego odcinka o znajdującym się nad jego lewym końcem. Niech brzeg, do którego należy odcinek s ma stopień d. Zachodzą teraz dwa przypadki:

- jeśli zwrot s jest taki sam, jak zwrot o (czyli na prawo), to umieszczamy w miotle odcinek o z numerem poziomu $d+1$. Oznacza to, że nasze wnętrze jest bezpośrednim podwnętrzem wnętrza ogarniętego przez brzeg, do którego należy odcinek s;
- jeśli zwrot s jest przeciwny do zwrotu odcinka o, to umieszczamy w miotle odcinek o z numerem poziomu d. Oznacza to, że nasze wnętrze jest wnętrzem sąsiednim do wnętrza ogarniętego przez brzeg, do którego należy odcinek s.

Pozostają jeszcze szczegóły implementacyjne. Miotła powinna się dać szybko zainicjalizować, a ponadto sprawnie wykonywać następujące operacje:

- odszukanie odcinka w miotle o największej współrzędnej nieprzekraczającej zadanego y, bądź stwierdzenie że takiego odcinka nie ma;
- wstawienie do miotły zadanego odcinka;
- usunięcie z miotły zadanego odcinka.

Musimy też pamiętać, żeby do miotły wstawiać odcinki z zaznaczeniem ich stopnia i zwrotu.

[1] Patrz „Łamane pł askie” —V OI, „Lodowisko” —VI OI, „Kopalnia zł ota” —VIII OI

Przyjmijmy następujące oznaczenia. Niech n oznacza liczbę linii brzegowych, k_i liczbę punktów tworzących i-tą linię brzegową, zaś $m = \sum_{i=1}^{n} k_i$ liczbę wszystkich punktów. Zauważmy, że w pesymistycznym przypadku liczba odcinków przechowywanych w miotle będzie rzędu m.

Jeśli zaimplementujemy miotłę na przykład w uporządkowanej tablicy, to choć samo wyszukiwanie odcinka względem posortowanych współrzędnych y zajmie nam czas rzędu $\log m$, to operacje wstawiania i usuwania odcinka w pesymistycznym czasie będą rzędu m. Przy $m/2$ odcinkach dostaniemy algorytm kwadratowy.

Podobne kłopoty będziemy mieli ze strukturami listowymi. Tam też nie sposób jest zrealizować w modelu list prostych wszystkich potrzebnych nam operacji w czasie mniejszym niż liniowy, a to oznacza, że koszt całego algorytmu też będzie kwadratowy.

Dobrą strukturą dla miotły są np. drzewa AVL, o których można się dowiedzieć np. z książki [14]. Implementacja ich jest jednak dość uciążliwa, choć niektórzy ćwiczą ją sobie w ramach przygotowań do olimpiady. Za pomocą drzew AVL każdą z potrzebnych operacji daje się zrealizować w pesymistycznym czasie rzędu $\log m$. Ponieważ dla każdego z m końców odcinków będziemy wykonywali każdą z tych operacji co najwyżej raz, więc łączny koszt tej fazy wynosi $O(m \log m)$. Wczytanie danych zajmuje czas liniowy ze względu na m, a początkowe posortowanie którymś z szybkich algorytmów sortujących daje się zrobić w czasie rzędu $m \log m$. Zatem taki jest rząd złożoności całego algorytmu. Oczywiście odcinki wkładamy do miotły z zaznaczeniem ich stopnia oraz zwrotu.

W programie wzorcowym, znajdującym się na płycie, zastosowany został wybieg implementacyjny znacznie upraszczający sprawę. Zastosowano bowiem strukturę `<map>` ze standardowej biblioteki STL języka `C++`. Tam po prostu za darmo dostajemy logarytmiczną implementację potrzebnych nam operacji. Oczywiście znajomość odpowiednich bibliotek oszczędzi nam też problemu szybkiego posortowania danych.

Testy

Opisy testów zawierają umowne nazwy figur.

nr	opis
0	przykładowy test z treści zadania
1	spirala z prostokącikami
2	mała szachownica
3	mały ciąg prostokątów zawartych w sobie - “rura”
4	spirala z prostokącikami
5	krzywa Hilberta st. 4 z prostokącikami
6	“drzewo” prostokątów
7	“rura”, w niej spirala, w spirali “rura”
8	duża szachownica, “klucz” i spirala
9	“klucz” i “drzewo” złożone z prostokątów, szachownic, spiral i “rura”
10	spirala, “rura” i klucz
11	“klucz” i spirala
12	“klucz” i spirala