Zajęcia 27

Temat: Algorytmy geometryczne

Czas trwania: 2x45 min

Cel zajęć:

projektuje i programuje proste problemy z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, tablice, rekurencję, pisze własne funkcje rekurencyjne, zbiory i operacje na zbiorach, struktury danych, biblioteka STL, algorytmy geometryczne: punkt, odcinek, prosta, wektor, wielokąt, iloczyn skalarny i wektorowy, otoczka wypukła, testuje poprawność programów dla różnych danych, posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Efekty:

- umie uruchomić potrzebne oprogramowanie,
- umie napisać program z wykorzystaniem algorytmów geometrycznych,
- zna iloczyn skalarny, wektorowy, otoczkę wypukłą

Formy i metody pracy: praca samodzielna, omówienie, wykład

Zadania do wykonania na zajęciach	Treści programowe
1. Bombardowanie	M.6, P.2.18, P.2.20, A.3.13,
2. Mur	M.6, P.2.18, P.2.20, A.3.13,

Materiały do zajęć:

http://informatykaplus.edu.pl/upload/list/czytelnia/Przeglad_podstawowych_algorytmow.pdf
ss. 29-32

Zadania do wykonania w domu:

Ołtarze (IV OI)

https://szkopul.edu.pl/problemset/problem/JR1lB-gdjNHcT5j3mtNRFhLa/site/

ZADANIA I ROZWIĄZANIA

Zadanie 1. Bombardowanie

Limit pamięci: 128MB

Stało się! Bitocja wypowiedziała wojnę Bajtocji! Wychodząc z założenia, że najlepszą formą obrony jest atak, już pierwszego dnia wojny Bajtocjanie wysłali swoje bombowce z bombami bitowymi nad sąsiednią wyspę. Bomby bitowe zmieniają wszystkie bity w urządzeniach elektronicznych o wartości 1 na losowe siejąc popłoch w dowództwie przeciwnika. Niestety, z powodu gęstej mgły piloci zrzucili swoje ładunki w różnych miejscach wyspy i teraz nie wiadomo, ile z nich na pewno trafiło w terytorium Bitocji. Ponieważ Bitocja ma kształt wielokąta wypukłego o znanych wierzchołkach, a Ty znasz miejsca zrzucenia ładunków, możesz określić tę liczbę!

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby całkowite n i m ($3 \le n, m \le 10^5$), odpowiednio liczba punktów określających kształt Bitocji oraz liczba bomb bitowych zrzuconych przez Bajtocję. W kolejnych n liniach znajdują się po dwie liczby całkowite x_i i y_i ($-10^9 \le x_i, y_i \le 10^9$) – współrzędne wierzchołków Bitocji. W następnych m liniach znajdują się po dwie liczby całkowite x_j i y_j ($-10^9 \le x_j, y_j \le 10^9$) – współrzędne zrzutów bomb bitowych.

Wyjście

Jedna liczba całkowita oznaczająca, ile bomb na pewno trafiło w terytorium Bitocji.

Przykład

Wejście	Wyjście
4 3 0 0 5 0 5 5 0 5 1 1 6 1 1 4	2

Rozwiązanie

Naszym zadaniem jest sprawdzenie, ile punktów leży wewnątrz wielokąta. Zauważmy, że wielokąt ten stanowi otoczkę wypukłą.

Rozwiązanie wolne: Dla każdego z punktów będziemy sprawdzać, czy leży on wewnątrz któregokolwiek z trójkątów zbudowanych z punktu pierwszego i każdej pary kolejnych dwóch wierzchołków wielokąta.

Rozwiązanie szybkie: Spróbujmy podzielić wielokąt na dwie podobnej wielkości części i sprawdźmy, w której z nich może być położony każdy sprawdzany punkt (na lewo bądź na prawo od prostej dzielącej nasz wielokąt). Podobnie postępujmy z nowym fragmentem tak długo, dopóki nie pozostanie nam jeden trójkąt (wyszukiwanie binarne po wyniku. Na koniec sprawdźmy, czy punkt ten leży wewnątrz trójkąta (suma pól trzech powstałych małych trójkątów, gdzie jednym z wierzchołków jest badany punkt, musi być równa polu „dużego" trójkąta).

```
det(punkt a, punkt b, punkt c) // iloczyn wektorowy
    zwróć (b.x-a.x)*(c.y-a.y) - (b.y-a.y)*(c.x-a.x)

wewnątrz_tr(punkt a, punkt b, punkt c, punkt x)
    jeżeli (|det(a,b,c)|=|det(a,b,x)|+|det(a,c,x)|+|(det(b,c,x)|
        zwróć PRAWDA
    zwróć FAŁSZ

punkt w[n] //wierzchołki wielokąta
wewnątrz (punkt x){
    L←1, P=n←1 //liczymy punkty od 0, ostatni jest n-1
    jeżeli
    jeżeli ( det(w[0],w[P],x)>0 ∨ det(w[0],w[L],x)<0) zwróć FAŁSZ
    // wyszukujemy binarnie
    dopóki (L<P-1)
        środek ← (L+P) div 2
        jeżeli (det(w[0],w[P],x)<=0 ∧ det(w[0],w[srodek],x)>=0)
            L←środek;
        przeciwnie P←środek;
    jeżeli (wewnątrz_tr(w[0],w[P],w[L],x)=PRAWDA) zwróć PRAWDA
    zwróć FAŁSZ
```

W głównej części programy wywołujemy funkcję `wewnątrz` dla każdej bomby i zliczamy.

Zadanie 2. Mur

Limit pamięci: 64MB

Kiedy w Bitomiu odkryto złoto, mieszkańcy miasta wpadli w popłoch. Do ich miasta zaczęły ciągnąć dziesiątki, później setki, a w końcu tysiące poszukiwaczy łatwego zarobku. Aby nie dopuścić do katastrofy, postanowiono ogrodzić Bitom murem. Jednocześnie postanowiono, że mur będzie możliwie najmniejszy.

Ktoś szybko naniósł współrzędne domów na mapę, ktoś drugi zaczął już wyrabiać cegły, inni zaczęli składać zamówienia na cement. Ale jaki ma być przebieg muru?

Zadanie

Napisz program, który:

- wczyta ze standardowego wejścia ilość domów i ich współrzędne,
- wyznaczy domy, obok których będzie bezpośrednio przebiegał mur,
- wypisze wynik na standardowe wyjście.

Wejście

W pierwszym wierszu standardowego wejścia zapisana jest jedna liczba całkowita n ($2 \le n \le 500\,000$) oznaczająca ilość domów. Kolejne n wierszy zawiera po dwie liczby całkowite a i b ($-10^9 \le a, b \le 10^9$) oznaczające współrzędne kolejnych domów. W części testów wartych około 40% punktów zachodzi warunek ($2 \le n \le 2000$) oraz ($-10000 \le a, b \le 10000$).

Wyjście

W pierwszym wierszu standardowego wyjścia Twój program powinien wypisać jedną liczbę całkowitą X, oznaczającą liczbę domów obok których stanie mur. W kolejnych X liniach znajdą się kolejne współrzędne domów. Jako pierwszy wypisz dom znajdujący się jak najbardziej na zachód. Jeśli jest takich kilka, wypisz ten, który znajduje się najbardziej na południu. Kolejne domy wypisuj w kolejności przeciwnej do ruchu wskazówek zegara.

Przykład

Wejście	Wyjście
10 1 4 3 3 4 2 3 5 1 2 5 4 1 0 6 6 5 3 6 4	8 1 0 4 2 5 3 6 4 6 6 3 5 1 4 1 2

Rozwiązanie

W zadaniu musimy wyznaczyć otoczkę wypukłą. Posortujmy wszystkie punkty od lewej do prawej (w przypadku takiej samej współrzędnej x posortujmy po y) w tablicy `punkty[]`. Dodajemy do otoczki dwa pierwsze punkty, a następnie dla wszystkich pozostałych punktów sprawdzamy, czy po ich dodaniu otoczka „skręca” w lewo (iloczyn wektorowy dla trzech ostatnich punktów musi być większa od 0). Jeżeli nie, usuwamy poprzednie punkty które w tym przeszkadzają. Działanie powtarzamy od prawej do lewej.

```
det(punkt a, punkt b, punkt c) // iloczyn wektorowy
    zwróć (b.x-a.x)*(c.y-a.y) - (b.y-a.y)*(c.x-a.x)

punkt otoczka[], punkty[]
buduj_otoczkę()
    otoczka[0] ← punkty[0]
    otoczka[1] ← punkty[1]
    ostatni ← 1 //ostatni dodany punkt do otoczki
    dla i=2,3,…,n-1 wykonuj
        dopóki (ostatni>0
        ∧ det(otoczka[ostatni-1],otoczka[ostatni],punkty[i])<0)
            ostatni ← ostatni - 1
        ostatni ← ostatni + 1
        otoczka[ostatni] ← punkty[i]
    dla i=n-2,n-3,…,0 wykonuj
        dopóki (ostatni>0
        ∧ det(otoczka[ostatni-1],otoczka[ostatni],punkty[i])<0)
            ostatni ← ostatni - 1
        ostatni ← ostatni + 1
        otoczka[ostatni] ← punkty[i]
```

Liczbę domów pamięta zmienna ostatni, wypisujemy zawartość tablicy `otoczka[]`.