Jakub Radoszewski — Treść zadania, Opracowanie
Anna Niewiarowska — Program

OI, Etap III, dzień pierwszy, 02.04.2008

Trójkąty

Mamy danych n parami różnych punktów na płaszczyźnie ($n \geqslant 3$). Istnieje $\frac{n \cdot (n-1) \cdot (n-2)}{6}$ trójkątów, których wierzchołkami są pewne parami różne spośród tych punktów (wliczając trójkąty zdegenerowane, tzn. takie, których wierzchołki są współliniowe).

Chcemy obliczyć sumę powierzchni wszystkich trójkątów o wierzchołkach w danych punktach.

Fragmenty płaszczyzny należące do wielu trójkątów liczymy wielokrotnie. Przyjmujemy, że powierzchnia trójkątów zdegenerowanych jest równa zero.

Zadanie

Napisz program, który:

- *wczyta ze standardowego wejścia współrzędne danych punktów na płaszczyźnie,*
- *wyznaczy sumę powierzchni wszystkich trójkątów o wierzchołkach w danych punktach,*
- *wypisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba całkowita n ($3 \leqslant n \leqslant 3\,000$), oznaczająca liczbę wybranych punktów. Kolejne n wierszy zawiera po dwie liczby całkowite x_i oraz y_i ($0 \leqslant x_i, y_i \leqslant 10\,000$), oddzielone pojedynczym odstępem i oznaczające współrzędne i-tego punktu (dla $i = 1, 2, \ldots, n$). Żadna para (uporządkowana) współrzędnych na wejściu nie powtarza się.

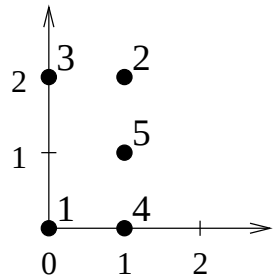
Wyjście

W pierwszym i jedynym wierszu wyjścia powinna się znajdować jedna liczba rzeczywista, równa sumie powierzchni wszystkich trójkątów o wierzchołkach w danych punktach. Wynik powinien być wypisany z dokładnie jedną cyfrą po kropce dziesiętnej i nie powinien się różnić od faktycznej wartości o więcej niż 0.1.

Przykład

Dla danych wejściowych:

```
5
0 0
1 2
0 2
1 0
1 1
```



poprawnym wynikiem jest:

```
7.0
```

Rozwiązanie

Rozwiązanie $O(n^3)$

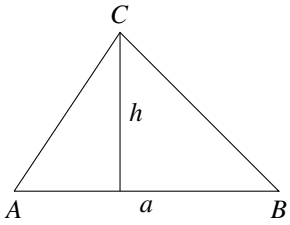
Zadanie polega na wyznaczeniu sumy pól wszystkich trójkątów, których wierzchołki pochodzą z zadanego zbioru n punktów. Najprostsze rozwiązanie to przejrzenie wszystkich trójkątów i zsumowanie ich pól — jego złożoność to $O(n^3)$. Pole trójkąta można policzyć na kilka sposobów.

Metoda I

Bodaj najbardziej znanym wzorem na pole trójkąta jest

$$S = \frac{ah}{2},$$

czyli połowa iloczynu długości podstawy i wysokości trójkąta. Podstawą trójkąta może przy tym być dowolny z jego boków.



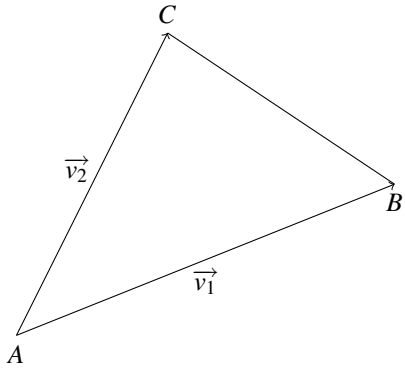
Metoda II

Aby policzyć pole trójkąta, można wybrać dwa jego boki i wyznaczyć połowę wartości bezwzględnej iloczynu wektorowego wektorów odpowiadających tym bokom:

$$S = \frac{|\overrightarrow{v_1} \times \overrightarrow{v_2}|}{2}.$$

Przypomnijmy, że wartość iloczynu wektorowego $\overrightarrow{v_1} = [x_1, y_1]$ oraz $\overrightarrow{v_2} = [x_2, y_2]$ to

$$\overrightarrow{v_1} \times \overrightarrow{v_2} = x_1 \cdot y_2 - x_2 \cdot y_1.$$

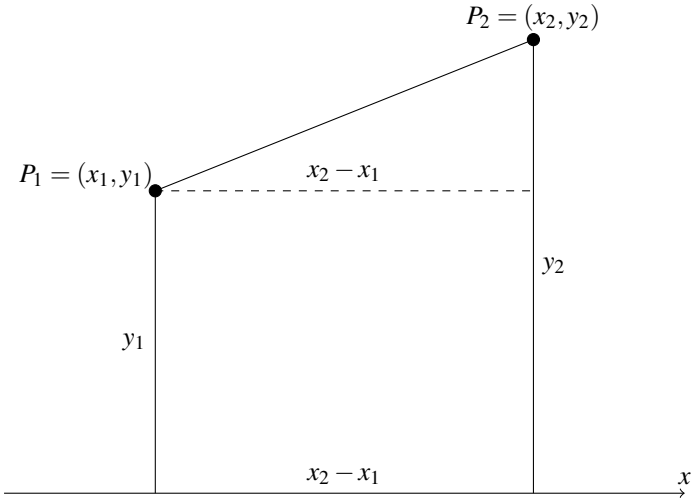


Metoda III

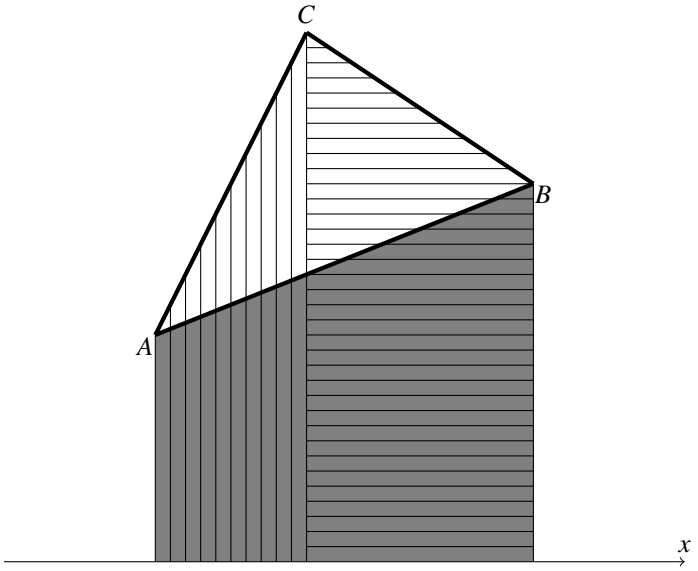
Pole trójkąta, podobnie jak i pole dowolnego innego wielokąta, można wyznaczyć za pomocą *metody trapezów*, czyli obchodząc boki trójkąta w kolejności zgodnej z kierunkiem ruchu wskazówek zegara i dodając do wyniku składowe odpowiadające kolejnym bokom. Składową dla boku łączącego punkty $P_1 = (x_1, y_1)$ i $P_2 = (x_2, y_2)$ jest „pole" trapezu ograniczonego tym bokiem, osią OX oraz liniami rzutów punktów P_1 i P_2 na oś OX. Wartość nazwana umownie „polem" wyraża się wzorem

$$\frac{(x_2 - x_1)(y_1 + y_2)}{2}$$

i może być ujemna (na przykład, jeżeli $x_1 > x_2$ oraz $y_1, y_2 > 0$).



Działanie metody trapezów dla całej figury ($\triangle ABC$) jest przedstawione na kolejnym rysunku:



Obchodząc ten trójkąt w kolejności zgodnej z kierunkiem ruchu wskazówek zegara, uzyskujemy „ścieżkę": $A \to C \to B \to A$. Składowe boków AC i CB (pola trapezów zakreskowanych na rysunku odpowiednio pionowo i poziomo) są dodatnie, natomiast składowa boku BA (zanegowane pole szarego trapezu na rysunku) jest ujemna. Zsumowanie wszystkich składowych daje więc w wyniku pole białego, zakreskowanego obszaru z rysunku, czyli faktycznie pole trójkąta $\triangle ABC$.

Jak to przyspieszyć?

Trzeba poszukać rozwiązania efektywniejszego niż działające w czasie $O(n^3)$, jeśli chcemy zmieścić się w limicie czasowym dla największych danych. Skoro sumowanie pól wszystkich trójkątów jest zbyt czasochłonne, to może powinniśmy spróbować zliczać pola trójkątów nie pojedynczo, a większymi partiami?

Możemy, na przykład, rozważyć wszystkie trójkąty zawierające jako bok określony odcinek AB. Wszystkich możliwych odcinków utworzonych z n punktów jest $O(n^2)$, więc daje to pewne szanse na rozwiązanie o lepszej złożoności niż poprzednie, jeśli tylko poradzimy sobie szybko z każdą grupą. Rzeczywiście, właśnie na tym pomyśle opierają się wszystkie rozwiązania wzorcowe niniejszego zadania[1].

Dla każdego *odcinka skierowanego* $\overrightarrow{AB}$, czyli wektora zaczepionego w pewnych dwóch spośród zadanych punktów, będziemy wyliczać jego *wagę* W_{AB}, czyli sumę zależnych od niego składników wchodzących w skład końcowego wyniku. Waga ta będzie miała

[1] Rozwiązania te różnią się przede wszystkim zastosowanym wzorem na pole trójkąta. Ponieważ każdy z Czytelników ma zapewne swój ulubiony, więc w dalszym ciągu opisu przedstawiamy „równolegle" rozwiązania wzorcowe dla każdej z trzech wcześniej przytoczonych metod wyznaczania pola trójkąta.

różną wartość i znaczenie w zależności od tego, z jakiej metody obliczania pola trójkąta zamierzamy skorzystać:

Metoda I — wagą odcinka skierowanego będzie suma pól trójkątów, w których jest on jednym z boków;

Metoda II — wagą wektora $\overrightarrow{AB}$ jest suma pól trójkątów, w których wektor $\overrightarrow{AB}$ pojawia się przy liczeniu pola za pomocą iloczynu wektorowego;

Metoda III — wagą wektora jest suma (po odpowiednich trójkątach) pól trapezów, które są ograniczone tym wektorem, osią OX i liniami rzutów punktów końcowych wektora na oś OX, z uwzględnieniem znaku; innymi słowy, będzie to iloczyn pola ze znakiem trapezu wyznaczonego przez ten wektor oraz liczby trójkątów, na których (odpowiednio skierowanym) obwodzie ten wektor występuje.

Zauważmy, że jeśli zsumujemy wagi W_{AB} dla wszystkich możliwych wektorów $A \neq B$, to pole każdego trójkąta policzymy kilkakrotnie. Dokładniej, w przypadku:

Metoda I — obwód każdego trójkąta składa się z trzech boków, z których każdy można skierować na dwa sposoby, a zatem pole trójkąta zostanie uwzględnione sześciokrotnie;

Metoda II — na sześć sposobów można wybrać parę boków skierowanych, czyli wektorów zaczepionych w jednym wierzchołku trójkąta (zauważ, że trzy spośród tych sześciu par będą różniły się od pozostałych trzech jedynie kolejnością wektorów);

Metoda III — skierowany zgodnie z kierunkiem ruchu wskazówek zegara obwód trójkąta składa się z 3 wektorów, stąd pole trójkąta zostanie wliczone do sumy końcowej trzykrotnie.

Skoro wiemy już, jak na podstawie sum wag wszystkich wektorów uzyskać wynik końcowy, to możemy zastanowić się, jak wyznaczyć W_{AB} w każdej z metod. Oznaczmy przez $P_1, P_2, \ldots, P_{n-2}$ wszystkie punkty dane w zadaniu, oprócz punktów A i B.

Metoda I — szukaną wagą odcinka skierowanego AB jest suma pól trójkątów:

$$W_{AB} = S(\triangle ABP_1) + S(\triangle ABP_2) + \ldots + S(\triangle ABP_{n-2}).$$

Wybierając w każdym z nich jako podstawę bok AB, dostajemy następujący wzór:

$$W_{AB} = \frac{ah_1}{2} + \frac{ah_2}{2} + \ldots + \frac{ah_{n-2}}{2} = \frac{a}{2}(h_1 + h_2 + \ldots + h_{n-2}),$$

gdzie $a = |AB|$, natomiast $h_1, h_2, \ldots, h_{n-2}$ oznaczają wysokości trójkątów $\triangle ABP_1, \triangle ABP_2, \ldots, \triangle ABP_{n-2}$ opuszczone na podstawę AB — inaczej mówiąc, są to odległości punktów $P_1, P_2, \ldots, P_{n-2}$ od prostej $pr(AB)$.

Jeżeli równanie tej prostej zapiszemy jako $ax + by + c = 0$, to odległość punktu $P_i = (x_i, y_i)$ od $pr(AB)$ wyraża się znanym wzorem:

$$h_i = d(P_i, pr(AB)) = \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}}.$$

Gdyby dla wszystkich punktów P_i wyrażenie $ax_i + by_i + c$ było dodatnie, to moglibyśmy bardzo ładnie uprościć wzór na W_{AB}, gdyż wówczas suma $h_1 + h_2 + \ldots + h_{n-2}$ wynosiłoby:

$$\frac{a(x_1 + \ldots + x_{n-2}) + b(y_1 + \ldots + y_{n-2}) + c}{\sqrt{a^2 + b^2}}. \tag{1}$$

Chociaż nie możemy liczyć na powyższy przypadek, to możemy jednak podzielić wszystkie punkty P_i na leżące po jednej stronie prostej $pr(AB)$ — gdzie rzeczywiście $ax_i + by_i + c \geqslant 0$, oraz na położone po drugiej stronie — gdzie $ax_i + by_i + c < 0$ (punkty leżące na prostej są i tak dla nas nieistotne, gdyż tworzą z A i B wyłącznie trójkąty zdegenerowane o zerowym polu). To spostrzeżenie pozwala przedstawić wagę wektora AB jako różnicę dwóch ułamków postaci analogicznej jak (1).

Metoda II — wagą odcinka skierowanego AB jest, tak samo jak w metodzie I, suma:

$$W_{AB} = S(\triangle ABP_1) + S(\triangle ABP_2) + \ldots + S(\triangle ABP_{n-2}).$$

Wyznaczając pole trójkąta $\triangle ABP_i$ w oparciu o skierowaną parę boków $\overrightarrow{AB}$ i $\overrightarrow{AP_i}$, dostajemy następujący wzór na W_{AB}:

$$W_{AB} = \frac{|\overrightarrow{AB} \times \overrightarrow{AP_1}|}{2} + \frac{|\overrightarrow{AB} \times \overrightarrow{AP_2}|}{2} + \ldots + \frac{|\overrightarrow{AB} \times \overrightarrow{AP_{n-2}}|}{2}.$$

Iloczyn wektorowy jest liniowy ze względu na każdy z dwóch argumentów. W szczególności, dla dowolnych wektorów $\vec{u}$, $\vec{v}$ i $\vec{w}$ mamy:

$$\vec{u} \times (\vec{v} + \vec{w}) = \vec{u} \times \vec{v} + \vec{u} \times \vec{w}.$$

Ponieważ we wzorze na W_{AB} występują wartości bezwzględne iloczynów wektorowych, aby skorzystać z powyższej własności, podzielimy punkty P_i na dwie grupy ze względu na znak iloczynu $\overrightarrow{AB} \times \overrightarrow{AP_i}$. Zauważmy, że jest on dodatni dla punktów P_i leżących na lewo od wektora $\overrightarrow{AB}$, czyli tworzących z nim kąt wypukły, natomiast niedodatni dla pozostałych punktów (punkty leżące na prostej $pr(AB)$ są, jak w metodzie I, nieistotne). Dla każdej z grup wystarczy wyznaczyć wartość postaci:

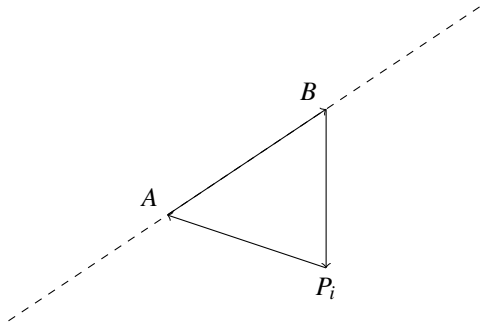
$$W_{AB} = \frac{1}{2}\overrightarrow{AB} \times (\overrightarrow{AP_{i_1}} + \overrightarrow{AP_{i_2}} + \ldots + \overrightarrow{AP_{i_k}}). \tag{2}$$

Wagę AB dostajemy jako różnicę tych wartości.

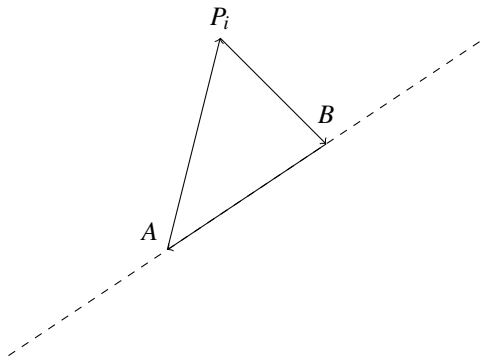
Metoda III — wagą odcinka skierowanego $\overrightarrow{AB}$ dla punktów $A = (x_A, y_A)$ i $B = (x_B, y_B)$ jest wartość:

$$W_{AB} = m \cdot \frac{(x_B - x_A)(y_A + y_B)}{2}, \tag{3}$$

gdzie m oznacza liczbę trójkątów $\triangle ABP_i$, na których obwodzie, skierowanym zgodnie z kierunkiem ruchu wskazówek zegara, występuje odcinek $\overrightarrow{AB}$. Zauważmy, że aby trójkąt $\triangle ABP_i$ miał tę własność, punkt P_i musi leżeć na prawo od wektora $\overrightarrow{AB}$, czyli wektor $\overrightarrow{BP_i}$ musi tworzyć z nim kąt skierowany wklęsły:



Dla punktów P_i położonych na lewo od wektora $\overrightarrow{AB}$, trójkąt $\triangle ABP_i$ nie zawiera na swoim (prawoskrętnym) obwodzie wektora $\overrightarrow{AB}$, lecz wektor $\overrightarrow{BA}$:



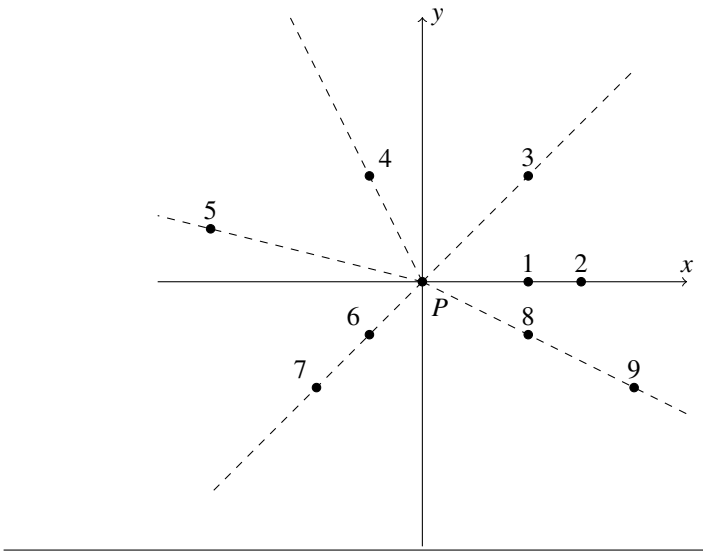
Punkty P_i leżące na prostej $pr(AB)$ nie mają, tak samo jak poprzednio, wpływu na wagę odcinka $\overrightarrow{AB}$. Parametr m ze wzoru (3) jest więc równy liczbie punktów P_i, które leżą na prawo od wektora $\overrightarrow{AB}$.

Analiza przeprowadzona dla wszystkich trzech metod wykazuje, że niezależnie od obranego wzoru liczenia pól trójkątów, musimy umieć dla wektora $\overrightarrow{AB}$ efektywnie podzielić wszystkie punkty P_i na leżące na prawo i na lewo od niego[2]. Dla każdej z tych grup musimy także wyznaczyć określoną wartość mającą wpływ na wagę wektora $\overrightarrow{AB}$ — w kolejnych metodach jest to: suma odciętych i rzędnych punktów z grupy (metoda I), suma wektorów o początku w A i końcach w punktach z grupy (metoda II) czy po prostu liczba punktów w grupie (metoda III). Do efektywnego rozwiązania zadania może więc doprowadzić wybranie takiej kolejności przeglądania wszystkich wektorów $\overrightarrow{AB}$, żeby podział punktów na leżące na lewo i na prawo od wektora był łatwy do przeprowadzenia lub zaktualizowania na podstawie podziału dla poprzednio rozważanego wektora.

[2]W przypadku metody I mówiliśmy raczej o punktach położonych nad oraz pod prostą $pr(AB)$. Zauważmy jednak, że punkty położone nad i pod prostą $pr(AB)$ odpowiadają punktom leżącym na lewo i na prawo od wektora $\overrightarrow{AB}$.

Zamiatanie kątowe

Dobrym kierunkiem poszukiwania właściwej kolejności analizowania wektorów wydaje się być rozważenie punktów *posortowanych kątowo (biegunowo)*. Niech $Z = \{P_1, P_2, \ldots, P_n\}$ oznacza zbiór wszystkich zadanych punktów. W porządku kątowym określamy środek — może to być dowolny punkt $P = (x, y)$ ze zbioru Z. Następnie sortujemy pozostałe punkty $Q \in Z$ niemalejąco względem kątów skierowanych, jakie tworzą z półprostą poziomą o początku w punkcie P, czyli względem $\angle P'PQ$, gdzie $P' = (x + \delta, y)$, $\delta > 0$. Kolejność punktów tworzących z półprostą taki sam kąt może być dowolna. Dla ustalenia uwagi możemy przyjąć, że w przypadku jednakowych kątów wcześniejszy jest punkt położony bliżej środka P.



Rys. 1: Przykład posortowania kątowego dziewięciu punktów wokół punktu P umieszczonego w środku układu współrzędnych (numery punktów odpowiadają kolejności w posortowanej sekwencji).

Punkty ze zbioru Z będziemy sortować kątowo kolejno wokół $P_1, P_2, \ldots, P_n$. Uporządkowanie Z wokół P_i pozwoli nam przeanalizować wszystkie wektory $\overrightarrow{P_iP_j}$ dla $j \neq i$ w kolejności zgodnej z porządkiem kątowym P_j i określić ich wagi zgodnie z metodą I, II lub III. Algorytm będzie zatem pewną formą klasycznej *metody zamiatania*, z tą tylko różnicą, że miotłą będzie półprosta obracająca się wokół punktu P_i.

Zamiatanie dla punktu P_i rozpoczynamy od półprostej poziomej p o początku w P_i, skierowanej ku rosnącym wartościom odciętej. Uzupełnienie p do prostej, czyli półprostą o początku w P_i skierowaną ku malejącym wartościom odciętej, nazwiemy q — oznaczenie to przyda nam się później, przy opisie algorytmu. Dla półprostej p łatwo jest wyznaczyć wszystkie punkty leżące na lewo (zbiór L) i na prawo od niej (zbiór P):

$$\begin{aligned} L &= \{P_j : |\angle P'P_iP_j| \in (0^\circ, 180^\circ)\} \\ P &= \{P_j : |\angle P'P_iP_j| \in (180^\circ, 360^\circ)\}, \end{aligned}$$

gdzie P' jest dowolnym, różnym od P_i punktem półprostej p.

Potrzebne do wyliczenia wagi wektora $\overrightarrow{P_iP'}$ wartości dla zbioru L (odpowiednio P) są wówczas równe:

Metoda I — sumom postaci:

$$\sum_{P_j=(x_j,y_j)\in L} x_j \quad \text{(odpowiednio} \quad \sum_{P_j=(x_j,y_j)\in P} x_j\text{)}$$

$$\sum_{P_j=(x_j,y_j)\in L} y_j \quad \text{(odpowiednio} \quad \sum_{P_j=(x_j,y_j)\in P} y_j\text{)}$$

Metoda II — sumie postaci:

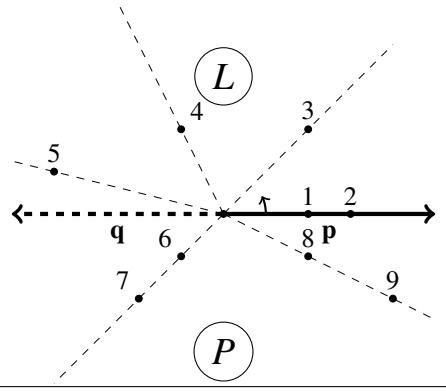
$$\sum_{P_j\in L} \overrightarrow{P_iP_j} \quad \text{(odpowiednio} \quad \sum_{P_j\in P} \overrightarrow{P_iP_j}\text{)}$$

Metoda III — rozmiarowi zbioru $|L|$ (odpowiednio $|P|$).

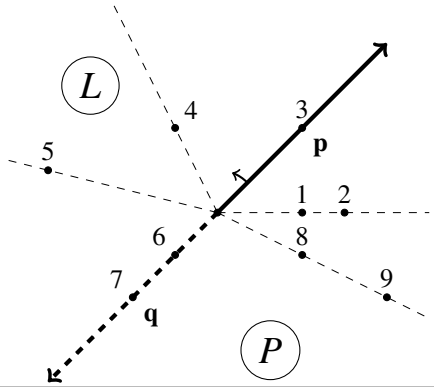
Znając powyższe zbiory i wartości dla początkowego położenia miotły, możemy przystąpić do zamiatania. Wykonamy półprostą p obrót o 360° wokół środka P_i, powiedzmy przeciwnie do kierunku ruchu wskazówek zegara. W trakcie obrotu na miotle p będą pojawiać się punkty zbioru Z w porządku kątowym względem środka P_i. Gdy na p znajdzie się punkt $P_j \in Z$, będziemy wyznaczać wagę odcinka skierowanego $\overrightarrow{P_iP_j}$. Po rozważeniu P_j punkt ten przerzucimy ze zbioru L do P, aktualizując jednocześnie interesujące nas wartości związane z L i P. Jeśli w trakcie obrotu miotły, na półprostej q znajdzie się jakiś punkt $P_k \in Z$, to punkt ten przenosimy z kolei ze zbioru P do L, także uaktualniając odpowiednie wartości służące do wyliczania wag.

Aby sprawnie wykonać obrót miotłą p, odnotowując wszystkie wystąpienia punktów ze zbioru Z na półprostej p lub q, dla każdej z tych półprostych będziemy pamiętać ostatni punkt ze zbioru Z, przez który ona przechodziła. Oznaczmy te punkty odpowiednio Q i R, a ich następniki w porządku kątowym — odpowiednio Q' i R'. Obrót prostej będziemy wykonywać zawsze o mniejszy z kątów: pomiędzy P_iQ i P_iQ' lub pomiędzy P_iR i P_iR'.

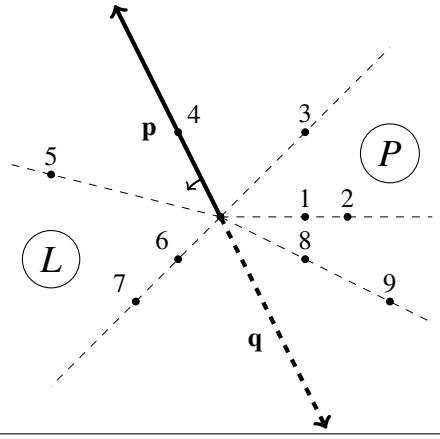
Rozważmy konkretny przykład, który pozwoli lepiej zrozumieć mechanizm opisanego zamiatania kątowego. Na rys. 2–6 przedstawiamy, jak kształtują się zbiory L i P przy kilku kolejnych obrotach półprostych p i q dla zbioru punktów z rys. 1.



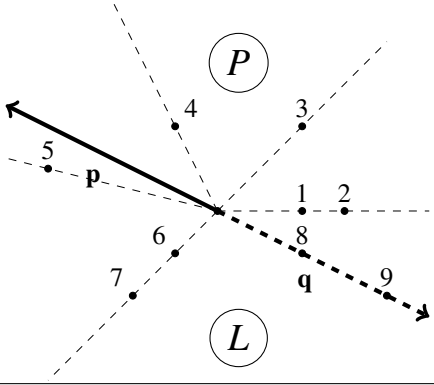
Rys. 2: Na samym początku mamy $L = \{3,4,5\}$, $P = \{6,7,8,9\}$. Tuż po tym, jak półprosta p zacznie się obracać, do zbioru P dołączą punkty o numerach 1 i 2.



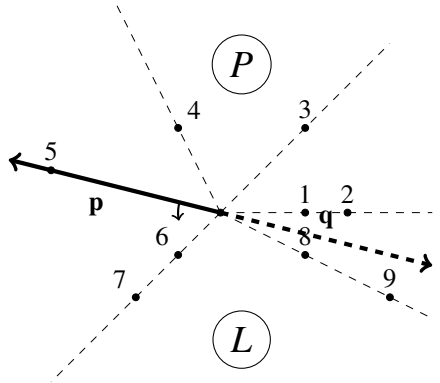
Rys. 3: W kolejnym kroku p napotyka na punkt 3, a q, równocześnie, na punkty 6 i 7. Wszystkie te punkty wskutek tego zdarzenia zmieniają swoje położenia w ramach zbiorów L i P. Dokładniej, zaraz po tym, jak półprosta zacznie się obracać, będzie $L = \{4,5,6,7\}$, a $P = \{1,2,3,8,9\}$.



Rys. 4: Po tym, jak p napotka punkt 4, nowe postaci zbiorów L i P to: $L = \{5,6,7\}$ i $P = \{1,2,3,4,8,9\}$.



Rys. 5: Przed tym, jak p napotka punkt 5, jej przedłużenie, czyli q, napotka punkty 8 oraz 9, które zostaną wówczas przerzucone z L do P. Nowe postaci zbiorów to: $L = \{5,6,7,8,9\}$, $P = \{1,2,3,4\}$.



Rys. 6: Tuż po napotkaniu przez p punktu 5 mamy $L = \{6,7,8,9\}$, $P = \{1,2,3,4,5\}$.

„Arytmetyka na kątach" — uwaga na pułapki!

Teoretycznie już wszystko wiemy — jak posortować punkty, jak je potem przetwarzać i jak wyznaczyć wynik końcowy (nawet na trzy sposoby). Pozostaje bardzo praktyczne pytanie: jak porównywać kąty i jak właściwie posortować punkty kątowo. Można spróbować się do tego zabrać na (co najmniej) dwa sposoby.

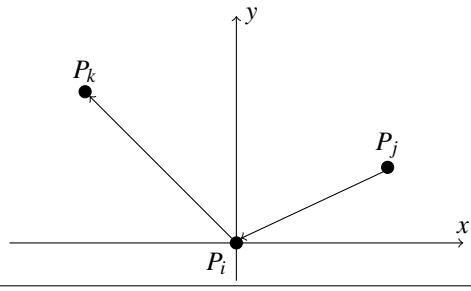
Pierwszy sposób to przypisanie punktom kątów, jakie tworzą one z wyjściową półprostą p, i sortowanie (porównywanie) względem tych kątów. Wartość kąta pomiędzy p i P_iQ możemy wyznaczyć za pomocą funkcji odwrotnych do funkcji trygonometrycznych (na przykład arcsin, arccos, ...). W obliczeniach komputerowych, szczególnie rozwiązując problemy geometryczne, staramy się jednak *unikać stosowania liczb zmiennoprzecinkowych*, gdy tylko da się zastąpić je obliczeniami na liczbach całkowitych. W obliczeniach na zmiennoprzecinkowych wartościach kątów, szczególnie z wykorzystaniem tak złożonych funkcji, jak odwrotne funkcje trygonometryczne, trudno bowiem uniknąć błędów zaokrągleń. Ich konsekwencją mogłaby być na przykład niewłaściwa kolejność w posortowanym ciągu punktów, które są „prawie współliniowe" z P_i. Taka pomyłka mogłaby oczywiście skończyć się złym wynikiem końcowym.

Spróbujmy więc obejść się bez funkcji trygonometrycznych (i ich odwrotności) i obliczeń zmiennoprzecinkowych. Możemy zaimplementować własną funkcję porównującą dwa punkty, tzn. zwracającą dla punktów P_j i P_k prawdę wtedy i tylko wtedy, gdy P_j powinien wystąpić w porządku przed P_k. Naturalne jest zastosowanie do tego celu iloczynu wektorowego, którego znak pozwala badać, czy przechodząc przez wspólny koniec dwóch odcinków skręcamy w lewo, w prawo, czy też idziemy prosto[3]. Na rysunku 7 możemy znaleźć przykład takiego porównania punktów.

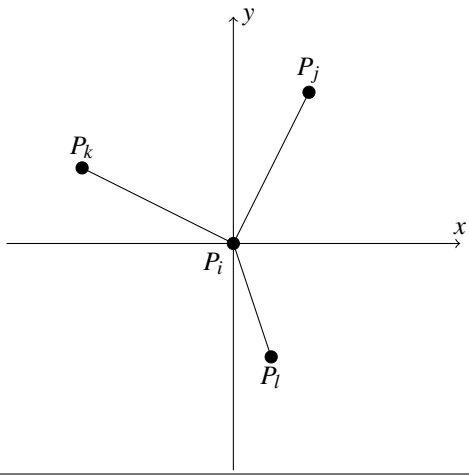
Reguła skrętu w prawo kryje jednak w sobie pułapki. Wyznacza ona bowiem porządek, w którym mogą wystąpić cykle (czyli tak naprawdę nie jest to porządek!). Przykład tej sytuacji przedstawiony jest na rysunku 8.

Jeśli tylko dostrzeżemy powyższy problem, to łatwo sobie z nim poradzić. Wystarczy w pierwszej kolejności zbadać, do której połowy układu współrzędnych należą rozważane punkty P_j i P_k. Jeśli oba należą do górnej lub oba należą do dolnej, to do ich porównania

[3]Dokładniejszy opis takiego testu można znaleźć w książce [20].



Rys. 7: Punkt P_j znajduje się w porządku kątowym względem środka P_i przed punktem P_k, gdyż idąc ścieżką $P_j \to P_i \to P_k$, skręcamy w prawo.



Rys. 8: Zgodnie z regułą, P_j jest przed P_k, P_k przed P_l, a P_l przed P_j, czyli rzeczywiście mamy cykl.

możemy zastosować iloczyn wektorowy. Jeśli natomiast należą do różnych półpłaszczyzn, to od razu decydujemy, że punkt z górnej półpłaszczyzny jest wcześniejszy w porządku od punktu z dolnej półpłaszczyzny.

To nie koniec pułapek kryjących się w sortowaniu kątowym. Kolejnym przypadkiem, w którym bardzo łatwo o błąd, są punkty leżące na osi poziomej, czyli na granicy półpłaszczyzny górnej i dolnej. By uniknąć problemów z określeniem ich położenia i porządku, najlepiej punkty postaci $(x,0)$ dla $x \geqslant 0$ zaliczyć do górnej półpłaszczyzny, natomiast pozostałe — do dolnej.

W ogólnym przypadku trzeba jeszcze zwrócić uwagę, w jakiej kolejności powinny występować punkty leżące na tej samej półprostej wychodzącej ze środka sortowania. Na szczęście w naszym przypadku nie jest to istotne.

Liczne przykłady implementacji sortowania kątowego można znaleźć w podręcznikach i innych opracowaniach. Tutaj ją pominiemy, polecając ją Czytelnikowi jako ćwiczenie. Z racji mnożących się pułapek i trudności w wychwyceniu ewentualnych błędów jest to ćwiczenie *bardzo pouczające*!

Złożoność rozwiązania

Najwyższa pora oszacować złożoność przedstawionych rozwiązań i ocenić ich efektywność. Posortowanie kątowe n punktów ma złożoność czasową taką samą, jak i każde inne sortowanie, gdyż potrafimy już porównać dwa punkty w czasie stałym. Stąd etap ten wymaga czasu $O(n \log n)$, oczywiście o ile tylko zastosujemy jakiś efektywny algorytm sortowania, np. przez scalanie. Koszt czasowy pojedynczego zamiatania kątowego jest, jak łatwo zauważyć, liniowy względem n. Złożoność czasowa całego algorytmu, który składa się z n takich faz, to zatem $O(n^2 \log n)$.

Czy jest to najefektywniejsza metoda rozwiązania naszego problemu? Okazuje się, że nie. Otóż istnieje bardzo skomplikowany algorytm jednoczesnego sortowania kątowego względem wszystkich punktów o złożoności czasowej $O(n^2)$ [4]. Nie był on jednakże brany pod uwagę przy sprawdzaniu efektywności rozwiązań niniejszego zadania, właśnie ze względu na duże skomplikowanie, ale także i niewielką różnicę rzędu złożoności czasowej w stosunku do rozwiązania wzorcowego.

Warto jednakże wspomnieć o ciekawym, a prostym pomyśle na usprawnienie opisanych rozwiązań wzorcowych. Nie poprawia on co prawda rzędu złożoności czasowej, ale polepsza stałą, co częściowo (bądź całkowicie) zapobiega wielokrotnemu wyliczaniu pól wszystkich trójkątów. Otóż po rozważeniu danego punktu jako środka sortowania P_i, można ten punkt po prostu wyrzucić ze zbioru Z. Dzięki temu, w kolejnych iteracjach maleje liczba punktów koniecznych do rozważenia, a trójkąty postaci $\triangle P_i P_j P_k$ dla $i < j < k$ są liczone tylko w iteracji algorytmu odpowiadającej P_i. To oznacza, że w przypadku metod I i II, pole każdego trójkąta jest w nowej wersji algorytmu wliczane zaledwie dwukrotnie, natomiast w przypadku metody III jest uwzględniane dokładnie raz!

Implementacje i inne rozwiązania

W rozwiązaniach wzorcowych zostały zaimplementowane metody oparte o dwa z opisanych sposobów wyznaczania pola trójkąta. W plikach `tro.cpp`, `tro0.pas` i `tro2.java` znajdują się implementacje rozwiązania wzorcowego oparte o metodę II, natomiast w pliku `tro1.java` — oparte o metodę III. Wobec wcześniejszych uwag na temat arytmetyki zmiennoprzecinkowej, nie powinno nikogo dziwić, że rozwiązanie oparte o metodę I nie zostało w ogóle zaimplementowane. W metodzie tej bowiem wielokrotnie zachodzi potrzeba operowania na liczbach rzeczywistych (pojawiają się w nim ułamki — na przykład przy wyznaczaniu parametrów prostej A, B i C — czy pierwiastki kwadratowe), co może powodować problemy z dokładnością.

Ciekawym spostrzeżeniem dotyczącym obliczeń w metodach II i III, o którym dotychczas nie wspominaliśmy, jest postać liczbowa rozważanych tam wag. Okazuje się, że pole trójkąta lub trapezu o wierzchołkach w punktach o całkowitoliczbowych współrzędnych musi być całkowitą wielokrotnością $\frac{1}{2}$. Dzięki temu, wszystkie obliczenia w rozwiązaniach opartych o metody II oraz III mogą być wykonywane na liczbach całkowitych, a dopiero na samym końcu trzeba wynik podzielić przez 2. To spostrzeżenie wyjaśnia także, dlaczego w zadaniu wystarczy wypisać wynik z dokładnością do zaledwie jednej cyfry po przecinku.

[4] Więcej szczegółów można znaleźć w artykule [35].

Testy

Rozwiązania zawodników były sprawdzane na 11 zestawach testów. Rozwiązania o złożoności $O(n^3)$ mieściły się w limicie czasowym tylko dla testów 1–3.

Nazwa	n	Opis
tro1a.in	10	prosty test poprawnościowy, losowy
tro1b.in	3	test poprawnościowy w kształcie małego trójkąta, odpowiedź niecałkowita
tro2a.in	10	prosty test poprawnościowy, dużo punktów na jednej prostej
tro2b.in	30	prosty test poprawnościowy, odpowiedź to 0.0
tro2c.in	5	prosty test poprawnościowy
tro2d.in	3	test poprawnościowy w kształcie małego trójkąta, odpowiedź niecałkowita
tro3a.in	20	prosty test poprawnościowy, losowy
tro3b.in	100	mały test w postaci prostokąta punktów kratowych, dużo trójek współliniowych punktów
tro3c.in	3	test poprawnościowy w kształcie małego trójkąta, odpowiedź niecałkowita
tro4.in	1 300	test średniej wielkości, losowy
tro5.in	1 500	test średniej wielkości, losowy
tro6.in	1 750	test duży, losowy
tro7.in	1 900	test duży, losowy
tro8a.in	2 000	test duży, losowy
tro8b.in	1 500	średni test w postaci prostokąta punktów kratowych, dużo trójek współliniowych punktów
tro9a.in	2 500	test duży, losowy
tro9b.in	2 000	duży test w postaci prostokąta punktów kratowych, dużo trójek współliniowych punktów
tro10.in	3 000	test duży, losowy
tro11a.in	3 000	test duży, losowy
tro11b.in	3 000	duży test w postaci prostokąta punktów kratowych, dużo trójek współliniowych punktów