

1. Sprawdzanie czy choć jedna para odcinków się przecina

Strukturą stanu jest zbiór SortedSet z biblioteki sortedcontainers (<http://www.grantjenks.com/docs/sortedcontainers/sortedset.html>) pozwalający na dodawanie, usuwanie i wyszukiwanie wartości w czasie $O(\log n)$. Zbiór zawiera odcinki klasy Line, które mogą być porównywane dzięki przeciążonym operatorom. Porównuje się współrzędną y-ową odcinków w obecnym położeniu miotły. Jako strukturę zdarzeń wybrałem listę końców wszystkich odcinków posortowaną rosnąco po x-owej współrzędnej, ponieważ tylko raz - na początku wykonywania algorytmu - wprowadzamy wszystkie zdarzenia i później nie potrzebujemy wprowadzać nowych, bo przy wykryciu pierwszego przecięcia przerywamy działanie programu i zwracamy informację o istnieniu przecięcia.

2. Wyznaczanie wszystkich przecięć

Należy zmienić strukturę zdarzeń, ponieważ teraz jest potrzebna możliwość wprowadzania nowych zdarzeń (przecięć odcinków) do struktury w czasie $O(\log n)$. Wykorzystałem w tym celu kopiec minimum z biblioteki heapq. Wprowadzam do struktury 3 informacje: współrzędną x-ową zdarzenia, informację czy jest to początek odcinka lub jego koniec lub przecięcie odcinków, listę odcinków biorących udział w zdarzeniu. Kluczem do porównywania wartości w kopcu jest współrzędna x-owa zdarzenia. Struktura stanu pozostała taka sama.

Istotną różnicą w porównaniu z algorytmem z punktu 1. jest potrzeba obsługi nowych zdarzeń - przecięć. Realizowana ona jest następująco: w algorytmach wyliczana jest dokładność (epsilon) zależna od wielkości przedziałów punktów. Odcinki przecinające się są uważane przez strukturę stanu za jednakowe gdy miotła jest na x-owej współrzędnej przecięcia. By rozwiązać ten problem zdecydowałem się w punkcie przecięcia cofać miotłę o wartość eps, wyszukiwać indeksy odcinków przecinających się a następnie odszukać w strukturze stanu odpowiednie co najwyżej 2 odcinki, które mogą później się przeciąć z aktualnie przecinającymi się odcinkami. Po sprawdzeniu ich przecięć usuwam ze struktury stanu odcinki, które aktualnie się przecinają, przesuwam miotłę o $2 \cdot \text{eps}$ w prawo (czyli jest teraz na pozycji o eps na prawo od punktu przecięcia) i ponownie wstawiam odcinki, które się przecinają. Robię to by zagwarantować ich poprawne ułożenie w strukturze stanu. Następnie powracam miotłą do punktu przecięcia i przechodzę do kolejnego zdarzenia. Pogarsza to dokładność algorytmu, ale czyni go działającym.

3. Kilukrotne wykrywanie przecięć

Program przeciwdziała wielokrotnemu wykrywaniu przecięć. W drugim algorytmie jest obecna kolejna struktura o nazwie intersections_pos, która jest również zbiorem SortedSet z biblioteki sortedcontainers. Zawiera on współrzędne x-owe punktów przecięć. Przy dodawaniu nowego przecięcia program sprawdza czy takie przecięcie już jest w intersections_pos w czasie $O(\log n)$.