Funkcje Matematyczne użyte do obliczeń w algorytmie:

- circle_minimum(p0,p1,p2)
  - Funkcja oblicza współrzędną y okręgu opisanego na trójkącie p0,p1,p2 taką, że ta współrzędna jest minimalna
  - Funkcja korzysta z rozwiązania układu równań
    $(a - p_{0x})^2 + (b - p_{0y})^2 = r^2 = (a - p_{1x})^2 + (b - p_{1y})^2$
    $(a - p_{0x})^2 + (b - p_{0y})^2 = r^2 = (a - p_{2x})^2 + (b - p_{2y})^2$
  - w celu wyznaczenia b oraz r (środek okręgu to (a,b), a jego promień to r)
  - By wyznaczyć y wystarczy od b_y odjąć r
  - Funkcja jest użyta w momencie sprawdzenia czy dane punkty mogą zostać dodane, jako zdarzenie okręgowe oraz w celu obliczenia priorytetu, z jakim takie zdarzenie ma być dodane do struktury zdarzeń (kolejki)
- get_vector(pl,pr)
  - Funkcja obliczająca wektor prostopadły do prostej przechodzącej przez punkty pl oraz pr w taki sposób aby wektor ten był zorientowany w lewo względem odcinka (pl,pr)
  - Dla pl = (x1,y1) oraz pr = (x2,y2) takim wektorem jest w szczególności (y1-y2,x2-x1) (trywialne)
  - Funkcja użyta przy obliczaniu punktów przecięcia odcinków 'niedokończonych' ze zbioru wynikowego z marginesem
- side_of_c(a,b,c)
  - Fukncja znajdująca przy pomocy wyznacznika czy punkt c znajduje się po lewej, na(lub na przedłużeniu), lub po prawej stronie odcinka (a,b).
  - Wyznacznik jest dodatni, gdy c leży po lewej stronie odcinka. W takiej sytuacji funkcja zwraca -1.
  - Jeśli jest ujemny to c jest po prawej stronie i funkcja zwraca 1
  - Jeśli c leży na (a,b) lub na jego przedłużeniu to zwraca 0
  - Funkcja użyta przy aktualizacji struktury stanu. Kiedy sprawdzane są potencjalne zdarzenia okręgowe. Jeśli idąc od lewej po ogniskach łuków idziemy zgodnie z ruchem wskazówek zegara oznacza to, że zdarzenie okręgowe istnieje i w jego centrum przecinają się dwa odcinki. W przeciwnym wypadku odcinki są rozbieżne lub ich punkt przecięcia znajduje się powyżej miotły
- get_y_intersect(y,vector,point)
  - funkcja znajdująca punkt przecięcia wektora vector zaczepionego w punkcie point z prostą o zadanej współrzędnej y
  - funkcja w celu znalezienia x korzysta z matematycznego faktu, że
    $$\frac{y - point_y}{x - point_x} = \frac{vector_y}{vector_x}$$
  - funkcja wykorzystywana przy wyznaczaniu punktów przecięcia odcinków 'niedokończonych' z marginesem do wizualizacji
- get_x_intersect(x,vectr,point)
  - działa analogicznie do get_y_intersect()
- find_end_box(point,vector,margin)
  - funkcja znajduje punkty przecięcia odcinków 'niedokończonych' z marginesem

    - funkcja w prosty sposób określa z którymi bokami prostokąta potencjalnie przetnie się wektor. Jeśli obie jego współrzędne są dodatnie to albo z górnym albo z prawym. Podobnie łatwo określić boki dla pozostałych możliwych znaków współrzędnych. Następnie sprawdzane jest przecięcie z tymi dwoma bokami i to, które nie wykracza poza prostokąt zostaje zwrócone, jako punkt przecięcia
    - funkcja obsługuje też zdarzenia w których współrzędne są zerowe
    - funkcja jest użyta w celu przetworzenia zbioru wynikowego na listę krawędzi do wyświetlenia w wizualizacji

Klasy użyte strukturze stanu

- edge
    - Klasa przestawiająca łuk w zbiorze wynikowym i w węzłach drzewa binarnego – strukturze stanu, w którym są one między parabolami(ich aktualny koniec jest punktem przecięcia parabol)
    - Przechowuje lewy i prawy punkt (punkty są rozróżnialne na prawy i lewy w celu możliwości określenia orientacji odcinka w funkcji vector – odcinki mają początek i koniec – są skierowane)
    - Przechowuje punkt początkowy oraz końcowy
    - Przechowuje swoje przeciwieństwo – odcinek o odwrotnych punktach lewym i prawym oraz o odwrotnych początku i końcu
- arc
    - Klasa przedstawiająca łuk paraboli
    - Przechowuje punkt będący ogniskiem paraboli
    - Przechowuje informację o tym czy łuk może być ‘centrum’ zdarzenia okręgowego (tzn. być tym z łuków zdarzenia okręgowego, który zanika)
    - Przechowuje informację o tym czy zdarzenie okręgowe, w którym w ogóle bierze udział(nie ważne jako który łuk) może się odbyć
- bst_node
    - klasa przedstawiająca węzeł w drzewie binarnym będącym strukturą stanu miotły
    - jeśli węzeł jest liściem to w polu arc przechowuje łuk a jeśli nim nie jest to w polu Edge przechowuje odcinek aktualnie kończący się w punkcie przecięcia paraboli przedstawianych przez najbliższy lewy i najbliższy prawy liść
    - przechowuje wskaźniki na lewe i prawe dziecko oraz na rodzica

Drzewo binarne – struktura stanu

- drzewo przechowuje dwie zmienne
    - l czyli aktualną pozycję miotły
    - root czyli ‘korzeń drzewa’
- pozycja miotły l jest aktualizowana za każdym razem, kiedy zmienia się w głównej funkcji
- Drzewo ma funkcję wypisującą jego kolejne liście. (Funkcja przydatna przy analizie algorytmu i debugowaniu nie jest konieczna w działaniu całego algorytmu)
- Funkcja get_arch_intersect(pl,pr) znajdująca matematycznie przecięcie 2 parabol o zadanych ogniskach przy konkretnej pozycji prostej l

    - Dla obu parabol funkcja wyznacza współczynniki a,b,c, odejmuje ich równania od siebie i znajduje jego odpowiednie miejsce zerowe
    - Przypadki, w których współrzędne y są równe lub jedno z ognisk znajduje się na linii są rozważane w podobny sposób jednak osobno by uniknąć dzielenia przez 0
    - Funkcja używana przy wyszukiwani binarnym oraz przy ustalaniu końców i początków odcinków
- Funkcja find(x)
    - Wyszukiwanie binarne paraboli znajdującej się nad punktem na prostej l o zadanej współrzędnej x
    - Dopóki nie zostanie znaleziony liść węzeł, który przetwarza funkcja zawiera odcinek, w którym przechowane są dwa punkty określające go. Są to dwa ogniska paraboli. Punkt przecięcia tych dwóch parabol(a dokładnie jego współrzędna x) kryterium względem, którego porównujemy
- Funkcja get_right_parent(node)
    - Funkcja dla zadanego liścia node zwraca węzeł, w którym znajduje się odcinek będący na prawym brzegu przechowywanego w tym liściu łuku. Znajduje się on w najbliższym nie-liściu po prawej stronie od liścia. Jest to elementarna operacja na drzewie binarnym niewymagająca tłumaczenia.
- Funkcja get_right_child(node)
    - Funkcja dla danego nie-liścia znajduje najbliższy liść po prawej stronie. W tym liściu znajduje się łuk którego lewym krańcem jest aktualny koniec odcinka przechowywanego w node. Operacja znajdowania najbliższego liścia po prawej jest elementarną operacją na drzewie binarnym nie wymagającą tłumaczenia.
- Funkcja get_next_leaf(node)
    - Funkcja znajduje najbliższy liść po prawej stronie od zadanego liścia node
    - Działa przez złożenie funkcji get_right_parent i get_right_child
    - Kiedy takiego liścia nie ma(zadany liść jest skrajnym prawym) funkcja zwraca None
    - Funkcja używana przy szukaniu nowych potencjalnych zdarzeń okręgowych
- Fukncje get_left_parent, get_left_child oraz get_prev_leaf działają analogicznie – odpowiedniki poprzednich 3 funkcji dla lewej strony
- Funkcja insert(p)
    - Funkcja będąca najważniejszą częścią obsługi zdarzenia punktowego
    - Funkcja dodaje nowy łuk do struktury stanu
    - Jako argument przyjmuje punkt p będący ogniskiem nowo dodawanego łuku
    - Jeśli struktura stanu jest pusta to po prostu tworzy nowy węzeł
    - W przeciwnym wypadku funkcja znajduje łuk znajdujący się dokładnie nad zadanym punktem, dezaktywuje go jako centrum zdarzenia okręgowego a następnie na jego miejsce wkłada nowe poddrzewo składające się z fragmentu tego łuku po lewej stronie od nowego łuku, nowego łuku, fragmentu łuku znajdującego się po prawej stronie nowego łuku oraz odcinków między nimi
    - Prawy i lewy z nowo powstałych łuków mają ognisko tam gdzie usuwany łuk a nowy łuk między nimi ma ognisko w zadanym punkcie p
    - Następnie funkcja sprawdza czy należy dodać zdarzenia okręgowe – szuka czy za nowo dodanym prawym łukiem znajduje się jakiś kolejny(przy pomocy funkcji

get_next_leaf) i jeśli tak to nowy łuk, prawy łuk i kolejny prawy łuk podlegają sprawdzeniu czy mogą zostać dodane, jako zdarzenie okręgowe. Sprawdzane są warunki, że środek okręgu pisanego na ich ogniskach znajduje się poniżej prostej l oraz czy idąc od lewej do prawej(według kolejności łuków) te ogniska są ułożone zgodnie z ruchem wskazówek zegara. Jeśli te warunki są spełnione to powstaje nowy zbiór składający się z lewego łuku, liścia z środkowym łukiem oraz prawego łuku. Jeśli zdarzenie nie powstaje to zamiast zbioru funkcja zwróci None
    - (dla łuków po lewej stronie funkcja postępuje analogicznie)
    - Zwracane są 2 zbiory oraz 2 nowo wstawione odcinki
- Funkcja delete(node)
    - Funkcja będąca najważniejszą częścią obsługi zdarzenia okręgowego
    - Funkcja usuwa łuk zanikający w zdarzeniu okręgowym.
    - Jako argument przyjmuje liść node będący centrum zdarzenia okręgowego (liściem który zawiera właśnie usuwany łuk)
    - Funkcja znajduje odcinki które przetną się w środku okręgu – znajduje je przy pomocy funkcji get_right_parent oraz get_left_parent
    - Znajduje punkt przecięcia łuków po prawej i lewej stronie usuwanego łuku – jest to nowy punkt przecięcia odcinków
    - Aktualizuje końcówki znalezionych odcinków, jako znaleziony punkt(a także początki odcinków przeciwnych)
    - Tworzy nowy odcinek oraz odcinek przeciwny do niego
    - Dezaktywuje wszystkie zdarzenia okręgowe, w których występuje usuwany łuk
    - Usuwa łuk oraz jego najbliższego rodzica a w drugim rodzicu zamienia pole edge na nowy odcinek. Pozostałe węzły zostają przepięte tak by nie uszkodzić poprawnej struktury drzewa
    - Następuje szukanie zdarzeń okręgowych tak jak w funkcji insert()
    - Zwracane są 2 potencjalne zbiory ze zdarzeniami okręgowymi oraz odcinek do dodania do zbioru wynikowego
- W idealnym rozwiązaniu drzewo powinno być równoważone, jednak nie miałem czasu na implementację

Zasadnicza część działania algorytmu

- Algorytm przyjmuje tablicę punktów w R^2
- Jeśli jest tylko jeden punkt lub nie ma żadnego algorytm zwraca None
- Algorytm znajduje maksymalne i minimalne współrzędne x-owe i y-owe punktów i ustala początkową pozycję miotły, jako największa pozycja y
- Inicjalizuje zbiór wynikowy, strukturę stanu, strukturę zdarzeń oraz wprowadza do struktury zdarzeń wszystkie zdarzenia punktowe według współrzędnej y. w strukturze zdarzeń – kolejce priorytetowej przechowywana jest też informacja w postaci wartości boolowskiej czy dane zdarzenie jest punktowe
- Funkcja handle_site_event otrzymuje punkt i dla tego punktu przeprowadza operację insert na strukturze stanu pobierając przy okazji nowe odcinki, które dodaje do zbioru wynikowego oraz potencjalne nowe zdarzenia okręgowe, które następnie dodaje do struktury zdarzeń przy pomocy funkcji add_circle_event

- Funkcja handle_circle_event otrzymuje zbiór reprezentujący potencjalne zdażenie okręgowe. Sprawdza pola active i circle_event_active w łukach w elementach zbioru i jeśli wszystkie są aktywne to pobiera środkowy element zbioru - wskaźnik na węzeł w strukturze stanu który ma usunąć i przeprowadza na niej operacje delete pobierając przy okazji nowy odcinek, który dodaje do zbioru wynikowego oraz potencjalne nowe zdarzenia okręgowe które następnie dodaje do struktury zdarzeń przy pomocy funkcji add_circle_event
- Funkcja add_circle_event otrzymuje na wejści potencjalny nowy zbiór reprezentujący potencjalne zdarzenie okręgowe. Jeśli ten zbiór jest wartością None to funkcja kończy działanie. Jeśli nie to dodaje go do struktury zdarzeń według najniższego punktu na okręgu opisanym na ogniskach łuków w zbiorze
- Algorytm osobo przetwarza pierwsze zdarzenie punktowe, a następnie w pętli while przetwarza wszystkie zdarzenia ze struktury zdarzeń do momentu, gdy będzie ona pusta.
- Tworzy margines do wizualizacji na podstawie maksymalnych i minimalnych współrzędnych odcinków w zbiorze wynikowym
- Tworzy sztuczne końcówki odcinków przy pomocy funkcji vector i find_end_box
- Przetwarza zbiór wynikowy na odcinki, które mogą zostać wyświetlone przez narzędzie graficzne(narzędzie pozwala przybliżyć obraz tak by potem nie widzieć ‘dalekich przecięć odcinków’)