Geometria obliczeniowa
Wykład 8

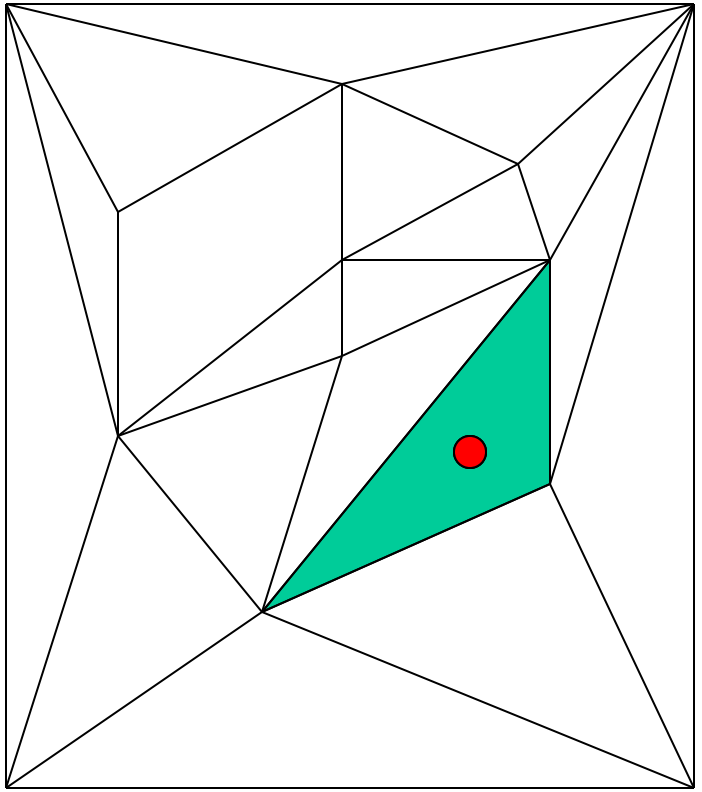
Lokalizacja punktu na płaszczyźnie

1. Metoda warstwowa
2. Metoda trapezowa
3. Metoda separatorów
4. Metoda doskonalenia triangulacji

Randomizowany algorytm znajdujący triangulację Delaunay.

Problem.

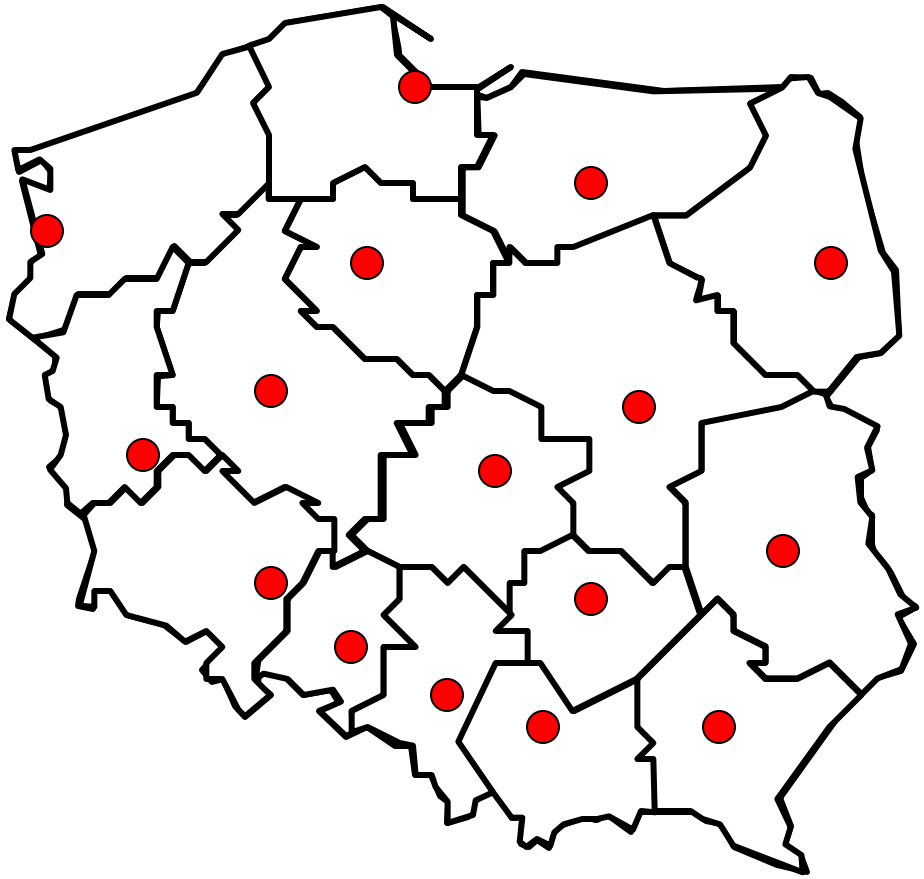
Dany jest podział D płaszczyzny (lub jej części) na wielokąty proste. Stwórz strukturę danych, która dla danego punktu p pozwoli szybko określać obszar zawierający ten punkt.



Ograniczony podział można umieścić w odpowiednio dużym wielokącie wypukłym W o małej liczbie boków i po striangulowaniu części wspólnej W i nieograniczonej części płaszczyzny możemy ograniczyć naszą uwagę do W.

Podobnie jak w przypadku lokalizacji punktu w obszarach prostokątnych, czas odpowiedzi na zapytanie o obszar będzie zależeć od rozmiaru struktury i wyboru metody jej konstrukcji.

Przykład. Lokalizacja punktów w podziale.

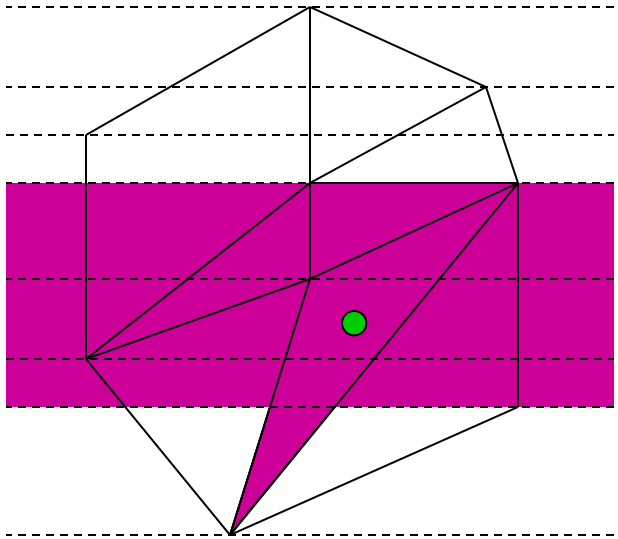


Metoda warstwowa.

Konstrukcja struktury:

Podziel przestrzeń na warstwy równoległymi prostymi przechodzącymi przez wierzchołki podziału D.

Stablicuj kolejne warstwy oraz ich podział.



Lokalizacja punktu:

Stosując przeszukiwanie binarne znajdź:

- warstwę zawierającą dany punkt,

- odpowiednią część warstwy zawierającą dany punkt.

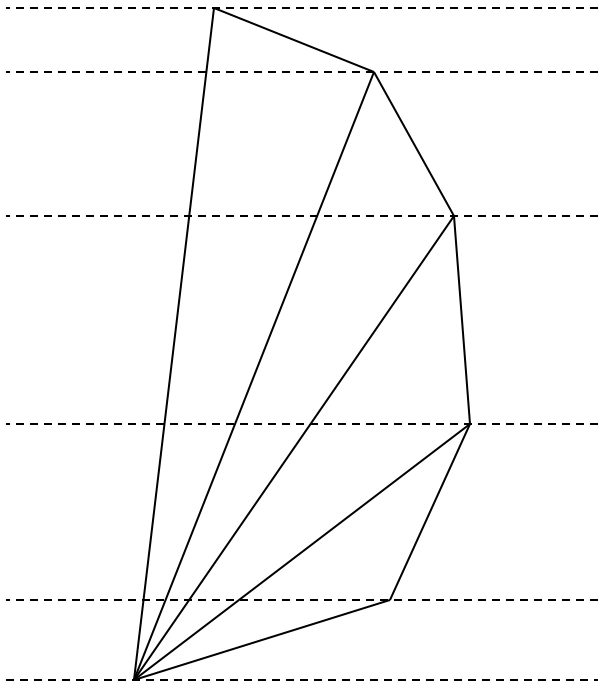
Określ obszar odpowiadający znalezionej części warstwy.

Lemat.

Struktura danych dla n-wierzchołkowego podziału płaszczyzny ma rozmiar $O(n^2)$ i można ją stworzyć, stosując np. algorytm zamiatania, w czasie $O(n^2)$.

Dowód.

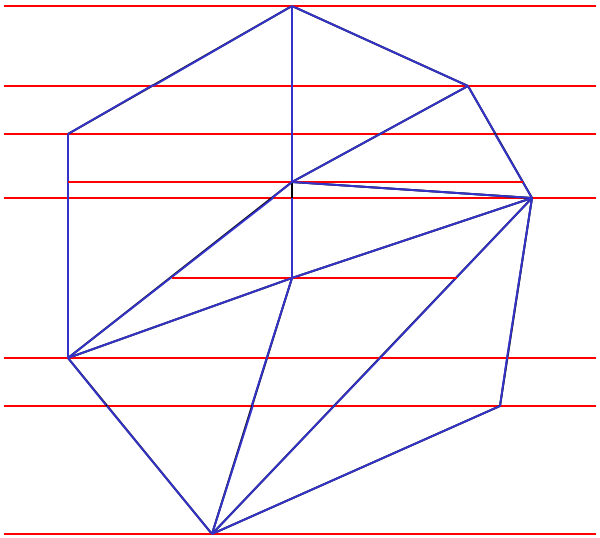
W algorytmie zamiatania strukturą zdarzeń jest uporządkowana lista wierzchołków podziału, a strukturą stanu - zrównoważone drzewo poszukiwań binarnych przechowujące uporządkowany ciąg krawędzi przecinanych przez miotłę. Łączny czas aktualizacji struktury stanu wynosi O(n log n). Czas tworzenia struktury danych dla zapytań zależy od rozmiaru struktury stanu (może być liniowy względem rozmiaru podziału – zależy od liczby przecinanych krawędzi).



Lemat.

Czas lokalizacji punktu w n-wierzchołkowym podziale płaszczyzny wynosi O(log n).

Metoda trapezów.



procedure TRAPEZ(D);

posortuj wierzchołki podziału D względem y-ów;

znajdź ich medianę y_m, prostą k ($y = y_m$) i stwórz dla niej węzeł drzewa z dwoma dowiązaniami;

rozbij D względem prostej k na D_1 i D_2;

for i := 1 ***to*** 2 ***do***

if istnieją krawędzie całkowicie przecinające D_i

then rozbij D_i wzdłuż tych krawędzi;

krawędziom przecinającym D_i przypisz węzły zrównoważonego poddrzewa węzła dla prostej k, którego liście odpowiadają tworzonym przez nie podziałom D_i (T_j) ;

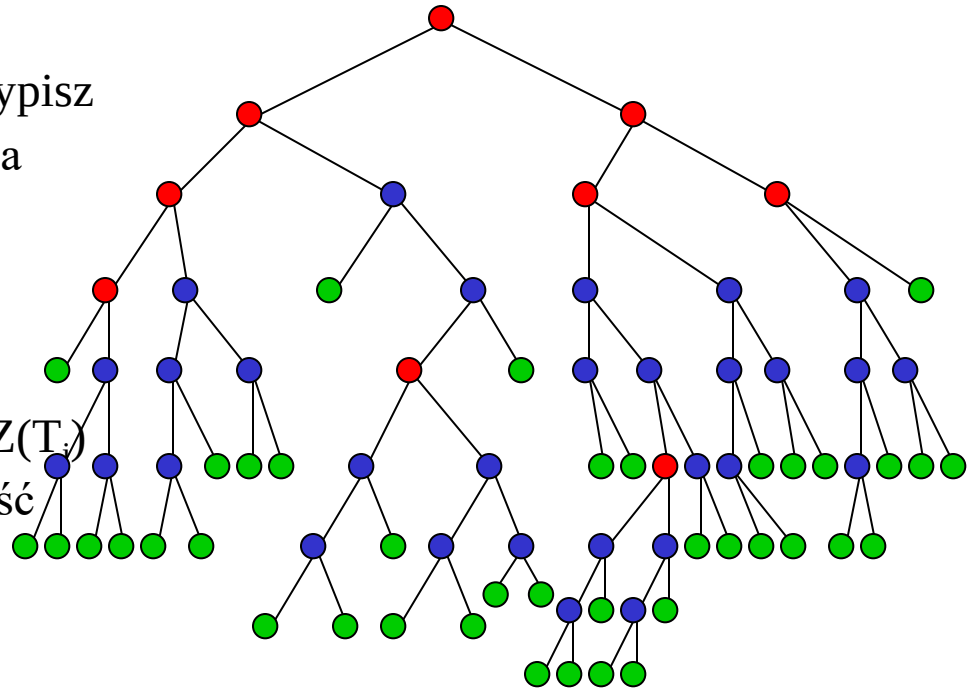
for każdy podział T_j ***do***

if T_j nie jest pusty ***then*** TRAPEZ(T_j)

else stwórz liść

else TRAPEZ(D_i);

return węzeł dla prostej k;



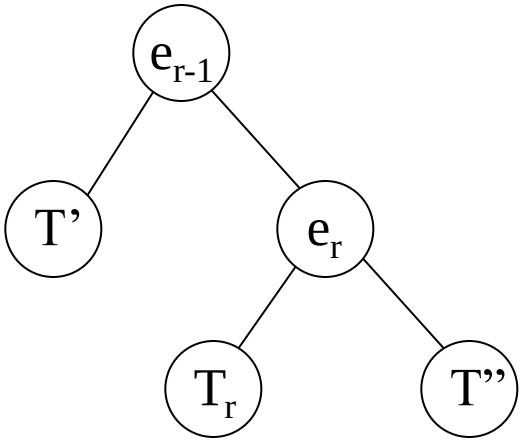
Równoważenie drzewa.

Niech $U = T_1 e_1 T_2 ... T_{k-1} e_{k-1} T_k$ oraz $W(U)$ będzie liczbą wierzchołków podziału płaszczyzny zawartych w U.

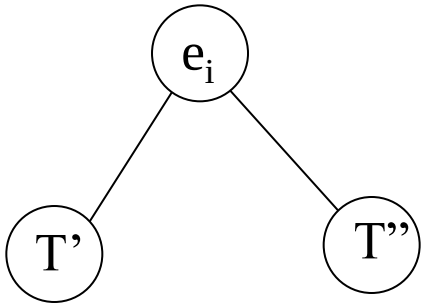
Niech r będzie taką liczbą, że $W(T_1) + ... + W(T_{r-1}) < W(U)/2$ oraz $W(T_1) + ... + W(T_r) \geq W(U)/2$.

Wtedy e_{r-1} jest korzeniem drzewa odpowiadającego U, e_r jego prawym synem, a lewym synem e_r jest poddrzewo odpowiadające T_r.

Pozostałe dwa poddrzewa odpowiadają początkowemu i końcowemu fragmentowi U.



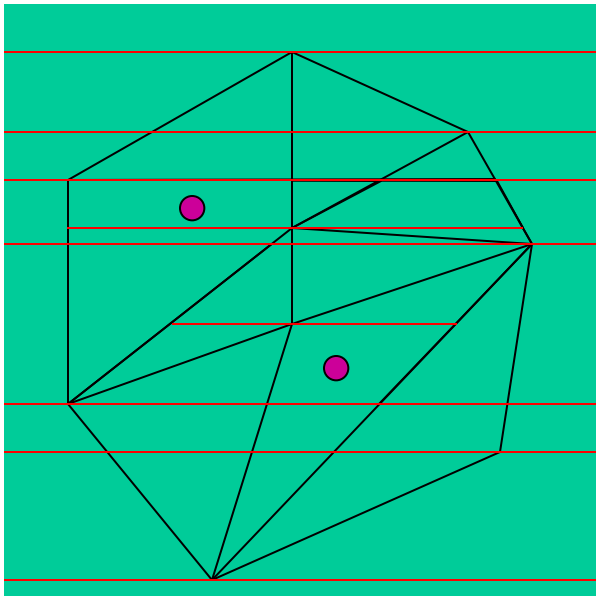
Gdy $W(U) = 0$, U odpowiada zrównoważone drzewo krawędzi e_i z liśćmi odpowiadającymi zbiorom pustych trapezów T’ i T” .



Lemat.

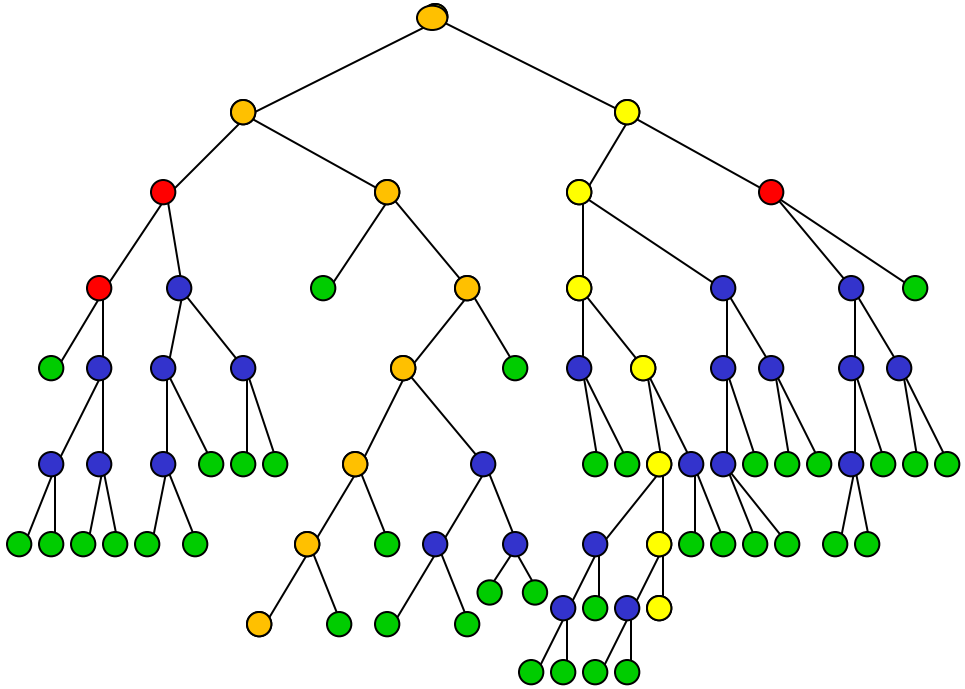
Niech U będzie pasem odpowiadającym ciągowi $T_1e_1T_2e_2 \ldots e_{k-1}T_k$, gdzie T_i oznacza i-ty fragment podziału pasa, a e_j j-tą krawędź rozdzielającą. Niech W(U) oznacza liczbę wierzchołków obszarów podziału płaszczyzny zawartych w U. Wtedy głębokość drzewa odpowiadającego pasowi U szacuje się przez $3\log W(U) + \lceil \log n \rceil + 3$, gdzie n jest całkowitą liczbą wierzchołków podziału płaszczyzny.

Dowód. Indukcja po W(U).



Twierdzenie.

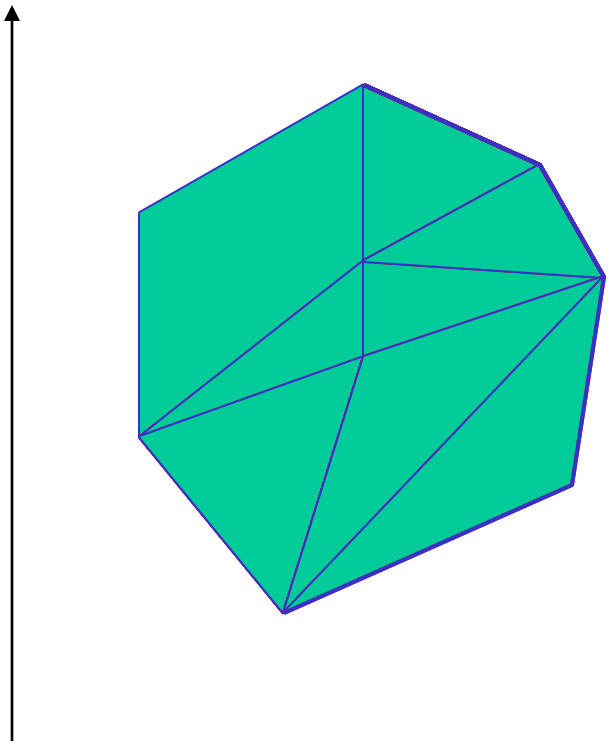
W n-wierzchołkowym podziale płaszczyzny lokalizujemy punkt w czasie O(log n) stosując strukturę danych o rozmiarze O(n log n) stworzoną w czasie O(n log n).



Metoda separatorów.

Definicja.

Separatorem nazywamy monotoniczną względem danego kierunku łamaną rozdzielającą podział i tworzoną przez jego krawędzie.

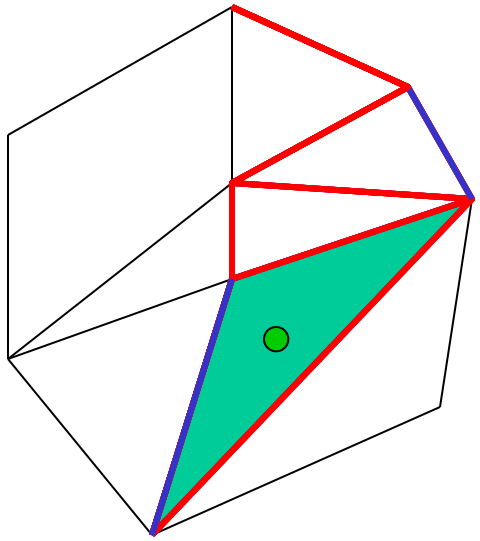


Naszym celem jest stworzenie ciągu separatorów (S_n) takiego, że :

- każde dwa sąsiednie separatory wyznaczają dokładnie jeden obszar należący do podziału D,
- każdy obszar podziału jest wyznaczany przez separatory,
- wszystkie wierzchołki separatorów o niższych numerach leżą po tej samej stronie separatora o numerze wyższym.

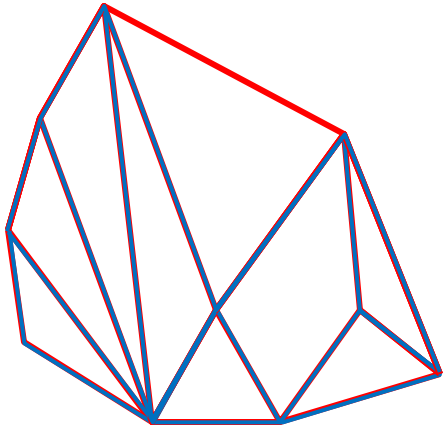
Załóżmy, że ciąg separatorów dla danego podziału jest stablicowany oraz że krawędzie każdego separatora są również stablicowane. Rozmiar struktury wynosi $O(n^2)$.

Wyszukujemy binarnie separator, który znajduje się powyżej/poniżej (z prawej/lewej strony) lokalizowanego punktu i taki, że poprzedzający go separator znajduje się poniżej/powyżej (z lewej/prawej strony) tego punktu (wyszukujemy odpowiednią krawędź separatora i sprawdzamy, po której stronie danego punktu się znajduje). Wyznaczamy w ten sposób obszar zawierający punkt.



Lemat.

Czas lokalizacji punktu w n-wierzchołkowym podziale płaszczyzny wynosi $O(\log^2 n)$.



Rozmiar struktury pozostaje kwadratowy, nawet gdy sąsiednie separatory mogą określać więcej niż jeden obszar podziału.

Wyznaczanie separatorów.

Definicja.

Niech zbiór wierzchołków podziału tworzy ciąg uporządkowany względem współrzędnej y-owej (w przypadku równych wartości - względem współrzędnej x-owej). Wierzchołek v_k jest ***regularny***, gdy istnieją krawędzie (v_i,v_k) i (v_k,v_j) dla $i < k < j$. Podział jest regularny, gdy wszystkie wierzchołki ciągu poza pierwszym i ostatnim są regularne.

Każdy podział można zregularyzować stosując algorytm podobny do algorytmu podziału wielokąta na wielokąty monotoniczne (łączymy krawędzią wierzchołek nieregularny v z pomocnikiem krawędzi sąsiadującej z v z lewej strony) lub triangulując otoczkę wypukłą podziału.

Załóżmy, że krawędzie podziału są skierowane od wierzchołka o mniejszym indeksie do wierzchołka o większym indeksie. Niech IN(v) i OUT(v) odpowiednio oznaczają zbiory krawędzi dochodzących do v i wychodzących z v.

Niech W(e) oznacza ***wagę*** krawędzi e, czyli liczbę separatorów, do których należy e.

Wtedy $W_{IN}(v) := \Sigma_{e \in IN(v)} W(e)$ oraz $W_{OUT}(v) := \Sigma_{e \in OUT(v)} W(e)$.

Aby wyznaczyć układ separatorów wystarczy pokazać, że wagi można tak dobrać, aby:

- waga każdej krawędzi była dodatnią liczbą całkowitą,
- $W_{IN}(v) = W_{OUT}(v)$ dla każdego v regularnego.

Pierwszy warunek zapewnia, że każda krawędź należy do co najmniej jednego separatora. Drugi zaś gwarantuje, że separatory są spójne i nie kończą się w wierzchołku regularnym.

procedure WAGI

for każda krawędź e do W(e) := 1;

for i := 2 to n-1 do

v := i-ty wierzchołek w ciągu;

$W_{IN}(v) := \Sigma_{e \in IN(v)} W(e)$;

d := pierwsza z lewej krawędź wychodząca z v;

if $W_{IN}(v) > W_{OUT}(v)$

then $W(d) := W_{IN}(v) - W_{OUT}(v) + W(d)$;

for i := n-1 downto 2 do

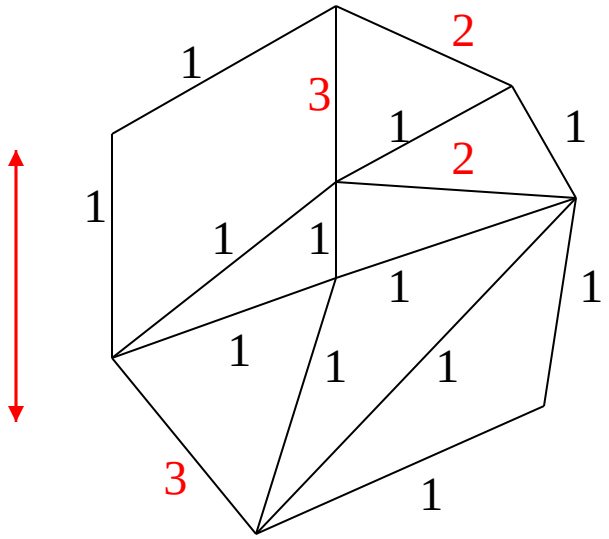
v := i-ty wierzchołek w ciągu;

$W_{OUT}(v) := \Sigma_{e \in OUT(v)} W(e)$;

d := pierwsza z lewej krawędź dochodząca do v;

if $W_{OUT}(v) > W_{IN}(v)$

then $W(d) := W_{OUT(v)} - W_{IN(v)} + W(d)$;

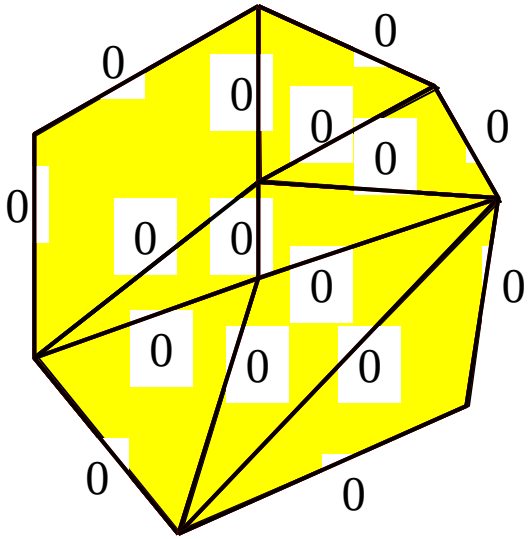


Przykład.

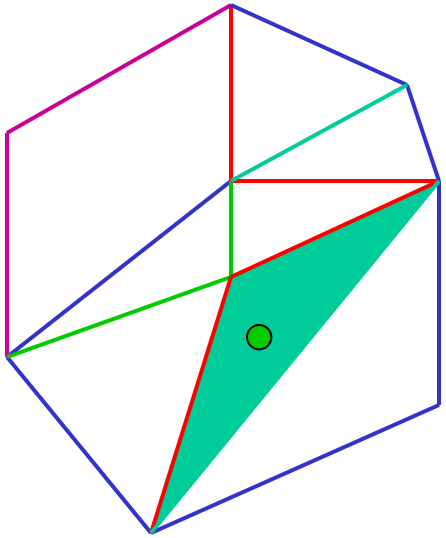
Układ separatorów określanych przez wagi krawędzi.

Między kolejnymi separatorami może być więcej niż jeden obszar

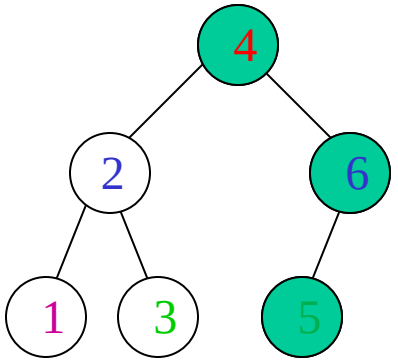
(jest 8 obszarów i 6 separatorów)



W poprzednim algorytmie przechowywaliśmy stablicowane separatory co znacznie obciążało pamięć.



Teraz stosujemy zrównoważone drzewo binarne, w którego węzłach przechowujemy stablicowany ciąg krawędzi danego separatora nie należących do separatorów odpowiadających przodkom tego separatora w drzewie. Jeśli w badanym ciągu nie ma krawędzi separatora leżącej nad lub pod lokalizowanym punktem, to wynik badania jest taki sam jak poprzednio.



Lemat.

Czas lokalizacji punktu w n-wierzchołkowym podziale płaszczyzny wynosi $O(\log^2 n)$, przy wykorzystaniu struktury o rozmiarze $O(n)$ tworzonej w czasie $O(n \log n)$.

Metoda doskonalenia triangulacji.

Załóżmy, że dany podział D ma kształt striangulowanego trójkąta.

procedure TRIANGLE(D)

T := zbiór trójkątów tworzących D;

V := {etykiety trójkątów}; E := ∅;

while |T| > 1 ***do***

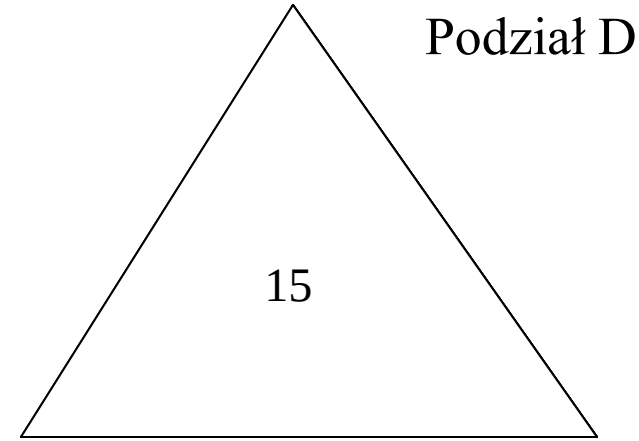
wybierz niezależny zbiór wewnętrznych wierzchołków stopnia mniejszego niż 12 ;

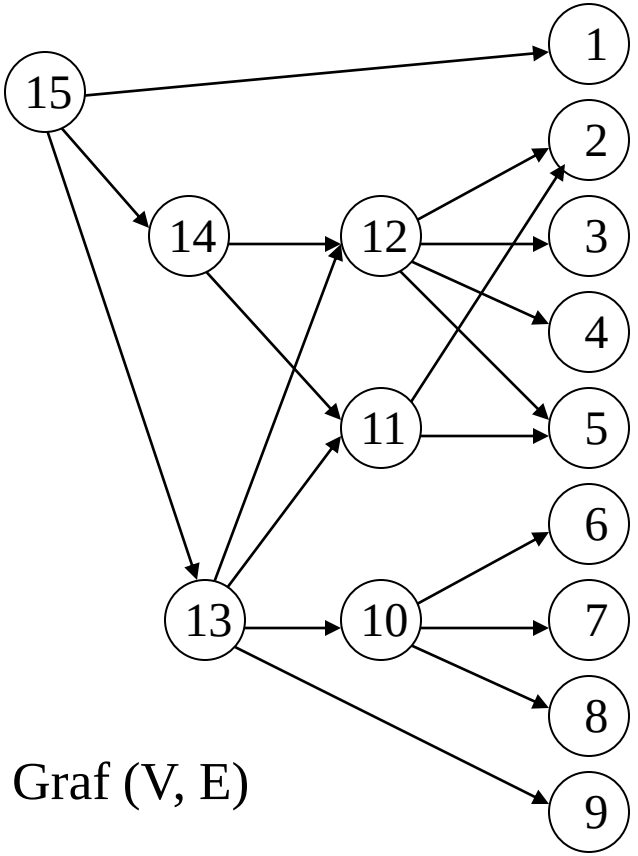
usuń z T trójkąty z tymi wierzchołkami;

usuń z D wybrane wierzchołki wraz z incydentnymi krawędziami, ponownie strianguluj D i dodaj nowe trójkąty do T;

V := V ∪ {nowe trójkąty}; E := E ∪ {(N,U): N-nowy, U-usunięty, N ∩ U ≠ ∅};

return (V, E);

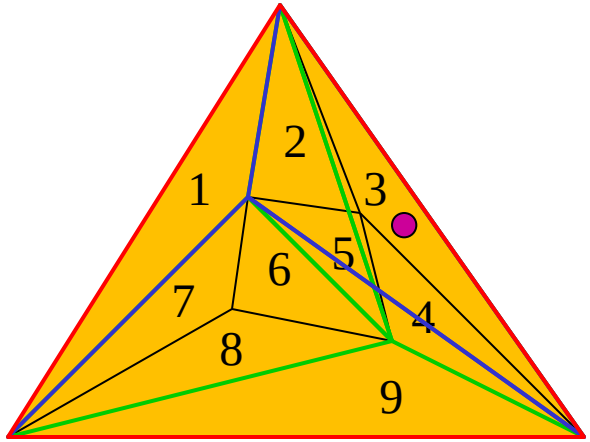




Graf (V, E)

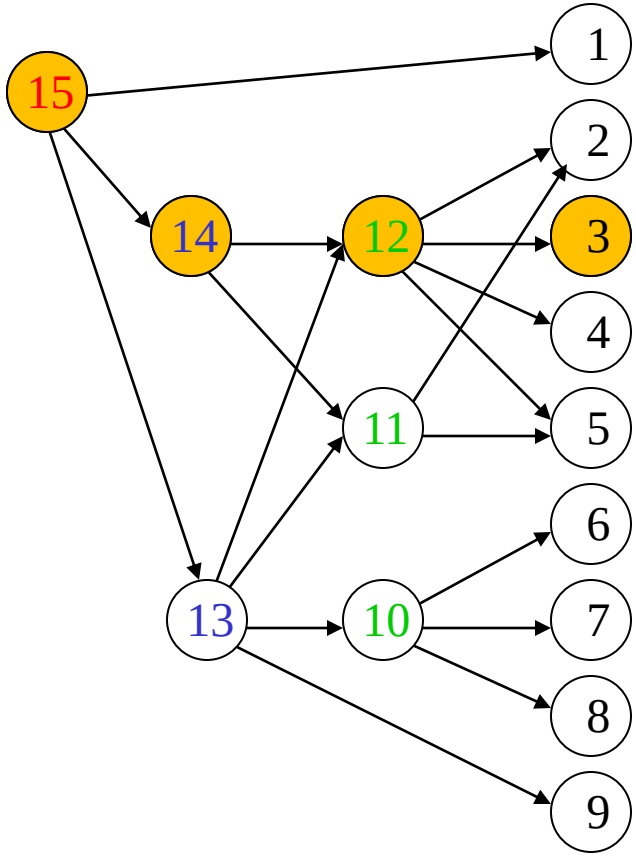
Lemat.

Liczba poziomów w grafie jest logarytmiczna, jego rozmiar jest liniowy względem liczby wierzchołków a każdy wierzchołek ma stopień ograniczony przez stałą.



Fakt.

Dla dowolnego podziału o n wierzchołkach czas preprocessingu wynosi O(n) (korzystając z algorytmu triangulacji Chazelle).



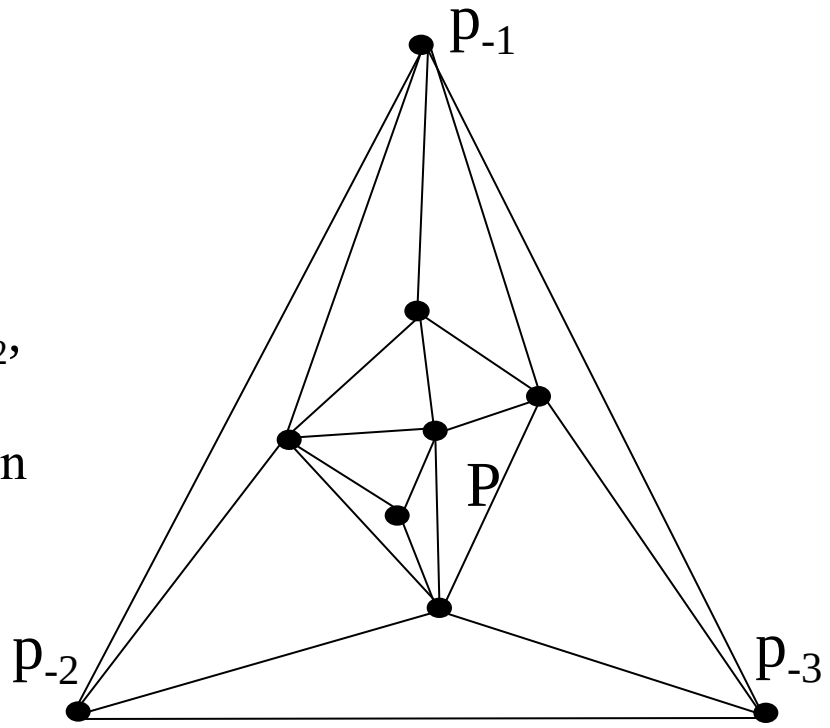
Twierdzenie.

W n-wierzchołkowym trójkątnym podziale możemy zlokalizować punkt w czasie O(log n) z pomocą struktury o rozmiarze O(n) stworzonej w czasie O(n).

Randomizowany algorytm znajdujący triangulację Delaunay.

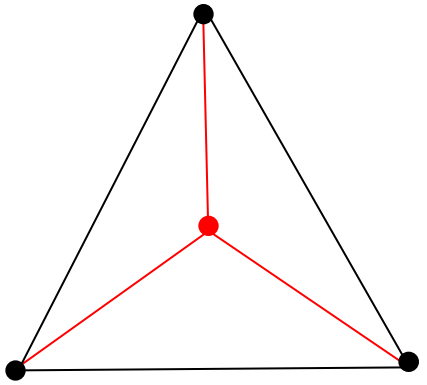
Załóżmy, że żadne cztery punkty z danego zbioru P = {p_1, …, p_n} nie są współokręgowe.

Zdefiniujmy też trzy dodatkowe punkty p_{-1}, p_{-2}, p_{-3}, które będą wierzchołkami trójkąta zawierającego zbiór P w swoim wnętrzu. W ten sposób będziemy mogli zastosować do opisu struktury podwójnie łączoną listę krawędzi.

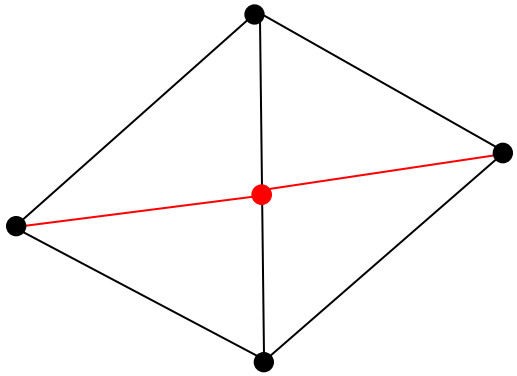


Podczas tworzenia triangulacji Delaunay trójkąty o wierzchołkach w punktach p_{-1}, p_{-2}, p_{-3}, traktujemy w szczególny sposób, aby nie wpływały na wynik. Po skończeniu pracy algorytmu usuwamy te punkty wraz z incydentnymi krawędziami.

W celu znalezienia triangulacji Delaunay stosujemy algorytm przyrostowy.

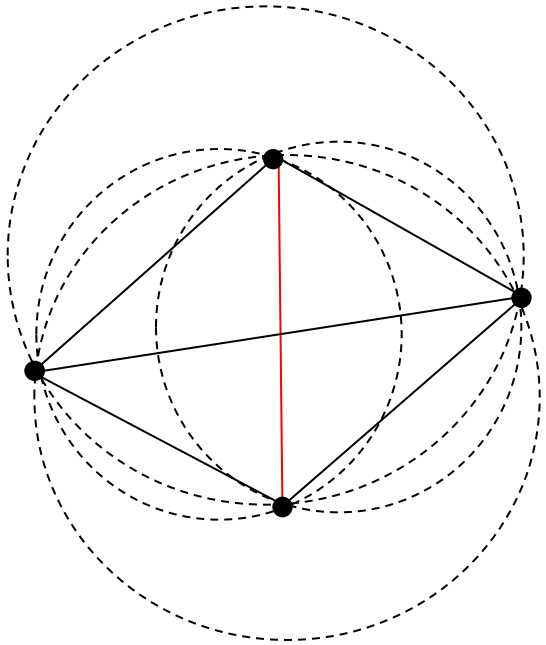


Po dodaniu kolejnego punktu mogą powstać trzy lub cztery nowe trójkąty, w zależności od tego, czy punkt ten znajduje się we wnętrzu aktualnego trójkąta, czy na jego krawędzi.



Definicja.

Niech $\Delta p_i p_j p_k$ oraz $\Delta p_i p_j p_l$ będą trójkątami należącymi do triangulacji. Krawędź $p_i p_j$ jest nielegalna (illegal edge), jeśli minimalny kąt w trójkątach $\Delta p_i p_j p_k$ i $\Delta p_i p_j p_l$ jest mniejszy od minimalnego kąta w trójkątach $\Delta p_i p_k p_l$ oraz $\Delta p_j p_k p_l$.



Krawędź legalizujemy zamieniając ją na drugą przekątną czworokąta tworzonego przez trójkąty przylegające do badanej krawędzi.

Jeśli któryś z nowopowstałych trójkątów nadal ma nielegalną krawędź, to kontynuujemy zamianę, aż do wyeliminowania wszystkich nielegalnych krawędzi. Ponieważ każda nowa krawędź ma koniec w dodawanym punkcie, więc proces ten jest skończony.

Algorytm (wstawianym punktem jest p_r, a badaną krawędzią p_ip_j)

LegalizeEdge(p_r,p_ip_j,T)

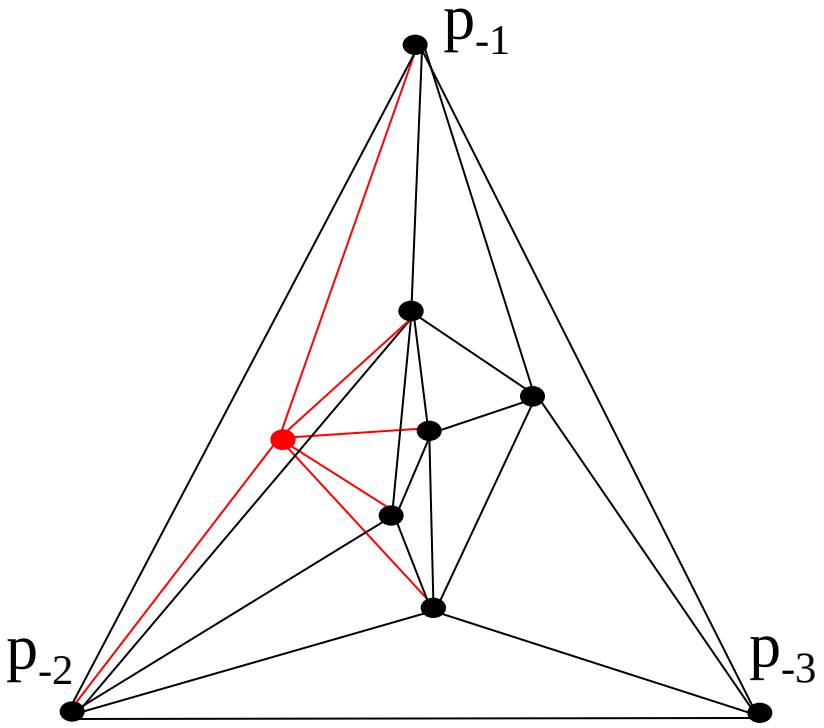
if p_ip_j jest nielegalna i $\Delta p_ip_jp_k$ należy do triangulacji

then

zastąp p_ip_j przez p_rp_k;

LegalizeEdge(p_r,p_ip_k,T);

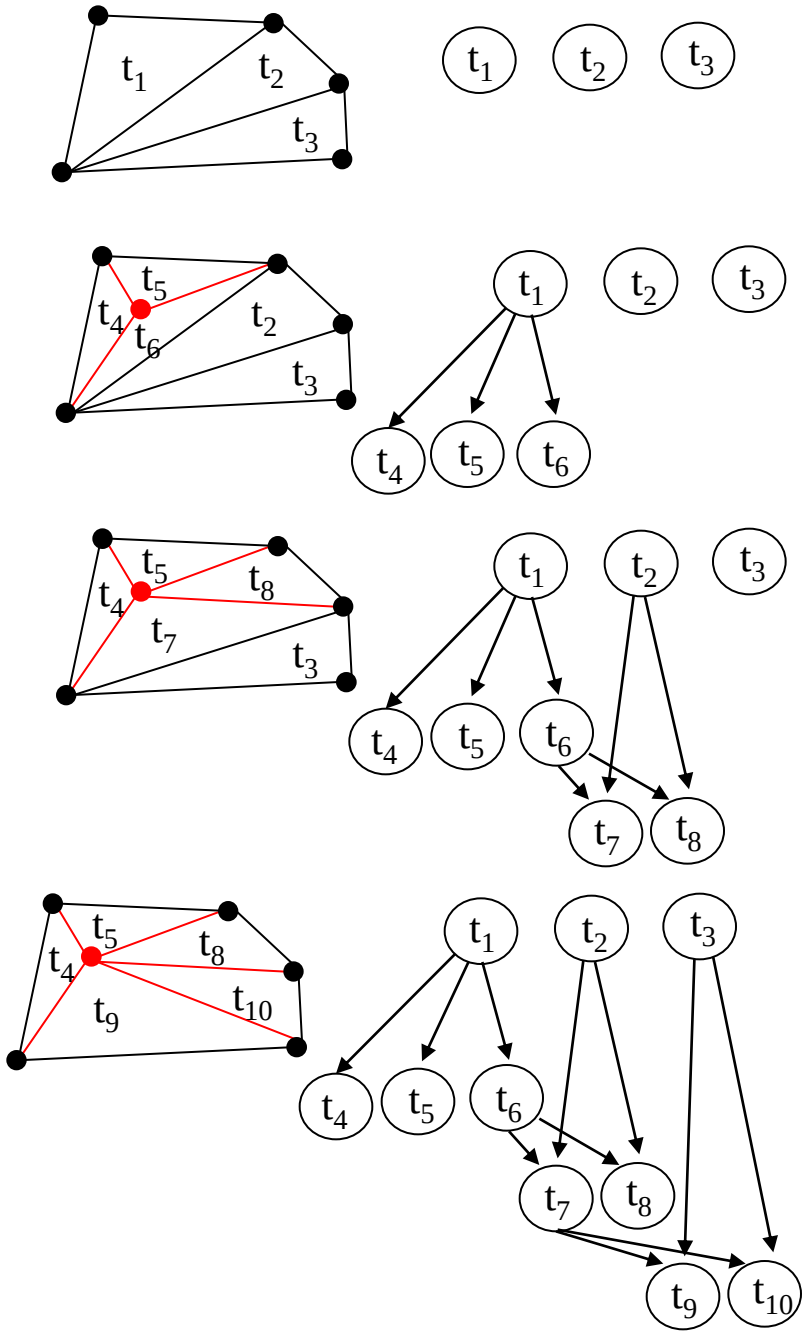
LegalizeEdge(p_r,p_kp_j,T);



Aby móc lokalizować trójkąt zawierający dodawany punkt tworzymy odpowiednią strukturę danych.

Jest to skierowany graf, w którym źródłem (wierzchołkiem o stopniu wejściowym 0) jest trójkąt $\Delta p_{-1}p_{-2}p_{-3}$, a przy dodawaniu kolejnych punktów powstają nowe wierzchołki odpowiadające trójkątom tworzonym zarówno przez dodanie punktu jak też przez zamianę krawędzi.

Każdy nowopowstały wierzchołek struktury danych jest końcem krawędzi łączących go z każdym wierzchołkiem odpowiadającym trójkątowi, z którego powstał.



Algorytm znajdujący triangulację Delaunay

zainicjuj triangulację T jako pojedynczy trójkąt $\Delta p_{-1}p_{-2}p_{-3}$ zawierający dany zbiór punktów P w swoim wnętrzu;

oblicz losową permutację $p_1, p_2, \ldots, p_n$ punktów z P;

for r:=1 **to** n **do**

znajdź trójkąt $\Delta p_i p_j p_k$ zawierający punkt p_r

if p_r leży wewnątrz trójkąta $\Delta p_i p_j p_k$

then legalizuj krawędzie triangulacji zaczynając od $p_i p_j$, $p_i p_k$ i $p_j p_k$

else legalizuj krawędzie triangulacji zaczynając od pozostałych krawędzi trójkątów, na których wspólnym boku leży p_r;

usuń z T punkty p_{-1}, p_{-2}, p_{-3} wraz ze wszystkimi incydentnymi krawędziami;

return T

Lemat.

Każda krawędź tworzona przy wstawianiu nowego punktu p_r jest krawędzią triangulacji Delaunay zbioru $\{p_{-1}, p_{-2}, p_{-3}, p_1, \ldots, p_{r-1}, p_r\}$.

Lemat.

Oczekiwana liczba trójkątów tworzonych przez algorytm wynosi 9n+1.

Dowód.

Dodawany punkt p_r ma początkowo stopień 3 lub 4 i tworzy tyle samo nowych trójkątów. Przy każdej zamianie krawędzi liczba trójkątów wzrasta o 2, a stopień punktu p_r o 1. Jeśli po wszystkich zamianach p_r ma stopień k, to powstały co najwyżej 2(k-3)+3 = 2k-3 nowe trójkąty.

Na mocy twierdzenia Eulera graf dla zbioru $\{p_{-1}, p_{-2}, p_{-3}, p_1, \ldots, p_{r-1}, p_r\}$ ma co najwyżej 3(r+3)-6 krawędzi. Trzy krawędzie należą do zewnętrznego trójkąta. Zatem suma stopni wierzchołków z P jest mniejsza niż 2(3(r+3)-9) = 6r. Czyli oczekiwany stopień punktu z P wynosi co najwyżej 6.

E(liczba trójkątów tworzonych w r-tym kroku) $\leq$ E(2deg p_r-3) = 2E(deg p_r)-3 $\leq$

$\leq 2\times6-3 = 9$.

Dodając trójkąt zewnętrzny otrzymujemy oczekiwaną liczbę trójkątów równą 9n+1.

Niech $K(\Delta)$ oznacza podzbiór punktów z P należący do koła opisanego na danym trójkącie Δ.

Lemat.

Jeśli P jest zbiorem punktów w położeniu ogólnym, to $E(\Sigma_\Delta \text{ card}(K(\Delta))) = O(n \log n)$, gdzie sumujemy po wszystkich trójkątach Delaunay stworzonych przez algorytm.

Dowód.

Niech P_r oznacza zbiór $\{p_1, \ldots, p_{r-1}, p_r\}$, a T_r triangulację zbioru $\{p_{-1}, p_{-2}, p_{-3}, p_1, \ldots, p_{r-1}, p_r\}$. Dla każdego r triangulacja jest jednoznacznie wyznaczona. Mamy $\Sigma_\Delta \text{ card}(K(\Delta)) = \Sigma^n_{r=1}(\Sigma_{\Delta \in Tr-T\{r-1\}} \text{ card}(K(\Delta)))$.

Niech $t(P_r,q)$ oznacza liczbę trójkątów $\Delta \in T_r$ takich, że punkt q należy do $K(\Delta)$, a $t(P_r,q,p_r)$ będzie liczbą trójkątów $\Delta \in T_r$ takich, że p_r jest wierzchołkiem trójkąta Δ wpisanego w koło zawierające q.

Każdy trójkąt stworzony w r-tej fazie algorytmu ma wierzchołek w p_r, więc $\Sigma_{\Delta \in Tr-T\{r-1\}} \text{ card}(K(\Delta)) = \Sigma_{q \in P-Pr} t(P_r,q,p_r)$.

Chwilowo ustalmy P_r. Wtedy wartość $t(P_r,q,p_r)$ zależy tylko od wyboru p_r. Ponieważ trójkąt $\Delta \in T_r$ ma wierzchołek w pewnym punkcie p należącym do P_r z prawdopodobieństwem nie większym niż 3/r, więc $E(t(P_r,q,p_r)) \leq 3t(P_r,q)/r$. Stąd

$E(\Sigma_{\Delta \in Tr\text{-}T\{r-1\}} \text{card}(K(\Delta))) = E(\Sigma_{q \in P\text{-}Pr} t(P_r,q,p_r)) \leq 3/r \, \Sigma_{q \in P\text{-}Pr} t(P_r,q)$.

Każdy $q \in P\text{-}P_r$ z jednakowym prawdopodobieństwem może stać się p_{r+1}, więc $E(t(P_r,p_{r+1})) = 1/(n\text{-}r) \, \Sigma_{q \in P\text{-}Pr} t(P_r,q)$, czyli

$E(\Sigma_{\Delta \in Tr\text{-}T\{r-1\}} \text{card}(K(\Delta))) \leq 3(n\text{-}r)/r \, E(t(P_r,p_{r+1}))$.

Ponieważ $t(P_r,p_{r+1})$ oznacza liczbę trójkątów $\Delta \in T_r$, dla których $p_{r+1} \in K(\Delta)$, więc są to dokładnie te trójkąty, które zostaną zmienione po dodaniu p_{r+1}.

Zatem $E(\Sigma_{\Delta \in Tr\text{-}T\{r-1\}} \text{card}(K(\Delta))) \leq 3(n\text{-}r)/r \, E(\text{card}(T_r\text{-}T_{r+1}))$.

Ale liczba trójkątów zmienionych po wstawieniu punktu p_{r+1} jest dokładnie o 2 mniejsza od liczby trójkątów stworzonych przez wstawienie tego punktu.

Stąd $E(\Sigma_{\Delta \in Tr\text{-}T\{r-1\}} \text{card}(K(\Delta))) \leq 3(n\text{-}r)/r \, (E(\text{card}(T_{r+1}\text{-}T_r)\text{-}2)$.

To samo zachodzi dla dowolnego zbioru P_r. Ponieważ oczekiwana liczba trójkątów tworzonych przez wstawienie p_{r+1} jest nie większa niż 6, więc

$E(\Sigma_{\Delta \in Tr\text{-}T\{r-1\}} \text{card}(K(\Delta))) \leq 12(n\text{-}r)/r$, czyli

$E(\Sigma_{\Delta} \text{card}(K(\Delta))) = E(\Sigma^{n}_{r=1}(\Sigma_{\Delta \in Tr\text{-}T\{r-1\}} \text{card}(K(\Delta)))) \leq 12 \, \Sigma^{n}_{r=1}(n\text{-}r)/r = O(n \log n)$.

Twierdzenie.

Triangulację Delaunay zbioru P zawierającego n punktów na płaszczyźnie można obliczyć w oczekiwanym czasie O(n log n) i z oczekiwanym rozmiarem pamięci O(n).

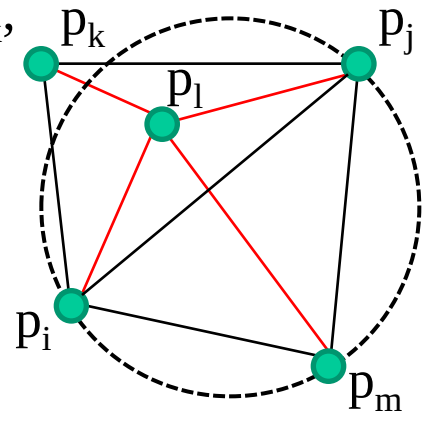
Dowód.

Oczekiwany rozmiar struktury danych jest liniowy.

Oczekiwany czas działania bez uwzględnienia czasu lokalizacji punktu jest również liniowy. Rozważmy lokalizację punktu p_r.

Czas lokalizacji wynosi O(1)+O(czas lokalizacji w trójkątach zawierających p_r, które zostały zmienione). Trójkąt $\Delta p_i p_j p_k$ może być usunięty z triangulacji, gdy:

- nowy punkt p_l wstawiono we wnętrzu lub na brzegu $\Delta p_i p_j p_k$,
- została zmieniona krawędź (np. $p_i p_j$) $\Delta p_i p_j p_k$.



W pierwszym przypadku $\Delta p_i p_j p_k$ był trójkątem wcześniejszej triangulacji, a w drugim - takim trójkątem był sąsiedni trójkąt ($\Delta p_i p_j p_m$) i wstawiono p_l. Wtedy okrąg opisany na $\Delta p_i p_j p_m$ zawiera p_l i p_r. Związmy koszt odwiedzenia trójkąta w trakcie lokalizacji p_r z trójkątem triangulacji zmienianym wraz z $\Delta p_i p_j p_k$.

Zatem takiemu trójkątowi przypisujemy czasy lokalizacji (jednostkowe) wszystkich punktów należących do koła opisanego na nim.

Dla trójkąta Δ takich punktów jest $\mathrm{card}(K(\Delta))$. Zatem oczekiwany czas lokalizacji wszystkich punktów wynosi (długość ścieżki wyszukiwania danego punktu = liczba kół zawierających dany punkt; liczba kół zawierających wszystkie punkty = liczba punktów zawieranych przez wszystkie koła):

$O(n) + O(\Sigma_{\Delta}\mathrm{card}(K(\Delta))) = O(n \log n)$.

Stąd oczekiwany czas działania algorytmu wynosi

$O(n) + O(n \log n) = O(n \log n)$.

Dziękuję za uwagę.

Ćwiczenia.

1. Udowodnij, że głębokość drzewa tworzonego w metodzie trapezowej jest nie większa niż $3\log W(U) + \lceil \log n \rceil + 3$.

2. Udowodnij, że rozmiar drzewa tworzonego w metodzie trapezowej wynosi $O(n \log n)$.

3. Podaj przykład podziału, dla którego drzewo tworzone w metodzie trapezowej ma rozmiar $\Omega(n \log n)$.

4. Wielokąt P nazywamy gwiaździstym, gdy zawiera punkt (centralny) p taki, że dla każdego punktu $q \in P$ odcinek pq jest zawarty w P. Wierzchołki wielokąta są kolejno stablicowane. W jakim czasie można sprawdzić, czy dany punkt należy do wielokąta gwiaździstego ? (znając punkt centralny lub nie)

5. Udowodnij, że liczba poziomów w grafie lokalizacji trójkątami jest logarytmiczna, jego rozmiar jest liniowy względem liczby wierzchołków, a każdy wierzchołek ma stopień ograniczony przez stałą.

6. Dany jest zbiór n punktów. Jak stworzyć strukturę (czas jej tworzenia jest nieistotny), dzięki której w czasie O(log n + k), gdzie k jest liczbą rozwiązań, będzie można zlokalizować wszystkie punkty oddalone od punktu zapytania o nie więcej niż d, gdzie k jest rozmiarem wyniku ?

7. Podaj przykład układu separatorów, które generują kwadratowy rozmiar odpowiedniej struktury względem liczby wierzchołków podziału płaszczyzny.

8. Jak stworzyć drzewo układu separatorów w czasie O(n log n) ?

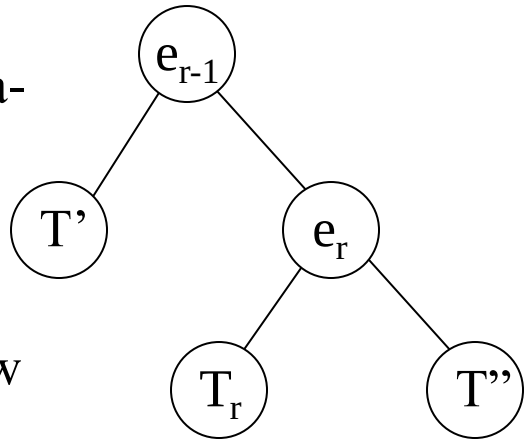
9. Udowodnij, że każda krawędź tworzona przy wstawianiu nowego punktu p_r jest krawędzią triangulacji Delaunay zbioru $\{p_{-1}, p_{-2}, p_{-3}, p_1, \ldots, p_{r-1}, p_r\}$.

1. Udowodnij, że głębokość drzewa tworzonego w metodzie trapezowej jest nie większa niż $3\log W(U) + \lceil \log n \rceil + 3$.

Dowód.

Indukcja po W(U).

Dla W(U) = 1 jedyny wierzchołek drzewa odpowiadający poziomej prostej przechodzącej przez wierzchołek podziału znajduje się w korzeniu T_r.



Poddrzewa T_r oraz T’ i T” są zrównoważonymi drzewami zawierającymi co najwyżej n wierzchołków odpowiadających trapezom.

Zatem głębokość całego drzewa wynosi co najwyżej $\lceil \log n \rceil + 3 = 3\log W(U) + \lceil \log n \rceil + 3$.

Załóżmy, że W(U) = k i dla wszystkich pasów U’ takich, że W(U’) < k teza indukcji jest prawdziwa.

Niech pas U_r odpowiada poddrzewu T_r. Mamy dwa przypadki.

1) $W(U_r) \leq k/2$. Ponieważ dla pasów U’ i U” odpowiadających poddrzewom T’ i T” mamy W(U’) < k/2 i W(U”) < k/2 ,

więc głębokości poddrzew T', T'' i T_r szacują się na mocy założenia indukcyjnego przez $3\log(k/2) + \lceil \log n \rceil + 3 = 3\log k + \lceil \log n \rceil$, co daje oszacowanie głębokości całego drzewa przez $3\log k + \lceil \log n \rceil + 2 \leq 3\log W(U) + \lceil \log n \rceil + 3$.

2) $W(U_r) > k/2$. Ponieważ $W(U') < k/2$ i $W(U'') < k/2$, więc dla T' i T'' mamy te same oszacowania co powyżej. Korzeniem poddrzewa T_r jest (podobnie jak dla przypadku $W(U) = 1$) wierzchołek odpowiadający poziomej prostej poprowadzonej przez medianę punktów podziału należących do U_r, więc wagi pasów odpowiadających lewemu i prawemu poddrzewu tego drzewa szacują się przez $W(U_r)/2 \leq k/2$.

Zatem na mocy założenia indukcyjnego ich głębokość szacuje się przez $3\log(k/2) + \lceil \log n \rceil + 3 = 3\log k + \lceil \log n \rceil$, co daje oszacowanie głębokości całego drzewa przez $3\log k + \lceil \log n \rceil + 3 \leq 3\log W(U) + \lceil \log n \rceil + 3$.