Triangulacja - porównanie metod

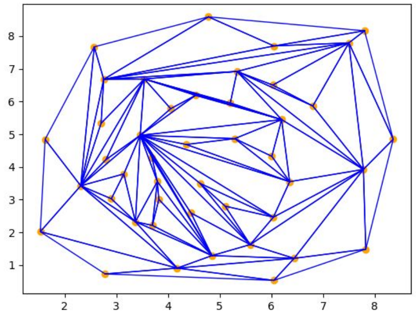
Tło z: https://www.shadertoy.com/view/MdfBzl

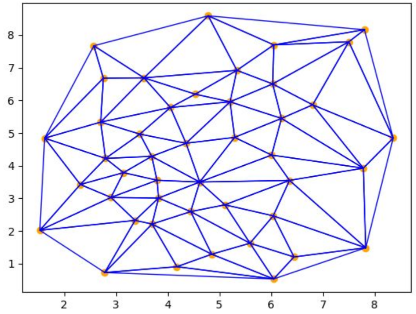
Triangulacja Delaunaya

Triangulacja zbioru punktów to proces dzielenia zbioru punktów na trójkąty, tak aby wszystkie punkty były w wierzchołkach tych trójkątów, a trójkąty się nie nakładały.

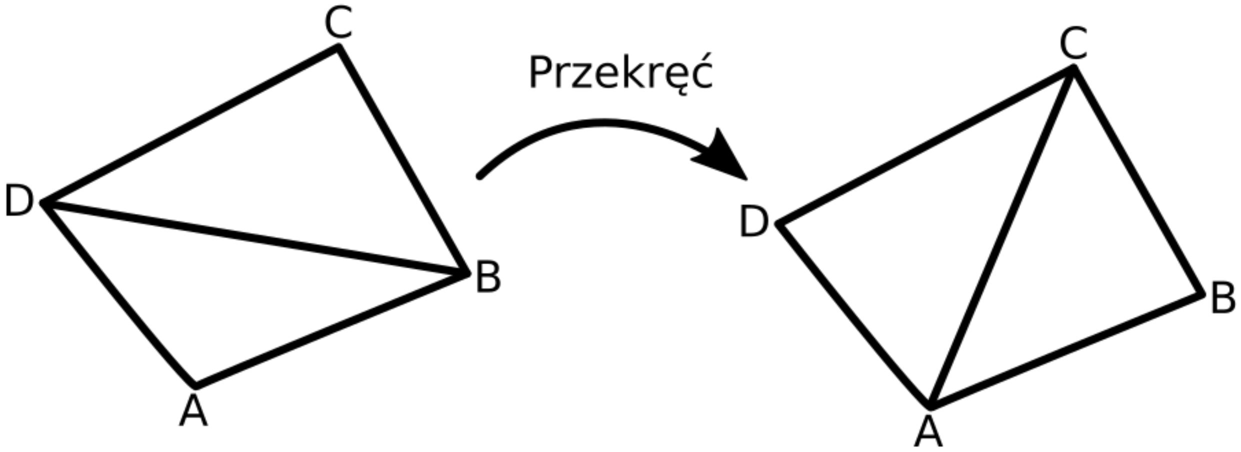
Triangulacja Delaunaya to szczególny rodzaj triangulacji, który maksymalizuje minimalny kąt w trójkątach, co zapobiega powstawaniu wąskich trójkątów.

Uwaga: zwykła triangulacja Delaunaya nie służy do triangulacji wielokątów, będzie to dopiero możliwe po dodaniu operacji odzyskiwania krawędzi.





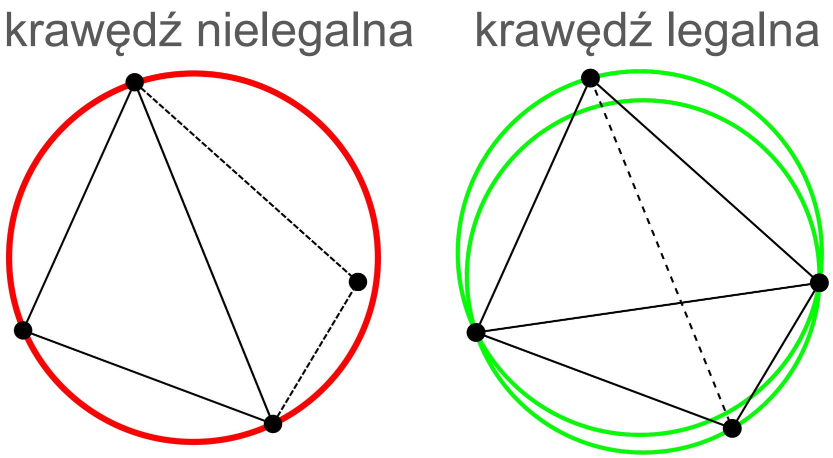
Przekręcanie krawędzi to operacja polegająca na zamienianiu przekątnej w czworokącie wypukłym.



Krawędź legalna i test koła opisanego

Krawędź legalna to taka krawędź, której przekręcenie nie zmieni lub zmniejszy najmniejszy kąt. Można udowodnić, że jeżeli wszystkie krawędzie w triangulacji są legalne, to jest to triangulacja Delaunaya.

Aby sprawdzić, czy krawędź jest legalna, wystarczy sprawdzić, czy koło opisane na trzech punktach czworokąta w tym obu punktach krawędzi zawiera czwarty punkt.



https://en.wikipedia.org/wiki/Delaunay_triangulation

Algorytm naiwny (Local Optimization Procedure)

Triangulację Delaunaya można uzyskać poprzez stworzenia dowolnej triangulacji zbioru punktów, a następnie przekręcaniu nielegalnych krawędzi tak długo, aż wszystkie będą legalne.

Ponieważ zmiana krawędzi na legalną zwiększa leksykograficznie wektor kątów triangulacji, to proces ten na pewno kiedyś się skończy.

Algorytm inkrementalny

Algorytm inkrementalny polega na dodawaniu kolejnych punktów i utrzymywaniu triangulacji Delaunaya. Zaczynamy od stworzenia supertrójkąta, w którym zmieszczą się wszystkie punkty.

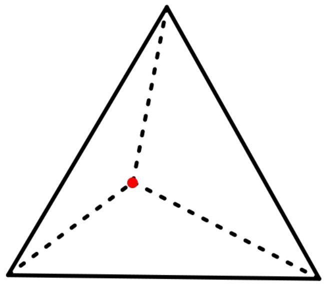
Następnie operacja dodania punktu polega na:

1. Odnalezieniu trójkąta, w którym znajduje się punkt.

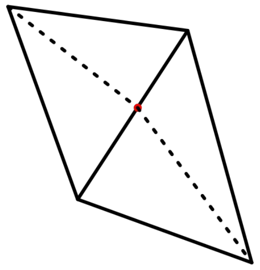
2. Dodaniu odpowiednich trójkątów powstałych na skutek wstawienia punktu.

3. Przywróceniu legalności krawędzi poprzez rekurencyjne przekręcanie krawędzi, które przestały być legalne.

Dodawanie punktu w trójkącie:



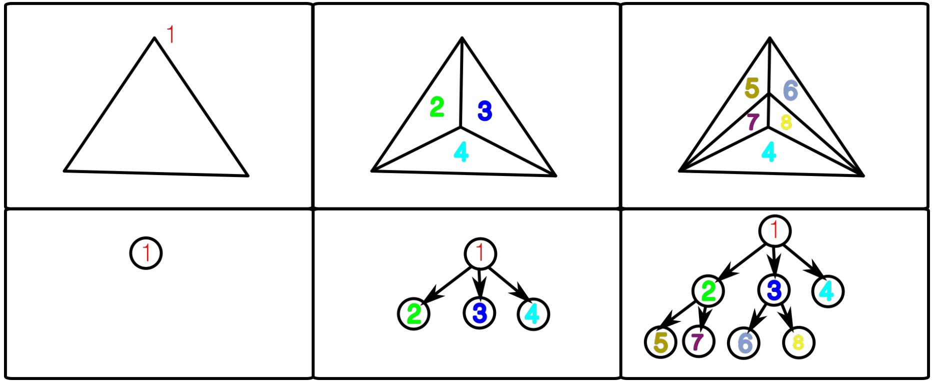
Dodawanie punktu na krawędzi:



Sposób odnajdywania trójkąta

Do lokalizacji trójkąta zawierającego punkt używany jest acykliczny graf skierowany, w którym występują trójkąty, które historycznie istniały podczas triangulacji, a krawędzie prowadzą od wcześniej istniejących trójkątów do nowo powstałych na ich miejscu.

Aby odnaleźć szukany trójkąt, zaczynamy od wierzchołka supertrójkąta i przechodzimy dowolną krawędzią prowadzącą do trójkąta zawierającego dodawany punkt. Ostatecznie, gdy stopień wyjściowy wierzchołka równy jest zero, znaleziony został odpowiedni trójkąt.



Złożoność

Dodatkowo, aby poprawić oczekiwaną złożoność tego algorytmu, punkty dodawane są w losowej kolejności. Skutkuje to oczekiwaną złożonością $\Theta(n\log n)$. Jednak pesymistyczna złożoność to $O(n^2)$. Wynika to z tego, że pesymistyczna złożoność zarówno operacji odnajdywania trójkąta, jak i wstawiania punktu to $O(n)$.

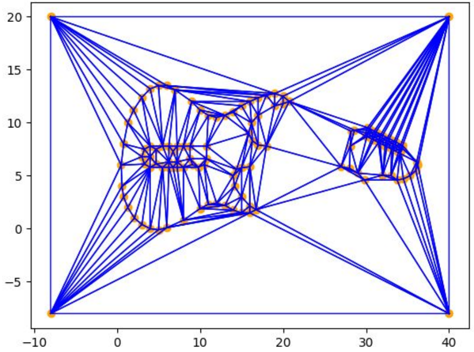
Triangulacja z ograniczeniami

Triangulacja Delaunaya z ograniczeniami dodaje jeszcze możliwość wymuszenia istnienia niektórych krawędzi.

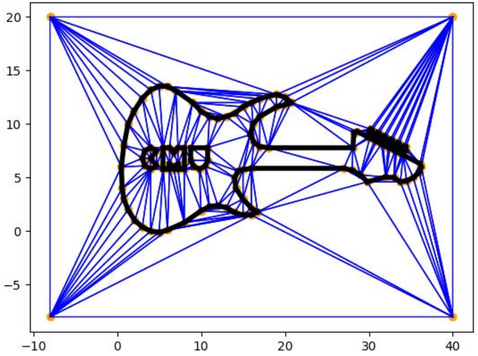
Wymuszone krawędzie nie mogą się przecinać (inaczej wymagałoby to dodania punktów).

Triangulacja Delaunaya z ograniczeniami nie jest triangulacją Delaunaya, jednak również maksymalizuje minimalne kąty, biorąc pod uwagę zadane ograniczenia.

Triangulacja Delaunaya bez ograniczeń:



Triangulacja Delaunaya z ograniczeniami:



Algorytm

1. Przeprowadzamy zwykłą triangulację Delaunaya.

Dla każdej wymuszonej krawędzi:

1. Jeżeli zadana krawędź znajduje się już w triangulacji, nie trzeba nic robić. W przeciwnym razie:

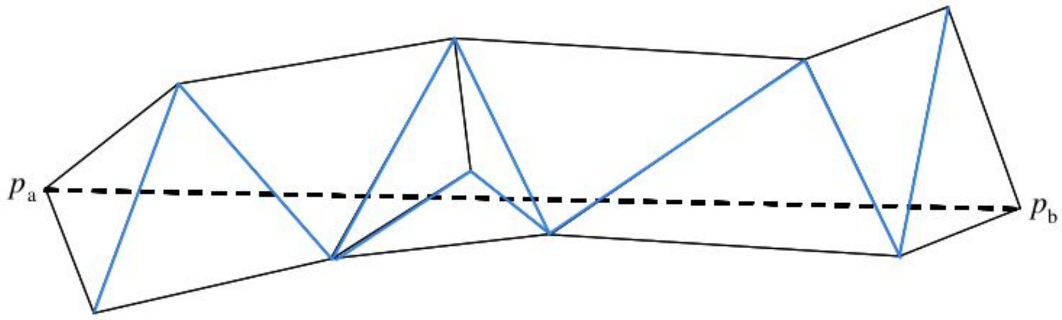
2. Znajdujemy krawędzie przecinające się z krawędzią wymuszoną.

3. Odpowiednio przekręcamy krawędzie, tak żeby przestały przecinać odzyskiwaną krawędź.

4. Przywracamy triangulację Delaunaya, nie przekręcamy wymuszonej krawędzi. Można udowodnić, że wystarczy jedynie sprawdzić legalność przekręconych krawędzi. Używamy do tego algorytmu LOP.

Znajdowanie krawędzi przecinających krawędź odzyskiwaną

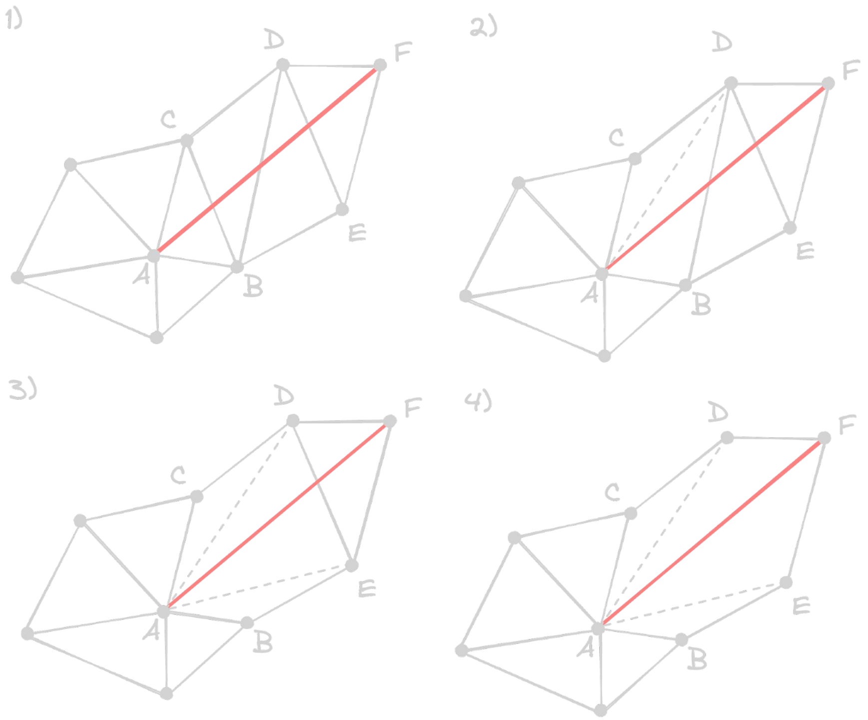
Aby znaleźć krawędzie przecinające się z zadaną krawędzią, najpierw przechodzimy przez krawędzie wychodzące z pierwszego wierzchołka wymuszonej krawędzi, w celu znalezienia pierwszej krawędzi przecinającej odzyskiwaną. Następnie przechodzimy po siatce do drugiego wierzchołka, znajdując po drodze wszystkie krawędzie przecinające zadaną krawędź.



Wymuszanie krawędzi za pomocą przekręcania

Dlaczego to jest możliwe: można udowodnić, że zawsze istnieje wierzchołek, który można pozbawić wszystkich krawędzi przecinających krawędź odzyskiwaną wychodzących z niego, za pomocą przekręceń. Można po kolei wyeliminować wszystkie takie wierzchołki i wtedy odzyskamy zadaną krawędź.

Faktyczna implementacja tego algorytmu jest jednak nieco inna, utrzymywana jest lista krawędzi, przecinających się z wymuszoną. Następnie po kolei przekręcane są krawędzie, jeżeli przekręcenie nie eliminuje przecięcia, to nowa krawędź jest dodawana z powrotem na koniec listy.



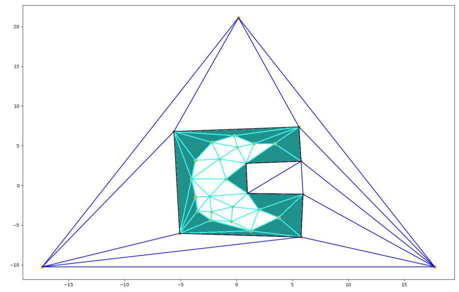
https://tchayen.com/constrained-delaunay-triangulation-from-a-paper

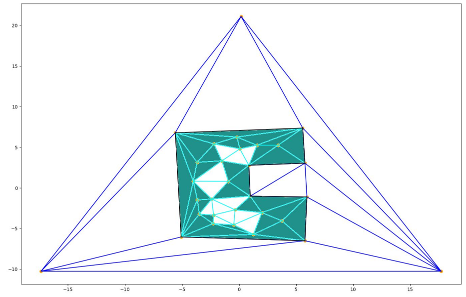
Usuwanie zewnętrznych

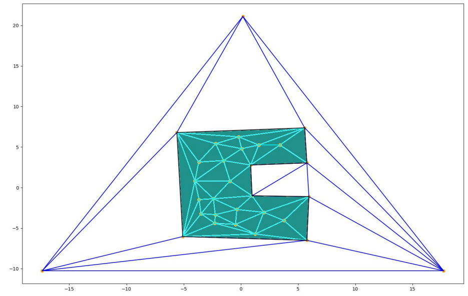
Przy usuwaniu trójkątów zewnętrznych, zakładamy, że z lewej strony od wymuszonych krawędzi znajdują się wnętrza wielokątów, natomiast z prawej zewnętrza.

Takie podejście pozwala na triangulację wielokątów podanych w kierunku przeciwnym do ruchu wskazówek zegara.

Aby znaleźć trójkąty wewnętrzne, używamy algorytmu BFS z wielu źródeł: z każdego trójkąta, będącego po lewej stronie od wymuszonej krawędzi. Następnie przechodzimy po sąsiadujących trójkątach, jednak nie przekraczamy wymuszonych krawędzi.







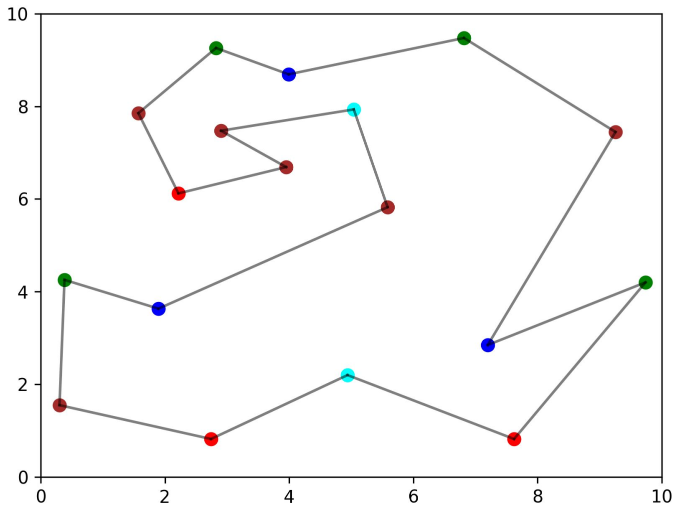
Triangulacja przez podział na wielokąty monotoniczne

Etapy algorytmu:

1. Klasyfikacja wierzchołków.
2. Algorytm zamiatania dzielący wielokąt na wielokąty monotoniczne.
3. Triangulacja monotonicznych wielokątów.

Klasyfikacja wierzchołków wielokąta

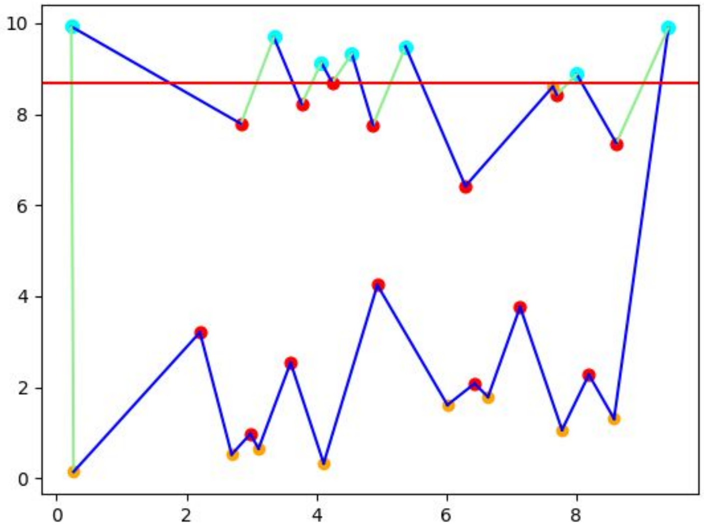
- w. początkowy - obaj sąsiedzi leżą poniżej, a kąt wewnętrzny wypukły,
- w. końcowy - obaj sąsiedzi leżą powyżej i kąt wewnętrzny wypukły,
- w. łączący - obaj sąsiedzi leżą powyżej i kąt wewnętrzny wklęsły,
- w. dzielący - obaj sąsiedzi leżą poniżej i kąt wewnętrzny wklęsły,
- w. prawidłowy - pozostałe przypadki (jeden sąsiad powyżej, drugi poniżej)



Triangulacja wielokątów monotonicznych

Elementy algorytmu zamiatania:

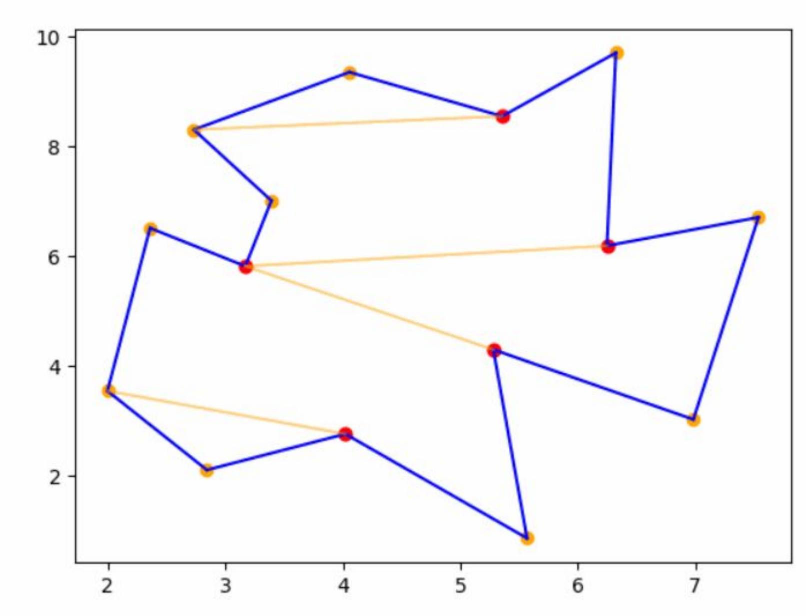
- miotła: prosta równoległa do osi OX, zamiata w dół;
- zdarzenia: wierzchołki wielokąta,
- struktura stanu: uporządkowany po x ciąg aktywnych krawędzi.



Obsługa zdarzeń

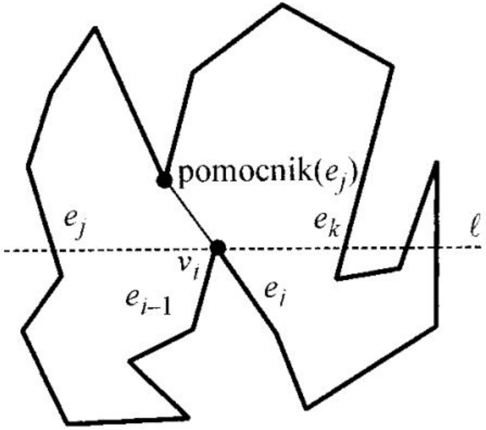
Idea algorytmu: eliminacja wierzchołków łączących i dzielących poprzez dodawanie przekątnych.

Miotła przechodząc przez płaszczyznę, napotyka się na kolejne wierzchołki. W zależności od ich typu wykonuje odpowiednią procedurę.



Obsługa zdarzeń - pojęcia i oznaczenia

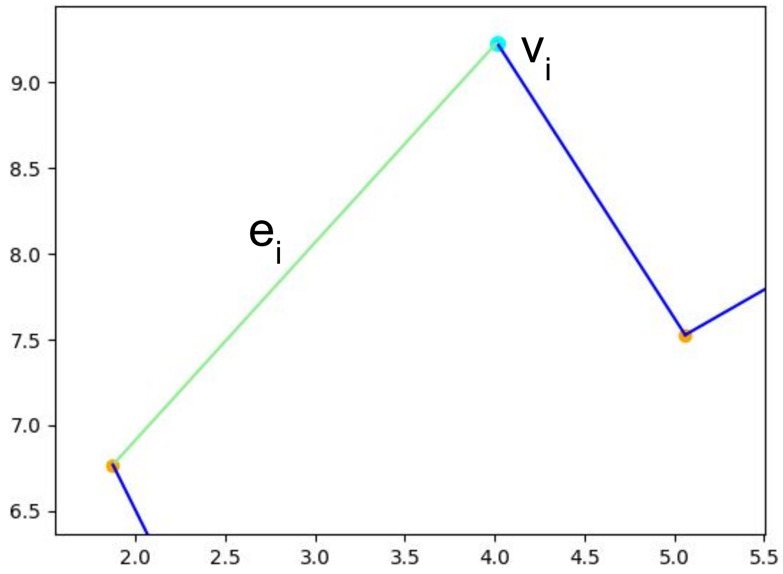
- pomocnik(e) - najniższy wierzchołek powyżej miotły taki, że odcinek poziomy łączący e z tym wierzchołkiem leży wewnątrz wielokąta,
- T - struktura stanu,
- P - wielokąt wejściowy,
- e_i - krawędź wychodząca z v_i w kolejności CCW.



Zdarzenia - wierzchołek początkowy

Procedura gdy v_i - początkowy:

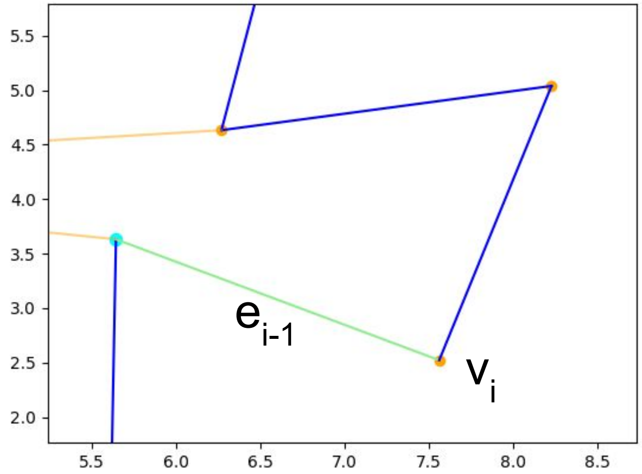
- wstaw e_i do T,
- ustaw pomocnika e_i na v_i.



Zdarzenia - wierzchołek końcowy

Procedura gdy v_i - końcowy:

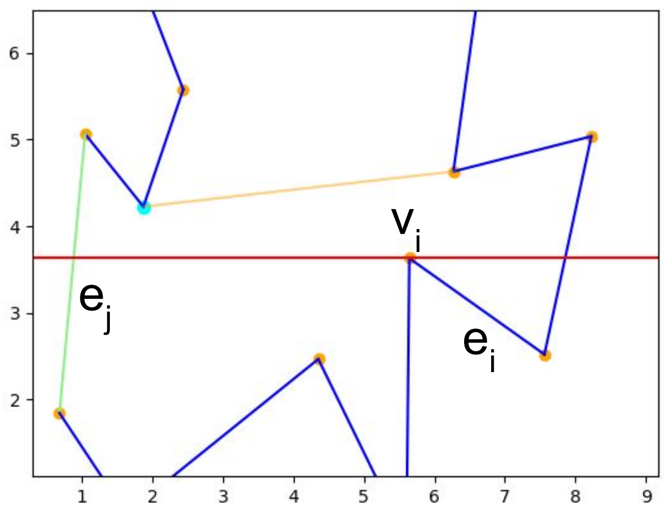
- jeżeli pomocnik(e_{i-1}) jest wierzchołkiem łączącym wstaw przekątną między v_i i pomocnik(e_{i-1}),
- usuń e_{i-1} z T.



Zdarzenia - wierzchołek dzielący

Procedura gdy v_i - dzielący:

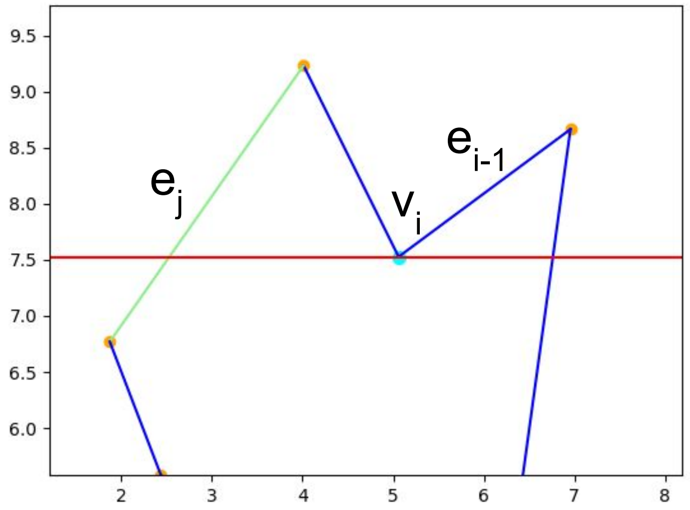
- szukaj w T krawędzi e_j bezpośrednio na lewo od v_i,
- wstaw przekątną łączącą v_i z pomocnik(e_j),
- ustaw pomocnik(e_j) na v_i,
- ustaw pomocnik(e_i) na v_i,
- wstaw e_i do T.



Zdarzenia - wierzchołek łączący

Procedura gdy v_i - łączący:

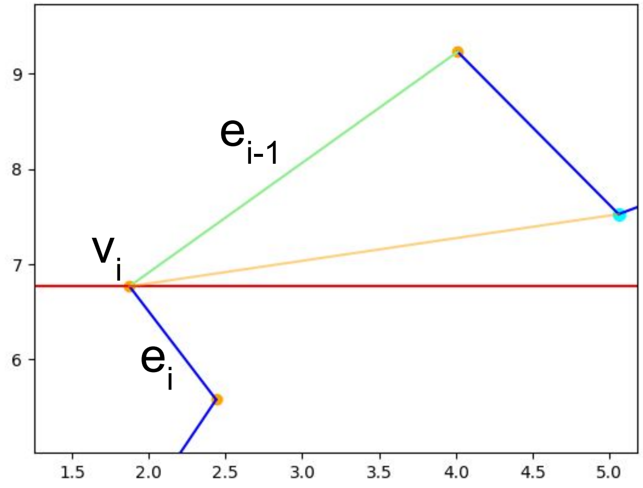
- jeżeli pomocnik(e_{i-1}) jest łączący to wstaw przekątną łączącą v_i z pomocnikiem(e_{i-1}),
- usuń e_{i-1} z T,
- szukaj w T krawędzi e_j bezpośrednio na lewo od v_i,
- jeżeli pomocnik(e_j) jest łączący to wstaw przekątną łączącą v_i z pomocnikiem(e_j),
- ustaw pomocnik(e_j) na v_i.



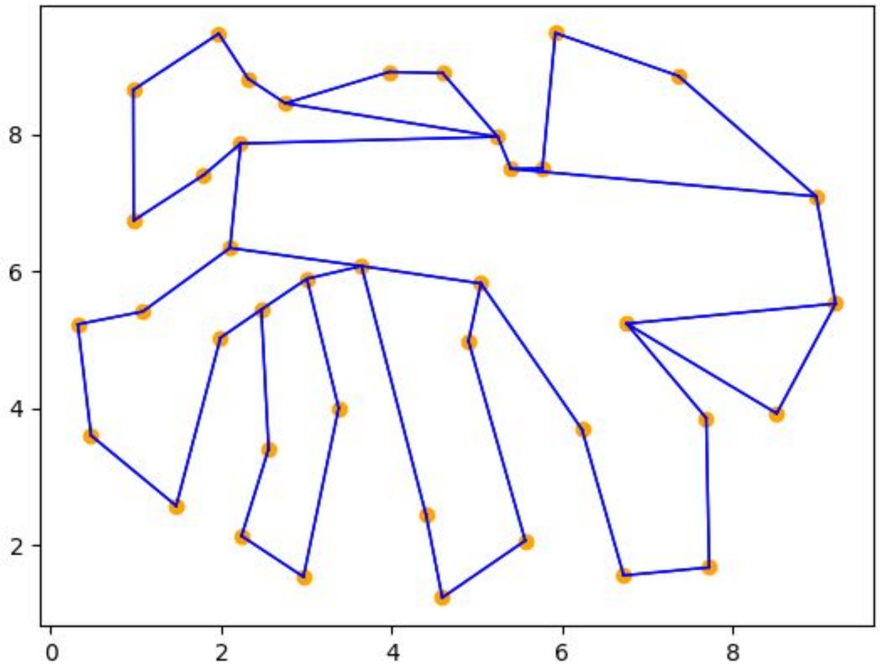
Zdarzenia - wierzchołek prawidłowy

Procedura gdy v_i - prawidłowy:

1. intP po prawej od v_i:
   • jeżeli pomocnik(e_{i-1}) jest łączący to wstaw przekątną łączącą v_i z pomocnikiem(e_{i-1}),
   • usuń e_{i-1} z T,
   • wstaw e_i do T i ustaw pomocnik(e_i) na v_i.
2. intP po lewej od v_i:
   • szukaj w T krawędzi e_j bezpośrednio na lewo od v_i,
   • jeżeli pomocnik(e_j) jest łączący to wstaw przekątną łączącą v_i z pomocnikiem(e_j),
   • ustaw pomocnika(e_j) na v_i.



Przykładowy rezultat podziału na wielokąty monotoniczne



Triangulacja wielokąta monotonicznego

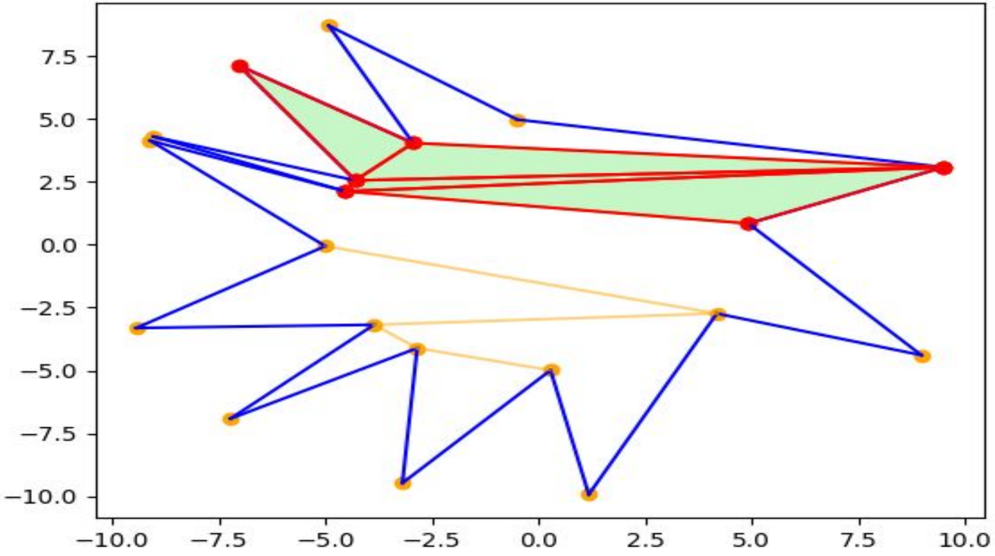
Etapy algorytmu:

1. Znalezienie lewego i prawego łańcucha wielokąta.
2. Sortowanie wierzchołków po rzędnych malejąco.
3. Przejście po punktach i tworzenie trójkątów.

Logika dodawania przekątnych podczas triangulacji wielokąta monotonicznego

- wstaw na stos dwa pierwsze wierzchołki z posortowanej po y malejąco listy wierzchołków,
- wykonuj następującą procedurę do wyczerpania wierzchołków:
  1. Jeżeli ostatni element stosu jest na tej samej gałęzi co aktualnie rozpatrywany wierzchołek:
     - dopóki trójkąt tworzony przez dwa ostatnie elementy stosu i rozpatrywany wierzchołek zawiera się w wielokącie dodawaj trójkąty do listy wynikowej i usuwaj wierzchołki ze stosu.
  2. Jeżeli ostatni element stosu jest na drugiej gałęzi:
     - dokładaj do listy wynikowej kolejne trójkąty tworzone przez rozpatrywany wierzchołek i dwa ostatnie elementy stosu, aż do wyczerpania wierzchołków na stosie. Na końcu wstaw dwa ostatnio usunięte wierzchołki z powrotem do stosu.

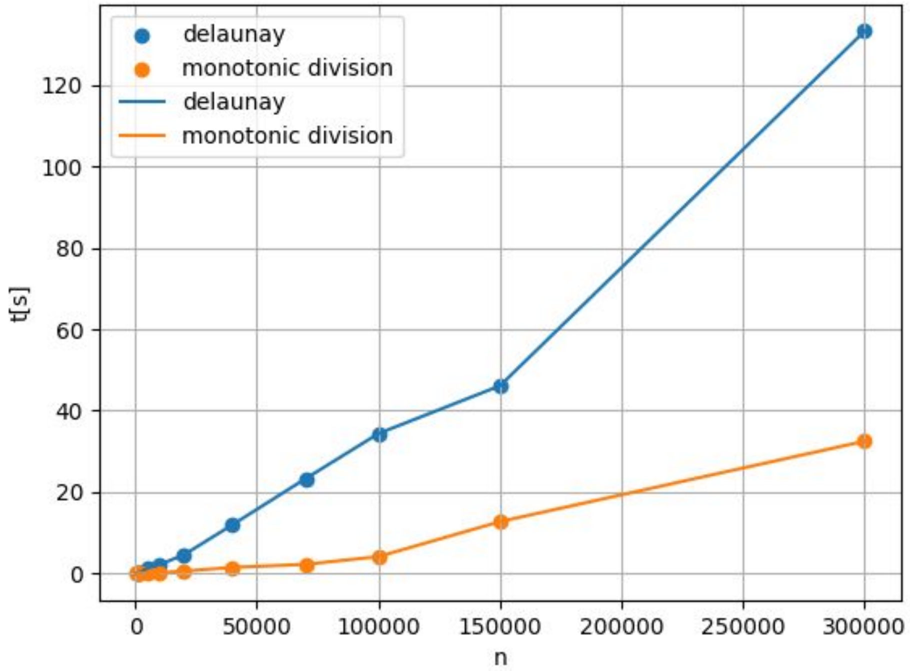
Przykładowy wielokąt po podziale na wielokąty monotoniczne i striangulowaniu jednego wielokąta monotonicznego



Porównanie wydajności

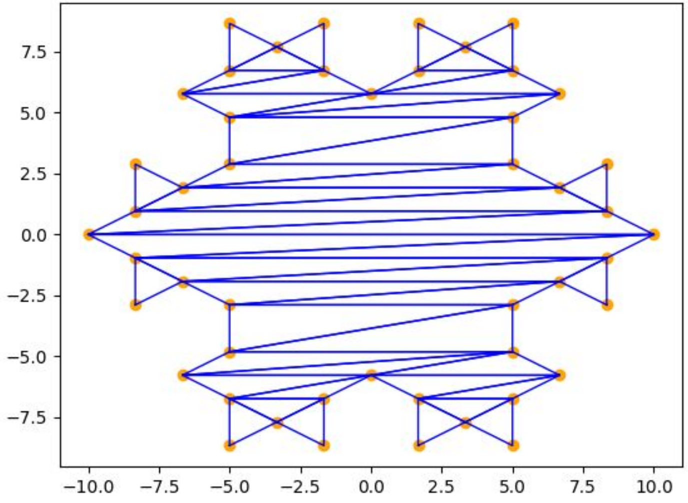
Zbiory testowe tworzone były na zasadzie losowego generowania punktów na dwóch okręgach o określonych promieniach i odpowiednim łączeniu ich.

Z wykresu wydajności można odczytać, że triangulacja za pomocą podziału na wielokąty monotoniczne jest od około 5 do 10 razy szybsza.

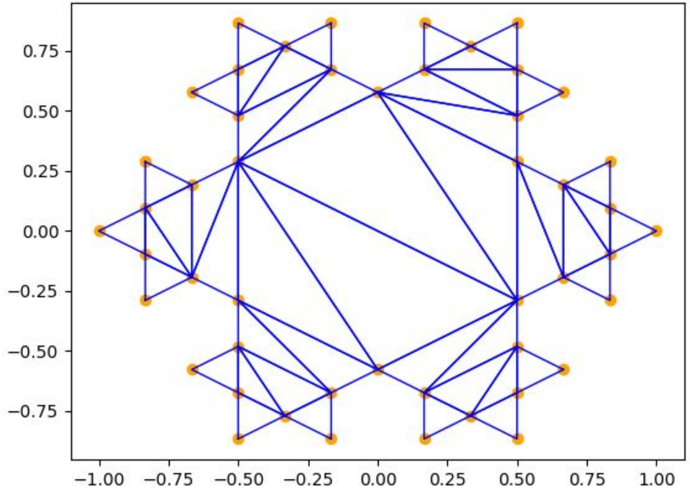


Porównanie jakości triangulacji - przykład

Jako kryterium oceny triangulacji wybrano średnią z minimalnych kątów dla każdego powstałego trójkąta.

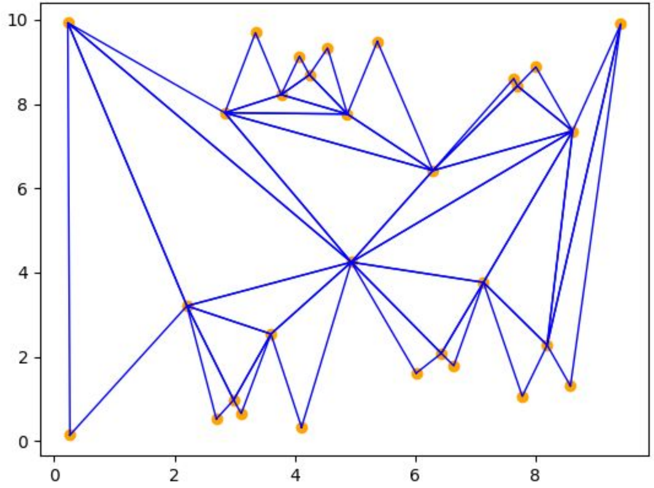


Triangulacja przez podział na wielokąty monotoniczne

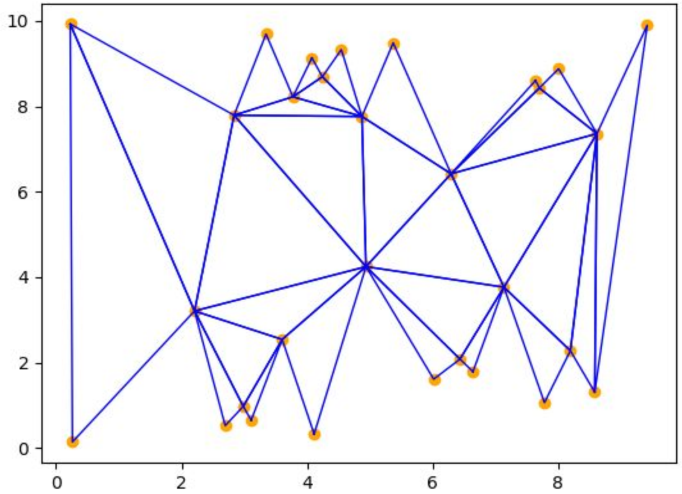


Triangulacja Delaunaya

Inne wielokąty

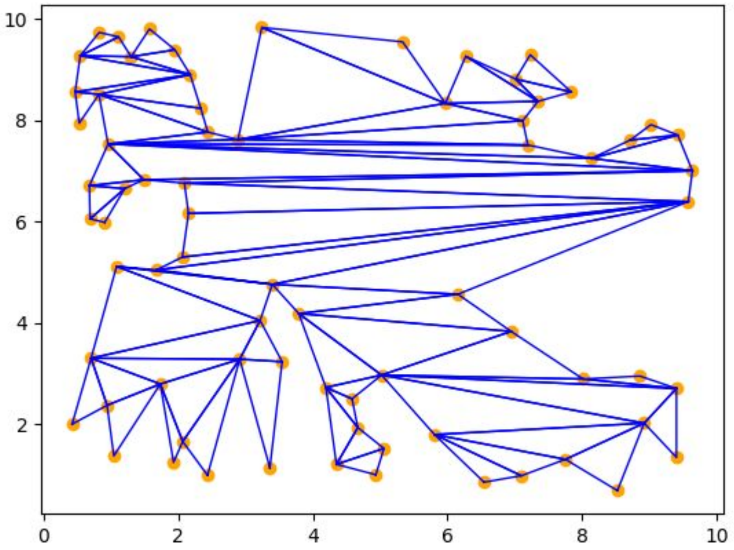


Triangulacja przez podział na wielokąty monotoniczne

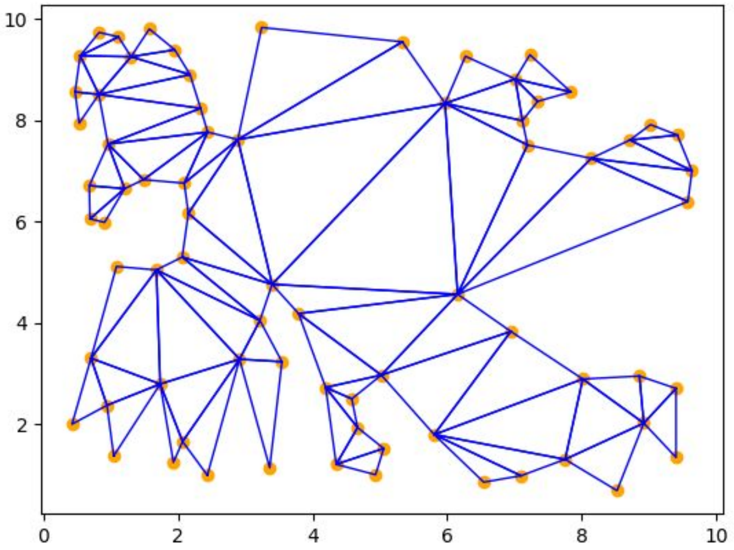


Triangulacja Delaunaya

Inne wielokąty



Triangulacja przez podział na wielokąty monotoniczne



Triangulacja Delaunaya

Tabela porównująca jakość triangulacji wg kryterium

Wielokąt	Delaunay (°)	Podział na monotoniczne (°)
A	31.02	21.81
B	24.19	21.30
C	38.90	22.98

Podsumowanie

1. Porównanie czasowe: triangulacja przez podział na wielokąty monotoniczne wypada znacznie lepiej (5-10 razy szybsza).
2. Porównanie jakościowe: triangulacja Delaunaya tworzy bardziej jakościowe trójkąty. Stara się ona unikać trójkątów wąskich i podłużnych (o małym kącie minimalnym).

Dziękujemy za uwagę!
Jan Chadziński i Emil Żychowicz

Źródła:

1. **Computational Geometry: Algorithms and Applications** autorstwa Marka de Berga, Otfrieda Cheonga, Marca van Krevelda i Marka Overmarsa (Trzecie wydanie).
2. **Wykłady Algorytmy Geometryczne** autorstwa dr Barbary Głut.
3. **Triangulations and Applications** autorstwa Øyvinda Hjelle i Mortena Dæhlena.
4. Artykuł o algorytmach geometrii płaskiej dostępny na stronie https://cp-algorithms.com/geometry/planar.html.
5. Wpis na blogu o triangulacji Delaunaya, dostępny pod adresem https://ianthehenry.com/posts/delaunay/.
6. Artykuł zatytułowany **A Fast Algorithm for Generating Constrained Delaunay Triangulations** autorstwa S. W. Sloana, dostępny pod tym linkiem.
7. Prezentacja na temat **Algorytmów Podziału Wielokątów**, dostępna pod adresem https://www.cs.jhu.edu/~misha/Spring16/04.pdf.
8. Artykuł omawiający triangulację Delaunaya z ograniczeniami, dostępny na stronie https://tchayen.com/constrained-delaunay-triangulation-from-a-paper.