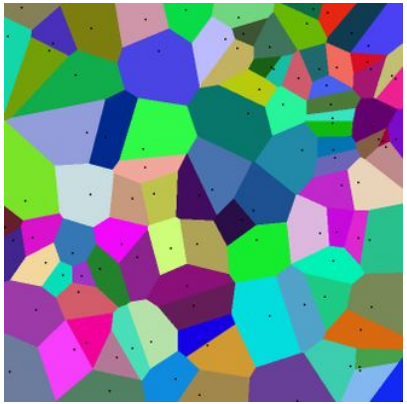
Wieloboki Voronoi

Jakub Ciszewski, Olaf Fertig

Czym jest diagram Voronoi?

Diagram Woronoja jest konstrukcją tworzoną ze zbioru punktów rozmieszczonych na określonej powierzchni. Polega na takim podziale badanego obszaru na części (komórki), że każdy punkt położony wewnątrz danej komórki diagramu leży bliżej węzła, znajdującego się w tej komórce, niż jakiegokolwiek innego węzła sieci.



Diagramy Voronoi dla różnych metryk

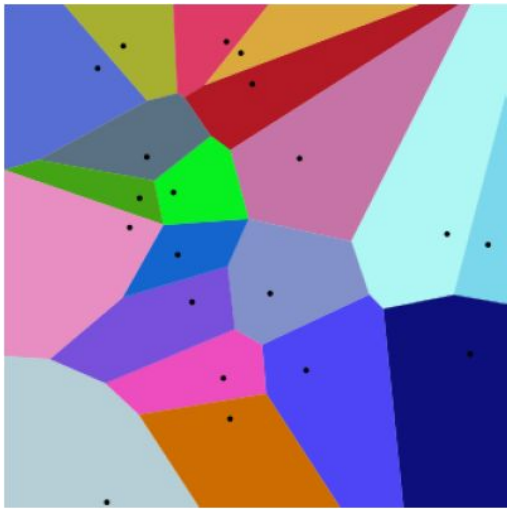


Diagram Voronoi dla metryki euklidesowej

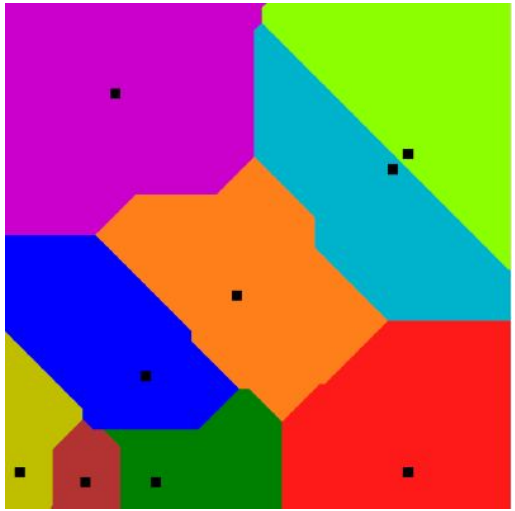


Diagram Voronoi dla metryki maksimum

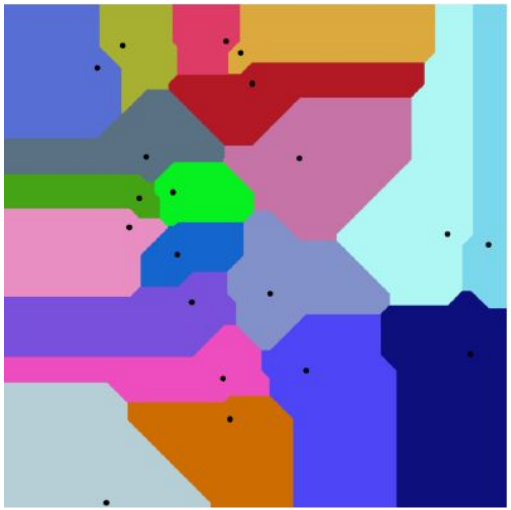
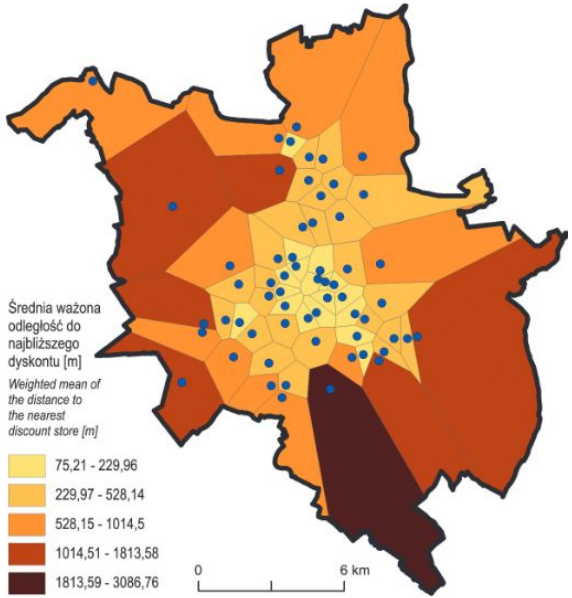


Diagram Voronoi dla metryki taksówkowej

Przykład zastosowania diagramu Voronoi

Przykładem zastosowania diagramu Voronoi może być podział pewnego obszaru przez sieć sklepów. Punktami, dla których obliczamy diagram, są położenia sklepów sieci na mapie, a wygenerowane komórki diagramu w połączeniu z gęstością ich zaludnienia pozwalają określić obszary, w których sklepów jest za mało, a także miejsca na mapie, w których sklepów jest za dużo.



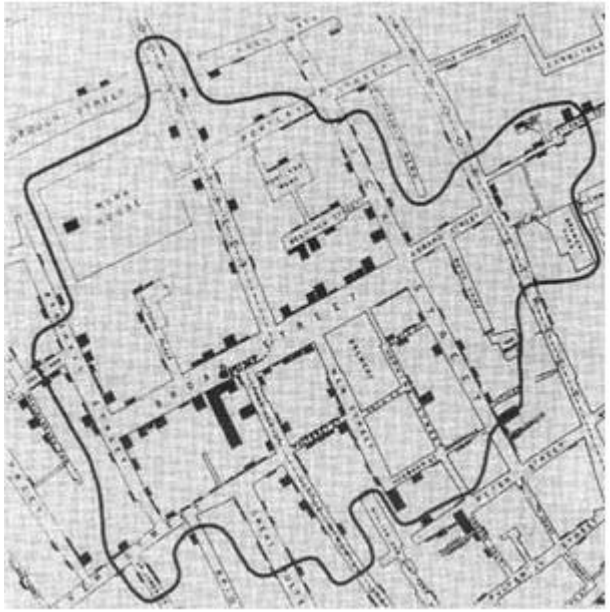
Zróżnicowanie przestrzenne przeciętnego dystansu do najbliższego dyskontu Biedronka z miejsc zamieszkania ludności na terenie Poznania

Zastosowania diagramu Voronoi

- Antropologia - badanie obszarów wpływów innych kultur
- Krystalografia - tłumaczenie struktur niektórych kryształów i metali
- Ekologia - badanie konkurencji między gatunkami roślin
- Ekonomia - tworzenie modeli rynkowych na rynku amerykańskim

Warto wspomnieć!

Jednym z pierwszych zastosowań diagramów Woronoja było stworzenie przez Johna Snowa diagramu korzystającego z podobnych założeń do diagramu Woronoja (było to przed wprowadzeniem pojęcia diagram Woronoja do literatury), który pozwolił na namierzenie źródła epidemii cholery na Broad Street w 1854 r. w Soho w Anglii. Wykazał on korelacje pomiędzy studnią, z których czerpali wodę mieszkańcy, a obszarami na których w wyniku epidemii zginęło najwięcej ludzi.

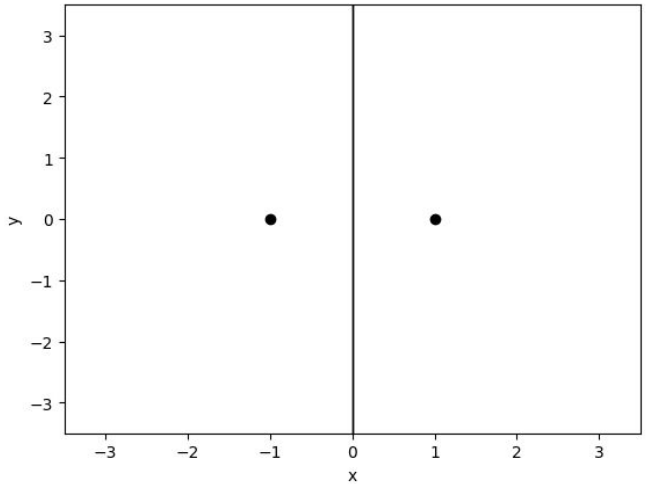


Mapa narysowana przez Snow'a tworząca komórkę diagramu Voronoi. Słupki reprezentują ilość śmierci w danym budynku

Algorytmy konstrukcji diagramów Voronoi

Algorytm naiwny

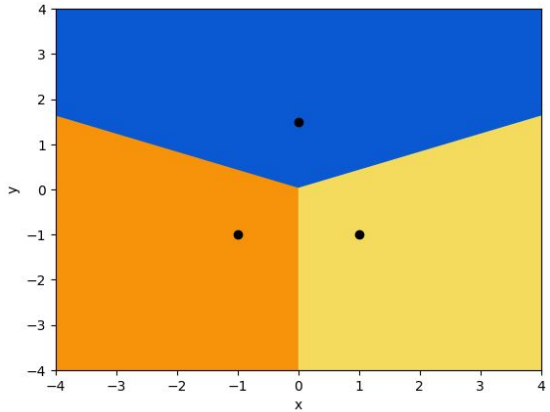
Diagram Voronoi dla dwóch punktów A, B można prosto skonstruować. Jest to symetralna odcinka AB.



Większa ilość punktów

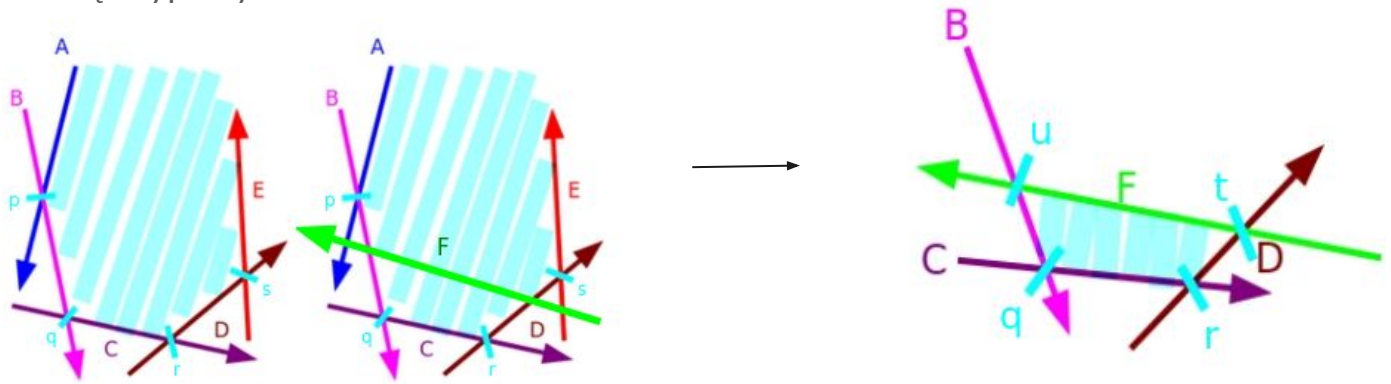
W ogólności dla n punktów komórkę diagramu Voronoi V (p) można skonstruować jako przecięcie n-1 półpłaszczyzn (zawierających punkt p) wyznaczonych przez symetralne odcinków wychodzących z p.

$$V_i(p_i) = \bigcap_{1 \le j \le n, j \ne i} h(p_i, p_j)$$



Przecięcie półpłaszczyzn [O(nlogn)]

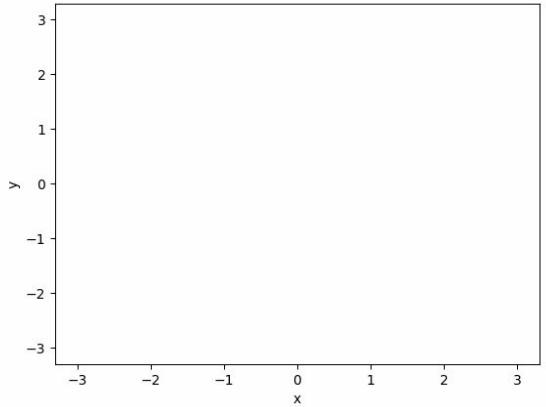
1.Sortujemy półpłaszczyzny względem kąta do Osi Ox.
2. Kolejne półpłaszczyzny dodawane są do dwukierunkowej kolejki, a nadmiarowe usuwane z końców kolejki.
3. Po przetworzeniu wszystkich półpłaszczyzn w kolejce powinny zostać półpłaszczyzny tworzące wielokąt wypukły



Implementacja algorytmu naiwnego [O((n^2)logn)]

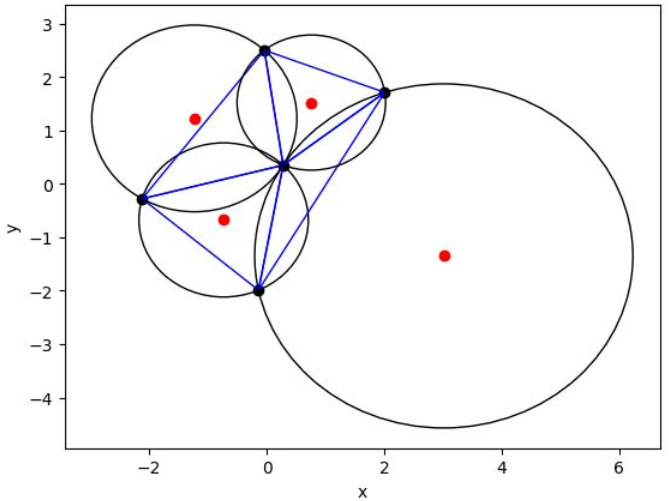
Wyznaczamy symetralne aktualnie analizowanego punktu z pozostałymi punktami.

Obliczamy przecięcie półpłaszczyzn wyznaczonych przez znalezione symetralne. Rezultatem tego przecięcia jest wielokąt Woronoja aktualnie przetwarzanego punktu



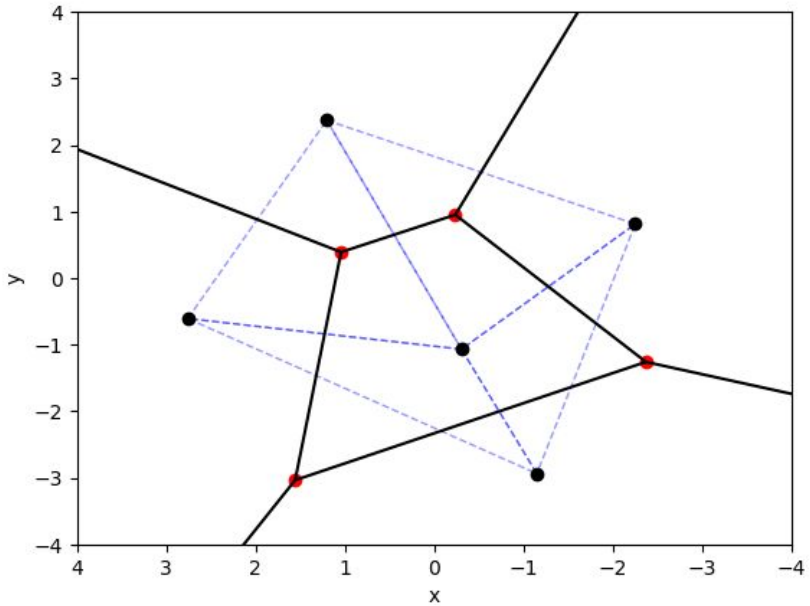
Graf dualny do triangulacji Delaunay

Triangulacja Delaunaya jest to taka triangulacja zbioru punktów P na płaszczyźnie, że żaden punkt ze zbioru P nie znajduje się wewnątrz okręgu opisanego na trójkącie należącym do triangulacji



Graf dualny do triangulacji Delaunay

Istotną z punktu widzenia diagramów Voronoi właściwością triangulacji Delaunaya jest fakt, że graf dualny triangulacji odpowiada diagramowi Voronoi dla tego samego zbioru punktów. Środki okręgów opisanych na trójkątach triangulacji odpowiadają wierzchołkom diagramu Voronoi, a odpowiednie krawędzie między tymi wierzchołkami można uzyskać biorąc pod uwagę sąsiedztwo trójkątów



Dlaczego dualny?

Punkt P jest środkiem zielonego okręgu opisanego na jednym z trójkątów należących do siatki => Punkt P jest równoodległy od punktów A,B,C => Jest wierzchołkiem diagramu Woronoja

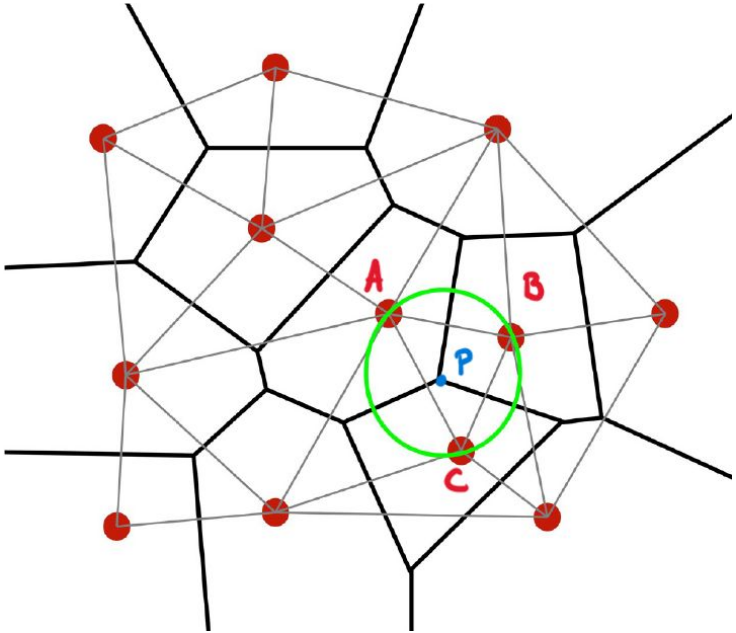
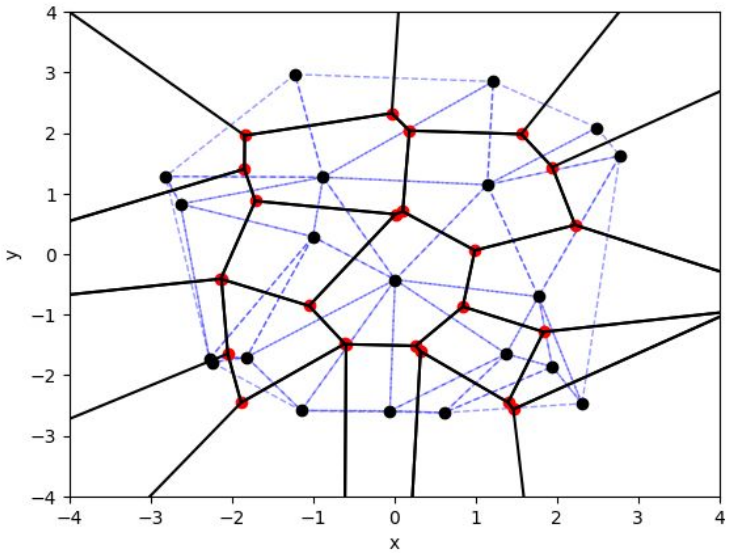


Diagram Voronoi z triangulacji Delone O(n^2)*

1.Na początku sprawdzamy czy wszystkie punkty nie leżą na jednej prostej. O(n)

2.Przeprowadzamy triangulacje punktów. O(n^2)

3. Dla każdego trójkąta w triangulacji należy obliczyć środek opisanego na
nim okręgu. Środki sąsiadujących ze sobą trójkątów tworzą odcinki, które
dodawane są do zbioru wynikowego E. Trójkąty należące do otoczki wypukłej
mają maksymalnie dwóch sąsiadów. Dla krawędzi trójkątów bez sąsiadów krawędź diagramu będzie półprostą. O(n)

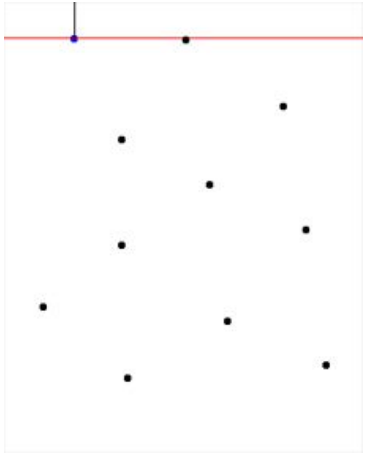


Algorytm Fortune'a

Algorytm bazujący na technice zamiatania wyznacza krawędzie diagramu na podstawie przecinających się paraboli tworzących "linie brzegową" (z ang. beachline), która zmienia się wraz z przesuwaniem miotły.

Algorytm przetwarza dwa rodzaje zdarzeń:

- zdarzenie punktowe (z ang. site event) - znane od początku
- zdarzenie okręgowe (z ang. circle event) - znajdowane z czasem działania algorytmu



Gif wizualizujący przebieg tworzenia diagramu przez algorytm Fortune'a

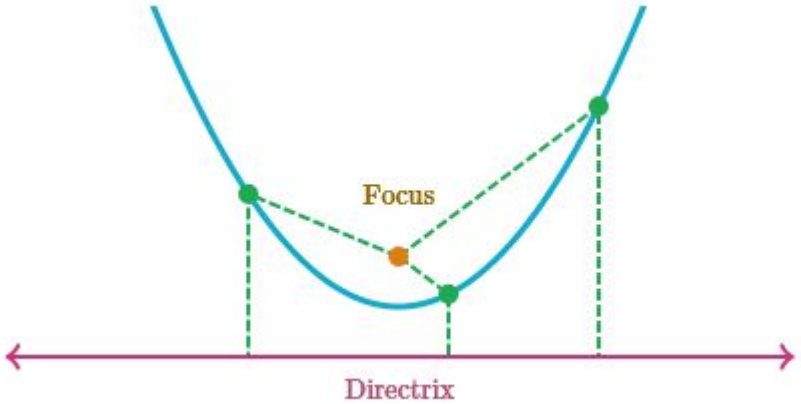
Steven Fortune - autor algorytmu

Określenie paraboli za pomocą punktu i prostej

Parabole w tym algorytmie są traktowane jako zbiór punktów równoodległych od prostej - miotły oraz punktów generujących(centrum) - punkty wejściowe.

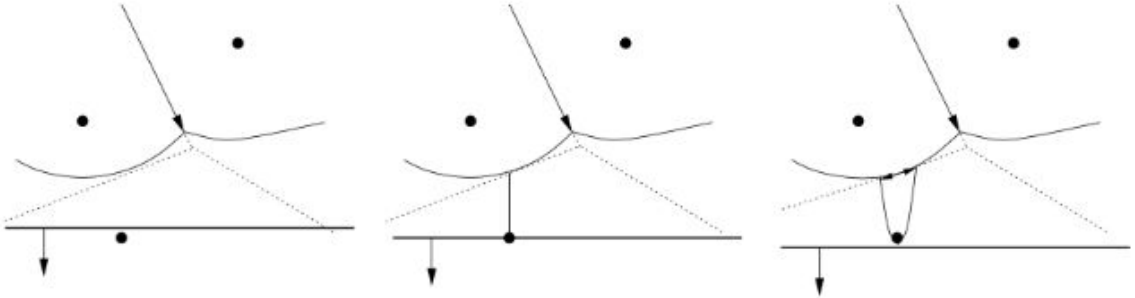
Równanie paraboli, gdzie
(a,b) - centrum
c - prosta y=c

$$(x-a)^2 + b^2 - c^2 = 2(b-c)y$$



Zdarzenie punktowe

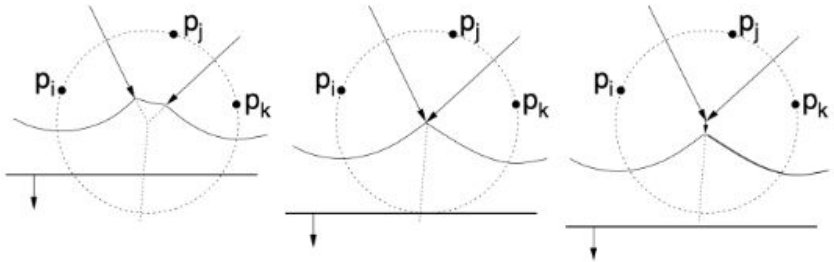
Zdarzenie punktowe pojawia się wtedy, gdy miotła natrafia na jeden z punktów wobec, których tworzony jest diagram. Parabola, która przed zdarzeniem znajdowała się nad punktem, zostaje rozdzielona na dwie części i rozpoczyna się tworzenie nowych krawędzi. Podczas obsługi tego zdarzenia wykrywane są też ewentualne zdarzenia kołowe.



Wizualizacja dodania nowego łuku do linii brzegowej

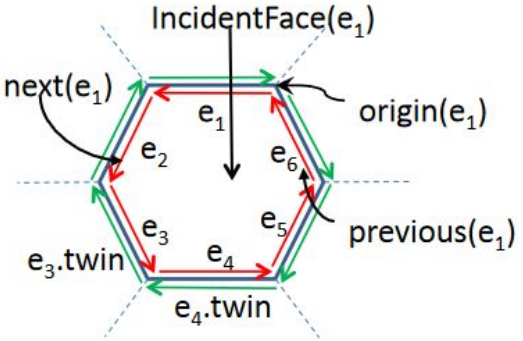
Zdarzenie okręgowe

Zdarzenie okręgowe określa punkt, w którym parabola zanika. Zdarzenia okręgowe tworzone są przez okręgi styczne do trzech punktów początkowych. Punktem zaniknięcia paraboli jest środek okręgu generującego zdarzenie okręgowe. Podczas tego zdarzenia z linii brzegowej usunięta zostaje zanikająca parabola oraz do diagramu zostaje dodany wierzchołek na miejscu środka badanego okręgu.



Wizualizacja zanikającego łuku

Struktury danych

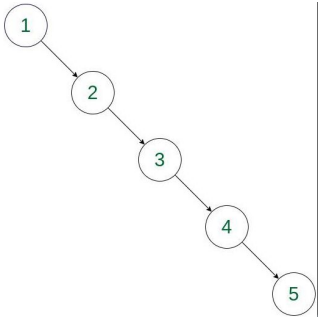


Wizualizacja elementów podwójnie łączonej listy krawędzi

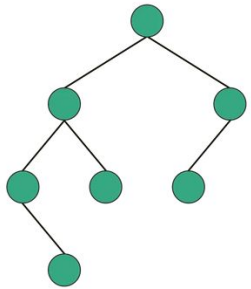
- Struktura zdarzeń - kolejka priorytetowa
  - Przechowuje zdarzenia punktowe i okręgowe
- Struktura stanu - zbalansowane drzewo wyszukiwań binarnych
  - Przechowuje stan linii brzegowej tworzącej diagram
- Struktura przechowująca diagram - lista podwójnie łączonych półkrawędzi
  - Przechowuje diagram

Struktura stanu

Struktura stanu to kluczowa struktura w algorytmie Fortune'a. Przechowuje informacje na temat paraboli, które tworzą diagram. Aby uzyskać pożądaną złożoność O(nlogn) struktura musi umożliwiać nam wyszukanie, wstawienie oraz usunięcie paraboli ze struktury stanu w O(logn). Nie możemy niestety użyć do tego zwykłego drzewa BST z racji na to, że istnieje szansa na to, że drzewo się zdegeneruje. Co niestety powoduje pogorszenie złożoności tych operacji do złożoności O(n) - przez co cały algorytm osiągnie złożoność O(n^2). Żeby zapobiec takiej sytuacji musimy utrzymywać balans struktury stanu.



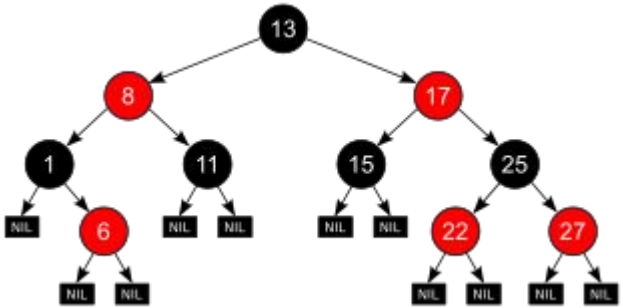
Zdegenerowane drzewo



Zbalansowane drzewo

Struktura stanu cd.

Jednym z podejść jest zastosowanie drzewa AVL, które rozróżnia dwa typy węzłów – wewnętrzne reprezentujące punkty przecięć paraboli oraz liście reprezentujące parabole. Lecz to podejście jest skomplikowane pod względem balansowania. Zamiast tego struktura stanu algorytmu może bazować na drzewie czerwono-czarnym zbudowanym tylko z węzłów reprezentujących parabole. Przy tym podejściu nie musimy martwić się o różne przypadki brzegowe. A większość implementacji możemy zaczerpnąć z literatury np. “Wprowadzenie do algorytmów” Cormen’a, Leiserson’a, Rivest’a, Clifford’a Stein’a.



Drzewo czerwono-czarne

Przebieg działania algorytmu

- Inicjalizacja struktury zdarzeń punktami początkowymi (jako zdarzenia punktowe)
- Inicjalizacja pustej struktury stanu
- Wyciąganie ze struktury zdarzeń kolejnych zdarzeń oraz ich przetwarzanie dopóki struktura zdarzeń nie będzie pusta

Parametry algorytmu

Złożoność obliczeniowa: **O(nlogn)**, gdzie n to liczba punktów, wobec której tworzony jest diagram.

Złożoność pamięciowa: **O(n)**, gdzie n to liczba punktów, wobec której tworzony jest diagram.

Porównanie czasowe algorytmów

	n	Naive [s]	Delauney [s]	Fortune [s]
0	10.0	0.03	0.00	0.00
1	100.0	0.74	0.01	0.01
2	1000.0	73.72	0.08	0.06
3	10000.0	inf	0.59	0.64
4	100000.0	inf	6.04	13.98

Bibliografia

https://www2.cs.sfu.ca/~binay/813.2011/DCEL.pdf

https://towardsdatascience.com/5-types-of-binary-tree-with-cool-illustrations-9b335c430254

https://en.wikipedia.org/wiki/Fortune%27s_algorithm

https://jacquesheunis.com/post/fortunes-algorithm/

https://www.varsitytutors.com/hotmath/hotmath_help/topics/finding-the-equation-of-a-parabola-given-focus-and-directrix

https://ufkapano.github.io/download/Mateusz_Malczewski_2021.pdf

https://9p.io/who/sjf/

https://www.ams.org/publicoutreach/feature-column/fcarc-voronoi

https://rcin.org.pl/igipz/Content/63054/PDF/WA51_82399_r2017-t89-z2_Przeg-Geogr-Kisiala.pdf

https://www.slideshare.net/OleksandrGlagoliev/fortunes-algorithm

https://github.com/fewlinesofcode/FortunesAlgorithm/blob/master/Sources/FortunesAlgorithm/FortuneSweep.swift

https://pvigier.github.io/2018/11/18/fortune-algorithm-details.html