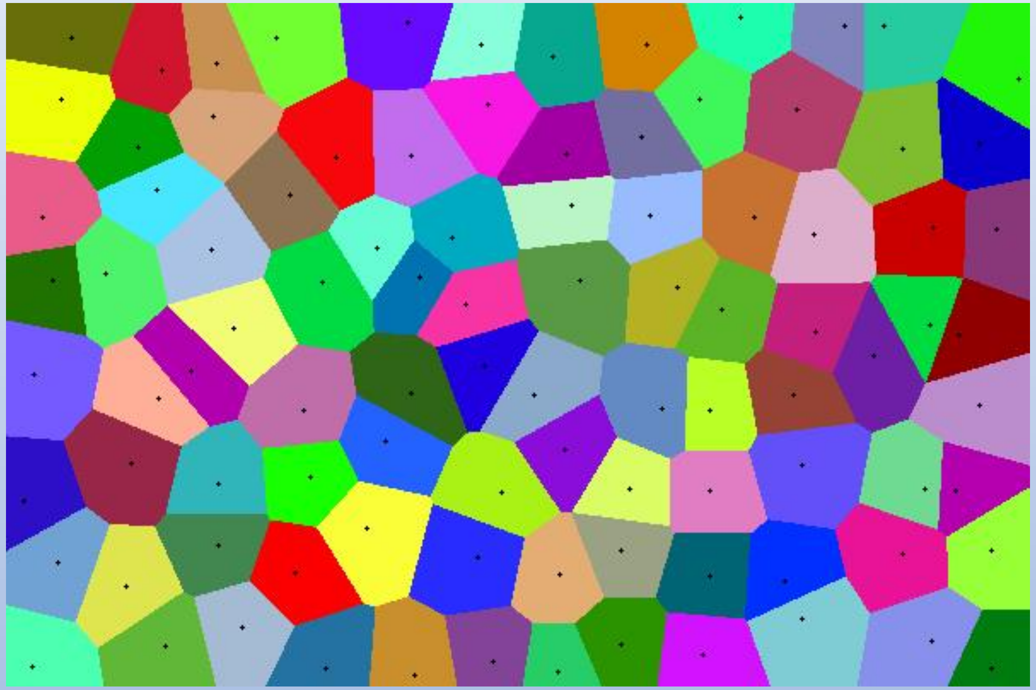
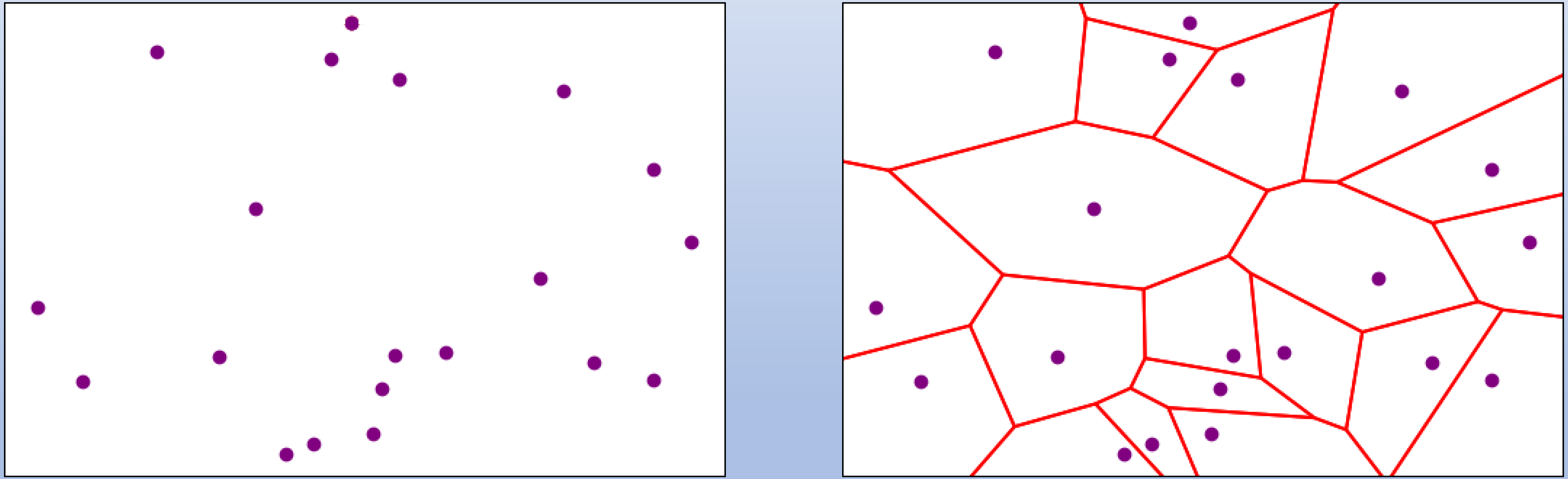
Wieloboki Voronoi

Jak wygląda diagram Voronoi?



Czym są wieloboki Voronoi?



Gieorgij Fieodosjewicz Woronoj

- Rosyjski matematyk pochodzenia ukraińskiego
- Żył w latach 1868-1908
- Studiował na Cesarskim Uniwersytecie Sankpetersburskim
- W roku 1894 został nauczycielem akademickim na Cesarskim Uniwersytecie Warszawskim
- Obejmował katedrę matematyki na Warszawskim Instytucie Politechnicznym
- Uczył Wacława Sierpińskiego oraz Borysa Delaunay'a

2 hrywny z 2008 roku

Definicja

Niech $P = \{p_1, p_2, \ldots, p_n\}$, gdzie $p_i \in \mathbb{R}^2$, $n \in \mathbb{N}$.

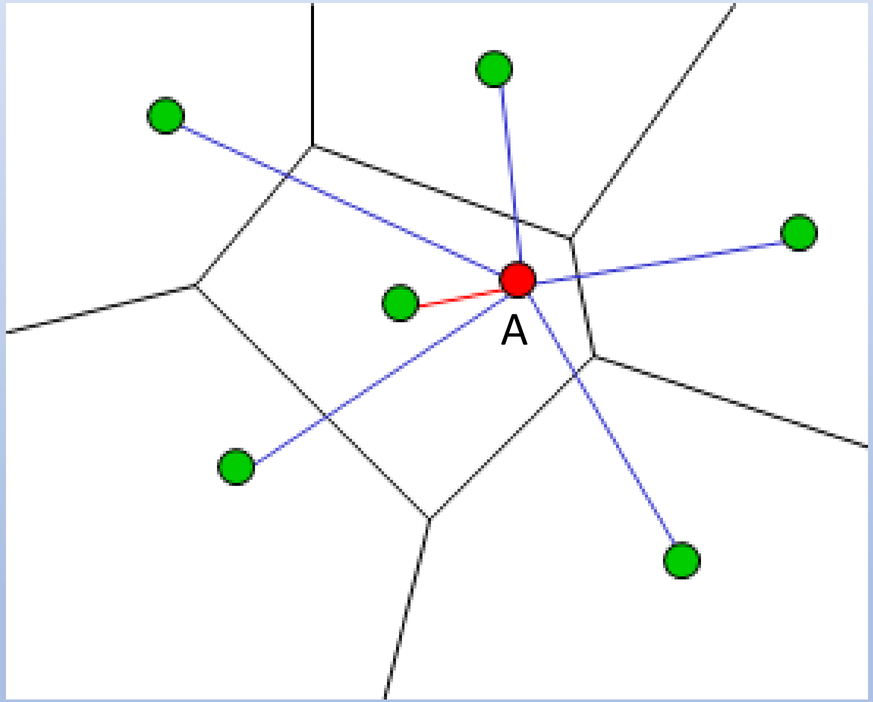
Komórką diagramu Voronoi dla punktu p_i nazywamy

$$V_P(p_i) = \{x : \forall_{j \neq i} d(x, p_i) \leq d(x, p_j)\},$$

gdzie d jest metryką euklidesową.

Diagramem Voronoi dla P nazywamy podział płaszczyzny na n komórek Voronoi $V_P(p_1), V_P(p_2), \ldots, V_P(p_n)$. Diagram Voronoi dla P oznaczamy przez $Vor(P)$.

Najbliższy punkt



Jeżeli wybierzemy jakikolwiek punkt A na płaszczyźnie, to punkt startowy znajdujący się najbliżej punktu A odpowiada temu samemu obszarowi, do którego należy punkt A

W jaki sposób powstają krawędzie?

Krawędzie Voronoi’a to „wycięte” symetralne odpowiednich punktów

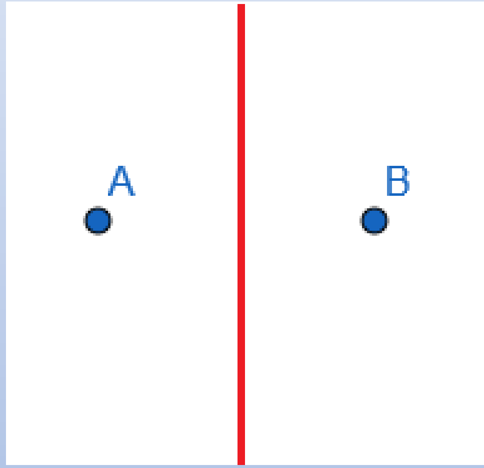


Diagram dla dwóch punktów

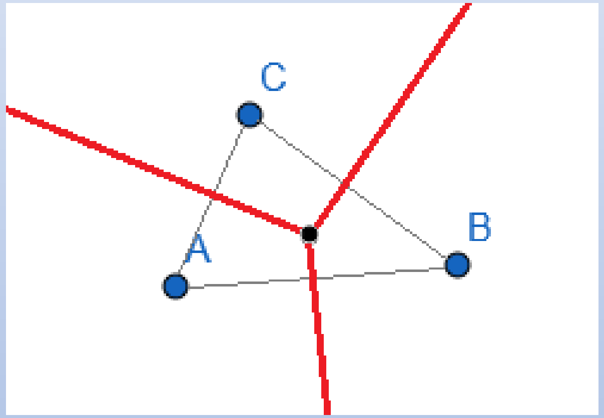


Diagram dla trzech punktów

Algorytm voronoi_diagram

- Dane wejściowe:

  zbiór punktów P = (x, y), gdzie x, y to współrzędne punktów
- Dane wyjściowe:

  diagram Voronoi ograniczony ramką jako podwójnie łączona lista krawędzi E

1. Inicjalizacja struktur

- Inicjalizacja kolejki zdarzeń Q wraz ze zdarzeniami początkowymi (kolejka priorytetowa z uszeregowana po współrzędnych y)
- Inicjalizacja pustej struktury drzewa czerwono-czarnego T (linia brzegowa)
- Inicjalizacja pustej listy krawędzi podwójnie łączonych E

Struktura zdarzeń

- W strukturze zdarzeń znajdują się zrówno zdarzenia punktowe, jak i zdarzenia okręgowe
- Zdarzenia są przechowywane w kolejce prorytetowej
- Na początku należy dodać wszystkie zdarzenia punktowe do kolejki
- Z kolejki wyjmowane są zdarzenia o największej współrzędnej y

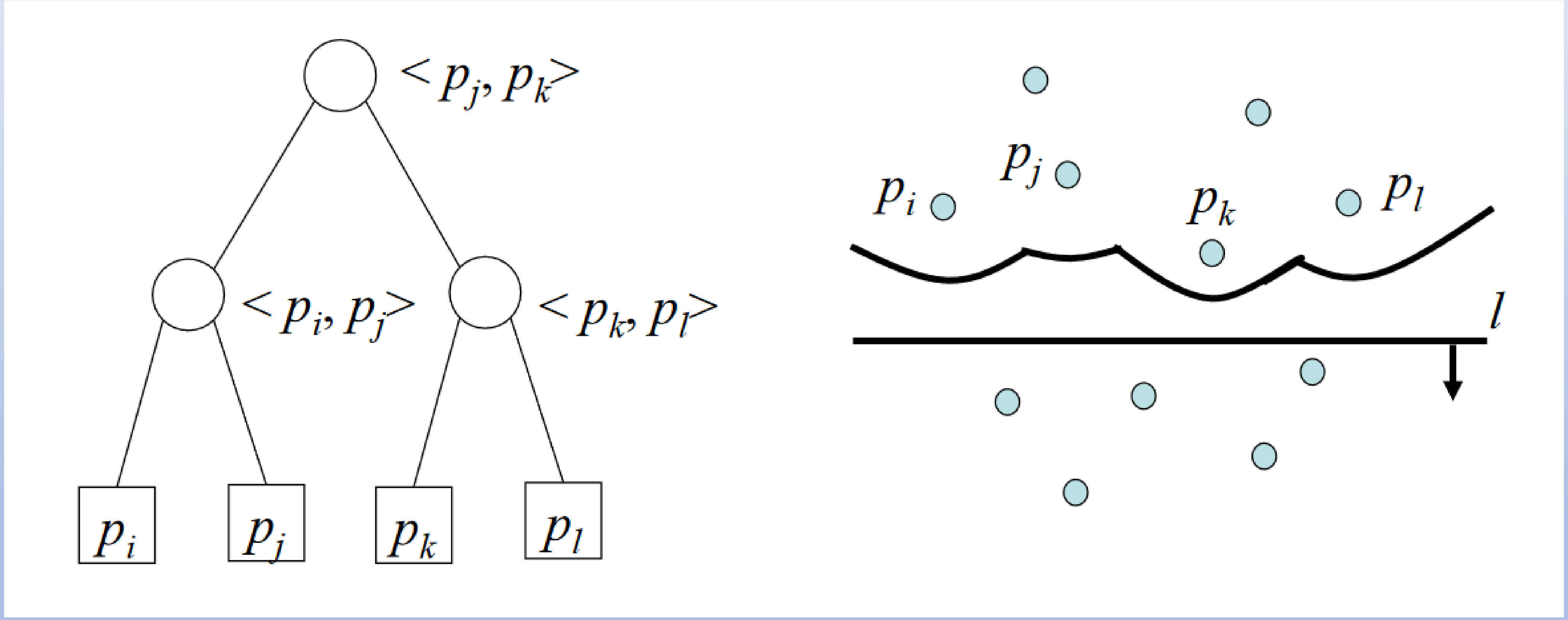
Zdarzenie punktowe

- Gdy miotła natrafi na punkt, który był wierzchołkiem startowym, wtedy mamy zgłoszenie zdarzenia punktowego

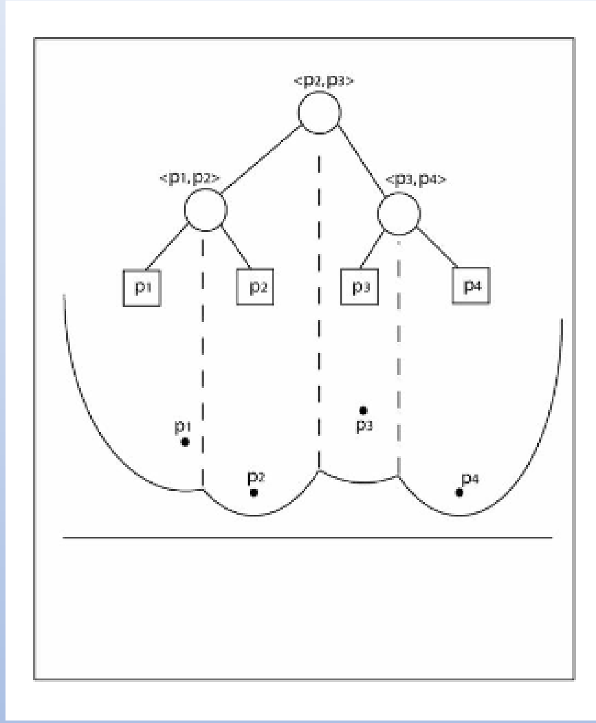
Zdarzenie okręgowe

- Zdarzeniem okręgowym jest punkt, który leży najniżej okręgu opisanego na wierzchołkach, odpowiadającym trzem kolejnym łukom
- Za każdym dodaniem lub usunięciem łuku sprawdzane jest, czy trzy kolejne punkty sąsiada prawego i lewego nie tworzą nowego zdarzenia okręgowego
- Należy dodać, że niektóre takie punkty znajdujące się najniżej okręgu mogły powstać już wcześniej, wtedy nie jest to zdarzenie okręgowe, a jedynie fałszywy alarm

Struktura drzewa – linia brzegowa

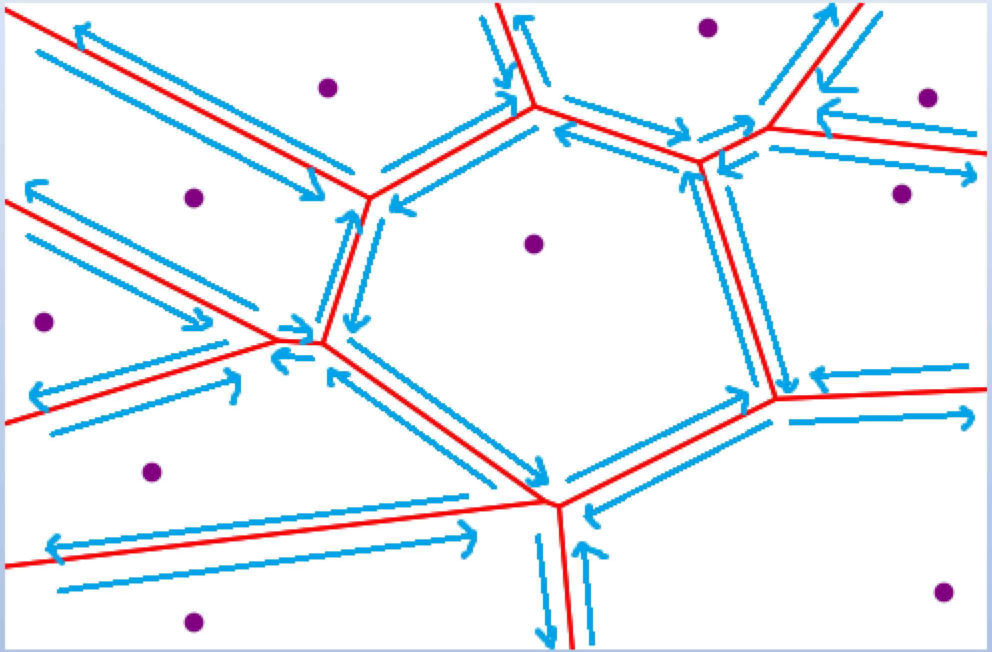


Struktura drzewa przechowująca łuki



Wszystkie łuki znajdują się w strukturze drzewa czerwono-czarnego, są przechowywane w kolejności przecięć się kolejnych parabol

Czym są krawędzie podwójnie łączone?



Każdą krawędź reprezentują dwie przeciwnie skierowane półkrawędzie
Posiada wierzchołek początkowy - *orient* oraz końcowy - *destination*

Algorytm Fortune'a

While Q nie jest pusta:

do usuń zdarzenie z największą współrzędną y z kolejki *Q*

if zdarzenie jest punktowe i wystąpiło w punkcie *s*:

then ***HandleSiteEvent***(*s*)

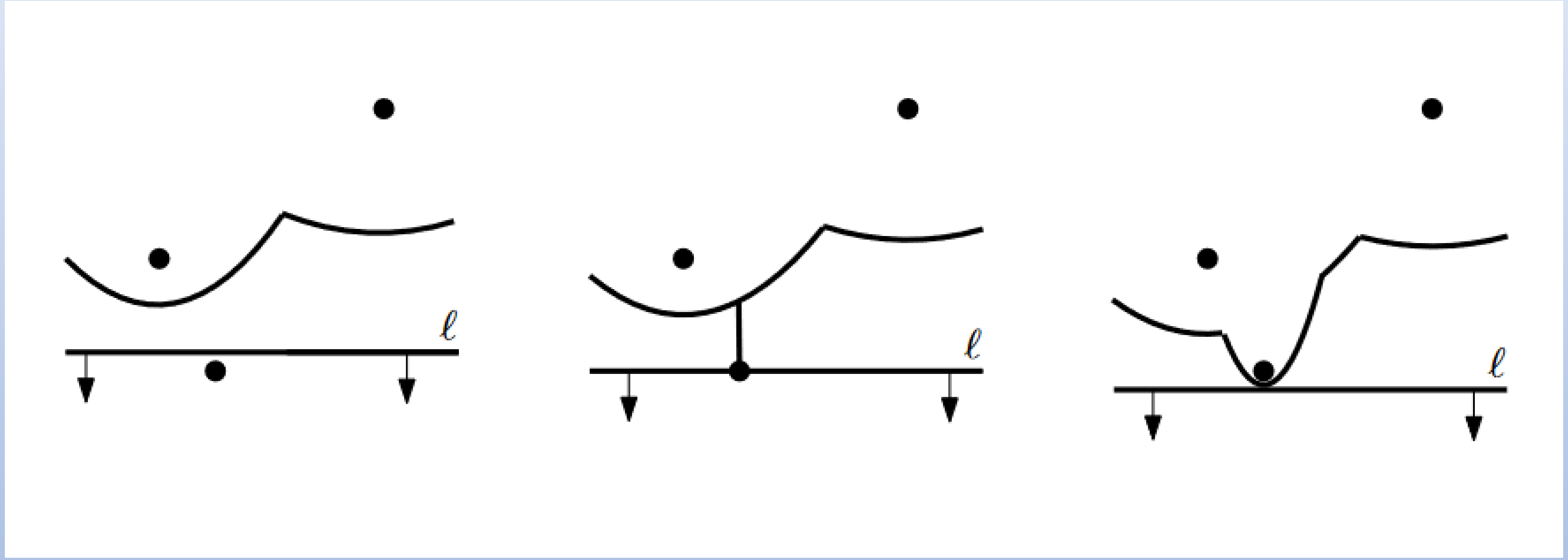
else ***HandleCircleEvent***(*c*), gdzie *c* jest liściem, przedstawiającym zanikający łuk

Na koniec oblicz zakresy ramki, zawierającej wszystkie wierzchołki startowe w jej wnętrzu, a następnie wyznacz miejsca przecięcia się półprostych z ramką

HandleSiteEvent

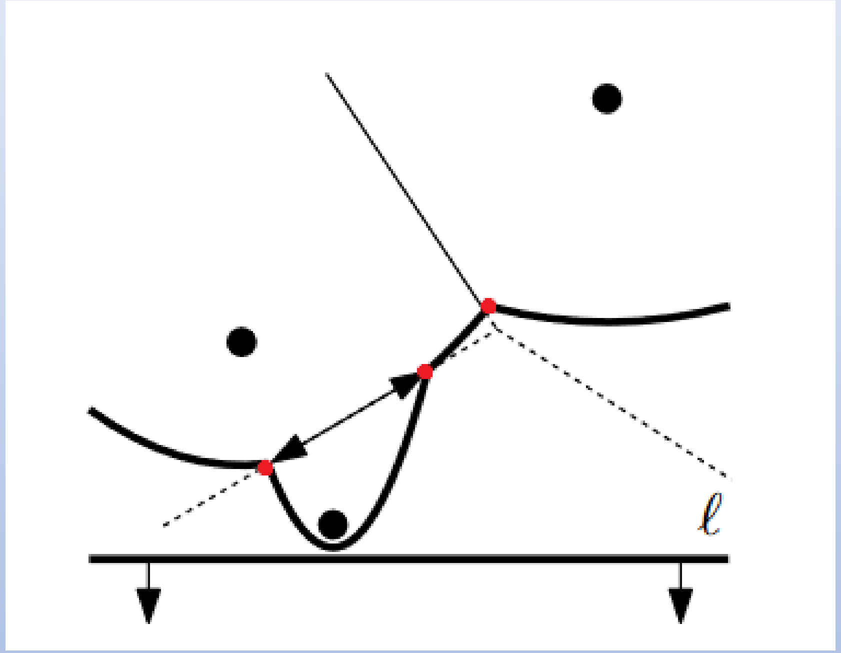
- 1. Najpierw należy znaleźć łuk (fragment paraboli) znajdujący się aktualnie nad punktem. Może się oczywiście zdarzyć, że takiego łuku nie będzie.
- 2. Następnie należy podzielić łuk na trzy mniejsze łuki parabol , środkowy łuk odpowiada punktowi z site eventu, pozostałe dwa pokrywają się z poprzednim łukiem (aktualizacja linii brzegowej)
- 3. Należy dodać do listy podwójnie łączonej nową półkrawędź
- 4. Jeśli zachodzi potencjalne zdarzenie kołowe to należy je dodać do kolejki priorytetowe zdarzeń (struktury zdarzeń)

Wykrywanie zdarzenia punktowego



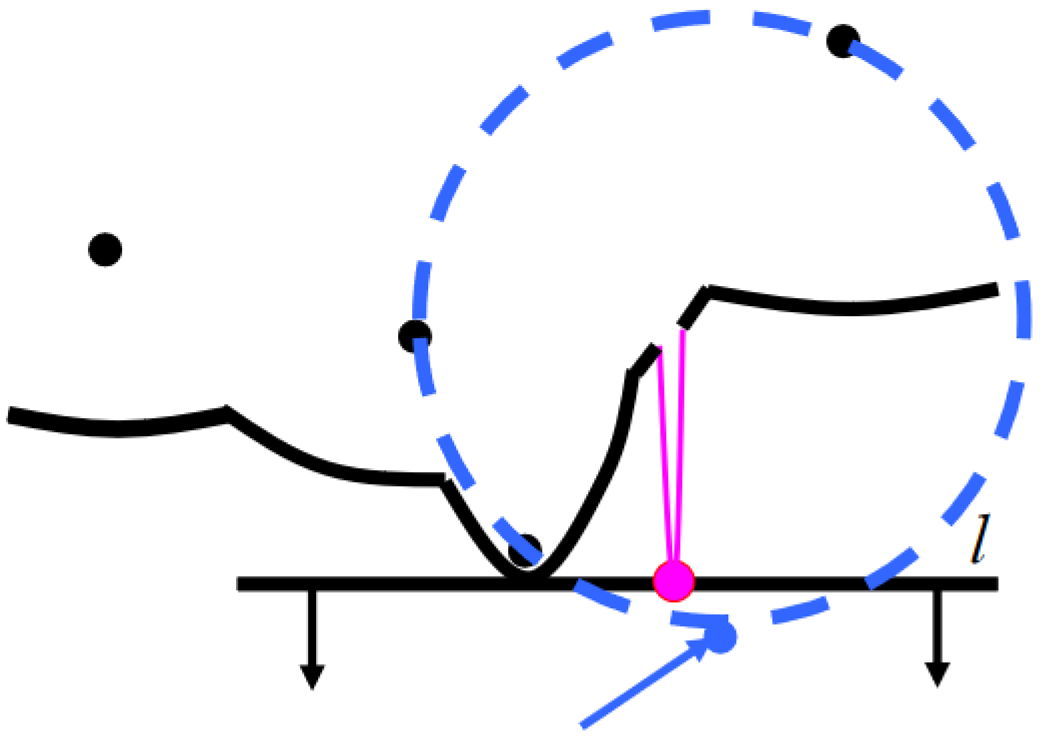
Miotła schodzi w dół, aż natrafi na zdarzenie punktowe, a nastęnie łuk, który znajduje się nad punktem rozbijany jest na trzy łuki. Środkowy łuk jest fragmentem paraboli równo odległym od punktu łuku oraz miotły.

Punkty przerwania łuków



Aby wiedzieć do jakiego łuku klasyfikować nowe punkty, będące zdarzeniami, potrzebujemy znać końce wszystkich łuków. Dlatego dla każdych dwóch sąsiadujących parabol należy znaleźć ich odpowiedni wierzchołek wspólny, który będzie punktem przerwania.

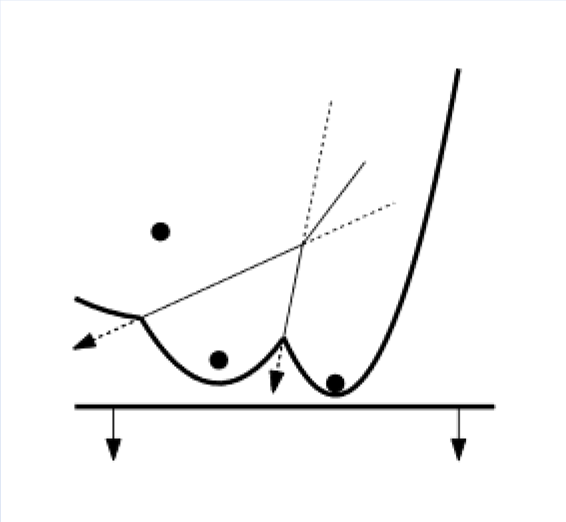
Obsługa fałszywego alarmu



Zdarzenie staje się fałszywym alarmem

Przed obsługą zdarzenia okręgowego może się zdarzyć, że poprzednie zdarzenie wskoczyło pomiędzy trójkę tworzącą zdarzenie okręgowe. Wtedy należy takie zdarzenie okręgowe usunąć z kolejki.

Obsługa fałszywego alarmu c. d.

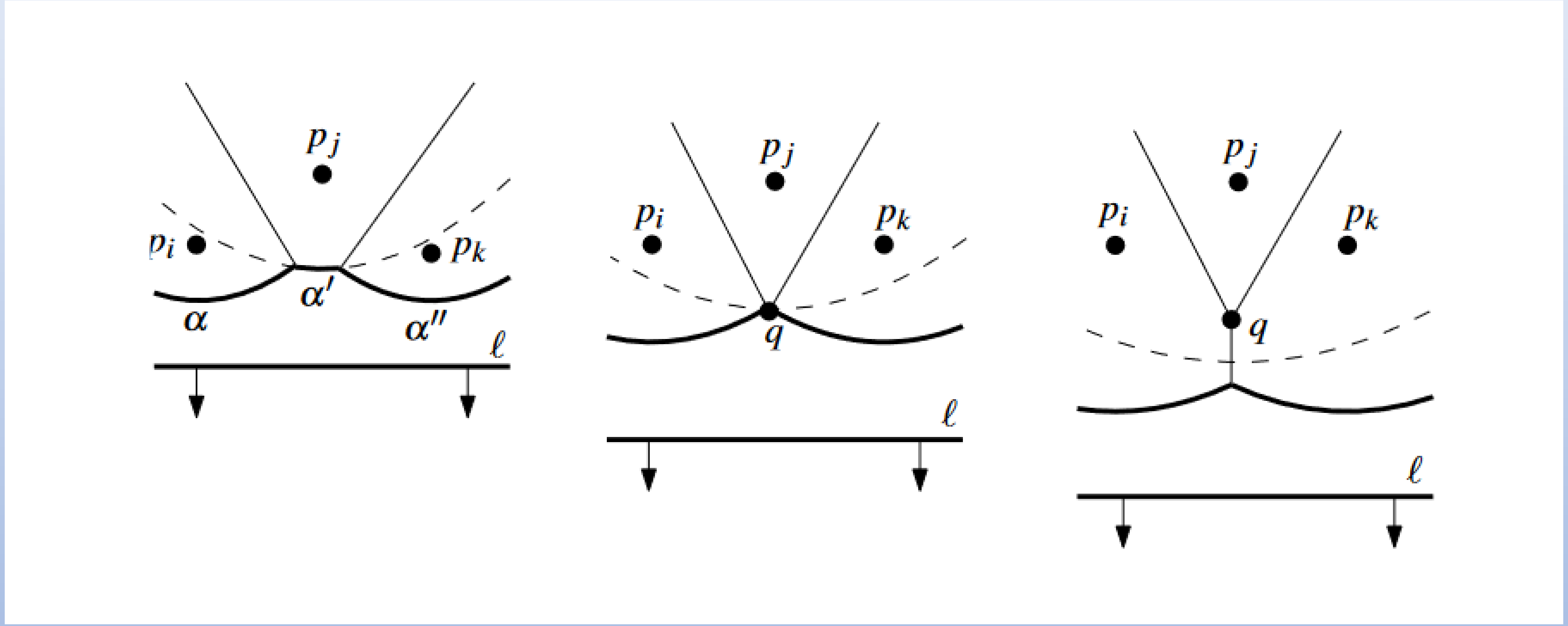


Może się zdarzyć, że punkty przerwania nie zbiegają do siebie, w takim przypadku trójka punktów nie definiuje potencjalnego zdarzenia okręgowego.

HandleCircleEvent

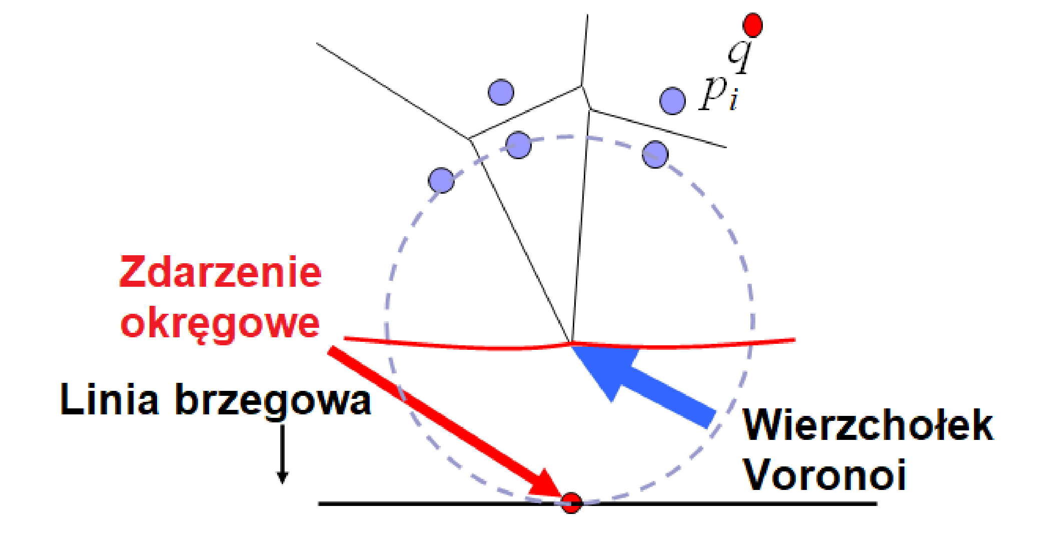
- 1. Najpierw należy dodać punkt, który powstał z zapadnięcia się łuku do punktu.
- 2. Należy usunąć znikający właśnie łuk ze struktury linii brzegowej
- 3. W tym momencie utworzyła się na pewno krawędź, więc trzeba ją dodać do listy, przechowującej diagram Voronoi
- 4. Podobnie jak w przypadku zdarzenia punktowe, zdarzenie okręgowe może spowodować powstanie kolejnego potencjalnego zdarzenia kołowego, które oczywiście trzeba dodać do struktury zdarzeń

Etapy zanikania łuku



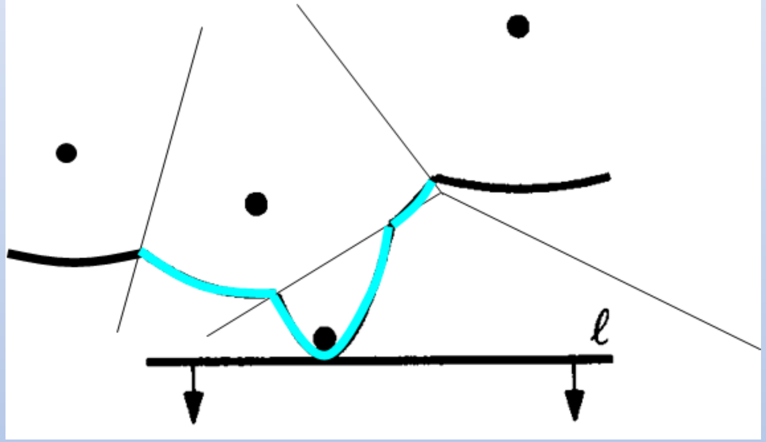
W momencie, w którym łuk zanika, powstaje nowy wierzchołek q, będący wierzchołkiem Voronoi

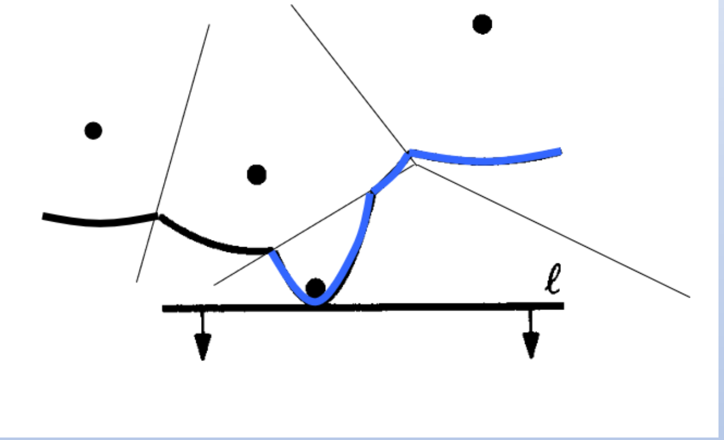
Powstawanie wierzchołka Voronoi



Wierzchołek Voronoi jest wykrywany w momencie, gdy miotła natrafi na punkt znajdujący się najniżej okręgu

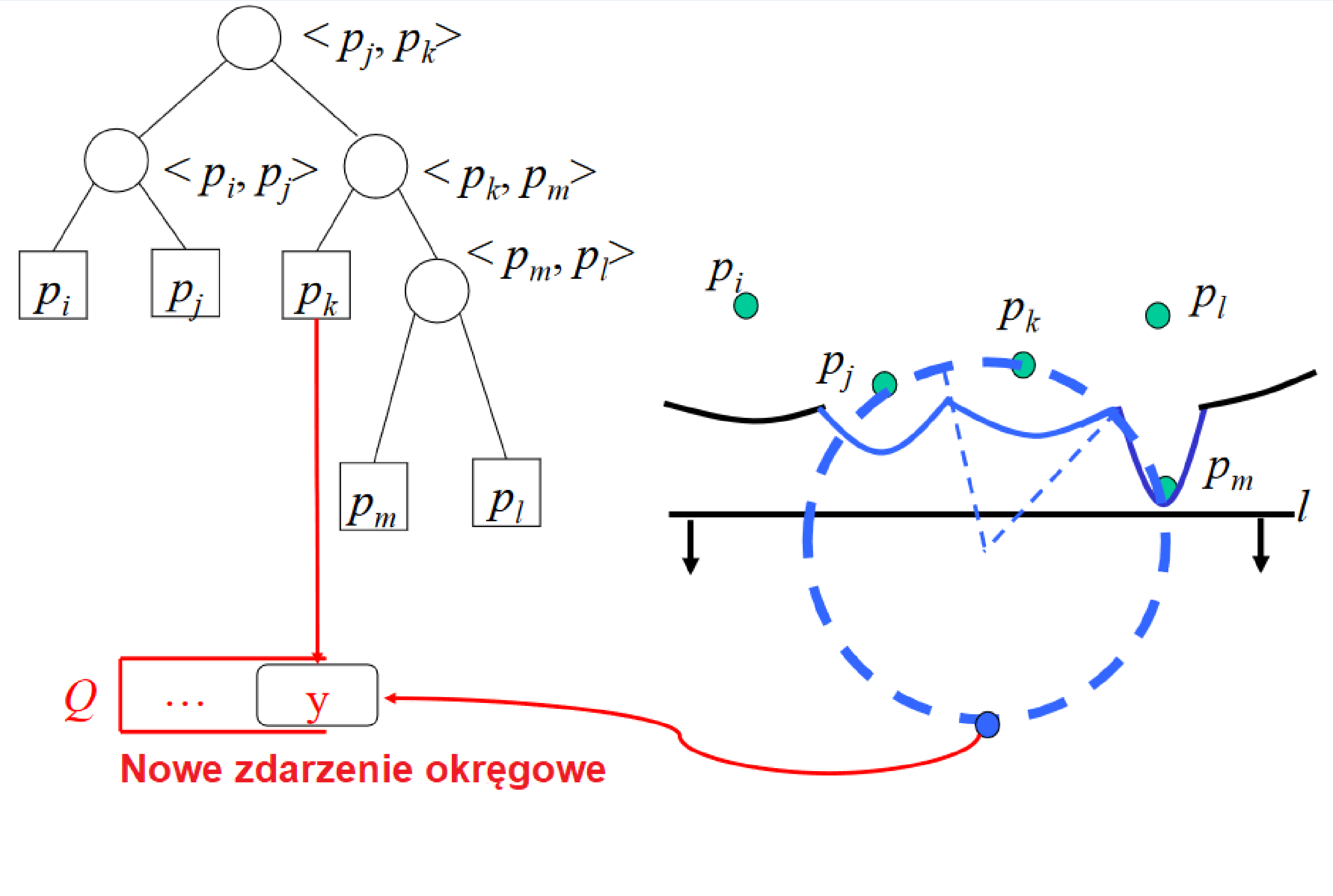
Wykrywanie zdarzeń okręgowych



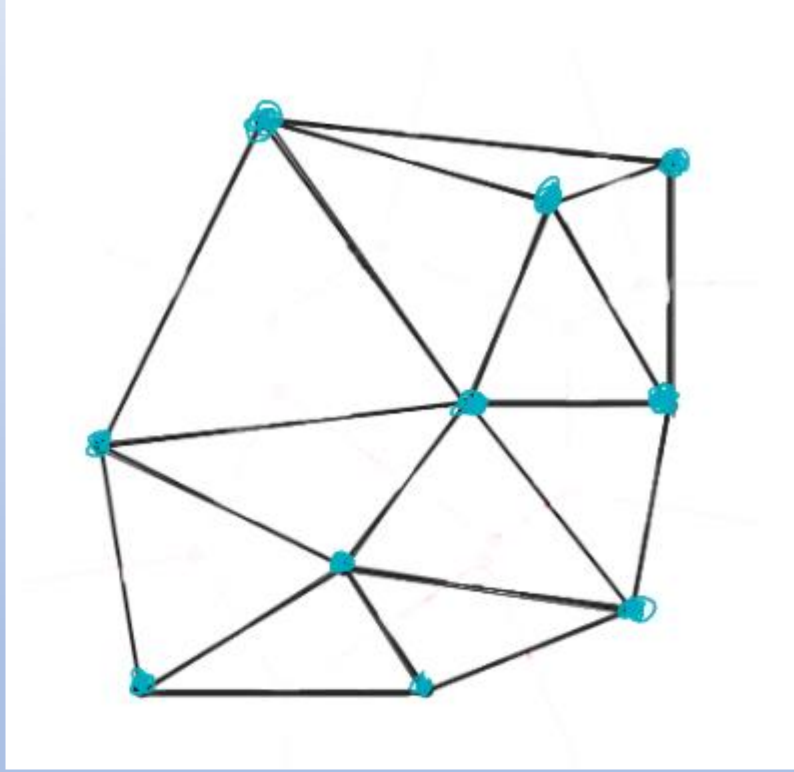


Dla każdych trzech kolejnych łuków sprawdzamy, gdzie leży środek okręgu opisanego na punktach, z kórych wychodziły łuki

Dodanie nowego zdarzenia okręgowego

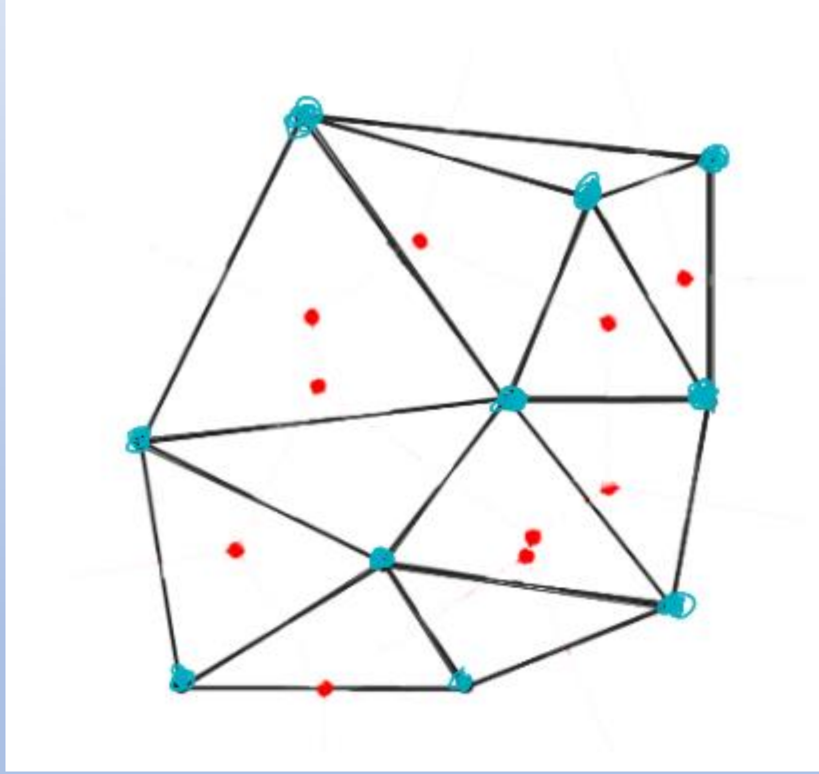


Algorytm wykorzystujący Triangulacje Delaunay'a



Na wejściu dostajemy zbiór punktów (niebieskie). Następnie na takich punktach wykonywana jest triangulacja Delaunay'a. Triangulacja ta polega na tym, aby wejściowe punkty stanowiły wierzchołki trójkątów triangulacji oraz aby kąty minimalne trójkątów były jak największe. W ten sposób otrzymujemy triangulacje (czarne trójkąty)

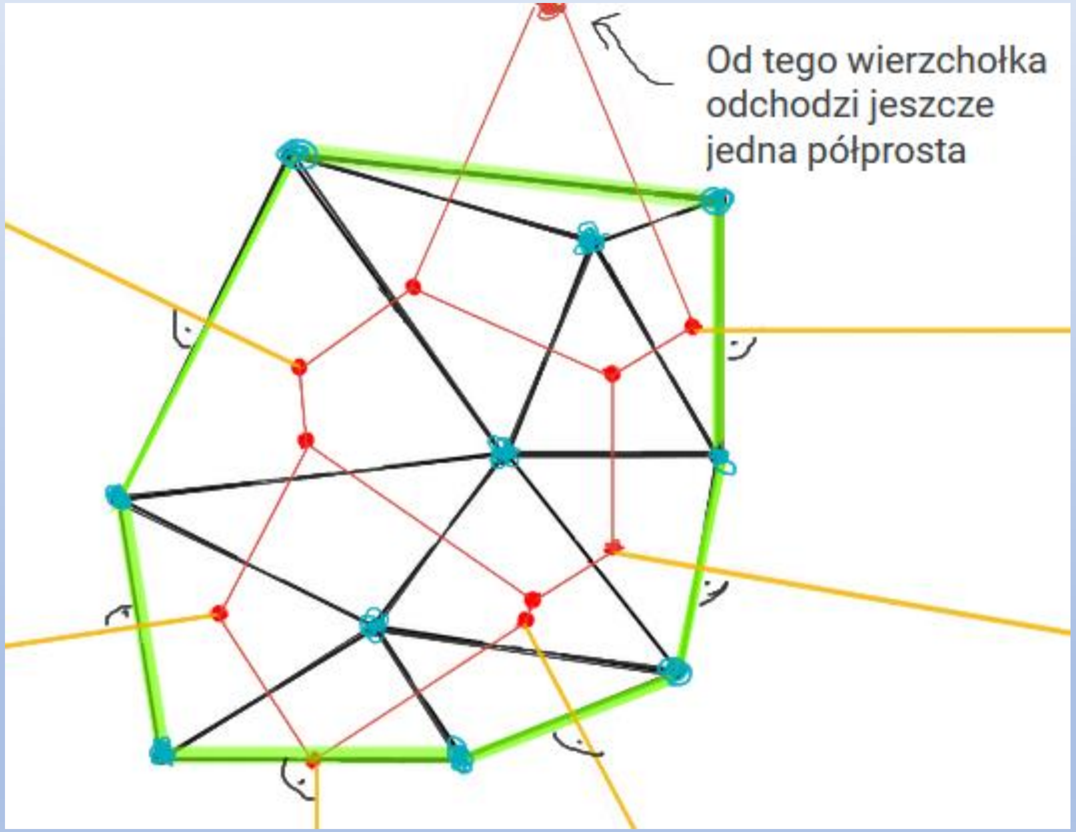
Algorytm wykorzystujący Triangulacje Delaunay'a cd.



Z racji, że graf diagramu Voronoi i Triangulacji Delauny'a są do siebie grafami dualnymi, to mając jeden graf, łatwo otrzymać drugi.
W naszym przypadku mając zadaną triangulacje, aby znaleźć wierzchołki Voronoi (czerwone), należy znaleźć środki okręgów opisanych na trójkątach triangulacji. Stanowią one wierzchołki Diagramu Voronoi.

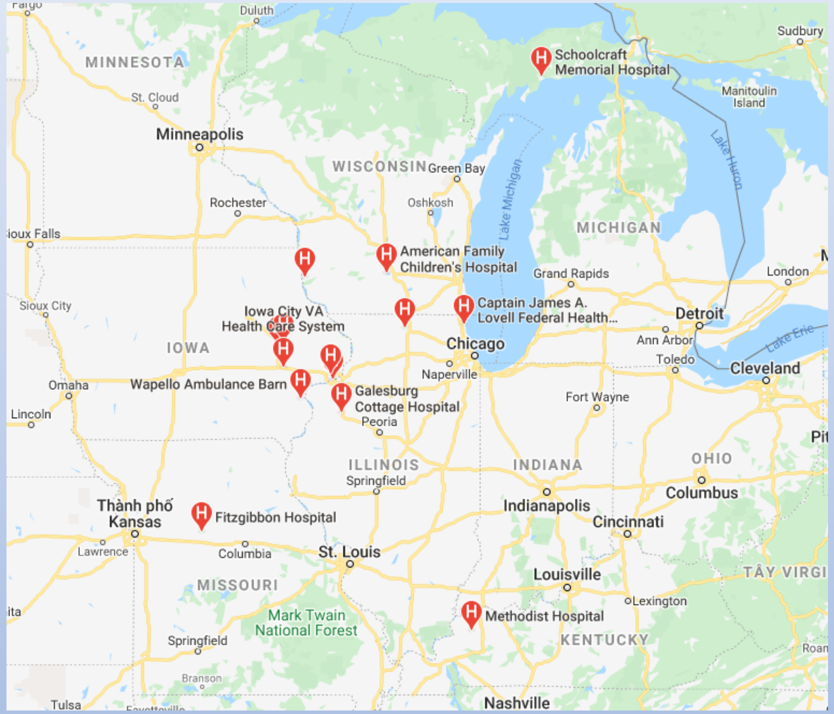
Uwaga, środek okręgu opisanego dla trójkąta na samej górze, nie zmieścił się na rysunku. Znajduje się on powyżej tego trójkąta na przecięciu symetralnych

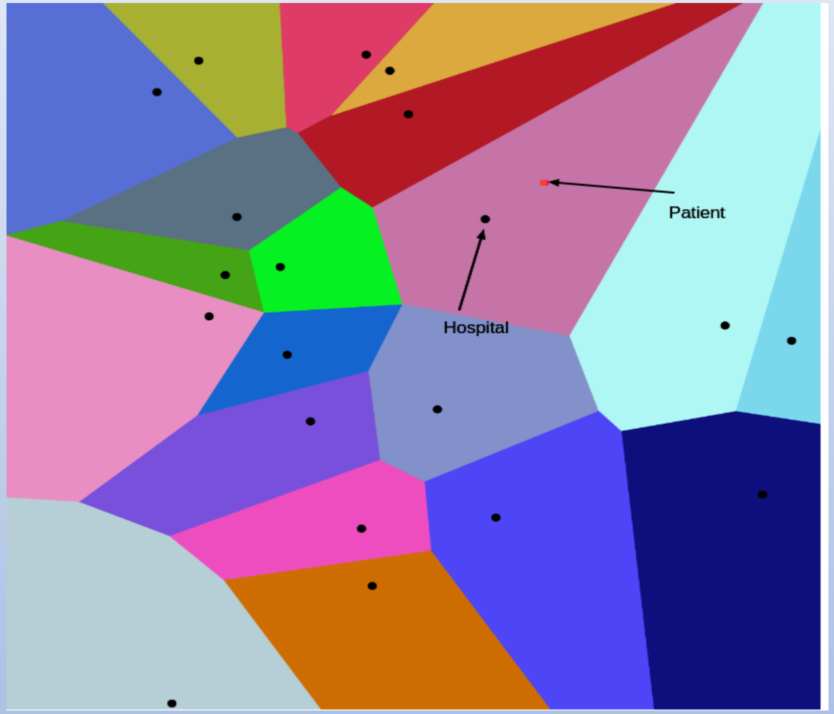
Algorytm wykorzystujący Triangulacje Delaunay'a cd.



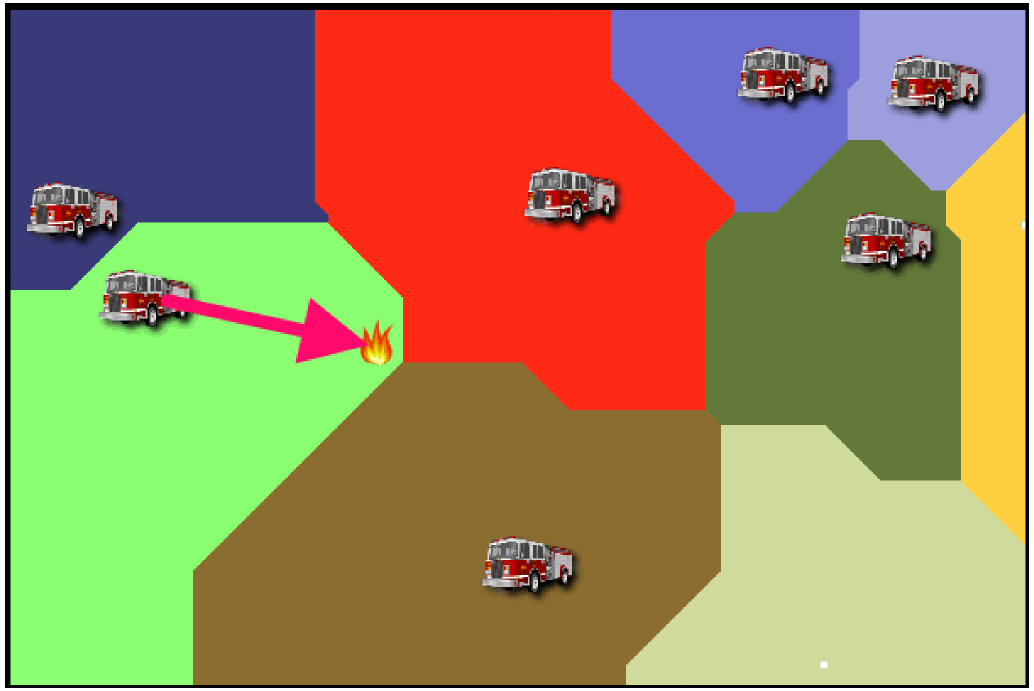
Mając już wierzchołki diagramu Voronoi należy połączyć ze sobą te, których trójkąty mają wspólne boki (czerwone krawędzie). Oprócz nich pojawiają się żółte półproste, które również należą do diagramu Voronoi. Aby je otrzymać musimy znaleźć takie trójkąty, które posiadają bok obwiedni triangulacji (zielone boki). Wówczas początkiem półprostej jest wierzchołek Voronoi związany z danym trójkątem. Sama półprosta jest tak poprowadzona aby była prostopadła do boku obwiedni i jednocześnie szła na zewnątrz obwiedni (patrz rysunek). W ten sposób otrzymujemy cały diagram Voronoi.

Zastosowania

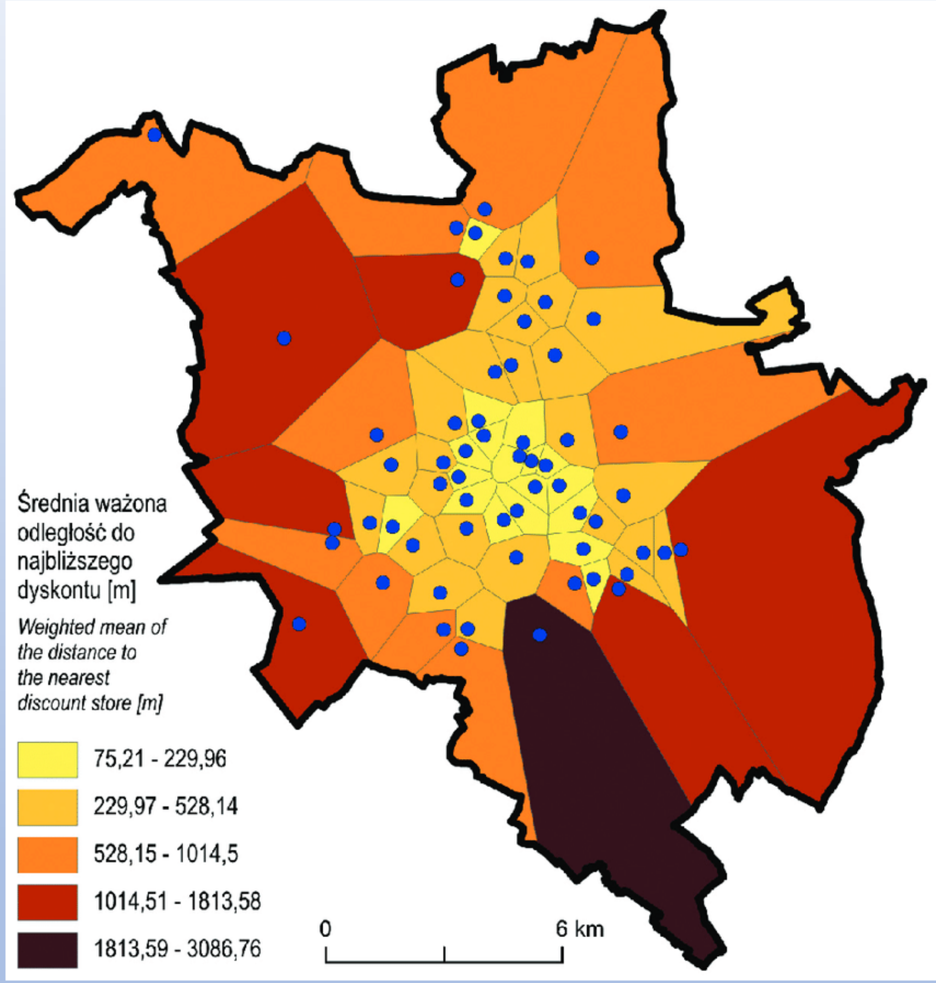


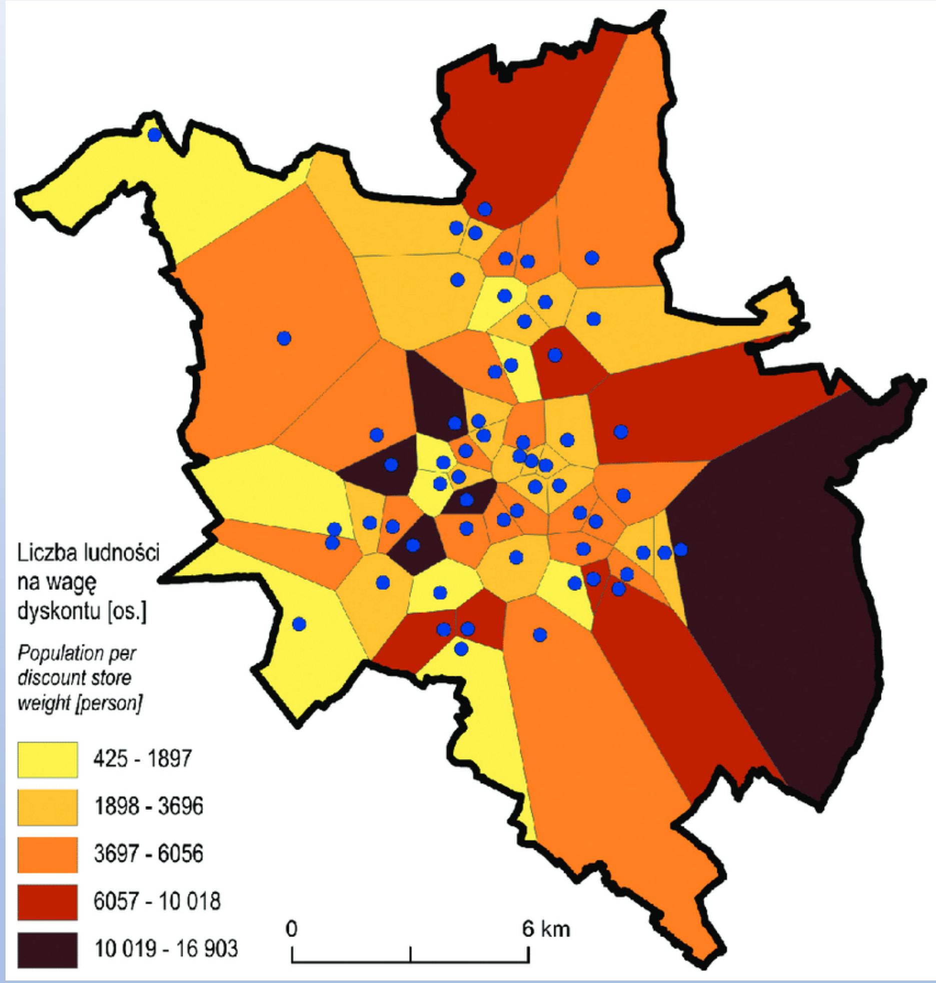


Do rannego jest wzywana najbliższa karetka pogotowia na podstawie miejsca adresu, a dzięki diagramom Voronoi wiemy, do którego szpitala ranny będzie miał najbliżej

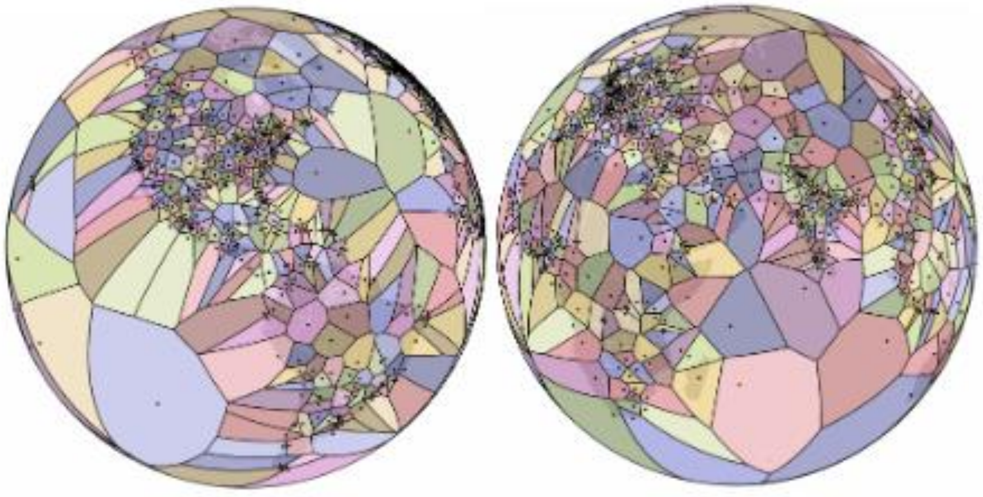


Gdy chcemy wysłać straż pożarną na miejsce pożaru w mieście, w którym ulice są do siebie prostopadłe, pomocna może okazać się metryka taksówkowa


Średnia ważona odległość do najbliższego dyskontu [m]
Weighted mean of the distance to the nearest discount store [m]
75,21 - 229,96
229,97 - 528,14
528,15 - 1014,5
1014,51 - 1813,58
1813,59 - 3086,76
0
6 km

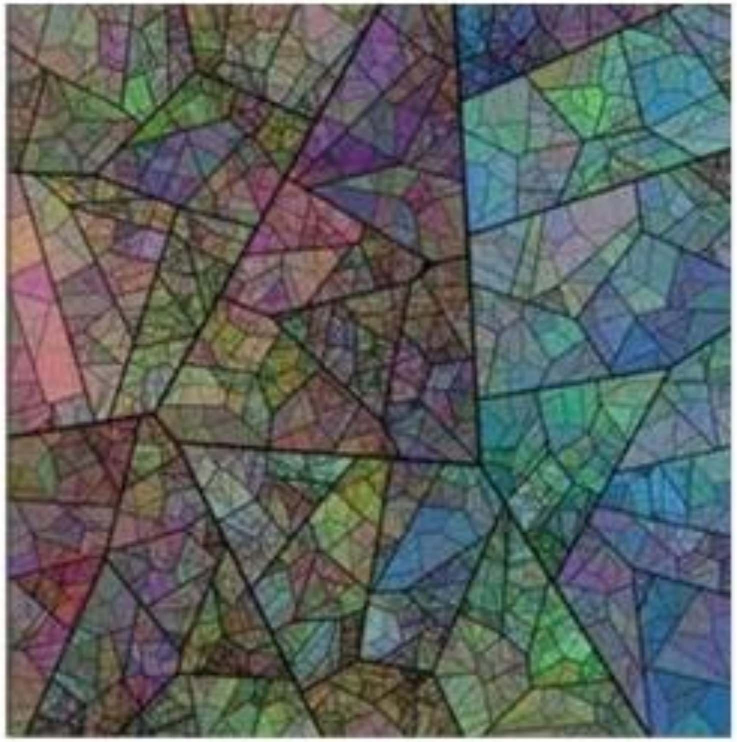

Liczba ludności na wagę dyskontu [os.]
Population per discount store weight [person]
425 - 1897
1898 - 3696
3697 - 6056
6057 - 10 018
10 019 - 16 903
0
6 km

Closest international Airport

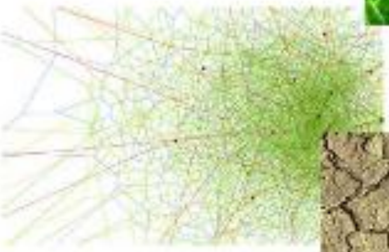


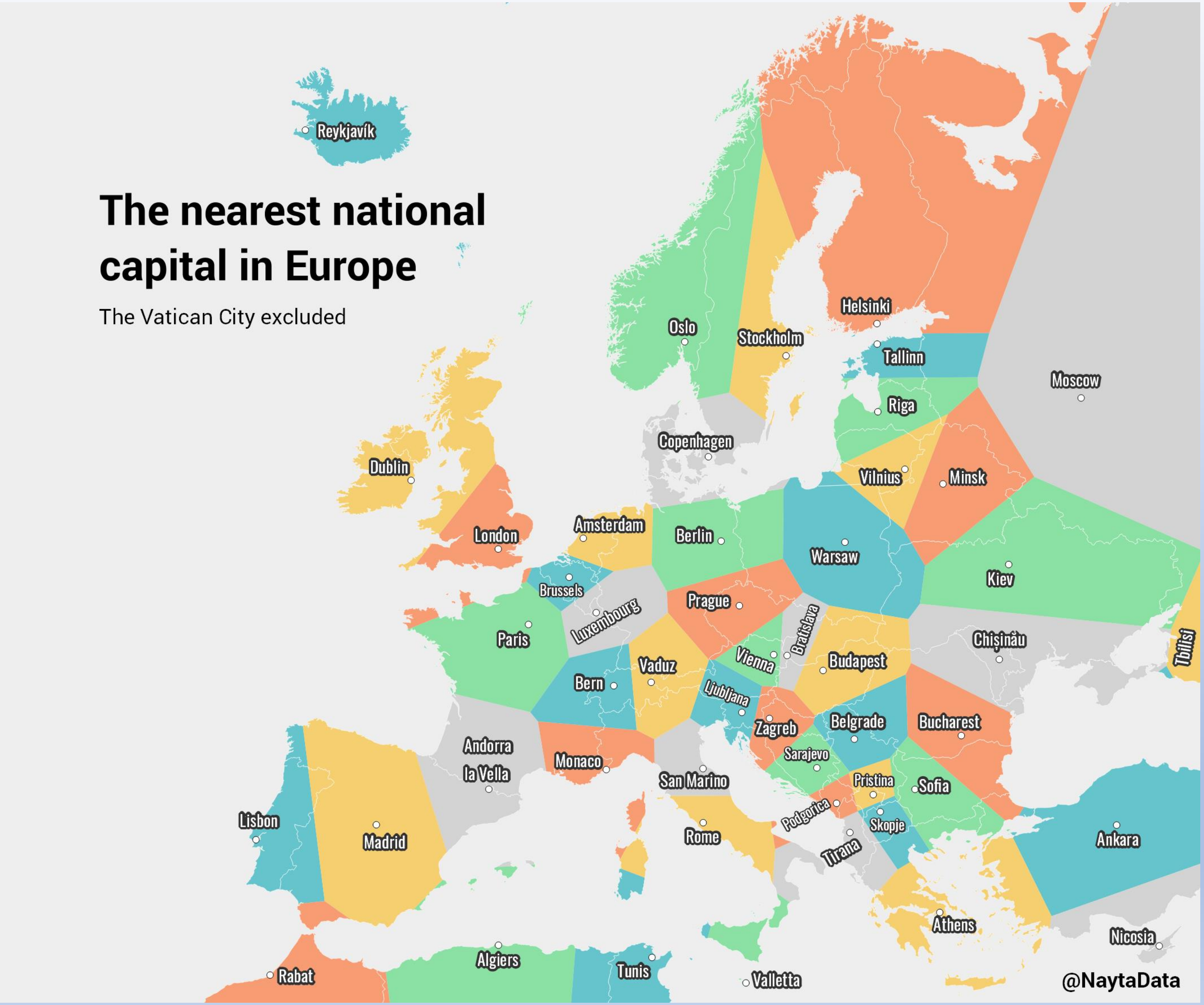
Ciekawostki

Voronoi art



Voronoi in nature




The nearest national capital in Europe
The Vatican City excluded
Reykjavík
Oslo
Stockholm
Helsinki
Tallinn
Riga
Moscow
Copenhagen
Dublin
Vilnius
Minsk
London
Amsterdam
Berlin
Warsaw
Kiev
Brussels
Luxembourg
Prague
Paris
Bratislava
Vienna
Budapest
Chișinău
Tbilisi
Vaduz
Bern
Ljubljana
Zagreb
Belgrade
Bucharest
Andorra la Vella
Monaco
San Marino
Sarajevo
Pristina
Sofia
Podgorica
Skopje
Lisbon
Madrid
Rome
Tirana
Ankara
Athens
Nicosia
Algiers
Tunis
Valletta
Rabat
@NaytaData

Zródła:

Zagadnienia teoretyczne projektu:

- 1. Mark de Berg - "Computational Geometry - Algorithms and Applications"
- 2. Barbara Głut – Prezentacje z wykładu poświęcone Diagramom Voronoi
- 3. Współrzędne środka okręgu opisanego na trójkącie https://en.wikipedia.org/wiki/Circumscribed_circle
- 4. Definicja paraboli jako zbioru punktów równoodległych od kierownicy i ogniska paraboli https://pl.wikipedia.org/wiki/Parabola_(matematyka)

Zagadnienia implementacyjne:

- 1. Triangulacja Delaunay'a z biblioteki Scipy https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.Delaunay.html
- 2. Diagramy Voronoi z biblioteki Scipy https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.Voronoi.html
- 3. Pomysł na realizacje parabol w narzędziu graficznym jako „gęstego" zbioru punktów https://katarzynabuzniak.pl/python-wykres-funkcji-kwadratowej/
- 4. Implementacja drzewa czerwono-czarnego z której skorzystaliśmy https://github.com/keon/algorithms/blob/master/algorithms/tree/red_black_tree/red_black_tree.py
- 5. Ogólne inspiracje, w szczególności obsługa łuków po obsłużenia struktury zdarzeń, znajdowanie łuku znajdującego się nad punktem https://github.com/pvigier/FortuneAlgorithm/blob/master/src/Beachline.cpp

Ciekawe zdjęcia i ilustracje:

- 1. https://vividmaps.com/nearest-national-capital-europe/
- 2. https://researchgate.net/figure/fig5_320027110
- 3. https://towardsdatascience.com/how-to-find-the-nearest-hospital-with-voronoi-diagram-63bd6d0b7b75
- 4. https://www.codingame.com/playgrounds/243/voronoi-diagrams/what-are-voronoi-diagrams
- 5. http://jwilson.coe.uga.edu/EMAT6680Fa08/Kuzle/Math%20in%20Context/Voronoi%20diagrams.html
- 6. https://perso.uclouvain.be/jean-francois.remacle/LMECA2170/slidesVoronoi.pdf

Prezentację przygotowali:

- Paweł Lewkowicz
- Mateusz Słuszniak