Das Voronoi Diagramm

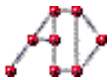
1. Definition

2. Eigenschaften

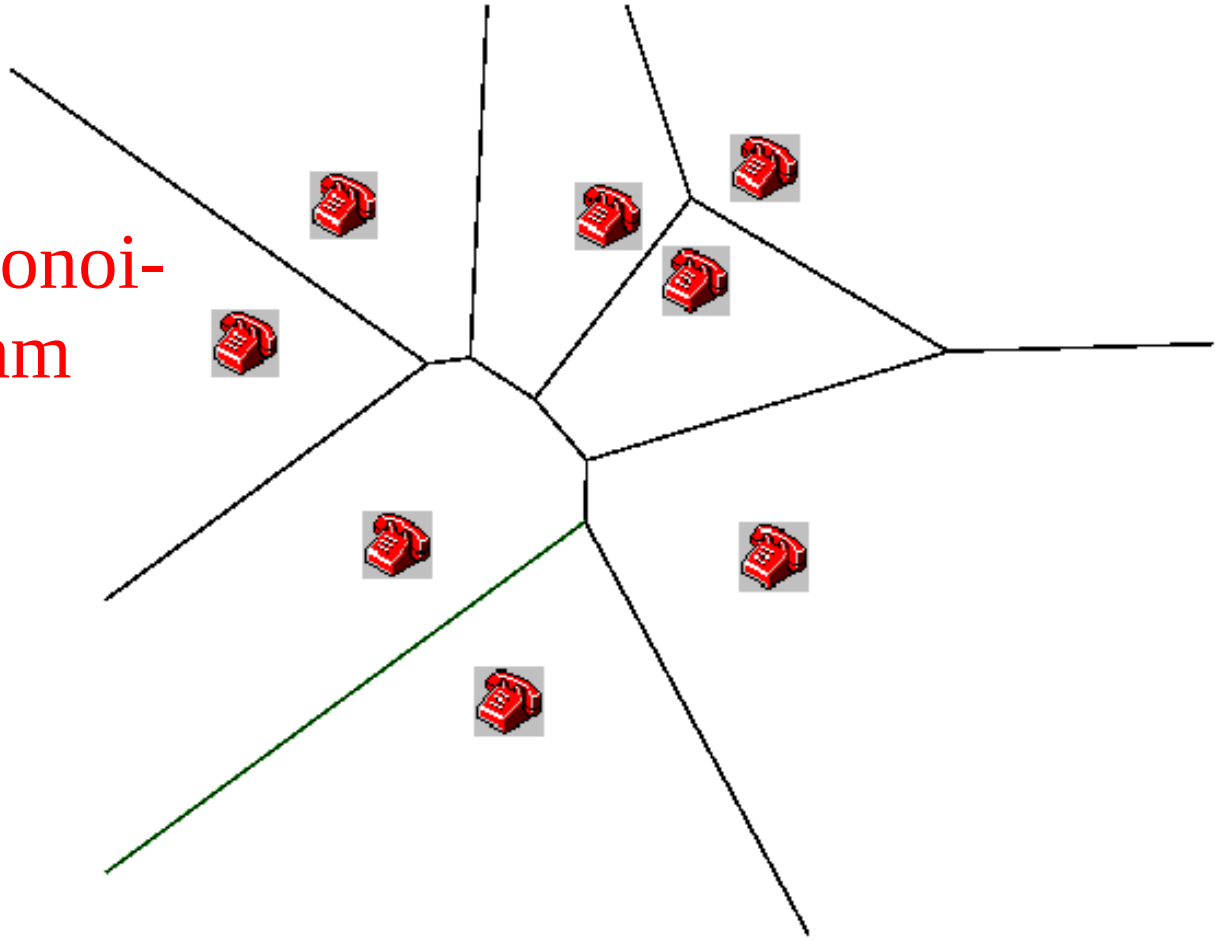
3. Größe und Speicherung

4. Konstruktion

5. Verwendung



Das Voronoi-Diagramm



Voronoi Regionen

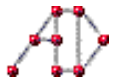
Euklidische Distanz:

$$d(p,q) = \sqrt{(px-qx)^2+(py-qy)^2}$$

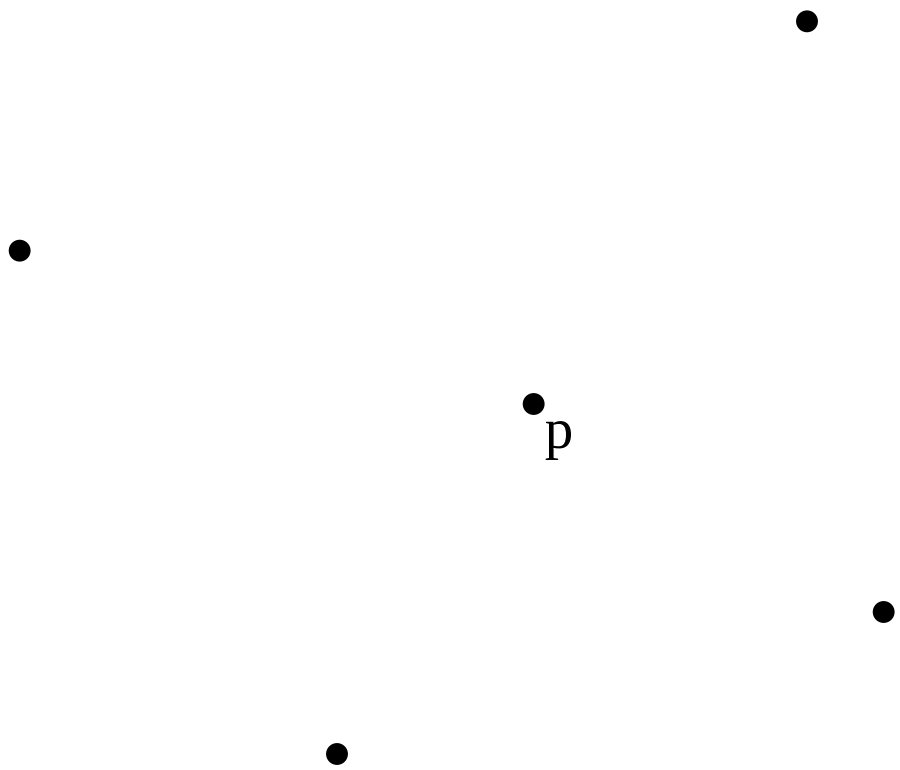
Das Voronoi-Diagramm VD(P) für eine Punktmenge P={p1,...,pn} ist eine Einteilung der Ebene in Gebiete gleicher nächster Nachbarn.

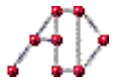
Für einen Punkt p ist die Voronoi Region V(pi) die Menge aller der Punkte, die näher zu pi als zu jedem anderen Punkt pj, j<>i, sind.

VD(P) besteht aus genau allen Voronoi Regionen V(p1),...,V(pn).



V(p) = H(p,q)

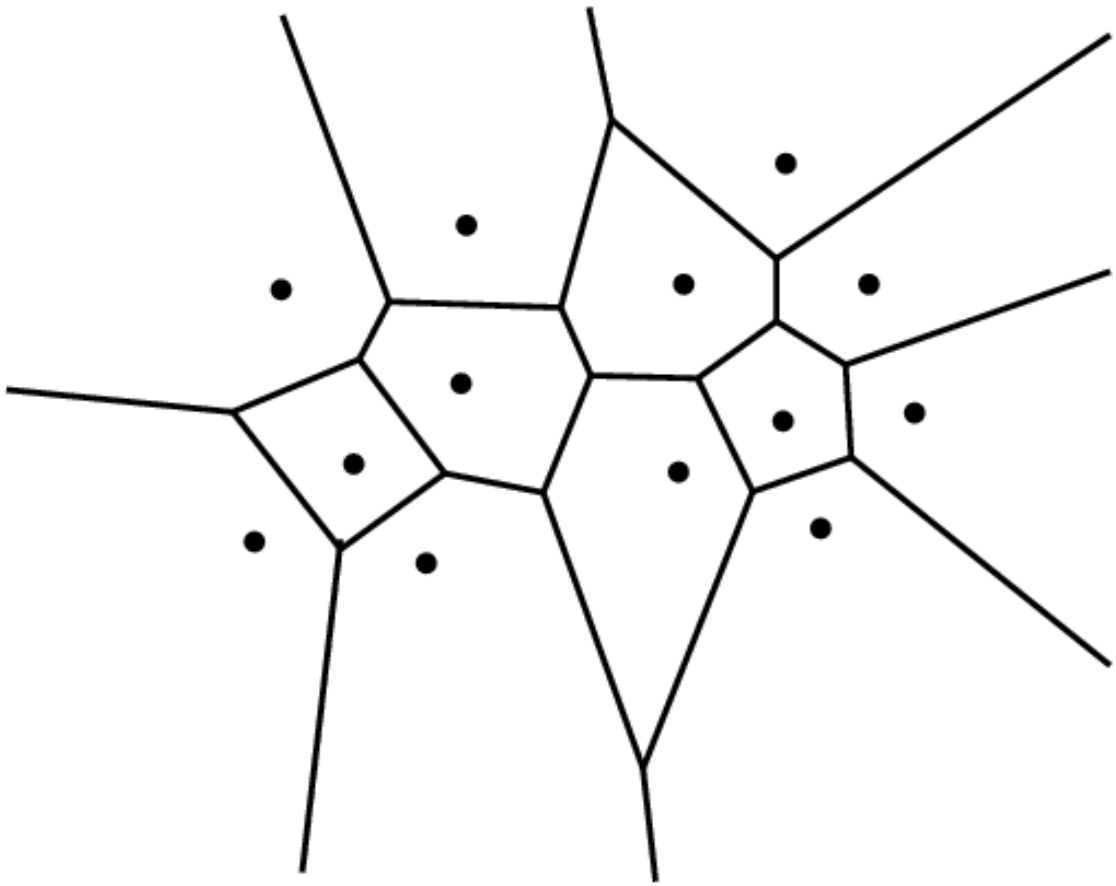




Berechnung des Voronoi-Diagramms

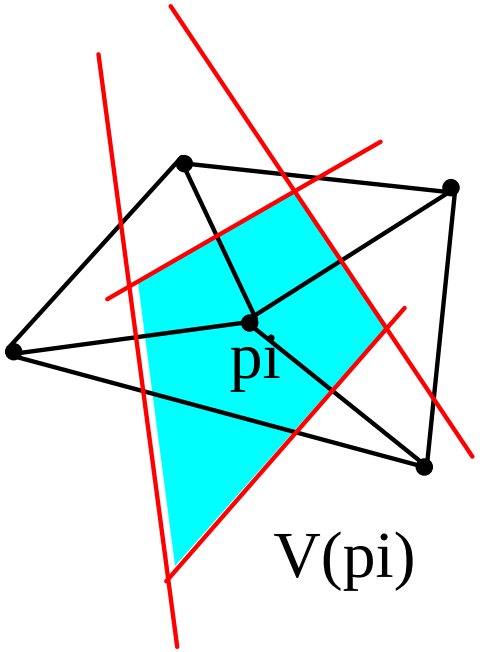
Gegeben: Eine Menge von Punkten (sites)

Gesucht: Eine Unterteilung der Ebene in Regionen gleicher nächster Nachbarn



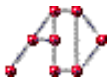
Animationen des Voronoi-Diagramms

http://wwwpi6.fernuni-hagen.de/java/JavaAnimation



haeppchen2.scr

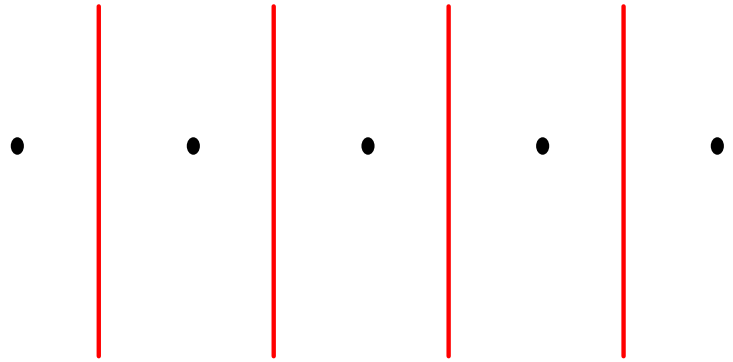
animation.scr



2. Eigenschaften des Voronoi Diagramms

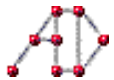
(1) Voronoi Regionen (Zellen) werden von Geradenstücken begrenzt.

Sonderfall:
Punkte kollinear

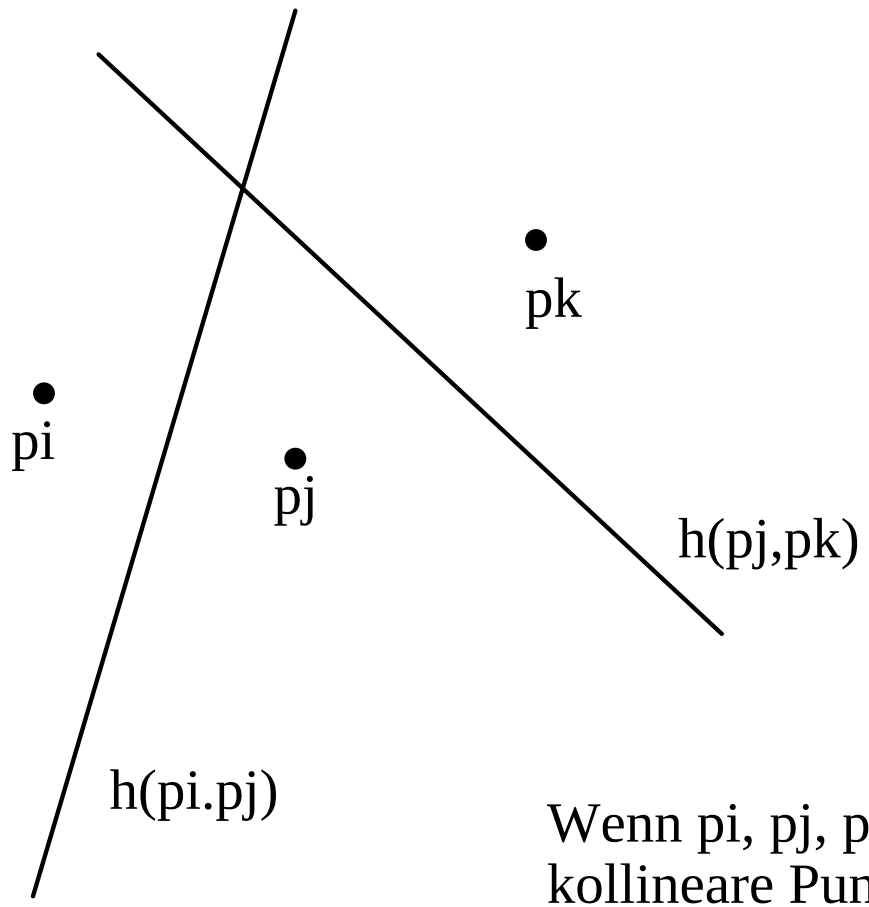


Sonst gilt:

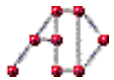
Satz: Das Voronoi Diagramm VD(P) einer Punktmenge P ist zusammenhängend und besteht aus Liniensegmenten und Halbgeraden.



Kanten von VD(P) sind Segmente oder Halbgeraden:



Wenn pi, pj, pk drei nicht kollineare Punkte sind, können h(pi,pj) und h(pj,pk) nicht parallel sein!

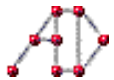


VD(P) ist zusammenhängend:

Ann.: VD(P) nicht zusammenhängend.

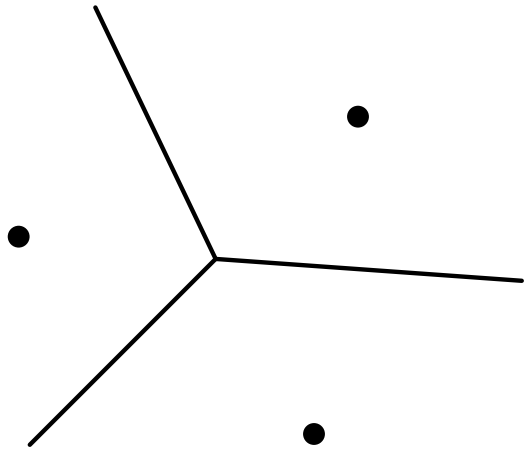
Dann muß aufspaltende, konvexe Voronoi Zelle V(pi) existieren.

=> V(pi) ist konvex
=> V(pi) wird von zwei Parallelen berandet ↯

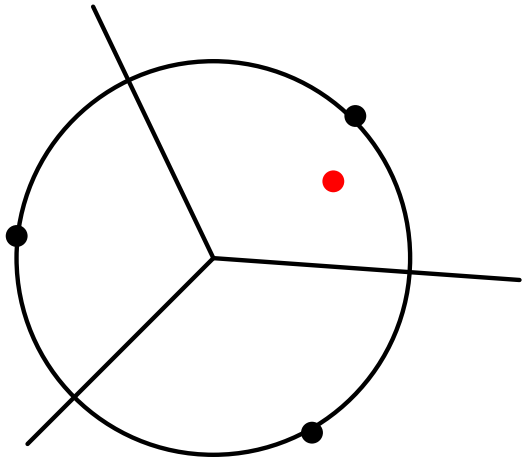


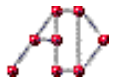
Weitere Eigenschaften (Annahme: Keine 4 Punkte auf einem Kreis):

(2) Jeder Knoten (Ecke) von VD(P) hat Grad 3



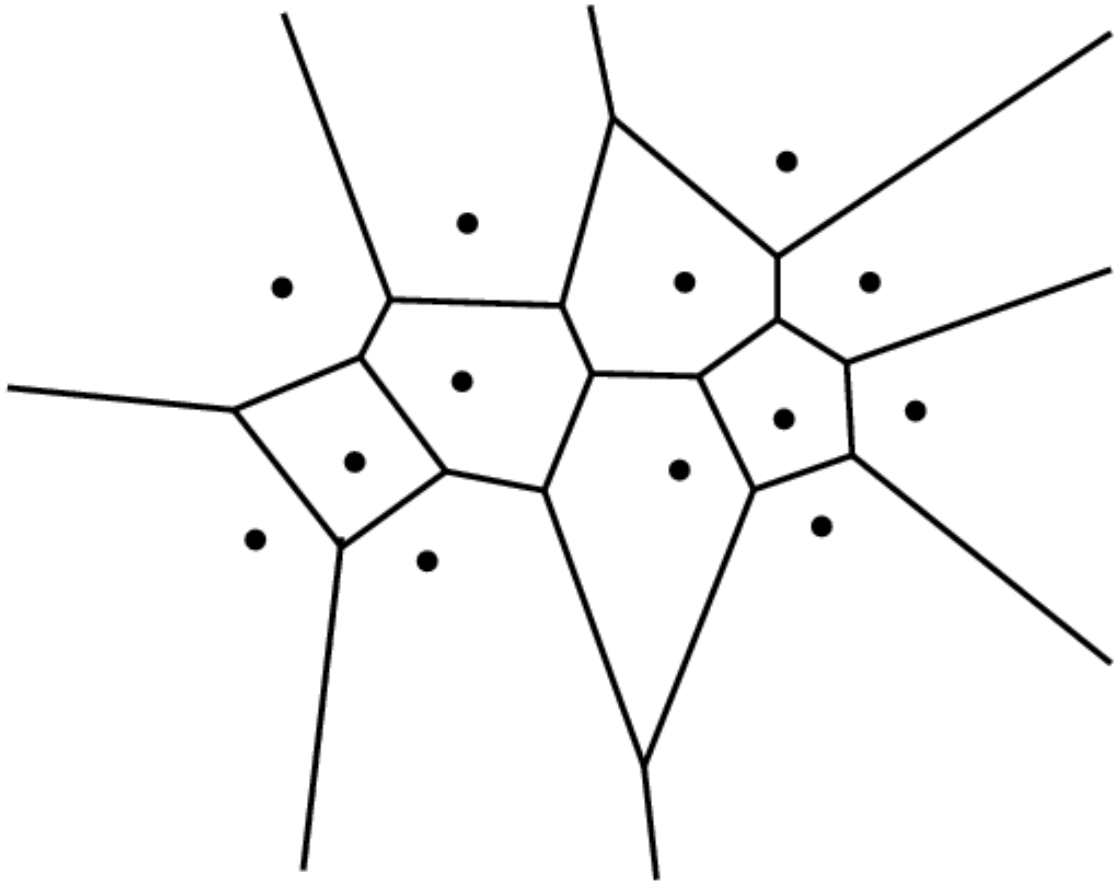
(3) In dem eine Voronoi Ecke definierenden Kreis liegt kein weiterer Punkt.





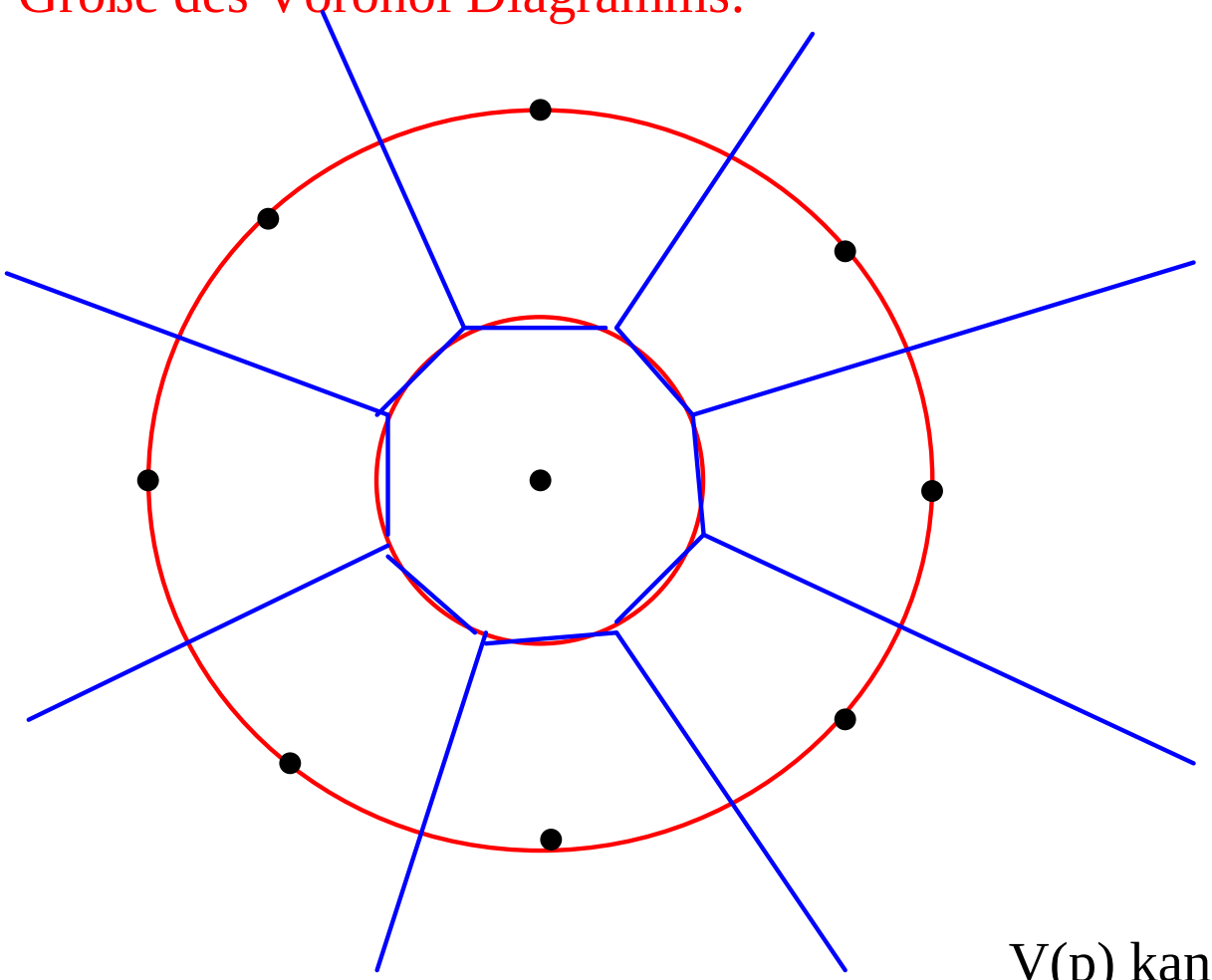
(4) Jeder nächste Nachbar eines Punktes definiert eine Kante der Voronoi Region des Punktes.

(5) Die Voronoi Region eines Punktes ist unbeschränkt genau dann, wenn der Punkt auf der konvexen Hülle der Punktmenge liegt.

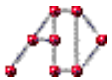


3. Größe und Speicherung

Größe des Voronoi Diagramms:



V(p) kann n Ecken haben!



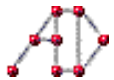
Satz: Das Voronoi Diagramm VD(P) für eine Menge von n Punkten in der Ebene hat höchstens 2n-5 Ecken und 3n-6 Kanten.

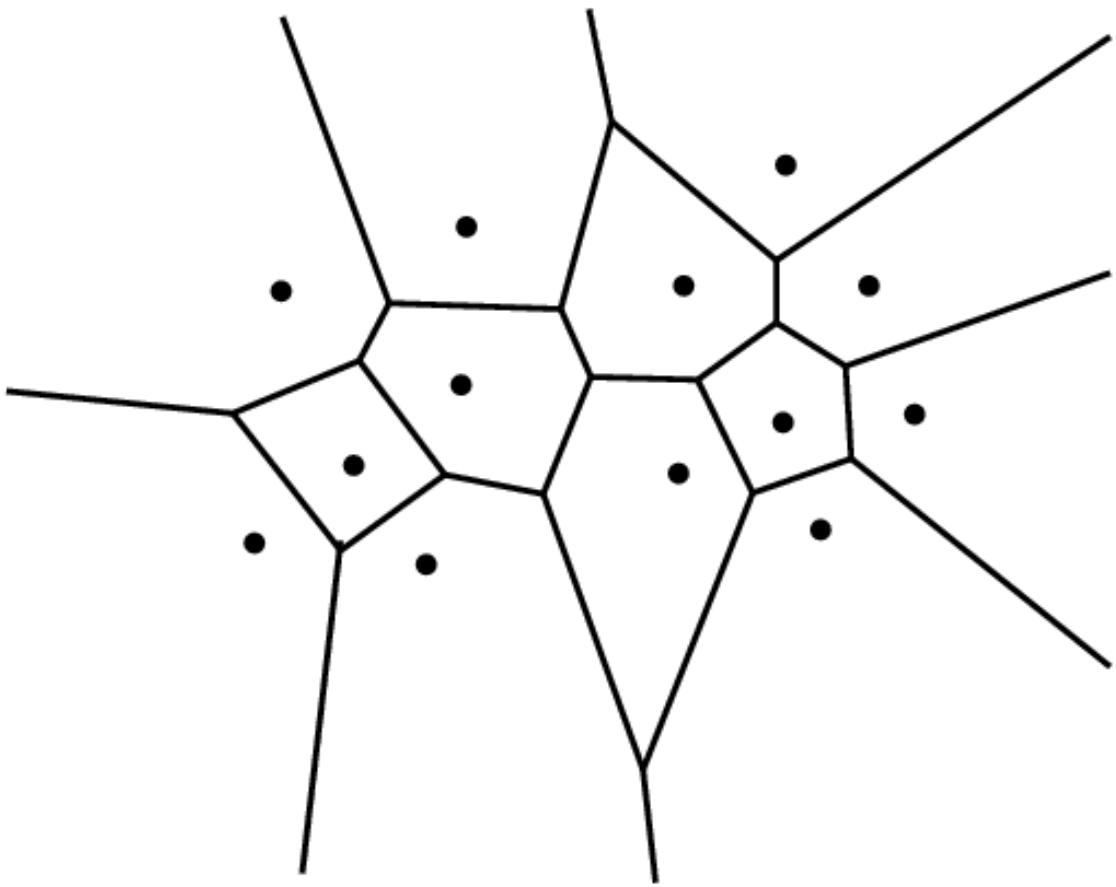
Verbinde alle Halbgeraden mit fiktivem Punkt ∞

Wende Formel von Euler an: $v - e + f = 2$

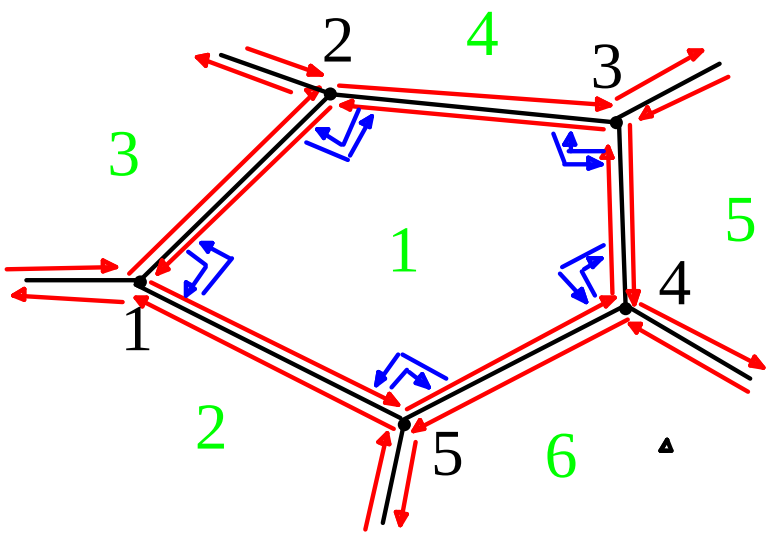
Für VD(P) + ∞ v = #Ecken von VD(P) + 1
e = #Kanten von VD(P)
f = #faces von VD(P) = n

Jede Kante in VD(P) +∞ hat genau zwei Ecken und
Jeder Knoten von VD(P) +∞ hat mindestens Grad 3:
=> Summe aller Knotengrade in Vor(P)∞
= 2 #Kanten von VD(P)
>= 3 (#Ecken von VD(P) + 1)





Speicherung des Voronoi-Diagramms



z.B.

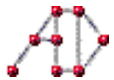
Knoten 1 = {(1,2) | 12}

Fläche 1 = { 15 | [] }

Kante 54 = { 4 | 45 | 1 | 43 | 15 }

3 Records:

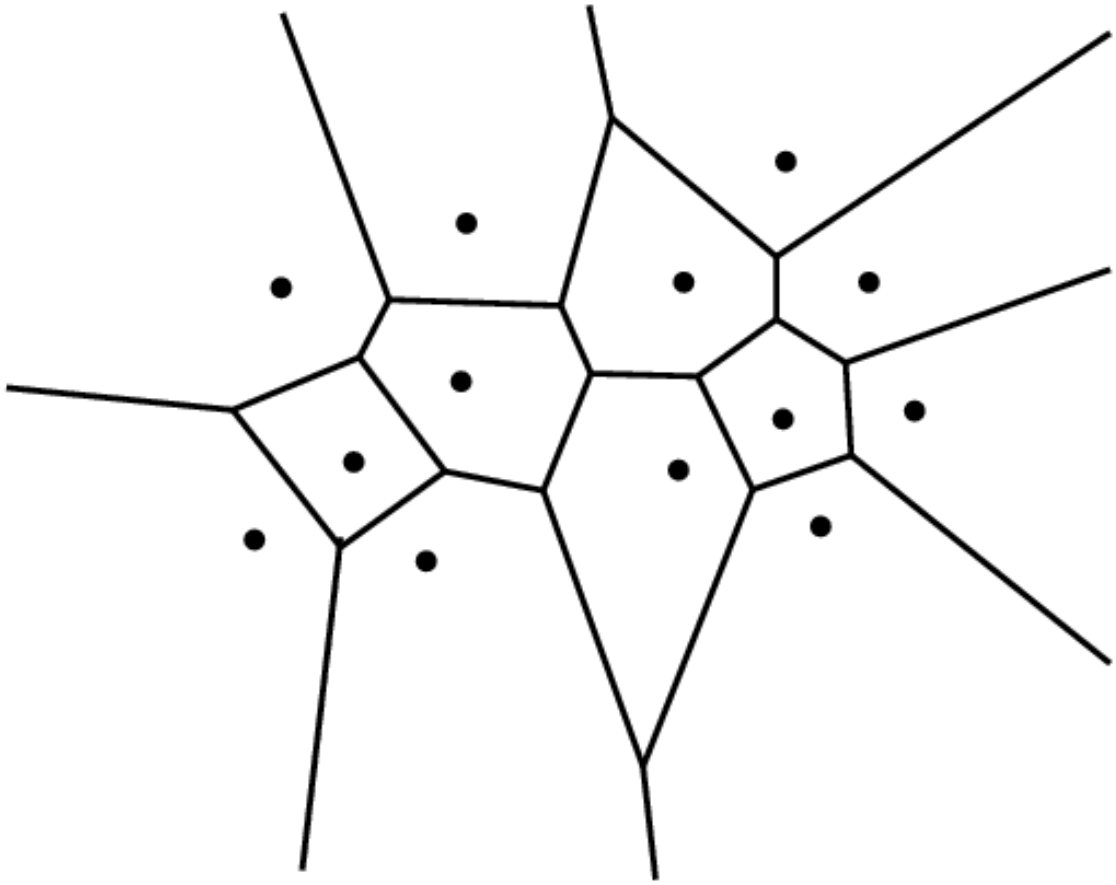
```
vertex {
    Coordinates
    Incident Edge
};
face {
    OuterComponent
    InnerComponents
};
halfedge {
    Origin
    Twin
    IncidentFace
    Next
    Prev
};
```

4. Berechnung des Voronoi-Diagramms

Gegeben: Eine Menge von Punkten (sites)

Gesucht: Eine Unterteilung der Ebene in Regionen gleicher nächster Nachbarn

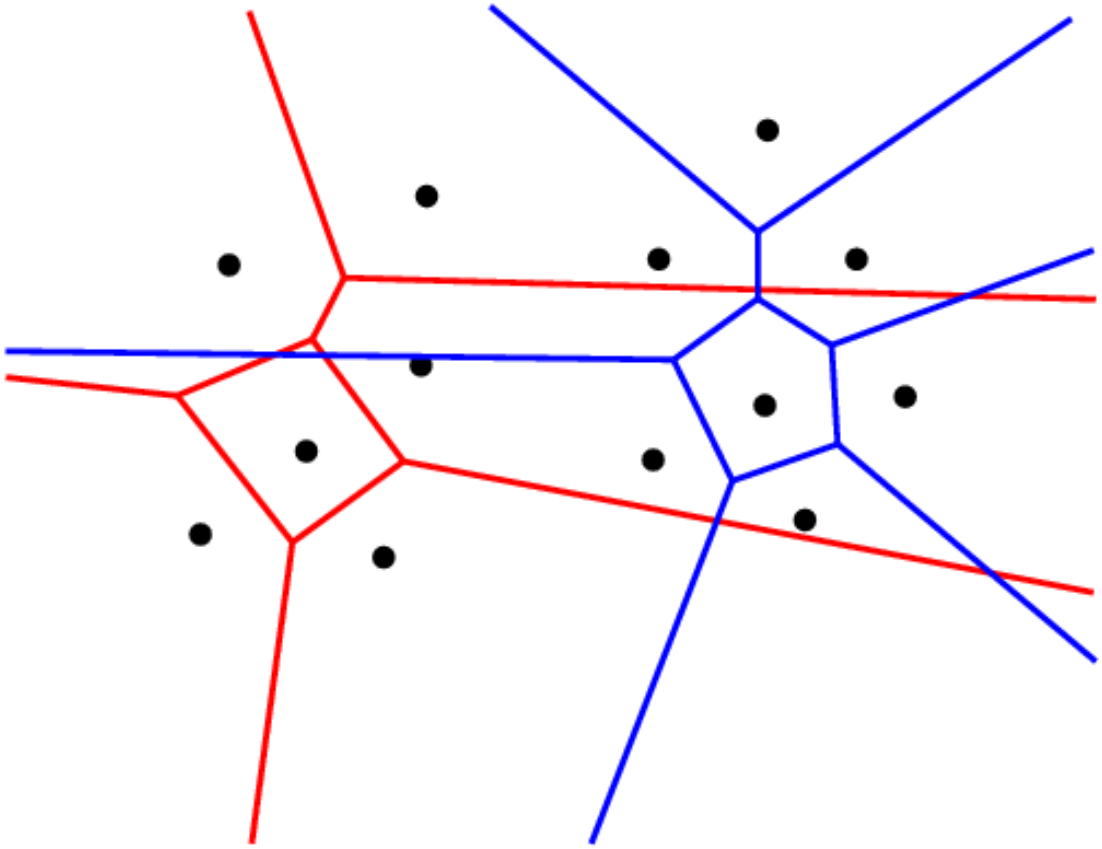


Divide: Einteilung der Punktmenge in zwei Hälften

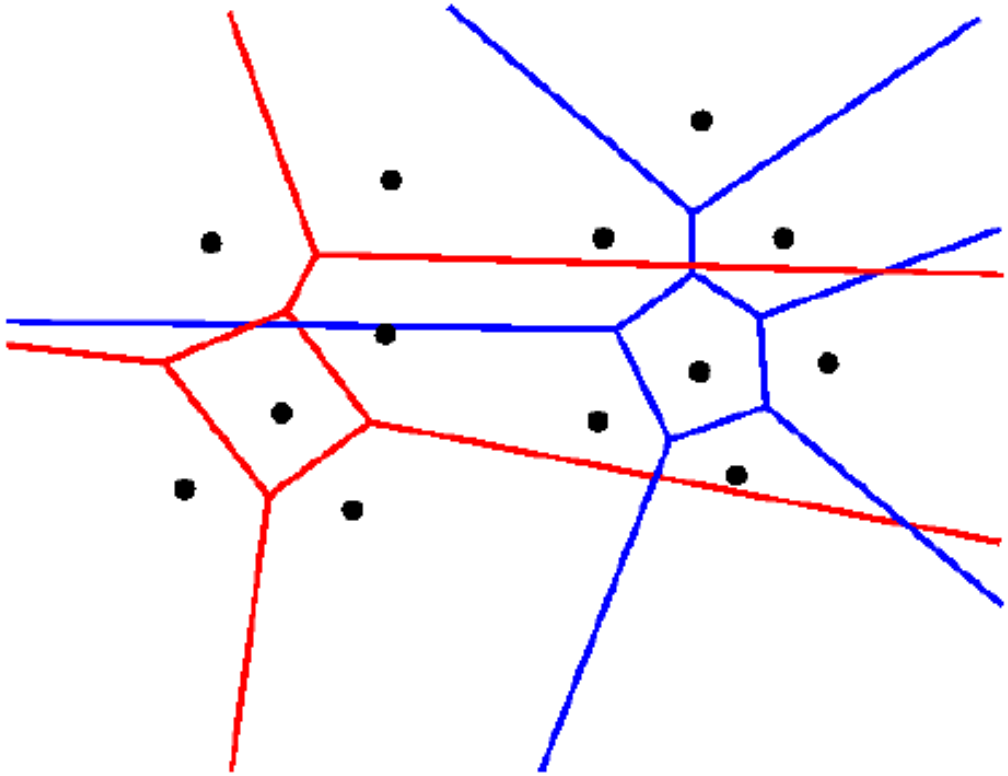
Conquer: Rekursive Berechnung der beiden kleineren Voronoi-Diagramme

Abbruchbedingung:

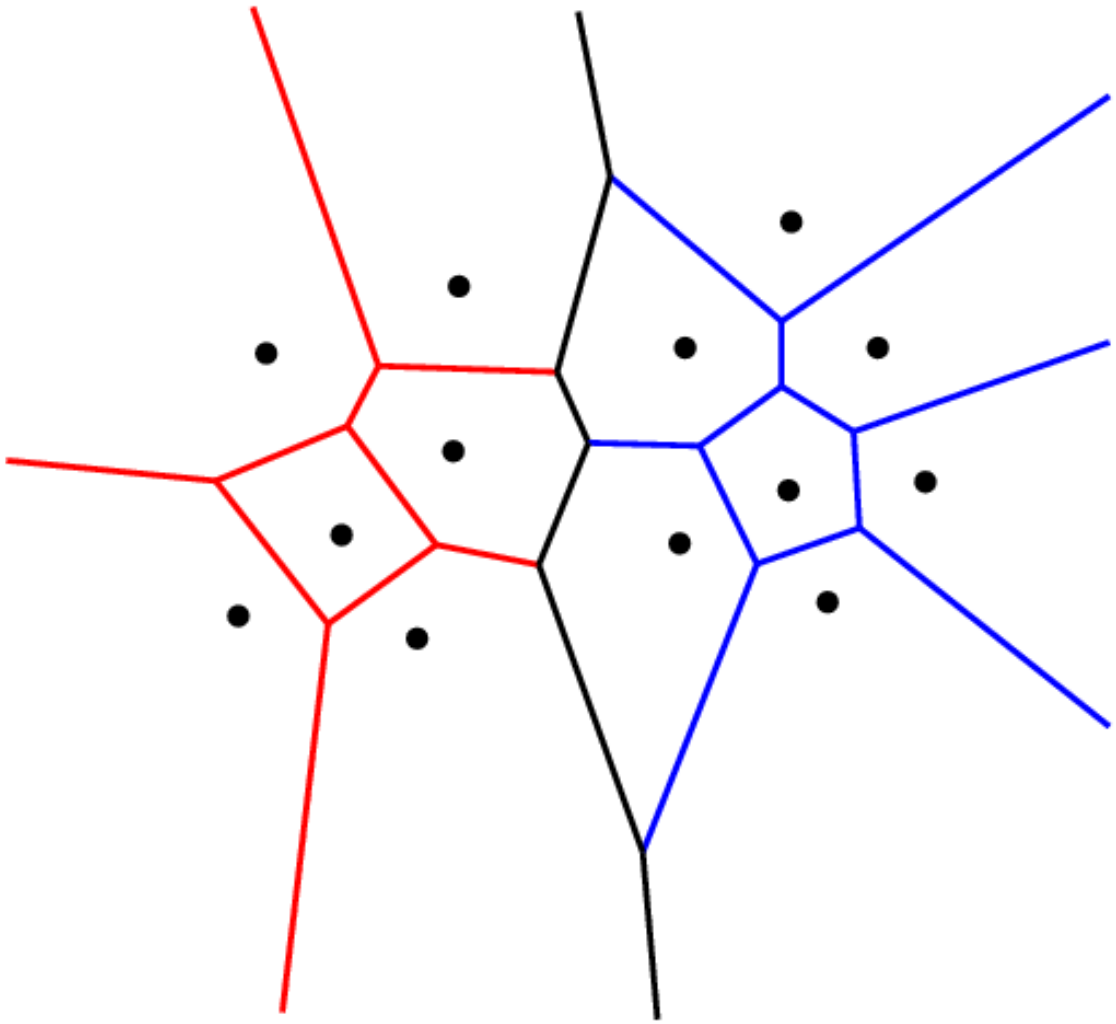
Voronoi-Diagramm eines einzelnen Punktes ist die gesamte Ebene.



Merge: Verbindung der Diagramme durch einen Kantenzug



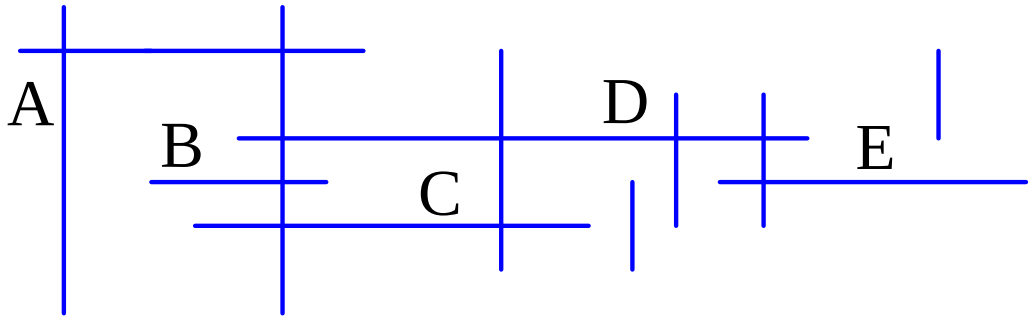
Ergebnis: Das fertige Voronoi-Diagramm

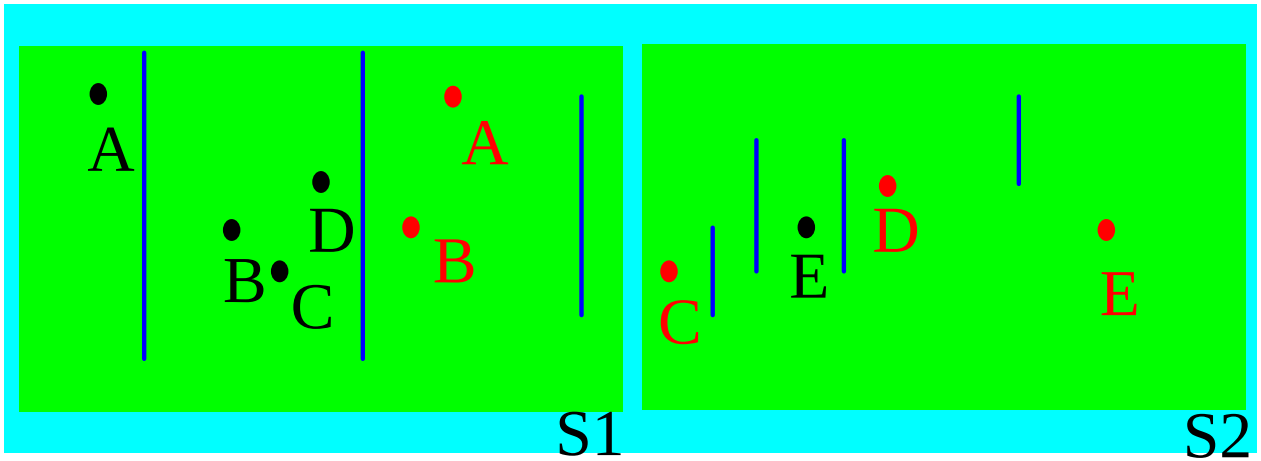


Laufzeit: Bei n gegebenen Punkten O(n log n)

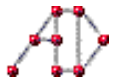
Geometrisches Divide-And-Conquer

Problem: Bestimme alle Paare sich schneidender Segmente



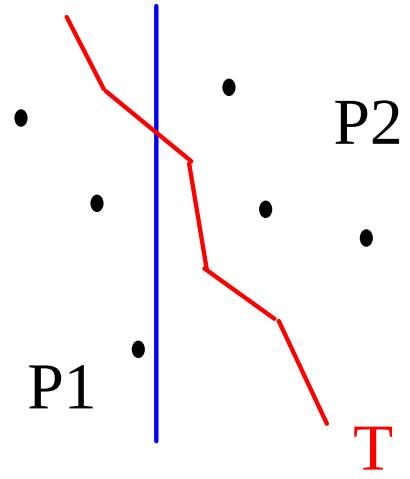


S



DAC-Konstruktion des Voronoi-Diagramms

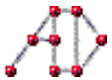
Devide:
Teile P durch eine vertikale Trennlinie T in 2 etwa gleich große Teilmengen P1 und P2. Falls |P|=1 => fertig



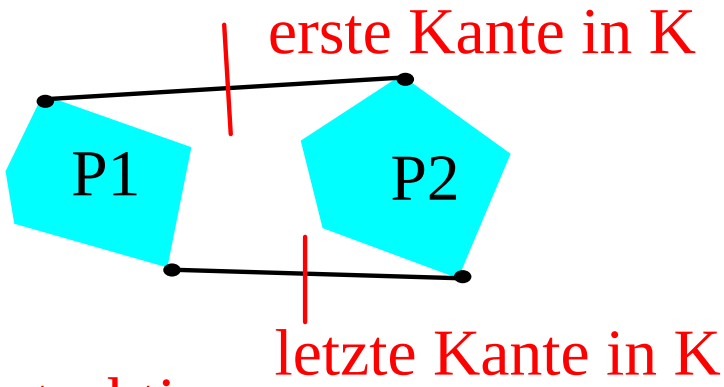
Conquer:
Berechne VD(p1) und VD(p2) rekursiv

Merge:
Berechne den P1 und P2 trennenden Kantenzug K
Schneide VD(P1) und VD(P2) mittels K ab
Veinige Vor(P1) und Vor(P2) und K

Satz: K in O(n) => Laufzeit T(n) = O(n log n)
Bew.: $T(n) = 2\,T(n/2) + O(n)$, $T(1) = O(1)$



Berechnung von K



4 Tangentialpunkte

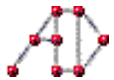
Beobachtung: K y-monoton

Inkrementelle (sweep-line) Konstruktion
(p1 in P1 und p2 in P2 mit Senkrechte m, Sweep l)

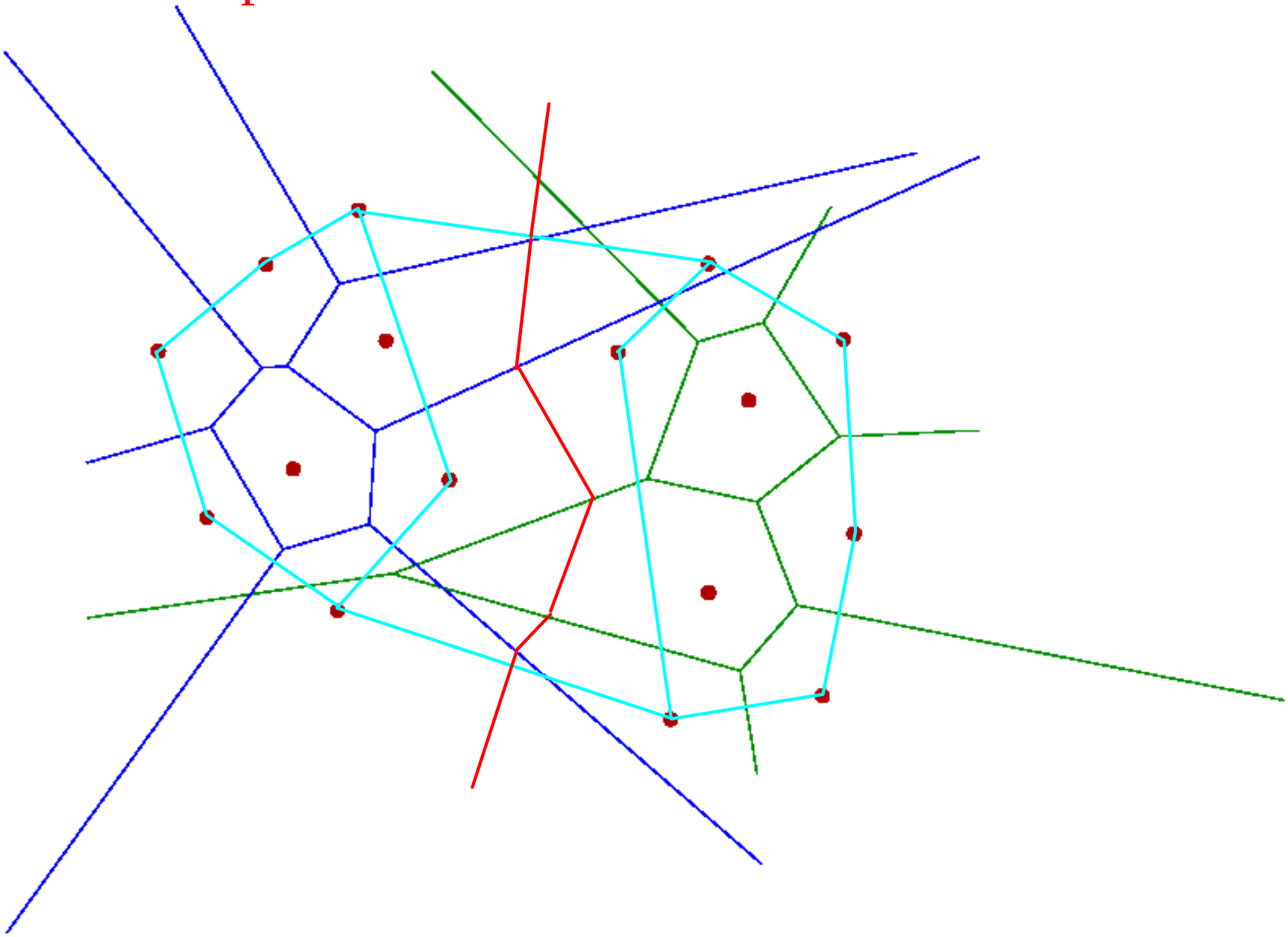
Ermittle Schnitt s1 von m mit VR(p1) unterhalb l
Ermittle Schnitt s2 von m mit VR(p2) unterhalb l

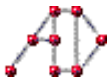
Erweitere K um Geradenstück l si
Setze l = si
Berechne neues K definierendes Paar p1,p2

Satz: Laufzeit O(n)
Bew: V(pi) sind konvex, demnach wird jede Vor-Kante
nur einmal besucht

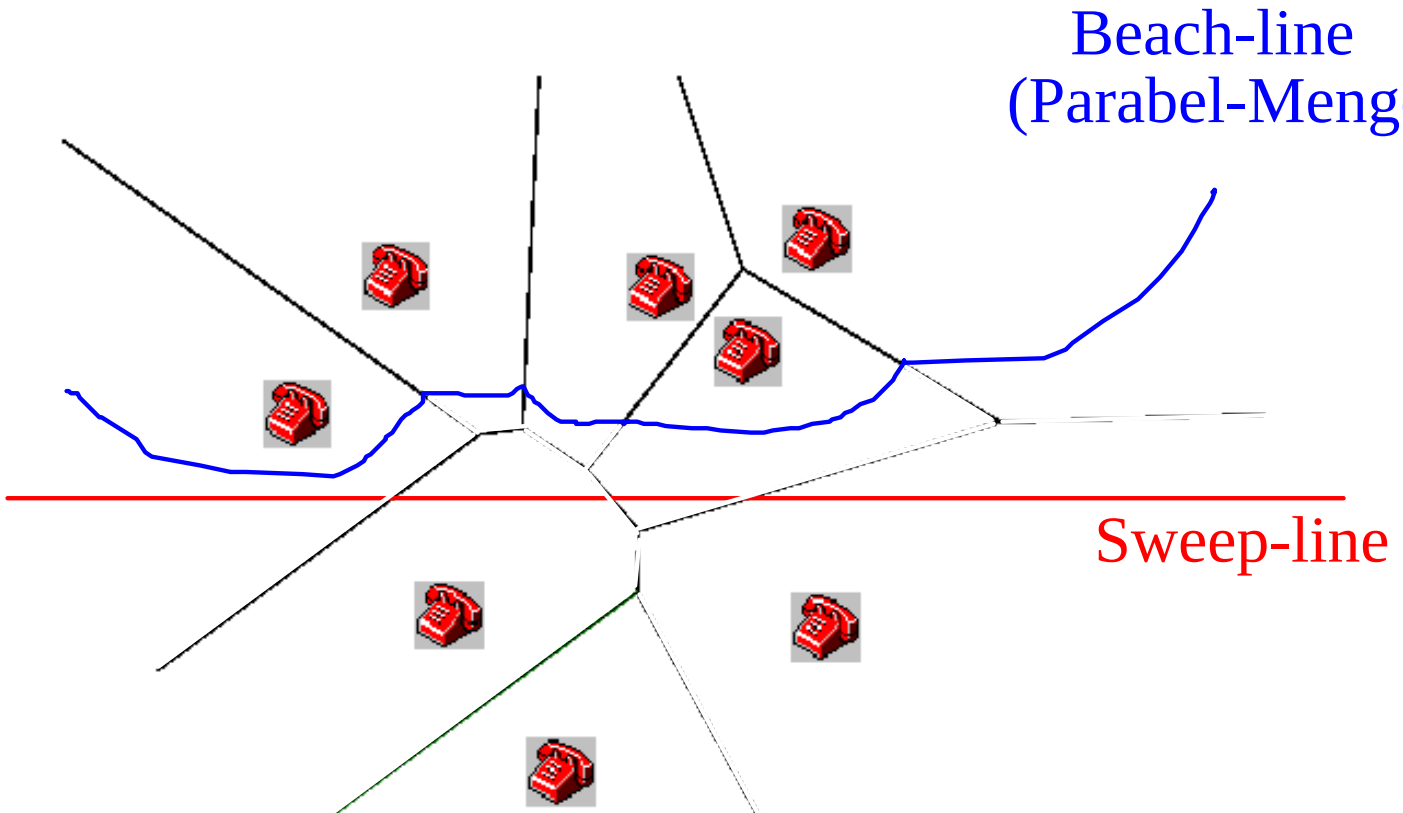


Ein Beispiel



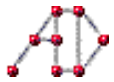


Alternative: Fortune's Algorithmus



Beobachtungen:

Schnitte der Parabeln definieren Kanten
Neue "Telefone" definieren neue Parabeln
Parabel-Schnitte verschwinden, wenn C(P,q) 3 Punkte hat



5. Verwendung (feste Objektmenge)

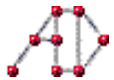
Dichtestes Punktepaar:
Durchlaufe Kantenliste für VD(P) und ermittle Minimum

Alle Nächsten Nachbarn
Durchlaufe Kantenliste für VD(P) für alle Punkte und bestimme jeweils nächste Nachbarn

Minimaler Spannender Baum (nach Kruskal)

1. Jeder Punkt p aus P definiert 1-elementige Menge
2. Solange mehr als eine Menge T existiert
   - 2.1 Finde p,p' mit p in T und p' nicht in T mit d(p,p') minimal
   - 2.2 Verbinde T und p' enthaltenes T' (Union)

Satz: Alle Verfahren in O(n log n)



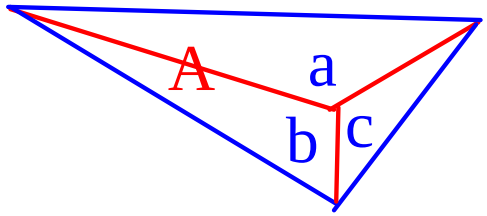
Anwendungen (dynamische Objektmenge)

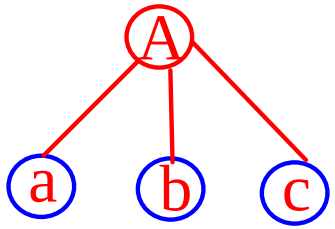
Suche nächsten Nachbarn

Idee: Hierarchische Untergliederung von VD(P)

1 Schritt: Triangulierung abgeschlossener Voronoi-Regionen
(einfach, da konvex, O(n) groß)

2. Schritt: Zusammenfassung von Dreiecken
und Aufbau eines
Suchbaumes





Regel von Kirkpatrick:
Entferne jeweils Punkte mit Grad < 12,
es sei denn sein Nachbar ist bereits entfernt.

Satz:
Unter Anwendung der Regel von Kirkpatrick entsteht
ein Suchbaum logarithmischer Tiefe