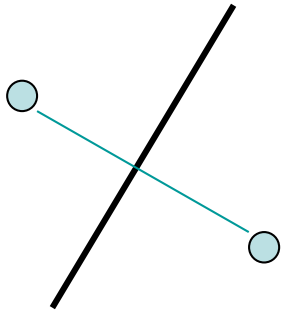
テーマ６：ボロノイ図とデローネイ三角形分割

ボロノイ図，デローネイ三角形分割

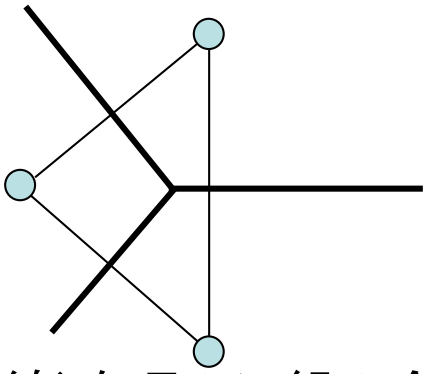
ボロノイ図とは

・平面上に多数の点が与えられたとき，平面をどの点に最も近いかという関係で分割したものをボロノイ図(Voronoi diagram)という.

・２点だけの場合
２点の垂直２等分線による分割



・３点の場合
３点で決まる三角形の外接円の中心から各辺に引いた垂直線による分割線



・２点からの等距離線（垂直２等分線）を巧みに組み合わせたものである.

ボロノイ図の応用

商圏の定義：社会地理学への応用
植物の勢力圏図
ロボティックスへの応用（最も安全な経路の発見）

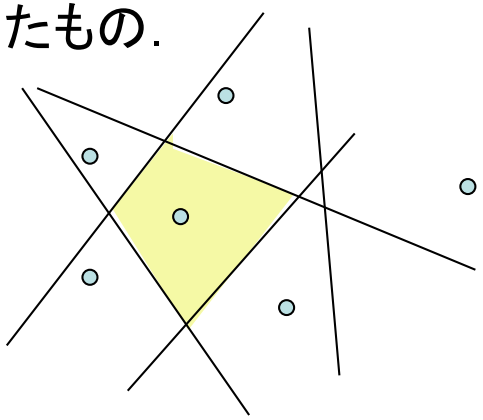
ボロノイ図の構成

・母点（サイト）　最初に与えられた点
・ボロノイセル　各母点の勢力圏（その点が最も近くなる領域）
・ボロノイ辺　　ボロノイセルの境界

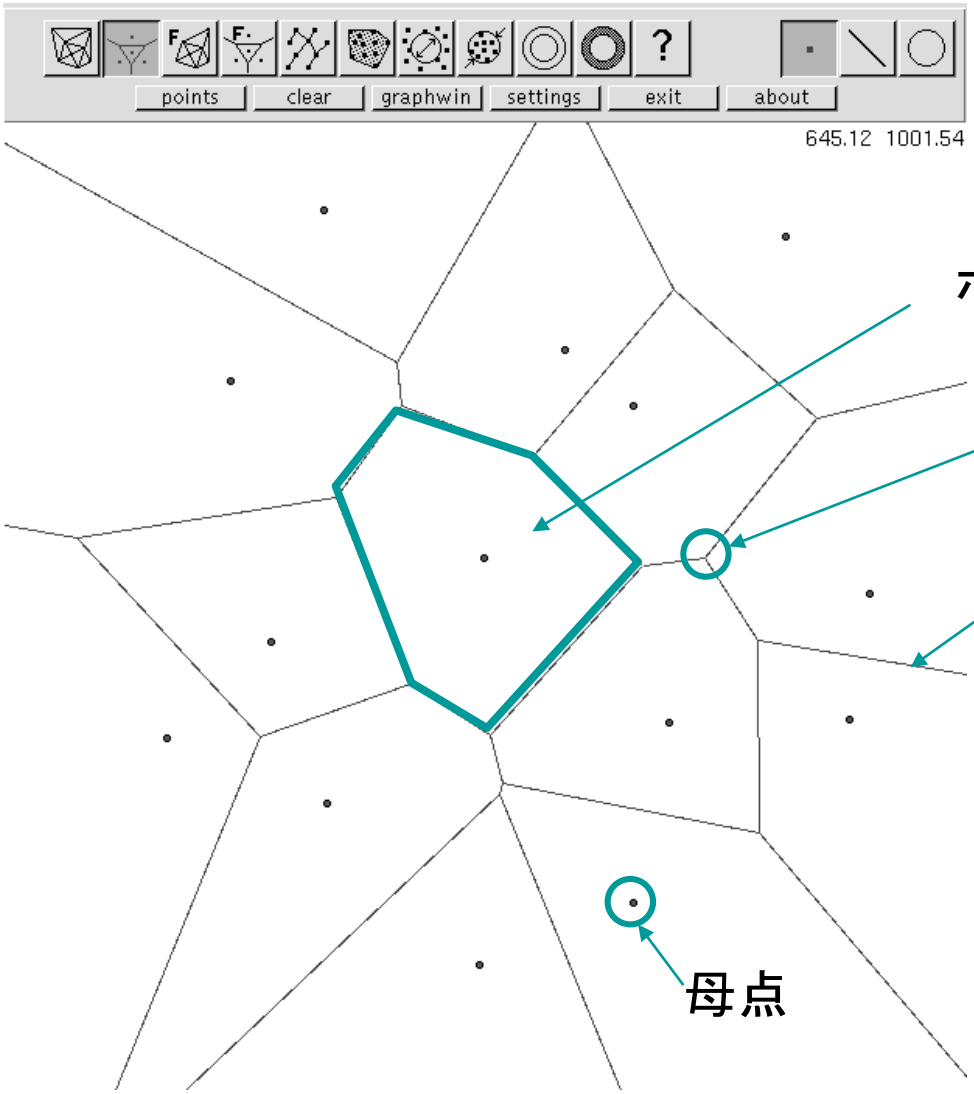
Pを入力点（母点）の集合とするとき，

Vor(P): Pのボロノイ図
$V(p_i)$: 母点p_iのボロノイセル
h(p, q): ２点pqの垂直２等分線で定まる2つの半平面のうち，点pを含むもの
このとき，
V(p)は，すべてのqについてh(p, q)の共通部分をとったもの．

$$V(p_i)=\bigcap_{j\neq i} h(p_i, p_j).$$



LEDAによるボロノイ図の描画

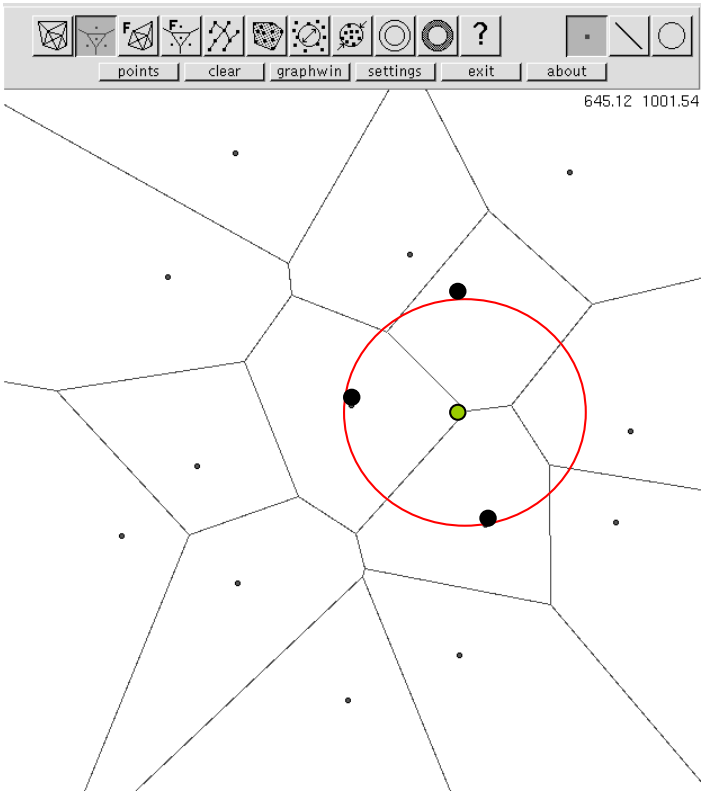


最近は
http://www.cs.cornell.edu/home/chew/Delaunay.html
など，Webでも数多く存在．

ボロノイ図の性質

縮退がなければ,

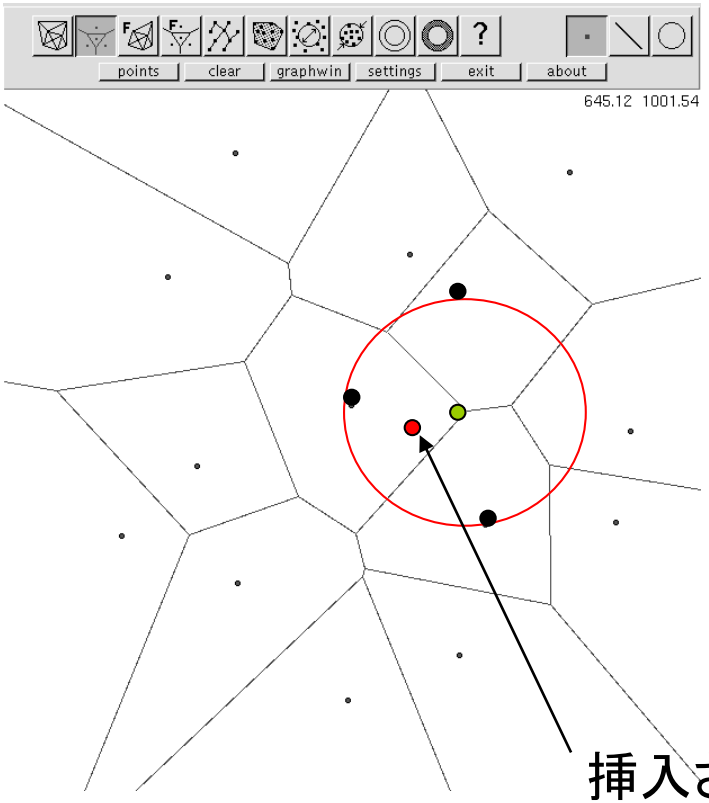
ボロノイ頂点は3つのボロノイセルの共通の頂点である.
それらの3点を通る円の中心はそのボロノイ頂点である.
その円の内部に他の母点は存在しない.

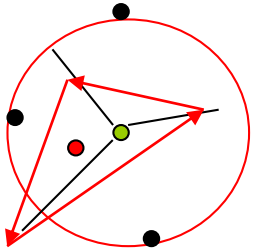


3つの母点を通る空円
中心はボロノイ頂点

新たな母点の挿入

新たな母点が挿入されたとき，まず，その母点を含むボロノイセルを求め，その周上で最も近いボロノイ頂点を求める．





そのボロノイセルの母点との垂直2等分線を求め，それがボロノイ辺と交わる最初の交点を求め，次のボロノイセルに移る．このようにして1周すれば，そのループの中の部分を切り取れば出来上がり．

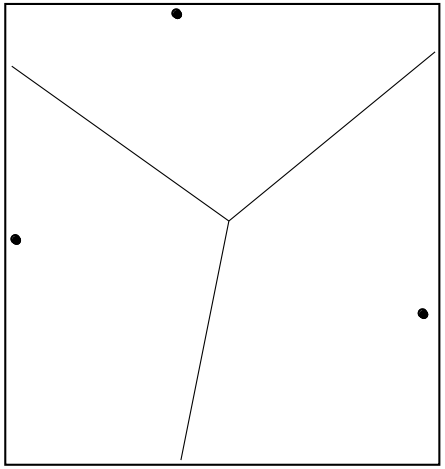
ボロノイ図構成のアルゴリズム

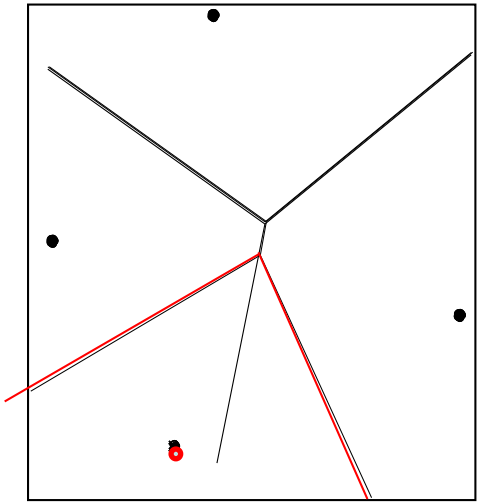
逐次添加法

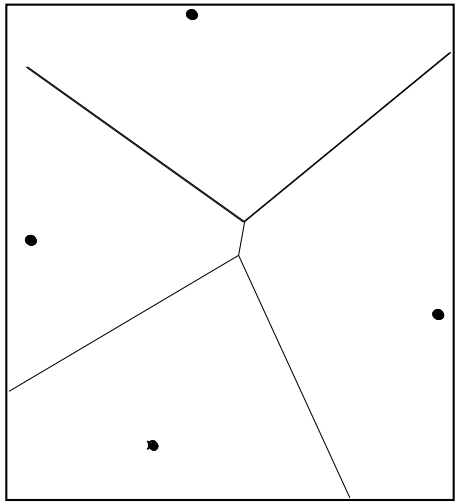
母点を適当な順序に並べた後，ボロノイ図を更新しながら1点ずつ加えていく方法.

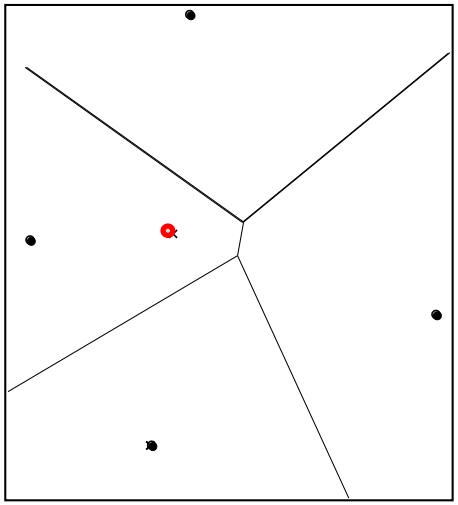
(1) 母点をランダムな順に並び替え，p[1], p[2], … p[n]とする.
(2) p[1], p[2], p[3]に対するボロノイ図を構成する. ➔ Vor(3).
(3) for i=4 to n do
(4)　Vor(i-1)において，母点p[i]を含むボロノイセルV[q]を求める.
(5)　点qと点p[i]との垂直2等分線が最初に交差するボロノイ辺を求める.
(6)　そのボロノイ辺に接する他方のボロノイセルV[r]を求める.
(7)　q=rとして上記の(5)(6)の操作を繰り返し，q=p[i]となれば終了.
(8)　上記の操作で得られたループをボロノイ図に加える.
(9)　ループの内部のボロノイ辺とボロノイ頂点を削除する.

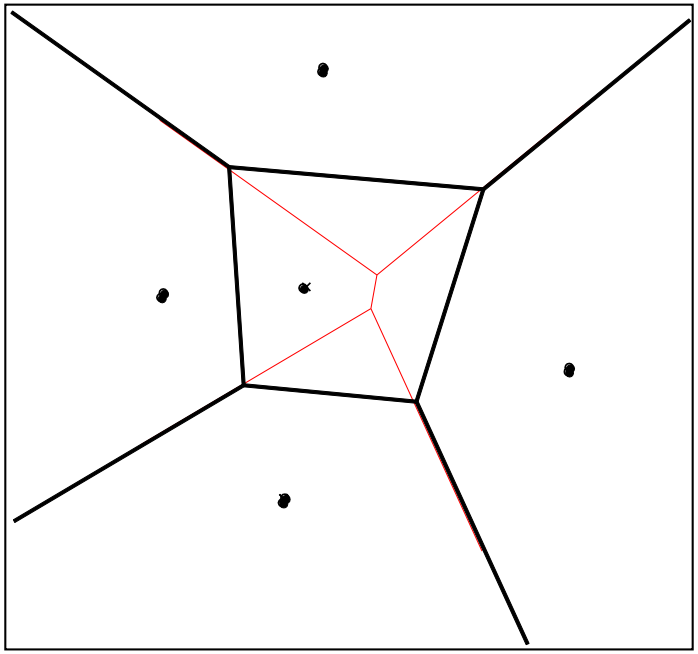
上記のランダム化アルゴリズムの計算時間の期待値は
O(n log n).
最初に母点をランダムな順序に並び替えるのが重要.



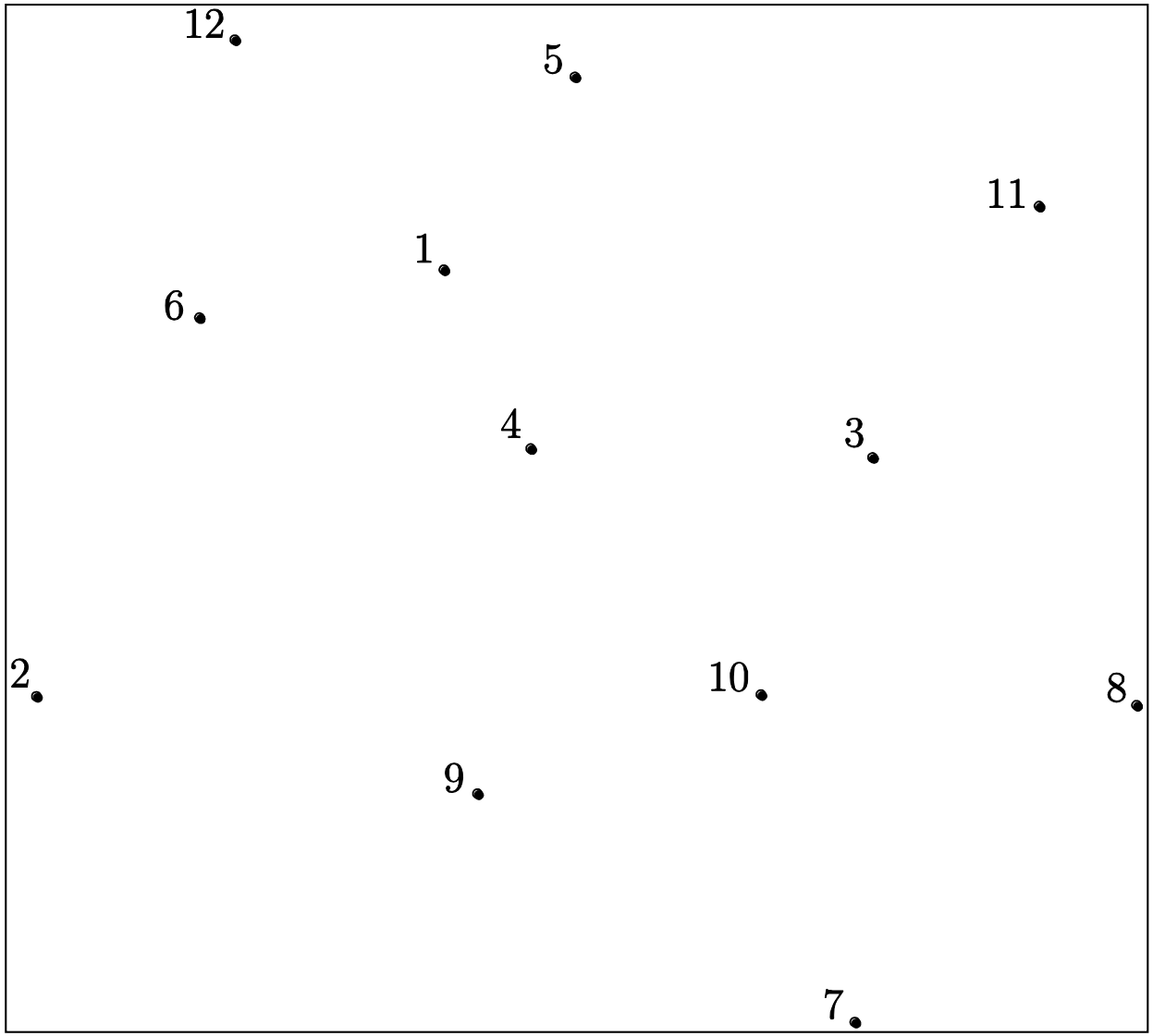








練習問題：下記の点集合に対するボロノイ図を描け.

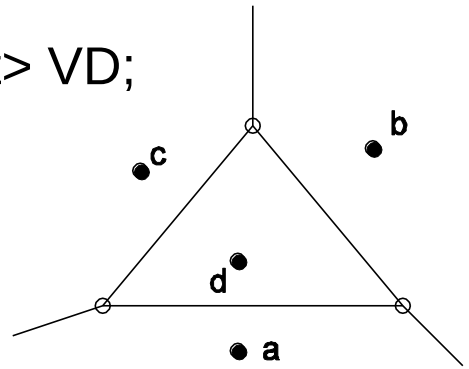


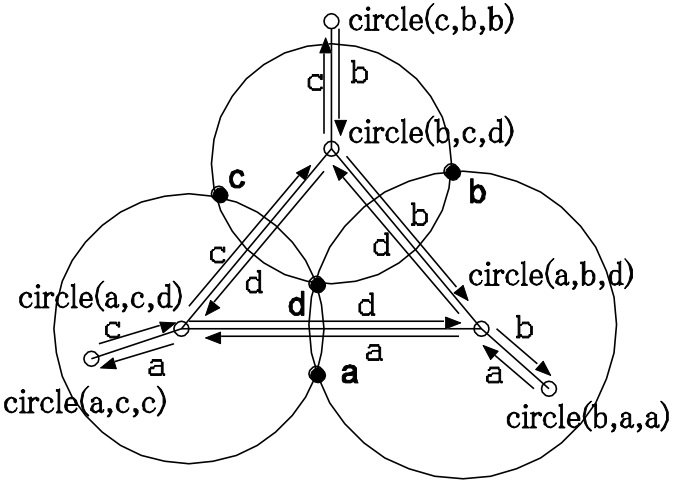
ボロノイ図の描画

LEDAを用いると，母点の集合を点のリストSとして与えると，
　　VORONOI(S, VD);
として関数VORONOI()を呼び出すだけでボロノイ図VDを
有向グラフの形で得ることができる.

表示するには幾何的情報を求める必要がある.

GRAPH<circle, point> VD;
list<point> S;
VORONOI(S, VD);





ボロノイ図の頂点➔円の情報を与える.
　　頂点で交わる3本のボロノイ辺に関連する3つの母点を通る円.
ボロノイ図の辺　➔点の情報を与える.
　　ボロノイ辺に対応する母点（辺の左にあるボロノイセルの母点）

ボロノイ図の描画

ボロノイ図の頂点➔円の情報を与える.
　　頂点で交わる3本のボロノイ辺に関連する3つの母点を通る円.
　　無限遠の頂点：　無限の半径をもつ円を対応させる

vをボロノイ頂点とすると
　VD[v].center(): ボロノイ頂点vに関連する円の中心.
　VD[v].point1(), VD[v].point2(), VD[v].point3():
　　ボロノイ頂点vを特徴付ける3つの母点.
　VD.outdeg(v): 頂点vに接続する辺の本数（1なら無限遠点）

```
node v;
forall_nodes(v, VD){
    if( VD.outdeg(v) < 2) continue; 無限遠の頂点は無視
    point p = VD.center();              ボロノイ頂点を特徴づける円の中心
    W.draw_circle(p, 1, green);       ボロノイ頂点を表示
}
```

ボロノイ図の描画

ボロノイ図の辺　➔点の情報を与える.

ボロノイ辺に対応する母点（辺の左にあるボロノイセルの母点）

有向辺であることに注意. 辺eの反対向きの辺はVD.reverse(e).

ボロノイ辺(u, v)の表示

・有限の辺の場合

線分segment(VD[u].center(), VD.center(v))を描画

・一方が無限遠点の場合(頂点vが無限遠点の場合）

条件：　VD.outdeg(u>1 && VD.outdeg(v)==1

頂点vに関連する3つの母点のうち1番目と3番目は必ず異なる（定義）

3番目の点から1番目の点に向かう線分を90度回転した方向に

頂点uから半直線を引けばよい.

vector vec = VD[v].point3() – VD[v].point1();

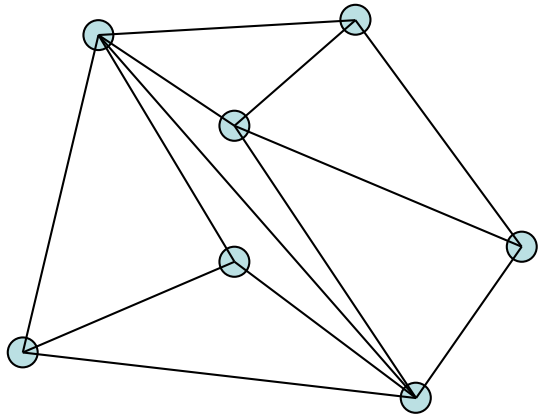
point cv = VD[u].center() + vec.rotate90();

ray(VD[u].center(), cv)を描画

```
if(VD.outdeg(u>1 && VD.outdeg(v)==1){
    vector vec = VD[v].point3() – VD[v].point1();
    point cv = VD[u].center() + vec.rotate90();
    W.draw_ray( VD[u].center(), cv, blue);
}
```

三角形分割

P:　平面上の点集合
すでに引いた線分と交差しない限りPの２点を結ぶ線分を引く。
このようにしてできた図形が三角形分割。



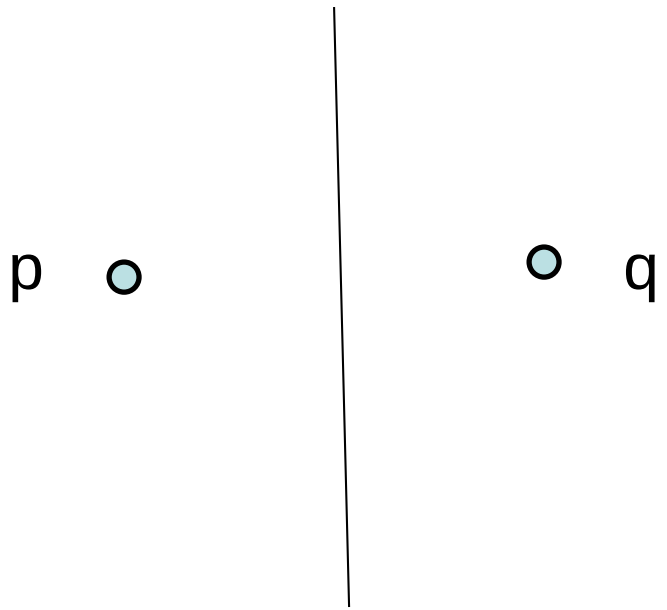
出来上がった図形は必ず三角形で構成されている。
４角形があれば、もう１本線分を引くことができるから。

問題：何通りも三角形分割は考えられるが、その中で最小の内角を最大にする三角形分割を求めよ。

実は、ボロノイ図の双対グラフとして得られる三角形分割が上の問題の解
➔ドローネイ三角形分割

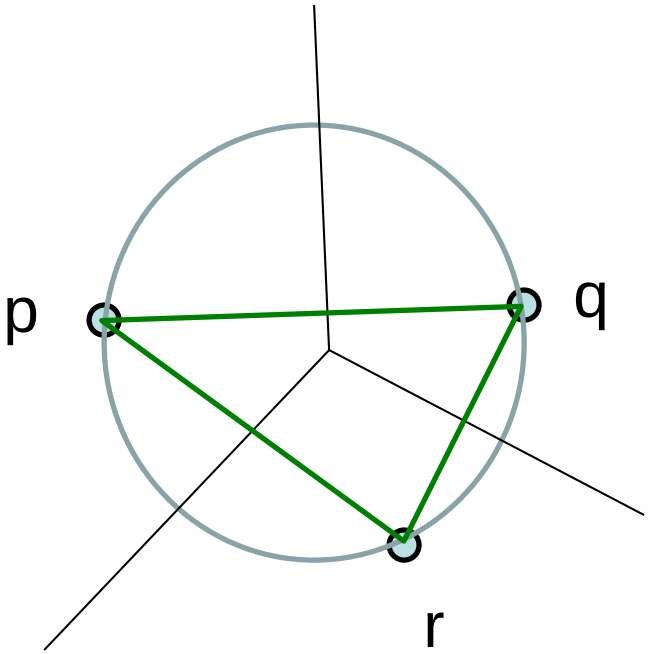
ボロノイ図とデローネイ三角形分割を求める定数作業領域アルゴリズム

ボロノイ図とデローネイ三角形分割



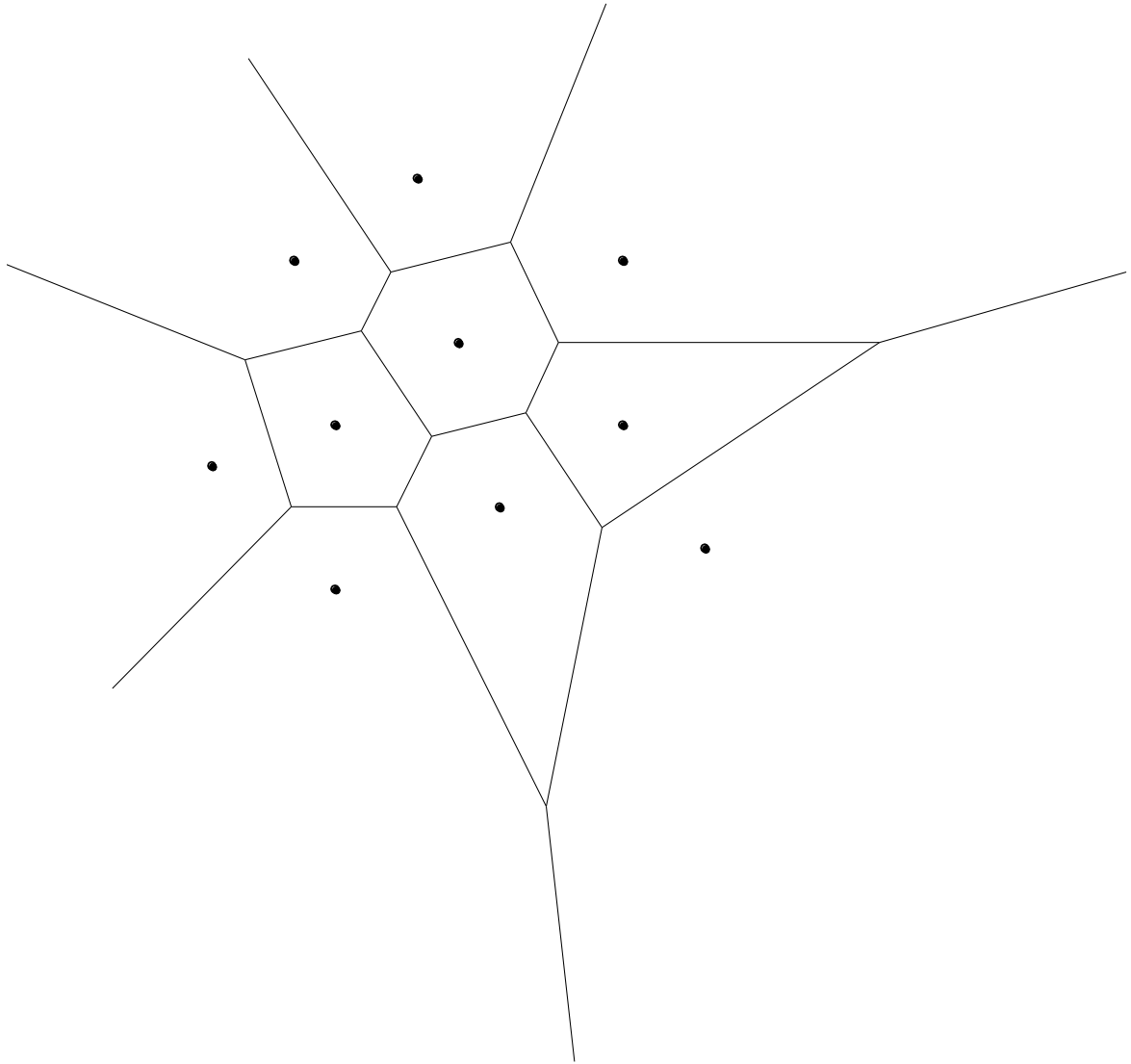
垂直2等分線

ボロノイ図とデローネイ三角形分割

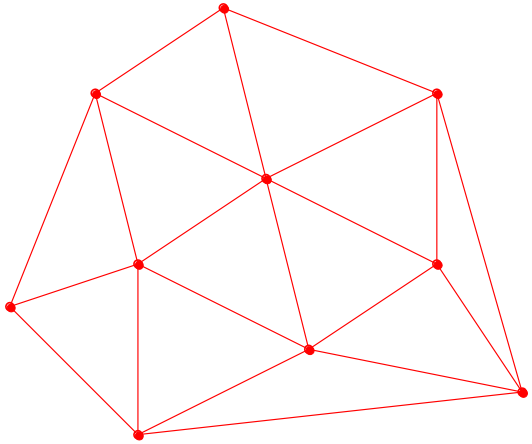


3本の垂直2等分線は1点で交わる

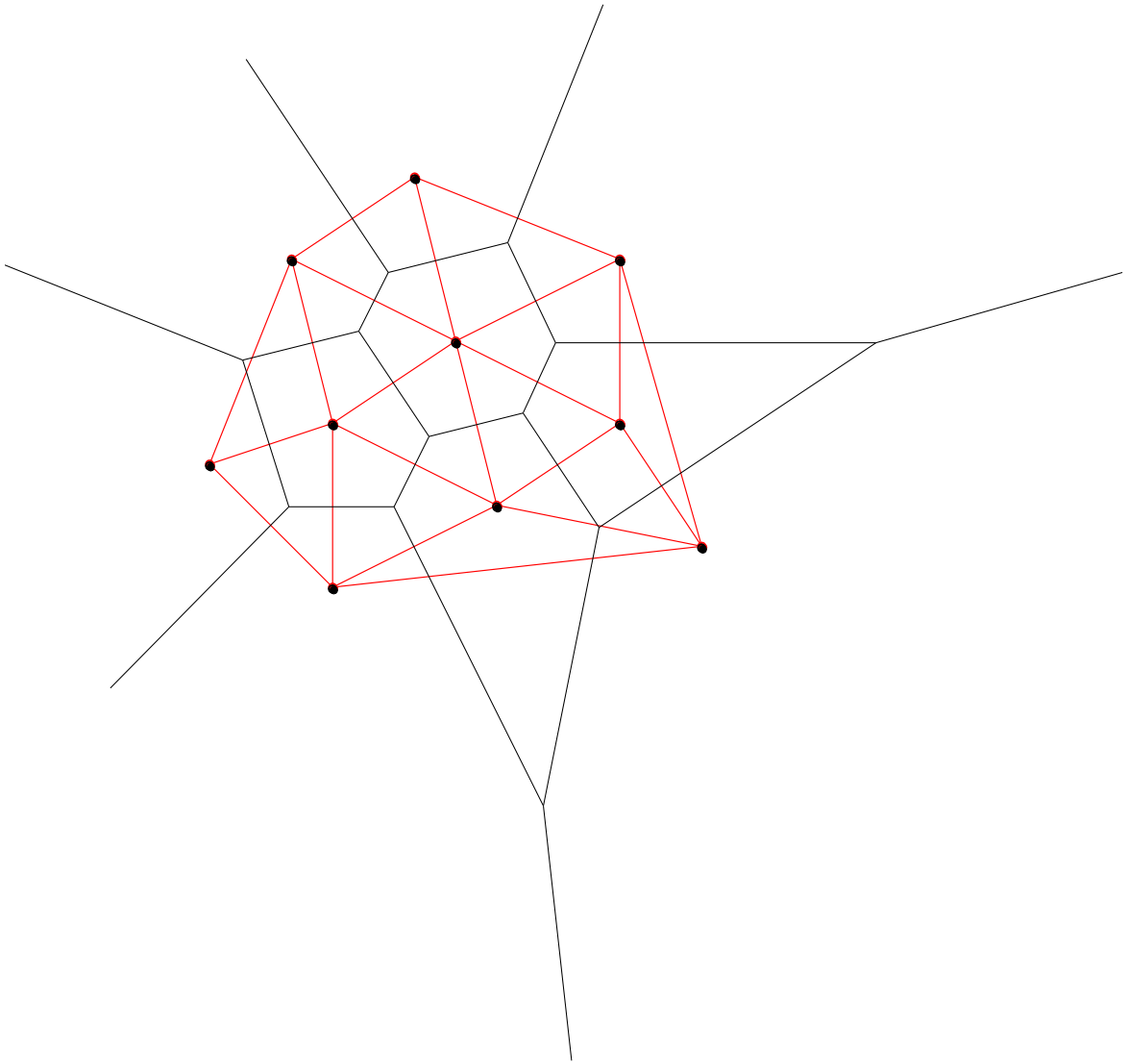
ボロノイ図



デローネイ三角形分割



ボロノイ図とデローネイ三角形分割



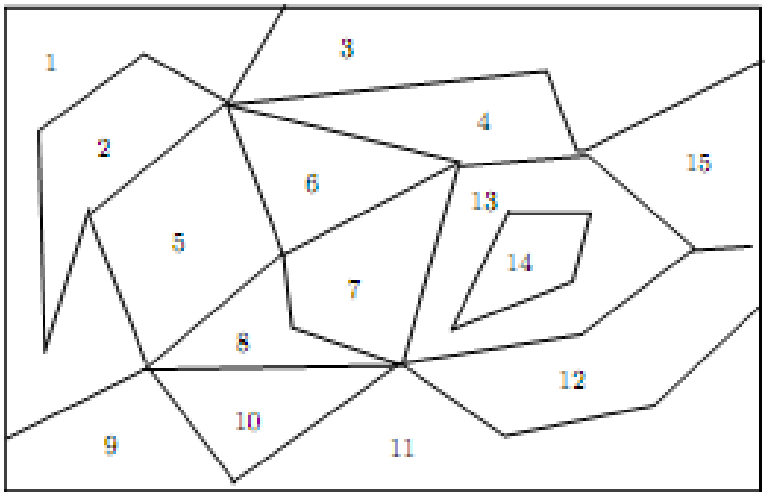
なぜデローネイ三角形分割を計算するのか？
計算されたデローネイ三角形分割をどう利用するか？

通常のシナリオ

平面上に点集合Pが与えられたとき，
Pのデローネイ三角形分割を計算し，
得られた三角形に関する幾つかの操作が効率よく行える
ようにデータ構造を構築する.

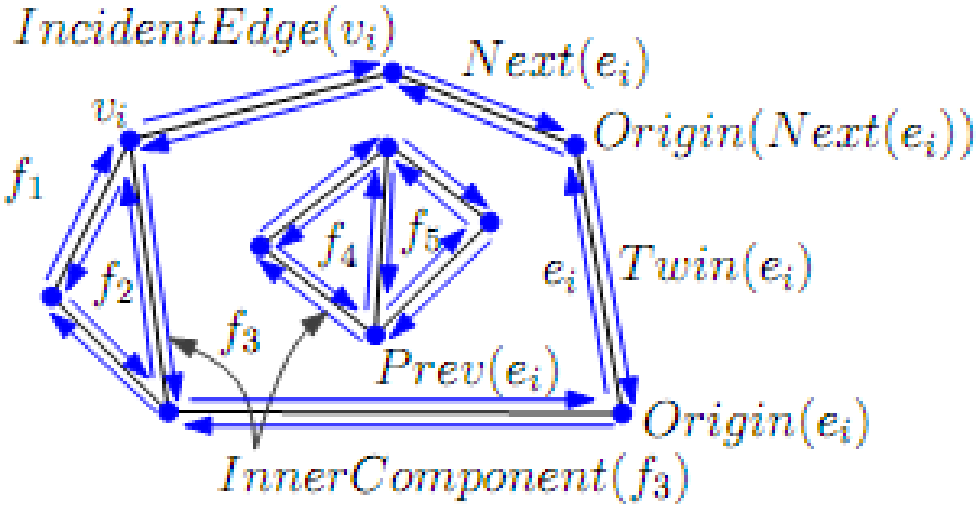
デローネイ三角形分割 = 平面グラフ（平面上に描かれたグラフ）
データ構造: 2重連結辺リスト (DCEL)
　DCEL内の各操作は定数時間で実行できる.

2重連結辺リスト



平面分割

可能な操作
1. 全ての頂点を列挙
2. 全ての辺を列挙
3. 面の境界を辿る
4. 各辺について，それに関連する情報を得る.



- IncidentEdge(v)
- Origin(e)
- Next(e)
- Prev(e)

・IncidentEdge(v): 頂点vに接続する任意の辺
・Origin(e) : 辺eの始点
・Next(e) : eが指す面の境界上でeの次の辺
・Prev(e) : eの前の辺

デローネイ三角形分割を求めて，それを2重連結辺リストの形で表現しておけば，上記の操作はO(1)時間で実行可能.

定数作業領域アルゴリズムの場合には，何も計算しないし，
データ構造も構成しない（メモリがないので）.

➔ **ゼロ・スペースのデータ構造**

何か操作が必要になれば，必要な計算を実行する．そのためにO(n)時間かかるが，メモリは節約できている.

デローネイ三角形分割に対するゼロ･スペースのデータ構造

入力データファイルを一度だけ走査することで実行できる操作:

(1) FindIncidentEdge(p):
　　点pに接続するデローネイ辺を一つ返す.

(2) **FindNextIncidentEdge(p, e)**
　　点pに接続するデローネイ辺の中でeの次の辺を求める.

(3) **FindNextEdgeOnFace(f, e)**
　　面fの境界上でeの次のデローネイ辺を求める.
　　(ここで面はデローネイ三角形を指す).

(4) **FindTwinEdge(e)**
　　eの双子辺(逆向きの辺)を返す.

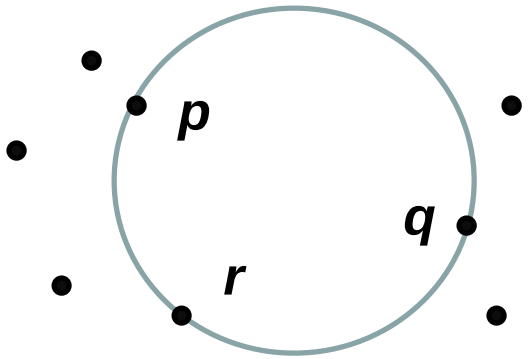
デローネイ三角形分割

デローネイ三角形分割は，すべての可能な三角形分割の中で，最小の内角が最大であるという意味で最適な三角形分割である．

観察1:

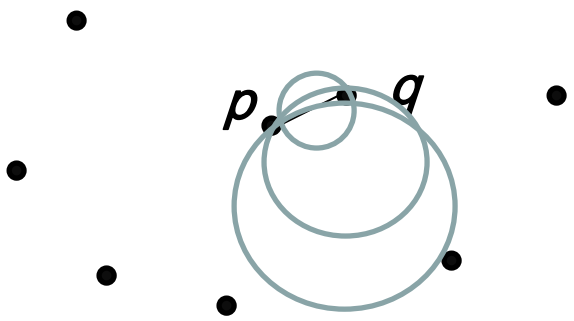
S: 平面上に与えられた点集合．

Sの２点**p**と**q**を結ぶ線分がデローネイ辺であるための必要十分条件は，**S**に点**r**が存在して，**p, q, r**で定まる円の内部に**S**の他の点が含まれないことである．



観察2:

S:平面上に与えられた点集合.
Sの点pがSの別の点qに最も近い点なら
(p, q) はデロネイ辺である .



事実: Sの 2 点p, qを通る空円が存在するなら , (p, q)はデロネイ辺である .

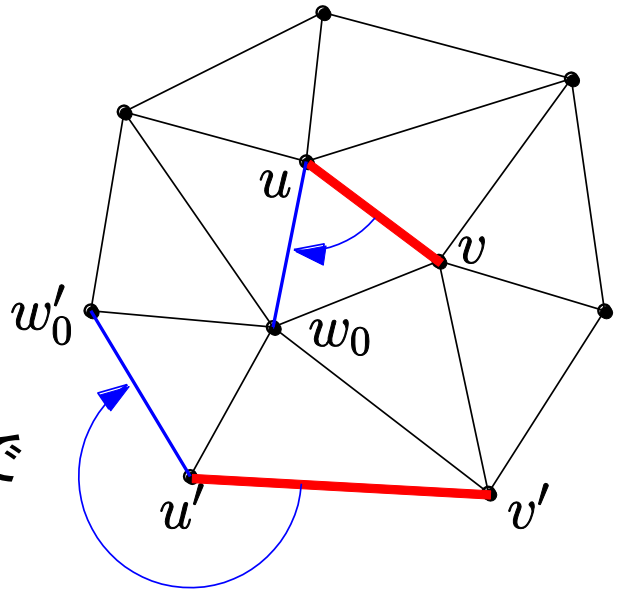
観察3:

Sの凸包上の辺はすべてデロネイ辺である .

デローネイ三角形分割を求めるアルゴリズム

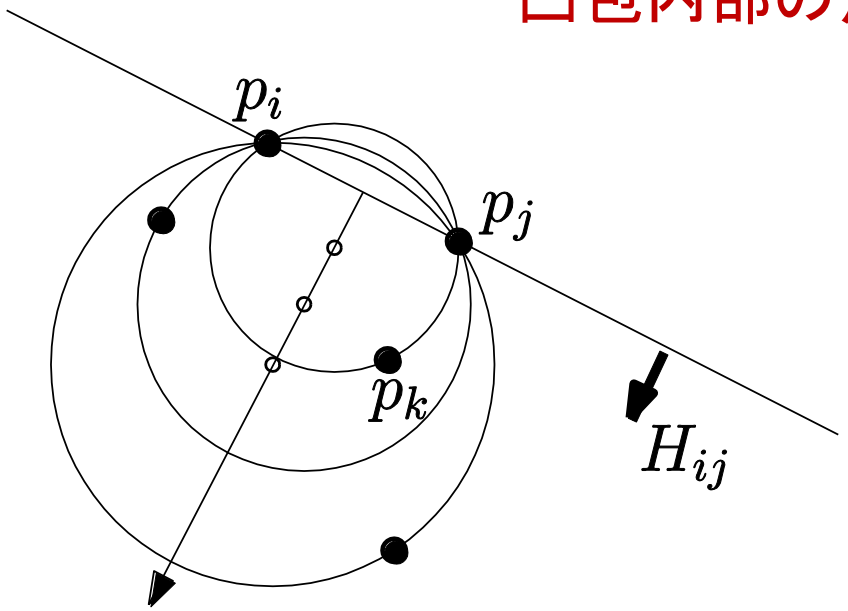
for **S**の各点 ***p***
　Sの中で点**p**に最も近い点**q**を求める.
　(u, v) = (p, q)
　repeat
　　デローネイ辺**(u, v)**を報告.
　　(u, v) = ClockwiseNextDelaunayEdge(***u, v***).
　until(***(u, v) = (p, q))***

凸包内部の辺と凸包上の
辺に対する時計回りの順で
次の辺



ClockwiseNextDelaunayEdge(p_i, p_j)

凸包内部の点に対して



次のような点 p_k を求めよ

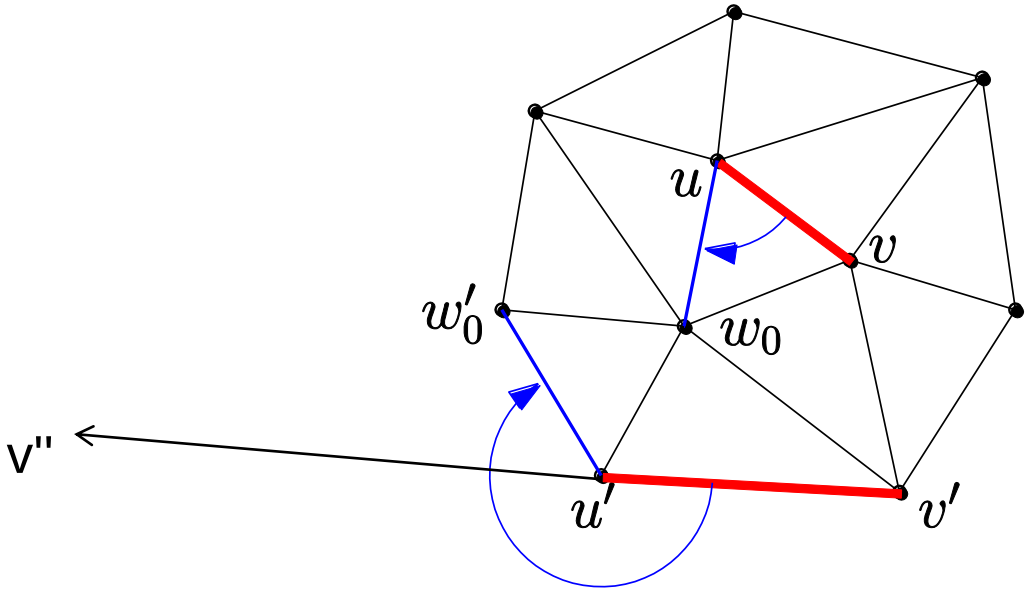
　(p_i, p_j, p_k) は時計回りの順, かつ

　これら3点を通る円が最小（半径の意味で）である.

……計算時間はO(n).

ClockwiseNextDelaunayEdge(p_i, p_j)

凸包上の辺について



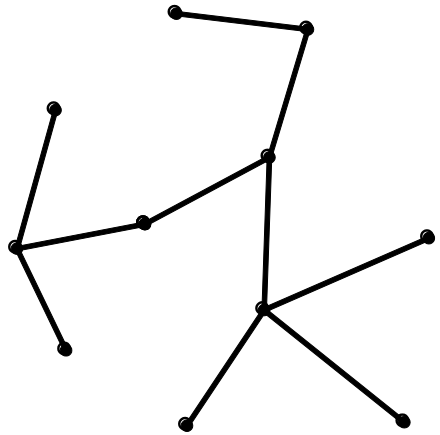
u'v'': 辺u'v'を逆方向に延長したもの
角wu'v''を最小にする点**w**を求める.
… 計算時間は O(n)

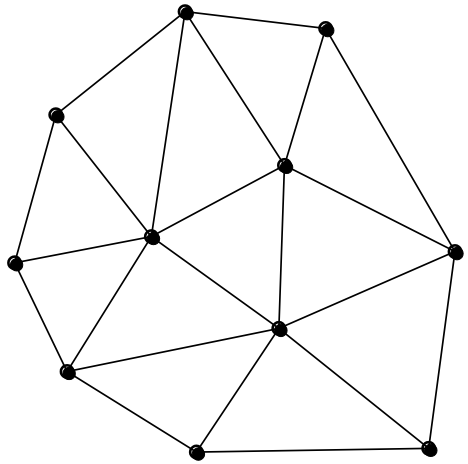
定理 1:
平面上にn個の点からなる集合が与えられたとき，定数の作業領域だけを用いて各デローネイ辺をちょうど1度ずつ$O(n^2)$の時間で報告するアルゴリズムが存在する.

証明:
デローネイ三角形分割はn頂点の平面グラフである.
したがって，デローネイ辺の本数は$O(n)$で抑えられる.
各デローネイ辺はO(n)時間で計算できるので，定理を得る.

最小全域木

定義:
点集合Sに対する最小全域木とは，すべての点を連結する木の中で辺長の総和が最小のものである.





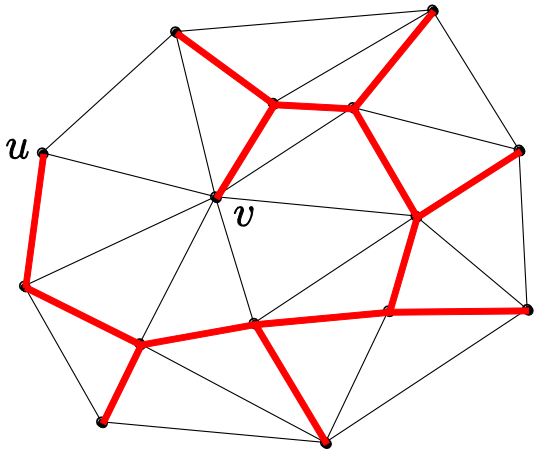
性質:
(ユークリッド) 最小全域木は，点集合に対するデローネイ三角形分割の部分グラフである.

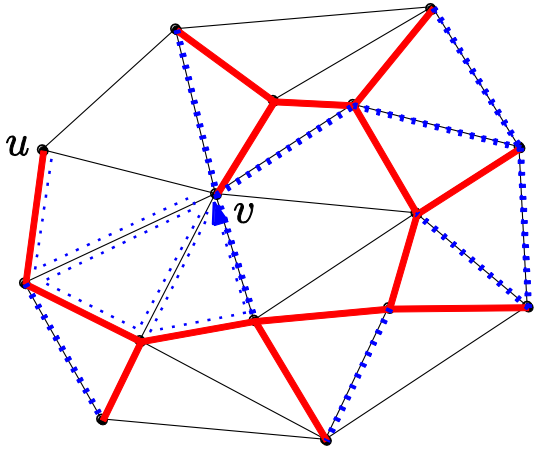
性質:

u, v: ***S***の２点

(u, v) がMST(***S***)の辺でない

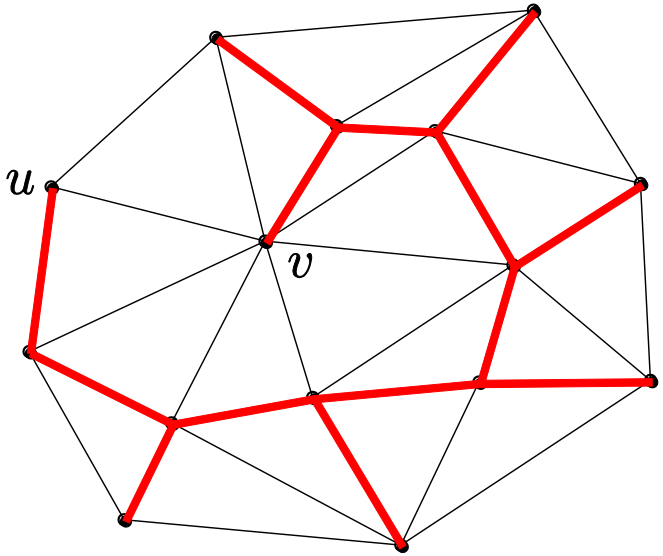
(u, v)よりも短いデローネイ辺だけからなる
デローネイ三角形分割の部分グラフで
これら２点が連結である.



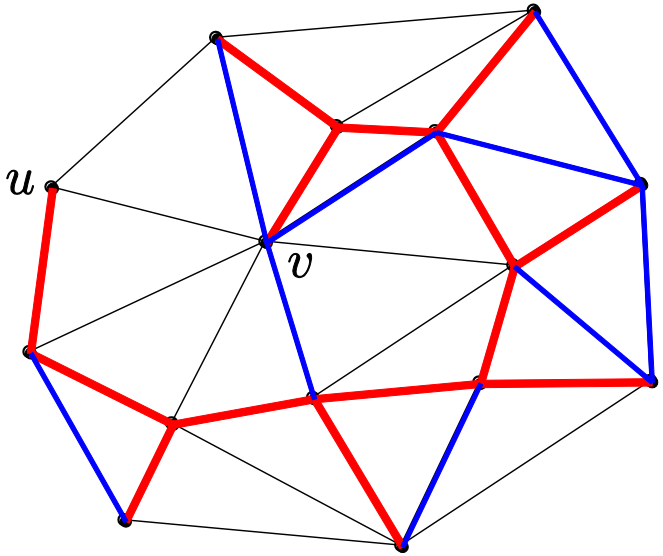


MSTを求めるアルゴリズム

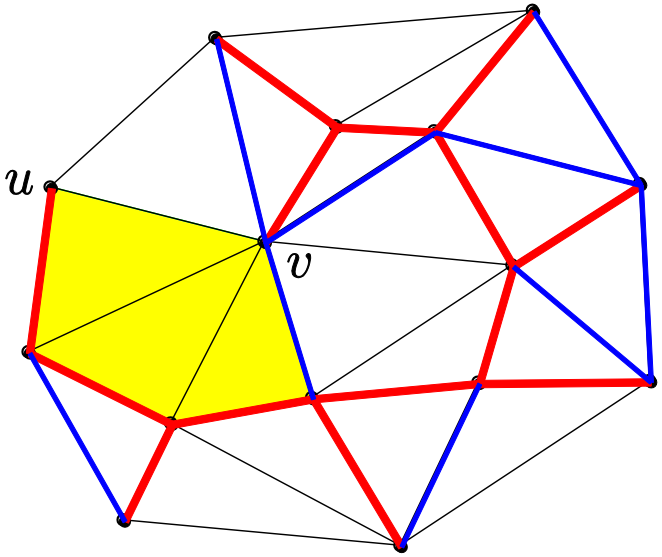
デローネイ辺を見つけるたびに，その辺より短いデローネイ辺だけからなる部分グラフにおいて，両端点が連結であるかどうかを判定



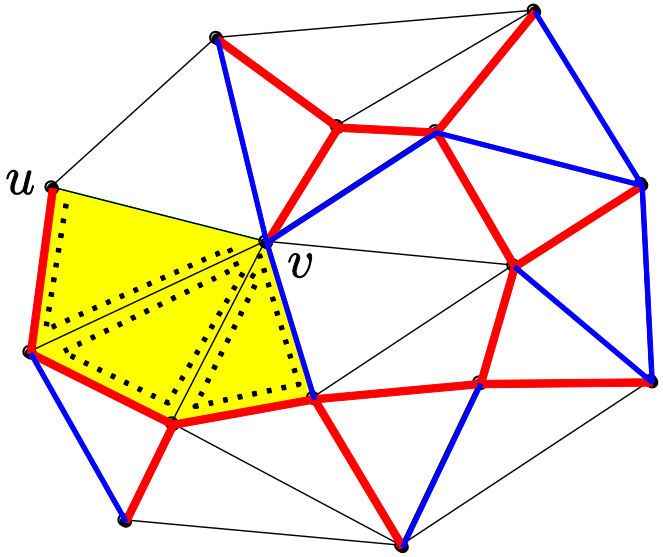
MST



DT(u, v)



DT(u, v) + (u,v)

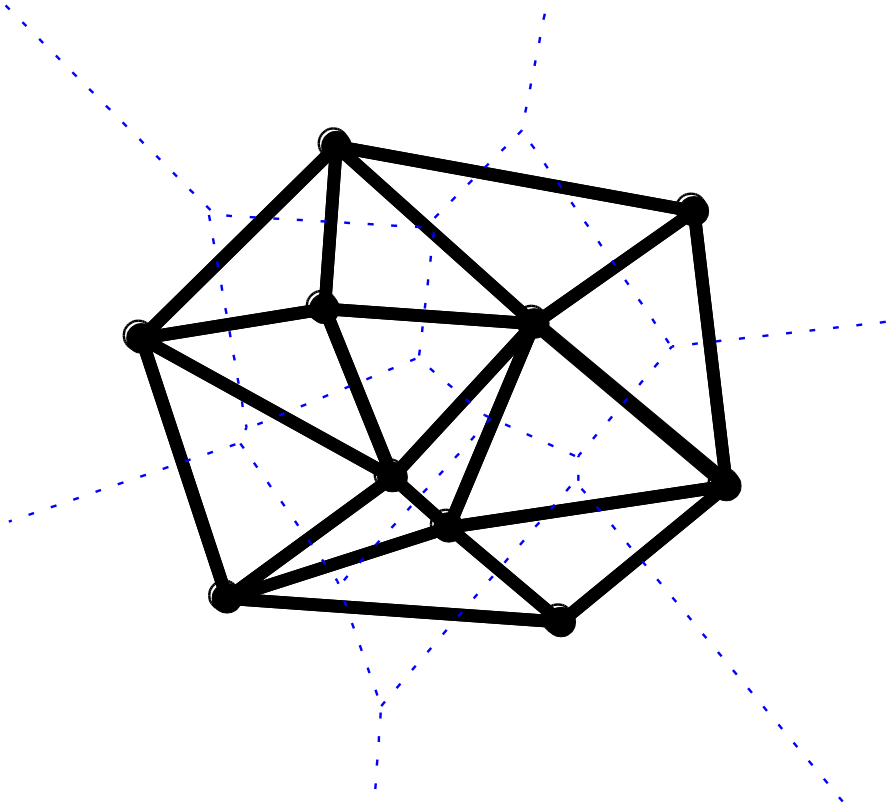


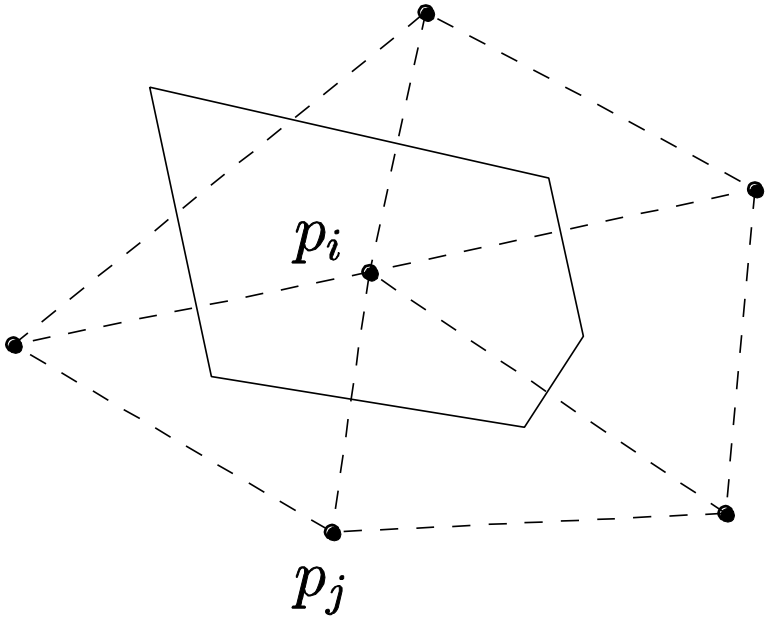
経路の探索

定理 3:
平面上に与えられたn点からなる点集合に対する最小全域木を定数作業領域だけで構成する$O(n^3)$ 時間のアルゴリズムが存在する.

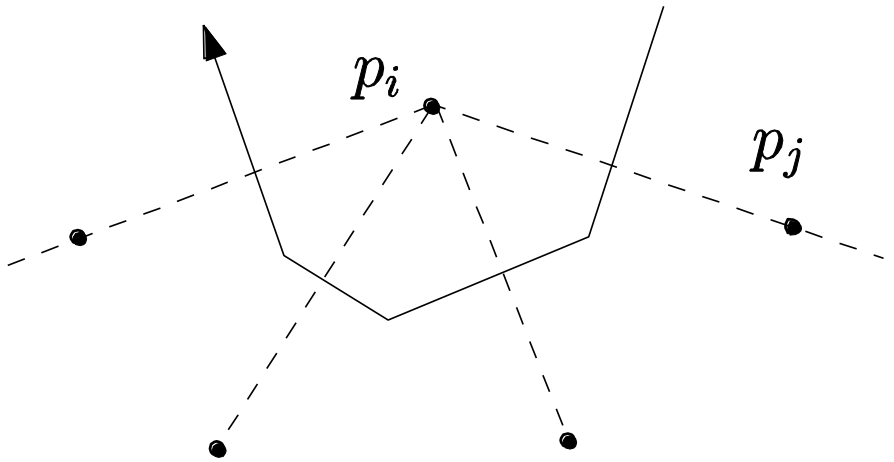
ボロノイ図を求める定数作業領域アルゴリズム

ボロノイ図はデローネイ三角形分割の双対である．





凸包内部の点



凸包上の点

点p_iに関するボロノイ領域を求めるには，点p_iに接続するデローネイ辺を時計回りの順に求めて，その垂直2等分線を次々と求めて行けばよい.

定理 2:
平面上のn点からなる点集合に対するボロノイ図を定数作業領域だけで構成する$O(n^2)$ 時間のアルゴリズムが存在する.