Geometria Computacional

Aula 8

Doubly-connected edge list - DCEL

Sec 2.2 do livro de de Berg e outros

Representação de mapas retilineares

Como acoplar o algoritmo de triangulação de polígonos monótonos ao algoritmo de Lee e Preparata?

Representação de mapas retilineares

Como acoplar o algoritmo de triangulação de polígonos monótonos ao algoritmo de Lee e Preparata?

Doubly-connected edge list – DCEL

Cada aresta aparece duas vezes nesta ED: uma vez em cada direção, uma cópia apontando para a outra (*twins* ou *half-edges*).

Representação de mapas retilineares

Como acoplar o algoritmo de triangulação de polígonos monótonos ao algoritmo de Lee e Preparata?

Doubly-connected edge list – DCEL

Cada aresta aparece duas vezes nesta ED: uma vez em cada direção, uma cópia apontando para a outra (*twins* ou *half-edges*).

Cada meia-aresta está em uma face do mapa, e aponta para a próxima meia-aresta e para a anterior na face.

Representação de mapas retilineares

Como acoplar o algoritmo de triangulação de polígonos monótonos ao algoritmo de Lee e Preparata?

Doubly-connected edge list – DCEL

Cada aresta aparece duas vezes nesta ED: uma vez em cada direção, uma cópia apontando para a outra (*twins* ou *half-edges*).

Cada meia-aresta está em uma face do mapa, e aponta para a próxima meia-aresta e para a anterior na face.

Pode se manter registro para cada face ou vértice do mapa.

Cada face/vértice tem apontador para uma das (meia-)arestas incidentes a ele.

Doubly-connected edge list – DCEL

Cada aresta $e = uv$ tem duas entradas na ED: (u, v) e (v, u).

Doubly-connected edge list – DCEL

Cada aresta $e = uv$ tem duas entradas na ED: (u, v) e (v, u).

Para cada meia-aresta (u, v):

$\texttt{twin}(u, v) = (v, u)$

$\texttt{prox}(u, v)$: próxima meia-aresta na face

$\texttt{prev}(u, v)$: meia-aresta anterior na face

Doubly-connected edge list – DCEL

Cada aresta $e = uv$ tem duas entradas na ED: (u, v) e (v, u).

Para cada meia-aresta (u, v):

$\texttt{twin}(u, v) = (v, u)$

$\texttt{prox}(u, v)$: próxima meia-aresta na face

$\texttt{prev}(u, v)$: meia-aresta anterior na face

Para cada face f, apontador para uma meia-aresta da face.

Doubly-connected edge list – DCEL

Cada aresta $e = uv$ tem duas entradas na ED: (u, v) e (v, u).

Para cada meia-aresta (u, v):
- $\texttt{twin}(u, v) = (v, u)$
- $\texttt{prox}(u, v)$: próxima meia-aresta na face
- $\texttt{prev}(u, v)$: meia-aresta anterior na face

Para cada face f, apontador para uma meia-aresta da face.

Revisite a primeira fase do algoritmo de Lee e Preparata (partição em monótonos) e a adapte para devolver essa representação da partição obtida em vez do conjunto de diagonais apenas.

Doubly-connected edge list – DCEL

Cada aresta $e = uv$ tem duas entradas na ED: (u, v) e (v, u).

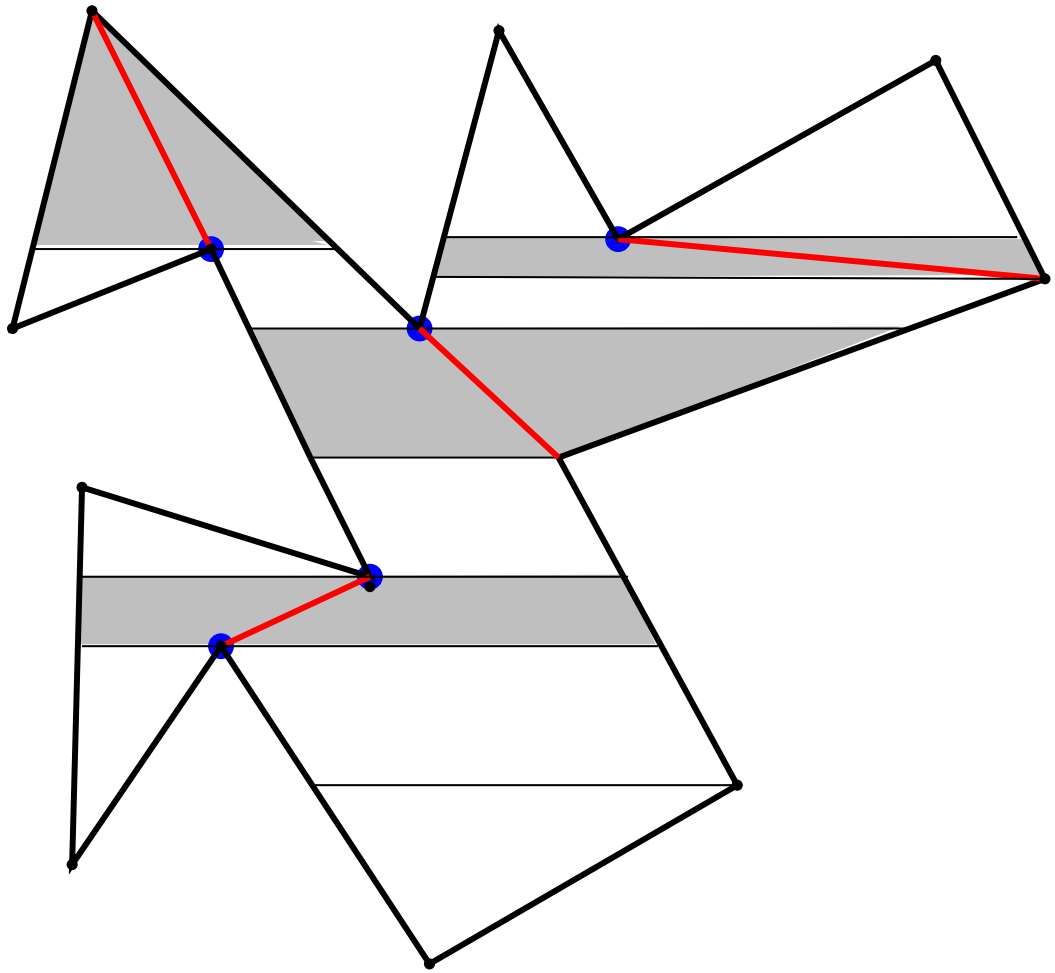
Para cada meia-aresta (u, v):
- $\texttt{twin}(u, v) = (v, u)$
- $\texttt{prox}(u, v)$: próxima meia-aresta na face
- $\texttt{prev}(u, v)$: meia-aresta anterior na face

Para cada face f, apontador para uma meia-aresta da face.

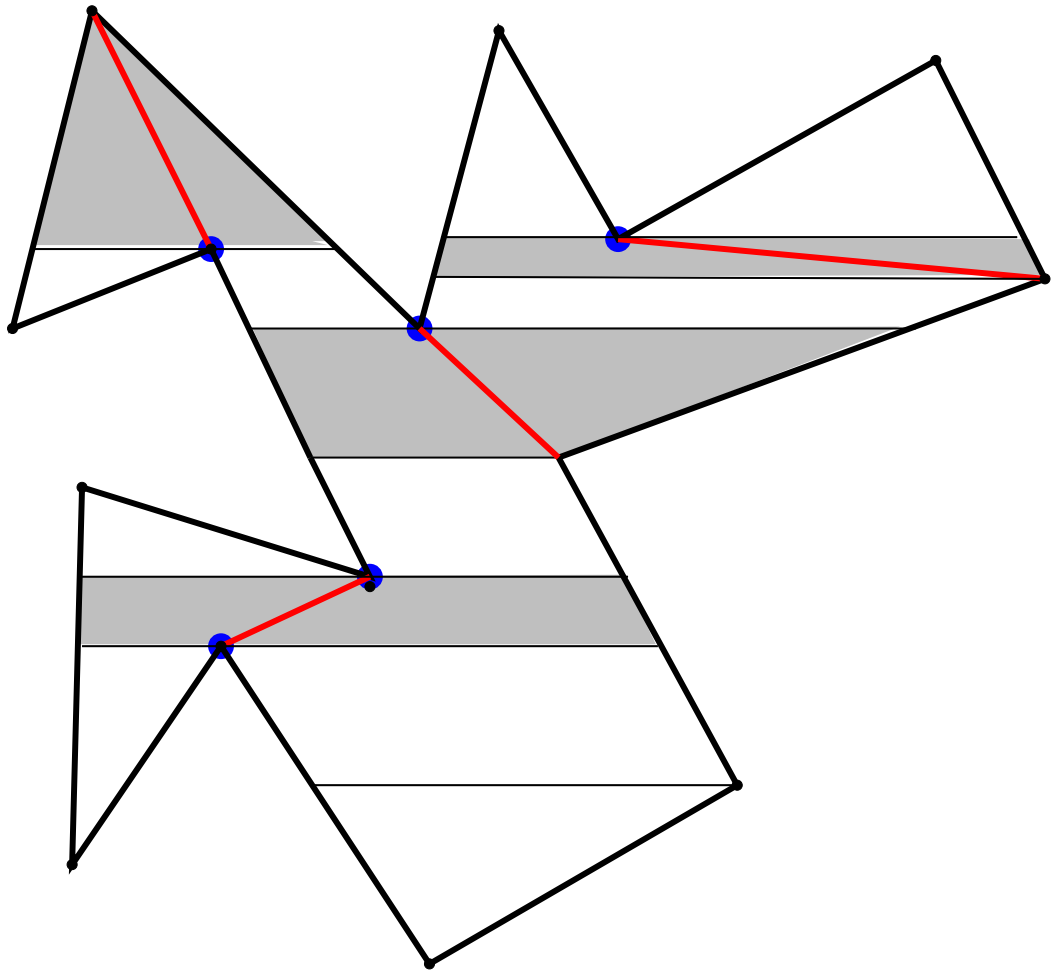
Revisite a primeira fase do algoritmo de Lee e Preparata (partição em monótonos) e a adapte para devolver essa representação da partição obtida em vez do conjunto de diagonais apenas.

Omita a face externa da representação, já que ela não é usada na segunda fase.

Partição em polígonos monótonos



Partição em polígonos monótonos



Qual é o grau máximo de um vértice no mapa retilinear final?

Algoritmo de Lee e Preparata

DivideEmMonótono-LP(X, Y, n)

1 para $k \leftarrow 1$ até n faça
2 $\quad E[k] \leftarrow k$
3 MergeSort$(Y, X, 1, n, E)$ $\quad \triangleright$ ordenação indireta decrescente de Y
4 $T \leftarrow$ Crie$ABBB$ $\quad t \leftarrow 0$
5 para $k \leftarrow 1$ até n faça
6 $\quad v \leftarrow E[k]$
7 $\quad v^- \leftarrow v - 1 \quad v^+ \leftarrow v + 1$
8 $\quad$ se $v^- = 0$ então $v^- \leftarrow n$
9 $\quad$ se $v^+ = n + 1$ então $v^+ \leftarrow 1$
10 $\quad$ se $Y[v^-] < Y[v] < Y[v^+]$ ou $Y[v^+] < Y[v] < Y[v^-]$
11 $\quad\quad$ então TrataCaso1$(T, v^-, v, v^+, Y, n, D, t)$
12 $\quad\quad$ senão se $Y[v^-] < Y[v]$
13 $\quad\quad\quad\quad$ então TrataCaso2(T, v^-, v, v^+, D, t)
14 $\quad\quad\quad\quad$ senão TrataCaso3(T, v, Y, n, D, t)
15 devolva (D, t)

Algoritmo de Lee e Preparata

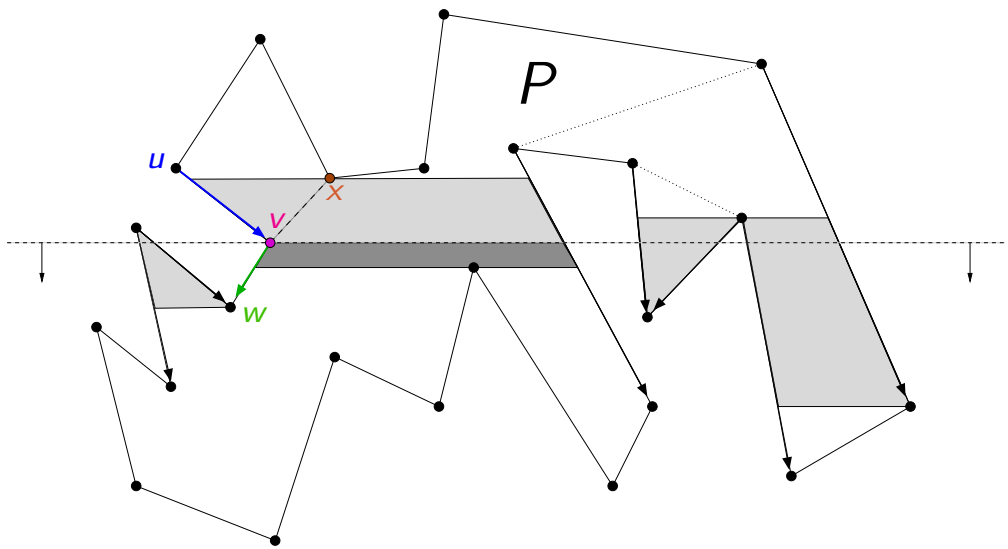
DivideEmMonótono-LP(X, Y, n)

1 para $k \leftarrow 1$ até n faça
2 $\quad E[k] \leftarrow k$
3 MergeSort$(Y, X, 1, n, E)$ $\quad \triangleright$ ordenação indireta decrescente de Y
4 $T \leftarrow$ Crie$ABBB$ $\quad D \leftarrow$ Crie$DCEL(P)$
5 para $k \leftarrow 1$ até n faça
6 $\quad v \leftarrow E[k]$
7 $\quad v^- \leftarrow v - 1 \quad v^+ \leftarrow v + 1$
8 $\quad$ se $v^- = 0$ então $v^- \leftarrow n$
9 $\quad$ se $v^+ = n + 1$ então $v^+ \leftarrow 1$
10 $\quad$ se $Y[v^-] < Y[v] < Y[v^+]$ ou $Y[v^+] < Y[v] < Y[v^-]$
11 $\quad\quad$ então TrataCaso1$(T, v^-, v, v^+, Y, n, D)$
12 $\quad\quad$ senão se $Y[v^-] < Y[v]$
13 $\quad\quad\quad\quad$ então TrataCaso2(T, v^-, v, v^+, D)
14 $\quad\quad\quad\quad$ senão TrataCaso3(T, v, Y, n, D)
15 devolva D

Caso 1

TrataCaso1(T, u, v, w, Y, n, D, t)

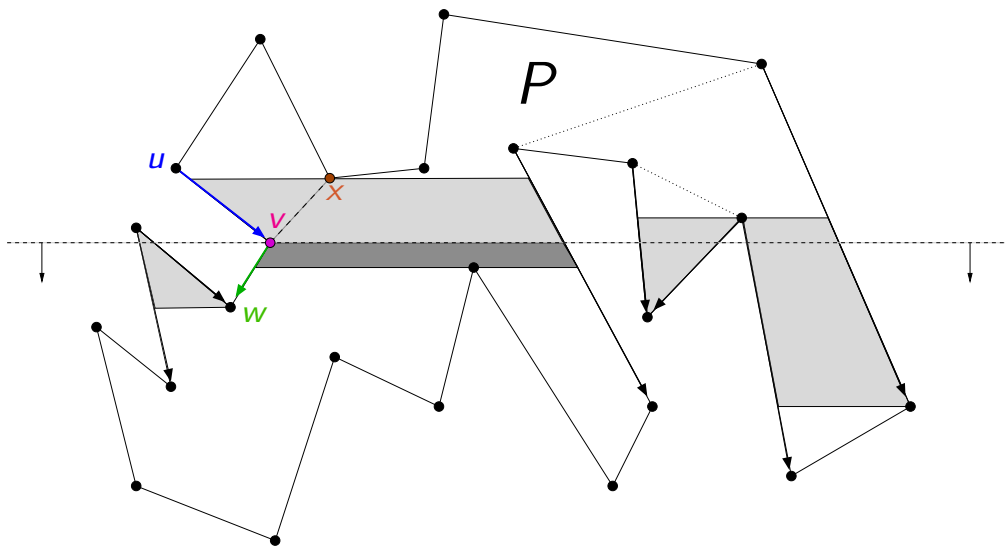
1 se $Y[u] < Y[w]$ então $u \leftrightarrow w$
2 $((i,j), x, (k,l)) \leftarrow$ Remove(T, v)
3 se $v = j$ ▷ o trapézio está à direita de v?
4 então Insere($T, (v,w), v, (k,l)$)
5 senão Insere($T, (i,j), v, (v,w)$)
6 se PontaParaBaixo(x, Y, n)
7 então $t \leftarrow t+1$ $D[t] \leftarrow (x, v)$



Caso 1

TrataCaso1(T, u, v, w, Y, n, D)

1 se $Y[u] < Y[w]$ então $u \leftrightarrow w$
2 $((i,j), x, (k,l)) \leftarrow$ Remove(T, v)
3 se $v = j$ ▷ o trapézio está à direita de v?
4 então Insere($T, (v,w), v, (k,l)$)
5 senão Insere($T, (i,j), v, (v,w)$)
6 se PontaParaBaixo(x, Y, n)
7 então ~~$t \leftarrow t+1$ $D[t] \leftarrow (x,v)$~~ ▷ adiciona xv à DCEL D



Consumo de tempo

Para criar a DCEL de P:

tempo linear no número de vértices de P;

cada vértice de P tem um apontador
para uma meia-aresta da DCEL incidente nele.

Para adicionar uma aresta xy à DCEL:

percorra a lista de arestas incidentes a x, em
sentindo horário, até as duas entre as quais y está;

faça o mesmo para y em relação a x;

crie duas novas meias arestas e as insira
nestes pontos das listas em torno de x e y;

consome tempo $\mathrm{O}(1)$, pois há no máximo
quatro arestas incidentes a cada vértice.

Geometria Computacional

Diagrama de Voronoi

Sec 7.1 do livro de de Berg e outros

Diagrama de Voronoi

Dados endereços de agências de correio, determinar qual é a região da cidade que fica mais próxima de cada agência.

Diagrama de Voronoi

Dados endereços de agências de correio, determinar qual é a região da cidade que fica mais próxima de cada agência.

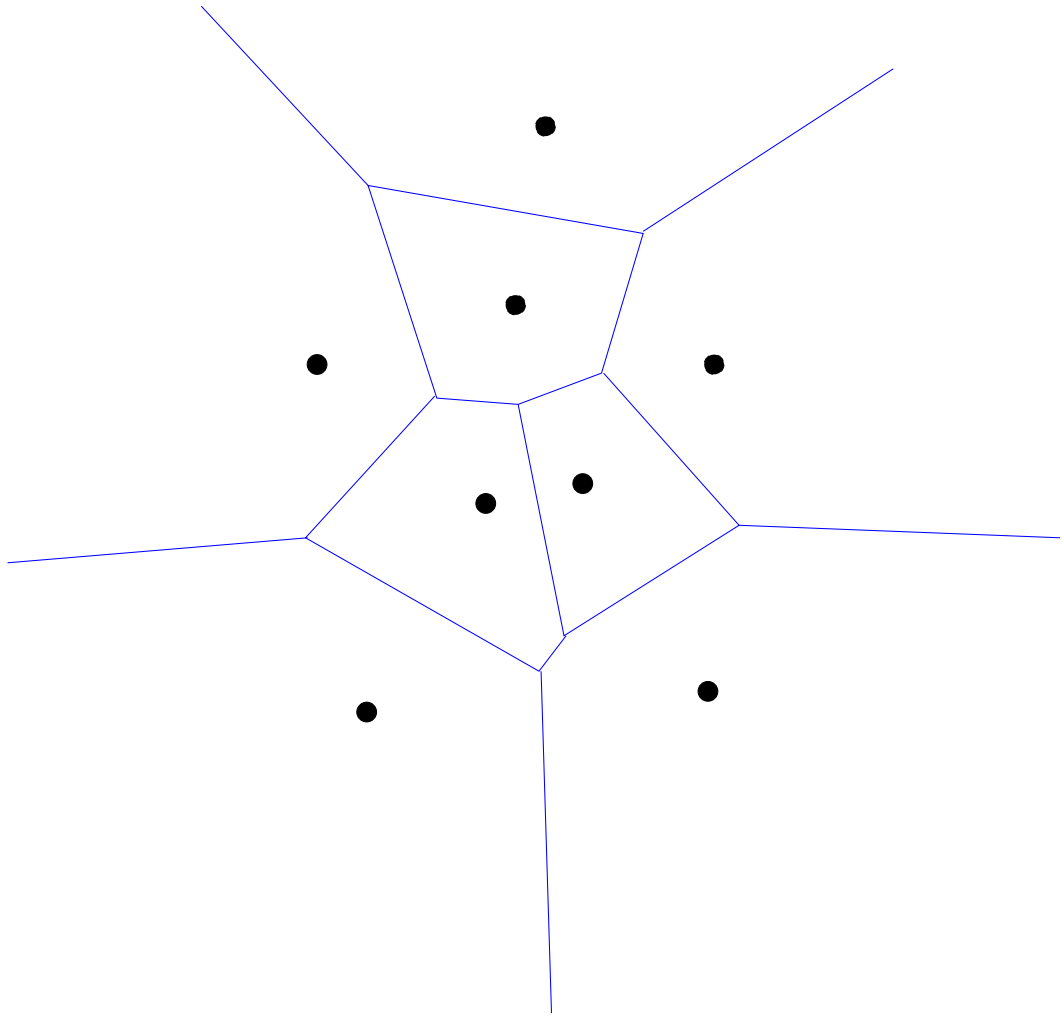
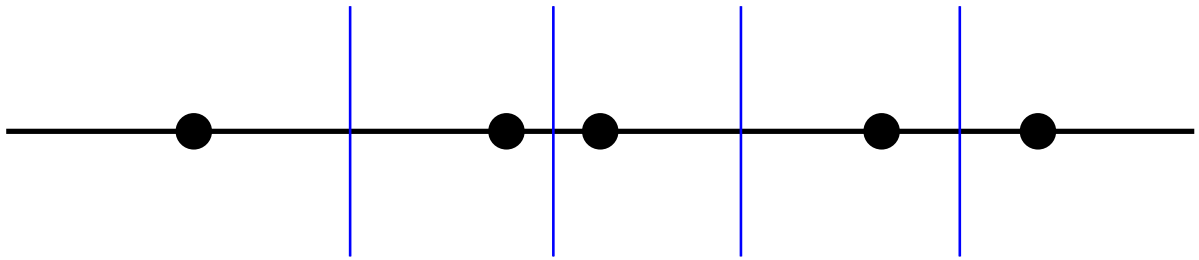


Diagrama de Voronoi

Dados endereços de agências de correio, determinar qual é a região da cidade que fica mais próxima de cada agência.

Versão unidimensional:

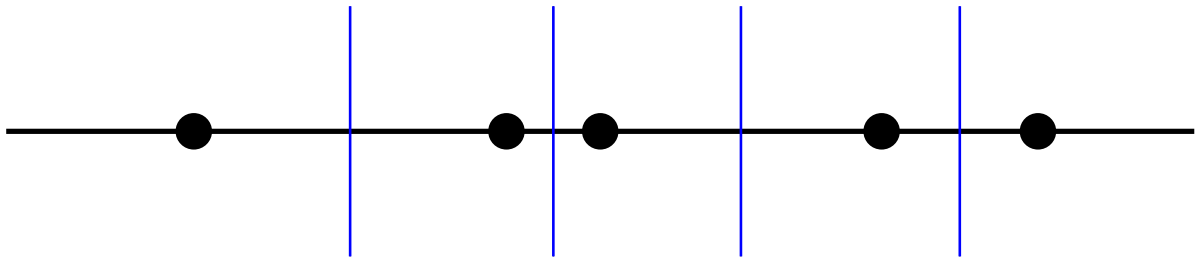


O diagrama são várias linhas paralelas.

Diagrama de Voronoi

Dados endereços de agências de correio, determinar qual é a região da cidade que fica mais próxima de cada agência.

Versão unidimensional:



O diagrama são várias linhas paralelas.

Versão bidimensional:

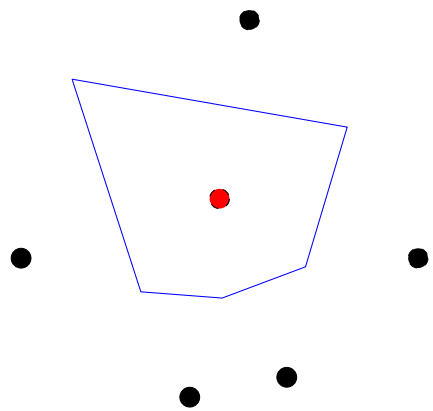
Pode ser construída em tempo $\mathrm{O}(n \lg n)$,
onde n é o número de pontos dados.

- Divisão e conquista: Shamos e Hoye (complexo)
- Linha de varredura: Fortune (elegante e simples)

Notação

$P := \{p_1, \ldots, p_n\}$

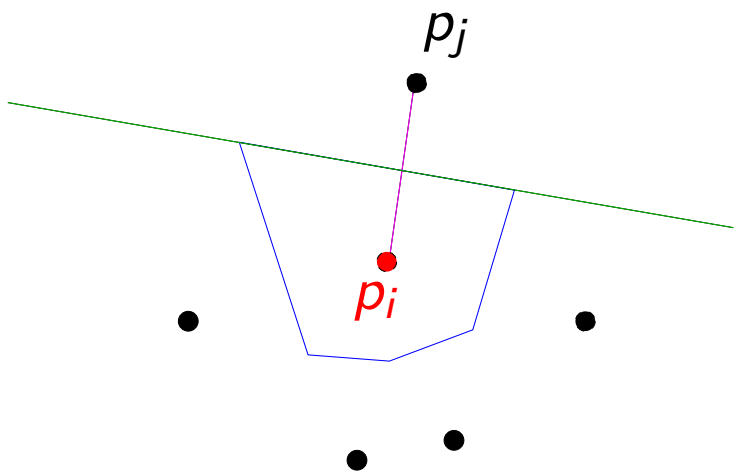
$\mathcal{V}(p_i) := \{q : \text{Dist}(q, p_i) < \text{Dist}(q, p_j) \text{ para todo } j \neq i\}$



Notação

$P := \{p_1, \ldots, p_n\}$

$\mathcal{V}(p_i) := \{q : \text{Dist}(q, p_i) < \text{Dist}(q, p_j) \text{ para todo } j \neq i\}$

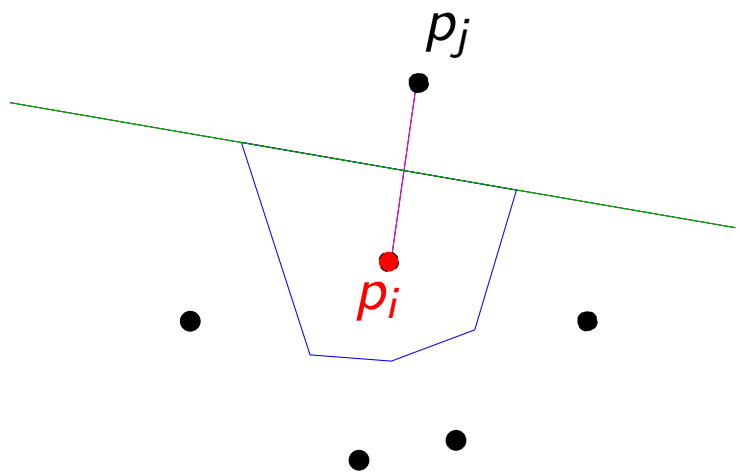


$h(p, q)$: semiplano determinado pela reta bissetora entre p e q que contém o ponto p.

$\mathcal{V}(p_i) = \cap_{j \neq i} h(p_i, p_j)$

Notação

$P := \{p_1, \ldots, p_n\}$
$\mathcal{V}(p_i) := \{q : \text{Dist}(q, p_i) < \text{Dist}(q, p_j) \text{ para todo } j \neq i\}$



$h(p, q)$: semiplano determinado pela reta bissetora entre p e q que contém o ponto p.

$\mathcal{V}(p_i) = \cap_{j \neq i} h(p_i, p_j)$

Logo $\mathcal{V}(p_i)$ é convexo (interseção de $n - 1$ semiplanos), com no máximo $n - 1$ arestas e vértices.

Diagrama de Voronoi

$P := \{p_1, \ldots, p_n\}$

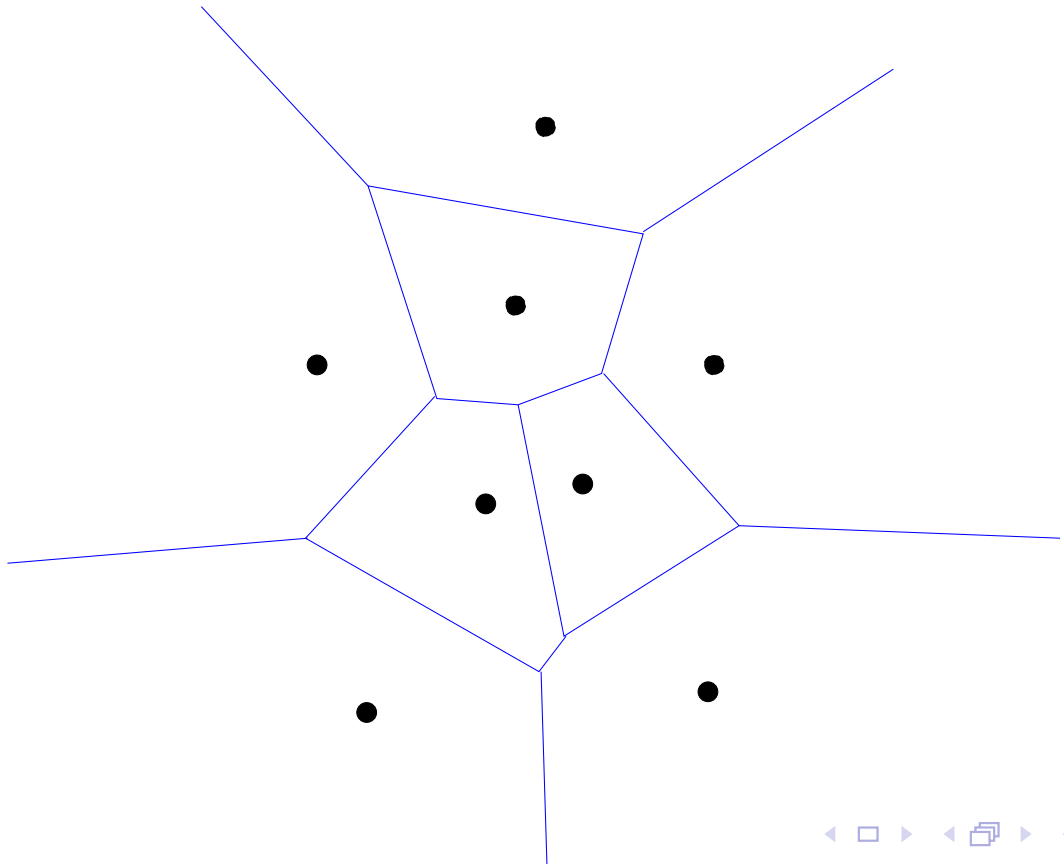
Célula de p_i:
$\mathcal{V}(p_i) := \{q : \text{Dist}(q, p_i) < \text{Dist}(q, p_j) \text{ para todo } j \neq i\}$

Diagrama de Voronoi

$P := \{p_1, \ldots, p_n\}$

Célula de p_i:
$\mathcal{V}(p_i) := \{q : \text{Dist}(q, p_i) < \text{Dist}(q, p_j) \text{ para todo } j \neq i\}$

Diagrama de Voronoi de P: $\text{Vor}(P)$
subdivisão do plano nas células $\mathcal{V}(p_1), \ldots, \mathcal{V}(p_n)$.



Complexidade do Diagrama de Voronoi

O diagrama tem vértices e arestas.

Qual pode ser o seu tamanho máximo em função de n?

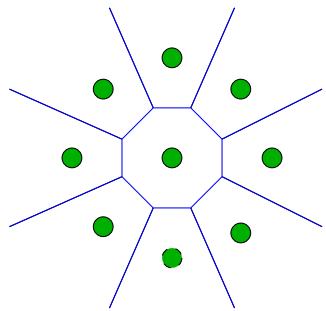
Complexidade do Diagrama de Voronoi

O diagrama tem vértices e arestas.

Qual pode ser o seu tamanho máximo em função de n?

Cada célula tem $\mathrm{O}(n)$ arestas.

Algumas podem ter $\Theta(n)$ arestas.



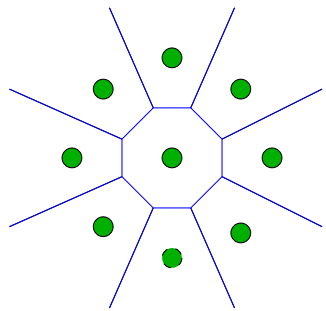
Complexidade do Diagrama de Voronoi

O diagrama tem vértices e arestas.

Qual pode ser o seu tamanho máximo em função de n?

Cada célula tem $\mathrm{O}(n)$ arestas.

Algumas podem ter $\Theta(n)$ arestas.



Teorema:
Para $n \geq 3$, o número de vértices no diagrama de Voronoi de um conjunto de n pontos no plano é no máximo $2n - 5$, e o número de arestas é no máximo $3n - 6$.

(Prova feita na aula.)

Arestas e vértices de $\mathrm{Vor}(P)$

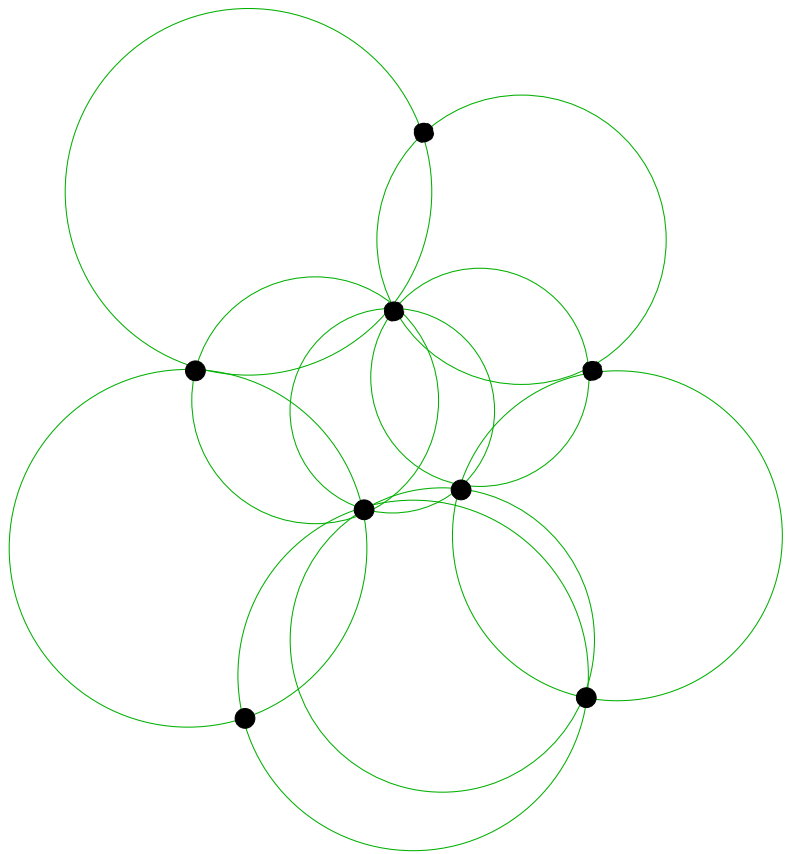
$C_P(q)$: maior círculo centrado em q que
não contenha pontos de P no seu interior.

Arestas e vértices de $\mathrm{Vor}(P)$

$C_P(q)$: maior círculo centrado em q que não contenha pontos de P no seu interior.

Teorema:

(i) Ponto q é vértice de $\mathrm{Vor}(P)$ sse $C_P(q)$ contém três ou mais pontos de P (em sua fronteira).

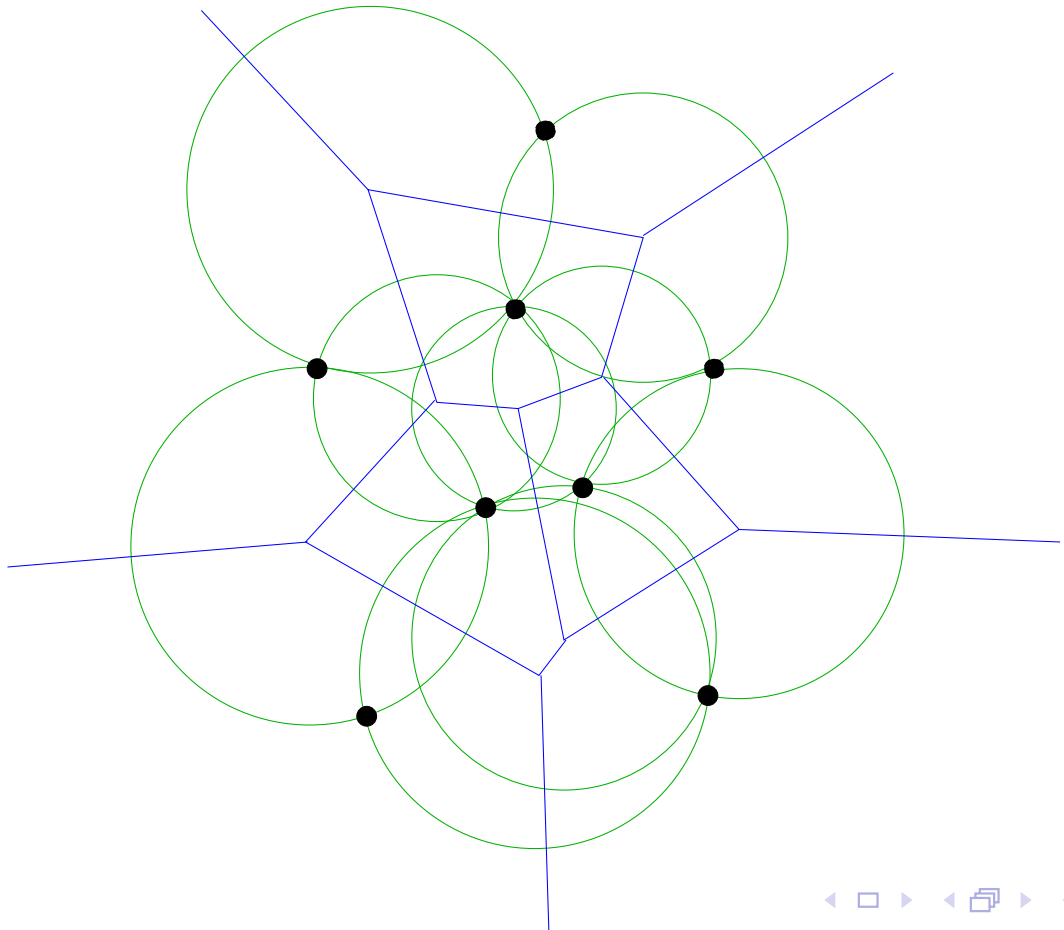


Arestas e vértices de $\mathrm{Vor}(P)$

$C_P(q)$: maior círculo centrado em q que
não contenha pontos de P no seu interior.

Teorema:

(i) Ponto q é vértice de $\mathrm{Vor}(P)$ sse $C_P(q)$ contém três ou mais pontos de P (em sua fronteira).

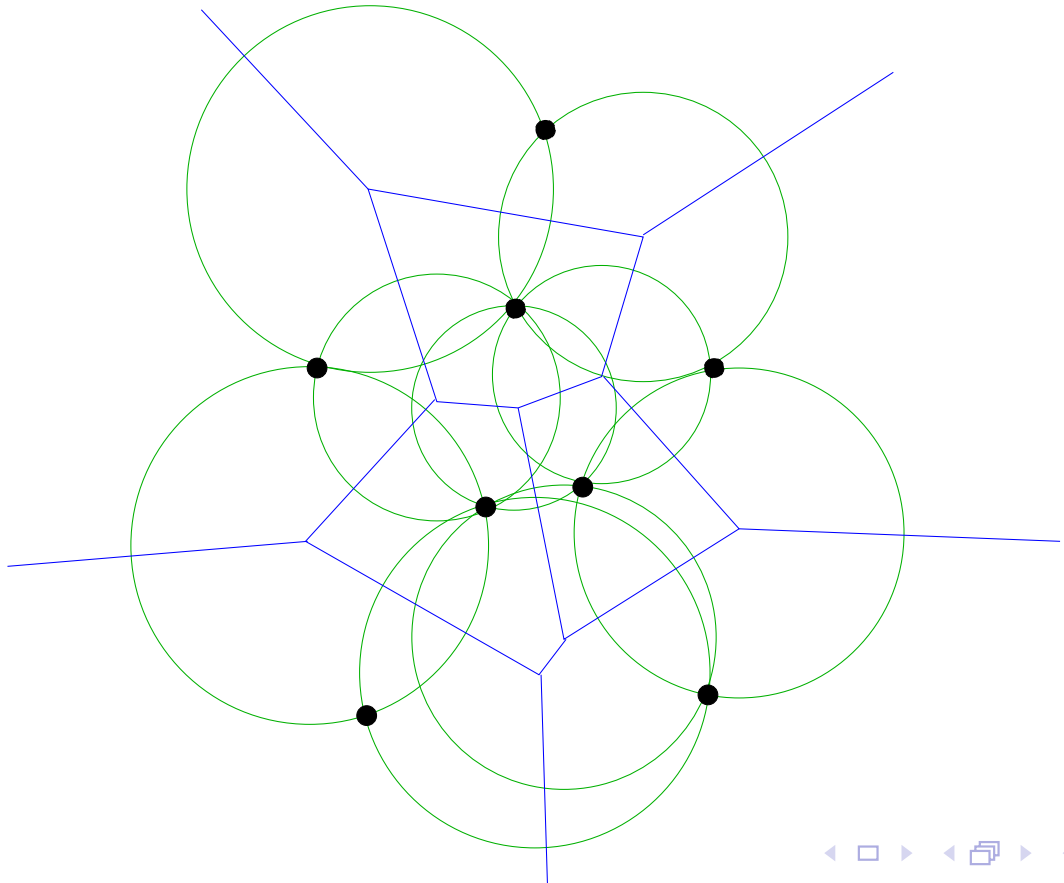


Arestas e vértices de $\mathrm{Vor}(P)$

Teorema:

(i) Ponto q é vértice de $\mathrm{Vor}(P)$ sse $C_P(q)$ contém três ou mais pontos de P (em sua fronteira).

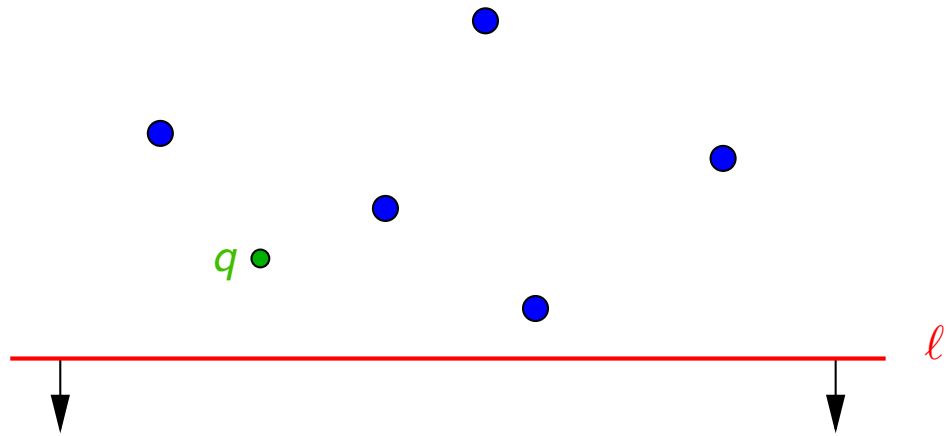
(ii) A reta bissetora entre os pontos p_i e p_j define uma aresta de $\mathrm{Vor}(P)$ sse existe um ponto q nela tq $C_P(q)$ contém p_i e p_j e apenas estes (em sua fronteira).



Algoritmo de Fortune

ℓ^+: semiplano acima da linha de varredura ℓ

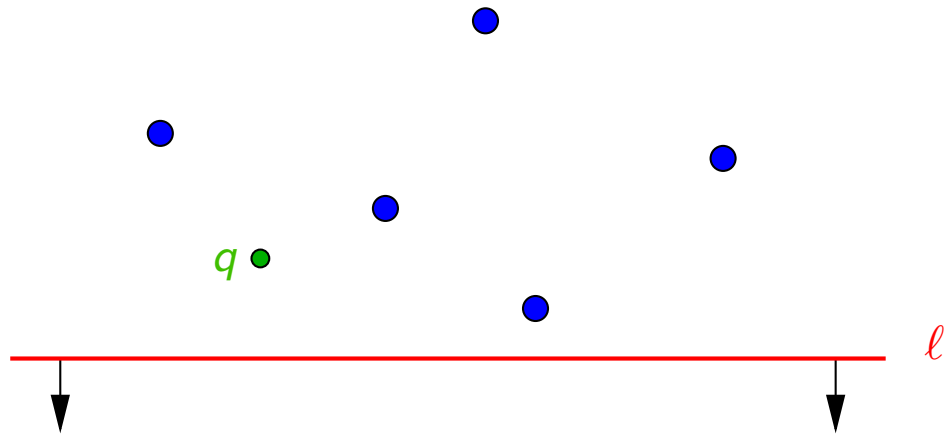
Para quais pontos q em ℓ^+
já conhecemos o ponto de P mais próximo a q?



Algoritmo de Fortune

ℓ^+: semiplano acima da linha de varredura ℓ

Para quais pontos q em ℓ^+
já conhecemos o ponto de P mais próximo a q?

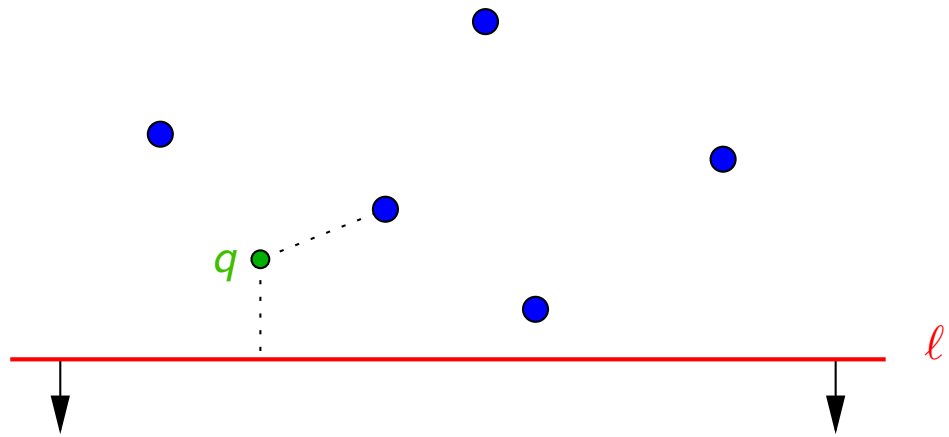


A distância de q a qualquer ponto abaixo de ℓ
é pelo menos a distância de q a ℓ.

Algoritmo de Fortune

ℓ^+: semiplano acima da linha de varredura ℓ

Para quais pontos q em ℓ^+
já conhecemos o ponto de P mais próximo a q?

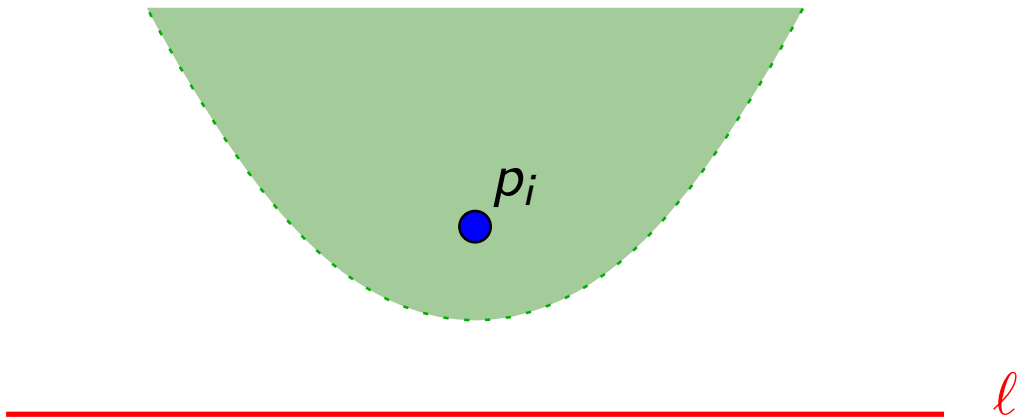


A distância de q a qualquer ponto abaixo de ℓ
é pelo menos a distância de q a ℓ.

Se q está mais próximo de um p_i acima de ℓ do que de ℓ,
então q está em $\mathcal{V}(p_i)$.

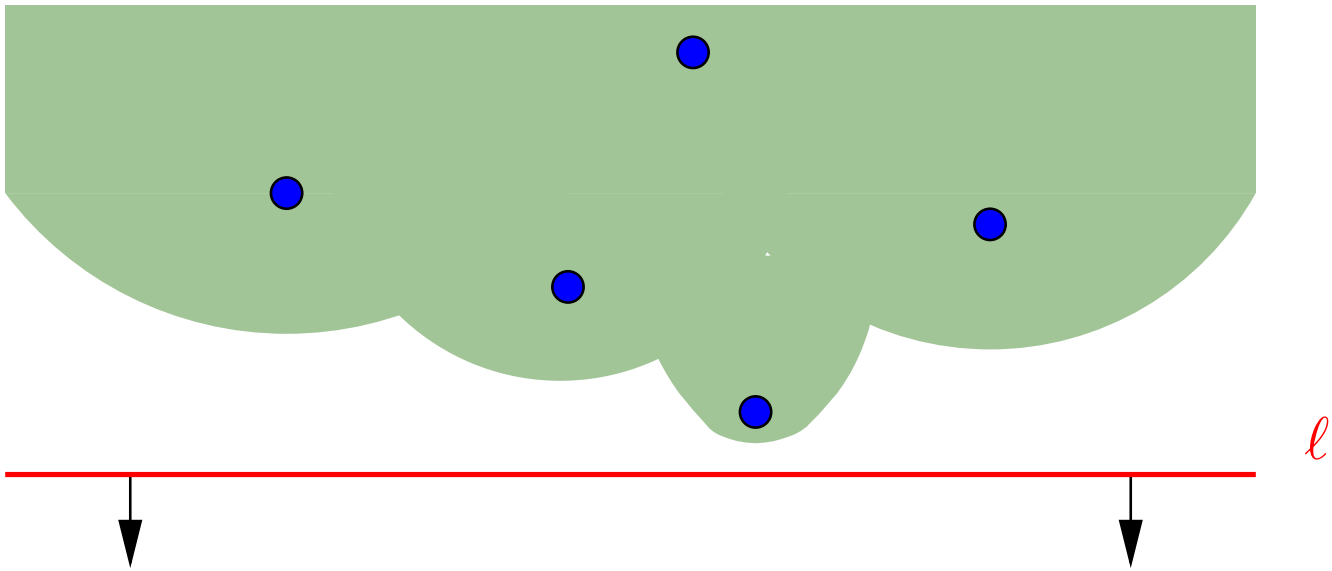
Algoritmo de Fortune

O conjunto dos pontos mais próximos a p_i do que ℓ é delimitado por uma parábola.



Algoritmo de Fortune

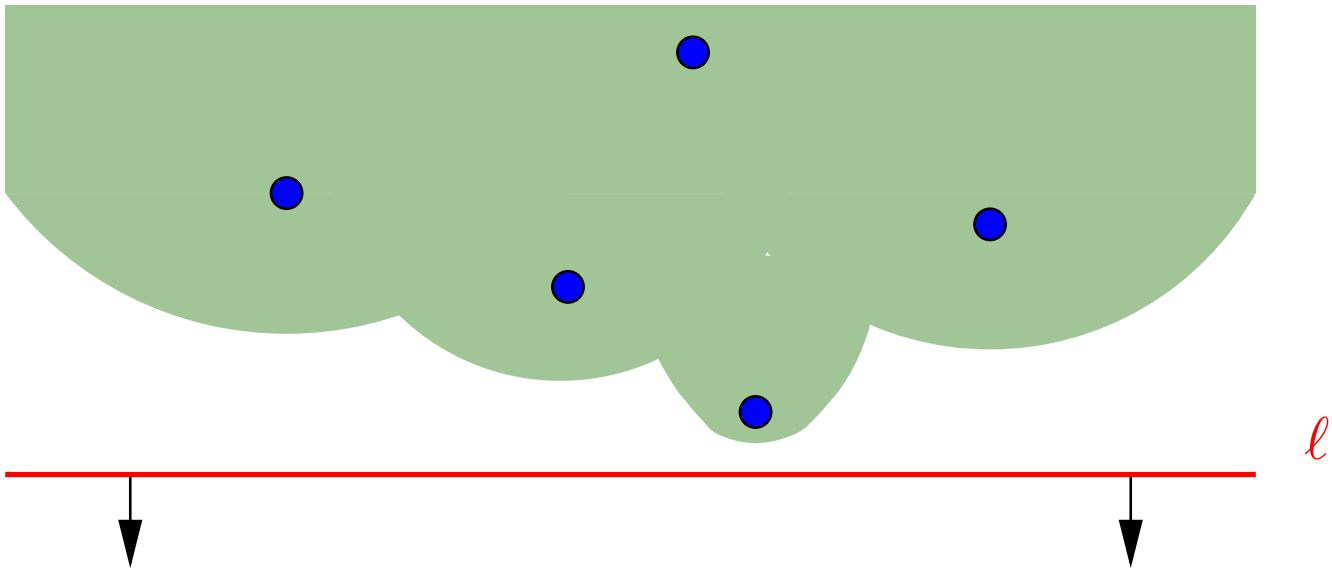
O conjunto dos pontos mais próximos a p_i do que ℓ é delimitado por uma parábola.



Assim, a região de ℓ^+ onde $\mathrm{Vor}(P)$ é conhecido é delimitada por um conjunto de parábolas,

Algoritmo de Fortune

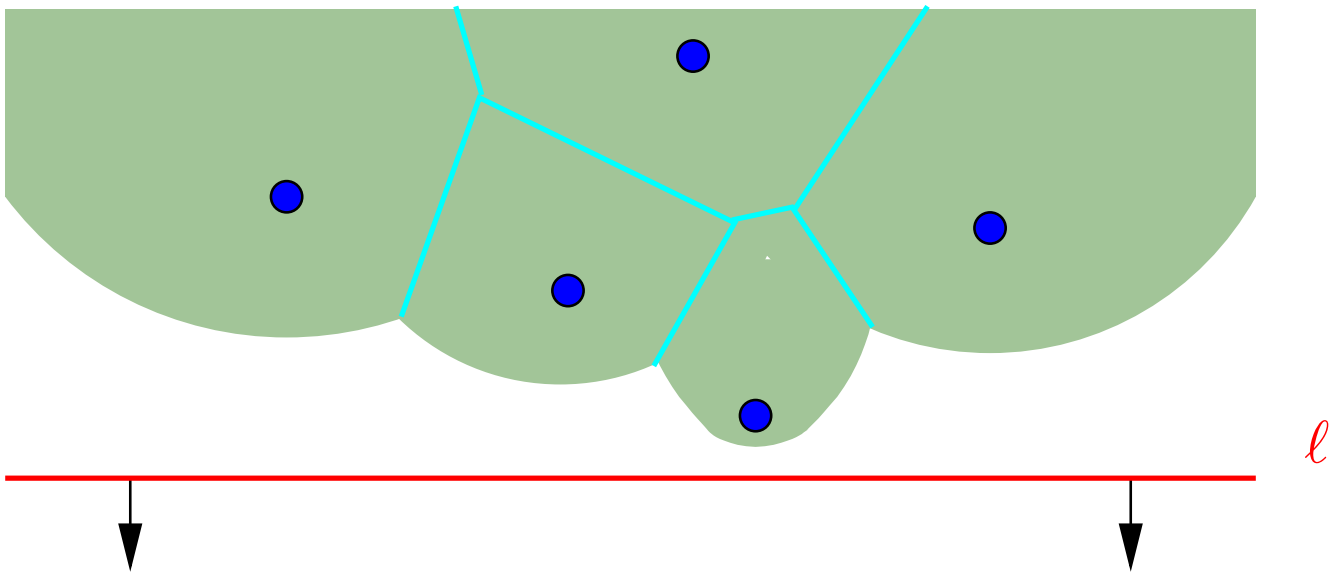
O conjunto dos pontos mais próximos a p_i do que ℓ
é delimitado por uma parábola.



Assim, a região de ℓ^+ onde $\mathrm{Vor}(P)$ é conhecido
é delimitada por um conjunto de parábolas,
ou arcos parabolóides,
que definem a chamada linha da praia.

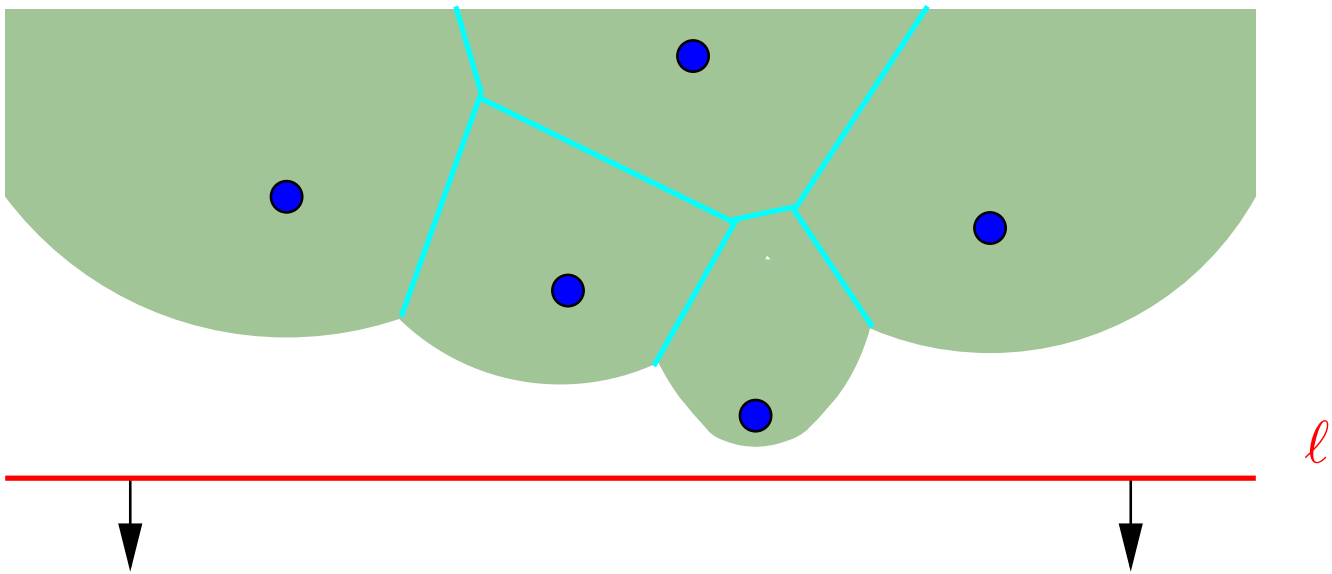
Algoritmo de Fortune

Pontos de encontro entre duas parábolas na linha da praia estão sempre sobre alguma aresta de $\mathrm{Vor}(P)$.



Algoritmo de Fortune

Pontos de encontro entre duas parábolas na linha da praia estão sempre sobre alguma aresta de $\mathrm{Vor}(P)$.



Esses pontos de encontro desenham $\mathrm{Vor}(P)$.
Vejam a animação do algoritmo.