Diagrame Voronoi

Mihai-Sorin Stupariu

Sem. I, 2019-2020

Problematizare

- Se consideră o mulțime de puncte (oficiile poștale) din plan. Care este cel mai apropiat?

Problematizare

- Se consideră o mulțime de puncte (oficiile poștale) din plan. Care este cel mai apropiat?
- Ideea de a delimita "zone de influență" a apărut cu multă vreme în urmă (de exemplu în lucrările lui Descartes, dar și în legătură cu alte probleme; este utilizată în mod curent în varii domenii. În plus, astfel de "împărțiri" apar în natură.

Formalizare

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de puncte din planul $\mathbb{R}^2$, numite **situri**.

Formalizare

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de puncte din planul $\mathbb{R}^2$, numite **situri**.
- **Diagrama Voronoi** a lui $\mathcal{P}$ (notată $\mathrm{Vor}(\mathcal{P})$) este o divizare a planului $\mathbb{R}^2$ în n celule $\mathcal{V}(P_1), \ldots, \mathcal{V}(P_n)$ cu proprietatea că

$$P \in \mathcal{V}(P_i) \Leftrightarrow d(P, P_i) \leq d(P, P_j),\ \forall j = 1, \ldots, n.$$

Formalizare

- Fie $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ o mulțime de puncte din planul $\mathbb{R}^2$, numite **situri**.
- **Diagrama Voronoi** a lui $\mathcal{P}$ (notată $\mathrm{Vor}(\mathcal{P})$) este o divizare a planului $\mathbb{R}^2$ în n celule $\mathcal{V}(P_1), \dots, \mathcal{V}(P_n)$ cu proprietatea că

$$P \in \mathcal{V}(P_i) \Leftrightarrow d(P, P_i) \leq d(P, P_j), \ \forall j = 1, \dots, n.$$

- Două celule adiacente au în comun o **muchie** sau un **vârf** (punct de intersecție a muchiilor).

Formalizare

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de puncte din planul $\mathbb{R}^2$, numite **situri**.
- **Diagrama Voronoi** a lui $\mathcal{P}$ (notată $\mathrm{Vor}(\mathcal{P})$) este o divizare a planului $\mathbb{R}^2$ în n celule $\mathcal{V}(P_1), \ldots, \mathcal{V}(P_n)$ cu proprietatea că

$$P \in \mathcal{V}(P_i) \Leftrightarrow d(P, P_i) \leq d(P, P_j),\ \forall j = 1, \ldots, n.$$

- Două celule adiacente au în comun o **muchie** sau un **vârf** (punct de intersecție a muchiilor).
- **Atenție!** Vârfurile lui $\mathrm{Vor}(\mathcal{P})$ sunt diferite de punctele din $\mathcal{P}$.

Formalizare

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de puncte din planul $\mathbb{R}^2$, numite **situri**.
- **Diagrama Voronoi** a lui $\mathcal{P}$ (notată $\mathrm{Vor}(\mathcal{P})$) este o divizare a planului $\mathbb{R}^2$ în n celule $\mathcal{V}(P_1), \ldots, \mathcal{V}(P_n)$ cu proprietatea că

$$P \in \mathcal{V}(P_i) \Leftrightarrow d(P, P_i) \leq d(P, P_j), \ \forall j = 1, \ldots, n.$$

- Două celule adiacente au în comun o **muchie** sau un **vârf** (punct de intersecție a muchiilor).
- **Atenție!** Vârfurile lui $\mathrm{Vor}(\mathcal{P})$ sunt diferite de punctele din $\mathcal{P}$.
- Uneori, prin abuz de limbaj, este precizată doar împărțirea în muchii / vârfuri.

Formalizare

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de puncte din planul $\mathbb{R}^2$, numite **situri**.
- **Diagrama Voronoi** a lui $\mathcal{P}$ (notată $\mathrm{Vor}(\mathcal{P})$) este o divizare a planului $\mathbb{R}^2$ în n celule $\mathcal{V}(P_1), \ldots, \mathcal{V}(P_n)$ cu proprietatea că

$$P \in \mathcal{V}(P_i) \Leftrightarrow d(P, P_i) \leq d(P, P_j), \ \forall j = 1, \ldots, n.$$

- Două celule adiacente au în comun o **muchie** sau un **vârf** (punct de intersecție a muchiilor).
- **Atenție!** Vârfurile lui $\mathrm{Vor}(\mathcal{P})$ sunt diferite de punctele din $\mathcal{P}$.
- Uneori, prin abuz de limbaj, este precizată doar împărțirea în muchii / vârfuri.
- Diagrame Voronoi pot fi construite pentru diverse funcții distanță (e.g. distanța Manhattan); forma celulelor depinde de forma "cercului" în raport cu funcția distanță respectivă.

Structura unei diagrame Voronoi

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de situri (puncte) din planul $\mathbb{R}^2$.

Structura unei diagrame Voronoi

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de situri (puncte) din planul $\mathbb{R}^2$.
- Celula asociată unui punct este o intersecție de semiplane:

$$\mathcal{V}(P_i) = \bigcap_{j \neq i} h(P_i, P_j),$$

unde $h(P_i, P_j)$ este semiplanul determinat de mediatoarea segmentului $[P_i P_j]$ care conține punctul P_i. În particular: fiecare celulă este o mulțime convexă. Aplicabilitate: algoritm (lent) de determinare a diagramei Voronoi.

Structura unei diagrame Voronoi

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de situri (puncte) din planul $\mathbb{R}^2$.
- Celula asociată unui punct este o intersecție de semiplane:
$$\mathcal{V}(P_i) = \bigcap_{j \neq i} h(P_i, P_j),$$
unde $h(P_i, P_j)$ este semiplanul determinat de mediatoarea segmentului $[P_i P_j]$ care conține punctul P_i. În particular: fiecare celulă este o mulțime convexă. Aplicabilitate: algoritm (lent) de determinare a diagramei Voronoi.
- Dacă toate punctele sunt coliniare, atunci diagrama Voronoi asociată $\mathrm{Vor}(\mathcal{P})$ conține $n-1$ *drepte paralele* între ele (în particular, pentru $n \geq 3$, ea nu este conexă).
- În caz contrar, diagrama este conexă, iar muchiile sale sunt fie *segmente*, fie *semidrepte* (cui corespund acestea?).

Structura unei diagrame Voronoi

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulțime de situri (puncte) din planul $\mathbb{R}^2$.
- Celula asociată unui punct este o intersecție de semiplane:
$$\mathcal{V}(P_i) = \bigcap_{j \neq i} h(P_i, P_j),$$
unde $h(P_i, P_j)$ este semiplanul determinat de mediatoarea segmentului $[P_i P_j]$ care conține punctul P_i. În particular: fiecare celulă este o mulțime convexă. Aplicabilitate: algoritm (lent) de determinare a diagramei Voronoi.
- Dacă toate punctele sunt coliniare, atunci diagrama Voronoi asociată $\mathrm{Vor}(\mathcal{P})$ conține $n-1$ *drepte paralele* între ele (în particular, pentru $n \geq 3$, ea nu este conexă).
- În caz contrar, diagrama este conexă, iar muchiile sale sunt fie *segmente*, fie *semidrepte* (cui corespund acestea?).
- **Propoziție.** Fie o mulțime cu n situri. Atunci, pentru diagrama Voronoi asociată au loc inegalitățile
$$n_v \leq 2n - 5, \qquad n_m \leq 3n - 6,$$

Legătura cu triangulările legale / unghiular optime

- Construcție:

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$
  - Diagrama Voronoi $\mathrm{Vor}(\mathcal{P}) \Longrightarrow$

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$
  - Diagrama Voronoi $\mathrm{Vor}(\mathcal{P}) \Longrightarrow$
  - Graful dual $\mathcal{G}(\mathcal{P})$. **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună $\Longrightarrow$

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$
  - Diagrama Voronoi $\mathrm{Vor}(\mathcal{P}) \Longrightarrow$
  - Graful dual $\mathcal{G}(\mathcal{P})$. **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună $\Longrightarrow$
  - Triangulare $\mathcal{T}_{\mathcal{P}}$ (numită **triangulare Delaunay**)

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$
  - Diagrama Voronoi $\mathrm{Vor}(\mathcal{P}) \Longrightarrow$
  - Graful dual $\mathcal{G}(\mathcal{P})$. **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună $\Longrightarrow$
  - Triangulare $\mathcal{T}_{\mathcal{P}}$ (numită **triangulare Delaunay**)
- **Propoziție.** *Fie $\mathcal{T}$ o triangulare a lui $\mathcal{P}$. Atunci $\mathcal{T}$ este o triangulare Delaunay dacă și numai dacă pentru orice triunghi din $\mathcal{T}$ cercul circumscris nu conține în interiorul său niciun punct al lui $\mathcal{P}$.*

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$
  - Diagrama Voronoi $\mathrm{Vor}(\mathcal{P}) \Longrightarrow$
  - Graful dual $\mathcal{G}(\mathcal{P})$. **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună $\Longrightarrow$
  - Triangulare $\mathcal{T}_{\mathcal{P}}$ (numită **triangulare Delaunay**)
- **Propoziție.** *Fie $\mathcal{T}$ o triangulare a lui $\mathcal{P}$. Atunci $\mathcal{T}$ este o triangulare Delaunay dacă și numai dacă pentru orice triunghi din $\mathcal{T}$ cercul circumscris nu conține în interiorul său niciun punct al lui $\mathcal{P}$.*
- **Teoremă.** *O triangulare este legală dacă și numai dacă este o triangulare Delaunay.*

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$
  - Diagrama Voronoi $\mathrm{Vor}(\mathcal{P}) \Longrightarrow$
  - Graful dual $\mathcal{G}(\mathcal{P})$. **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună $\Longrightarrow$
  - Triangulare $\mathcal{T}_{\mathcal{P}}$ (numită **triangulare Delaunay**)
- **Propoziție.** *Fie $\mathcal{T}$ o triangulare a lui $\mathcal{P}$. Atunci $\mathcal{T}$ este o triangulare Delaunay dacă și numai dacă pentru orice triunghi din $\mathcal{T}$ cercul circumscris nu conține în interiorul său niciun punct al lui $\mathcal{P}$.*
- **Teoremă.** *O triangulare este legală dacă și numai dacă este o triangulare Delaunay.*
- **Teoremă.** *Orice triangulare unghiular optimă este o triangulare Delaunay. Orice triangulare Delaunay maximizează cel mai mic unghi, comparativ cu toate triangulările lui $\mathcal{P}$.*

Legătura cu triangulările legale / unghiular optime

- Construcție:
  - Mulțime de puncte $\mathcal{P}$ în planul $\mathbb{R}^2 \Longrightarrow$
  - Diagrama Voronoi $\mathrm{Vor}(\mathcal{P}) \Longrightarrow$
  - Graful dual $\mathcal{G}(\mathcal{P})$. **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună $\Longrightarrow$
  - Triangulare $\mathcal{T}_{\mathcal{P}}$ (numită **triangulare Delaunay**)
- **Propoziție.** *Fie $\mathcal{T}$ o triangulare a lui $\mathcal{P}$. Atunci $\mathcal{T}$ este o triangulare Delaunay dacă și numai dacă pentru orice triunghi din $\mathcal{T}$ cercul circumscris nu conține în interiorul său niciun punct al lui $\mathcal{P}$.*
- **Teoremă.** *O triangulare este legală dacă și numai dacă este o triangulare Delaunay.*
- **Teoremă.** *Orice triangulare unghiular optimă este o triangulare Delaunay. Orice triangulare Delaunay maximizează cel mai mic unghi, comparativ cu toate triangulările lui $\mathcal{P}$.*
- **Întrebare:** Cum "funcționează" această construcție când punctele din $\mathcal{P}$ sunt (de exemplu) vârfurile unui pătrat?

Algoritmul lui Fortune [1987]

- Metodă clasică (și eficientă) de determinare a diagramei Voronoi pentru o mulțime de puncte din planul $\mathbb{R}^2$. Complexitate: $O(n \log n)$. Detalii: http://www.ams.org/samplings/feature-column/fcarc-voronoi

Algoritmul lui Fortune [1987]

- Metodă clasică (și eficientă) de determinare a diagramei Voronoi pentru o mulțime de puncte din planul $\mathbb{R}^2$. Complexitate: $O(n \log n)$. Detalii: http://www.ams.org/samplings/feature-column/fcarc-voronoi
- **Principiu (paradigmă):** sweep line / paradigma dreptei de baleiere.

Algoritmul lui Fortune [1987]

- Metodă clasică (și eficientă) de determinare a diagramei Voronoi pentru o mulțime de puncte din planul $\mathbb{R}^2$. Complexitate: $O(n \log n)$. Detalii: http://www.ams.org/samplings/feature-column/fcarc-voronoi
- **Principiu (paradigmă):** sweep line / paradigma dreptei de baleiere.
- **Inconvenient:** la întâlnirea unui vârf al diagramei, dreapta de baleiere nu a întâlnit încă toate siturile (puncte din $\mathcal{P}$) care determină acest vârf!

Algoritmul lui Fortune [1987]

- Metodă clasică (și eficientă) de determinare a diagramei Voronoi pentru o mulțime de puncte din planul $\mathbb{R}^2$. Complexitate: $O(n \log n)$. Detalii: http://www.ams.org/samplings/feature-column/fcarc-voronoi
- **Principiu (paradigmă):** sweep line / paradigma dreptei de baleiere.
- **Inconvenient:** la întâlnirea unui vârf al diagramei, dreapta de baleiere nu a întâlnit încă toate siturile (puncte din $\mathcal{P}$) care determină acest vârf!
- **Adaptare:** nu reținem informația legată de intersecția dintre linia de baleiere și diagramă, ci doar informația legată de partea diagramei care <u>nu</u> mai poate fi influențată de punctele situate de dincolo de linia de baleiere. Din punct de vedere practic, apare o reuniune de arce de parabolă (**curbă parabolică)**, ceea ce este situat deasupra acestei curbe nu mai poate fi influențat de evenimentele nedetectate.

Despre curba parabolică (I)

- **Curba parabolică (*beach line*):**

Despre curba parabolică (I)

- **Curba parabolică (*beach line*):**
  - Este o reuniune de arce de parabolă.
  - Un punct de pe curba parabolică este egal depărtat de situl care determină arcul de parabolă și dreapta de baleiere. Presupunem că dreapta de baleiere d are ecuația $y = 0$, iar situl P_i, situat deasupra lui d (adică $y_i > 0$) are coordonatele (x_i, y_i). Locul geometric al punctelor egal depărtate de d și P_i are ecuația
$$y = \frac{1}{2y_i}x^2 - \frac{x_i}{y_i}x + \frac{x_i^2 + y_i^2}{2y_i}.$$

Despre curba parabolică (I)

- **Curba parabolică (*beach line*):**
  - Este o reuniune de arce de parabolă.
  - Un punct de pe curba parabolică este egal depărtat de situl care determină arcul de parabolă și dreapta de baleiere. Presupunem că dreapta de baleiere d are ecuația $y = 0$, iar situl P_i, situat deasupra lui d (adică $y_i > 0$) are coordonatele (x_i, y_i). Locul geometric al punctelor egal depărtate de d și P_i are ecuația
  $$y = \frac{1}{2y_i}x^2 - \frac{x_i}{y_i}x + \frac{x_i^2 + y_i^2}{2y_i}.$$
  - Punctele de racord ale arcelor de parabolă aparțin muchiilor diagramei Voronoi;

Despre curba parabolică (I)

- **Curba parabolică (*beach line*):**
  - Este o reuniune de arce de parabolă.
  - Un punct de pe curba parabolică este egal depărtat de situl care determină arcul de parabolă și dreapta de baleiere. Presupunem că dreapta de baleiere d are ecuația $y = 0$, iar situl P_i, situat deasupra lui d (adică $y_i > 0$) are coordonatele (x_i, y_i). Locul geometric al punctelor egal depărtate de d și P_i are ecuația
  $$y = \frac{1}{2y_i}x^2 - \frac{x_i}{y_i}x + \frac{x_i^2 + y_i^2}{2y_i}.$$
  - Punctele de racord ale arcelor de parabolă aparțin muchiilor diagramei Voronoi;
  - Curba parabolică este x-monotonă, adică orice dreaptă verticală o intersectează *exact* într-un punct (la ce folosește această proprietate?).

Despre curba parabolică (II)

- Modificarea curbei parabolice:

Despre curba parabolică (II)

- Modificarea curbei parabolice:
  - **Site event / eveniment de tip locație.** (i) La trecerea printr-un sit apare un arc de parabolă (care “la început” este degenerat) și, reciproc, apariția unui nou arc este posibilă doar la trecerea printr-un sit. (ii) În consecință, la un moment fixat, curba parabolică are maxim $(2n-1)$ arce.

Despre curba parabolică (II)

- Modificarea curbei parabolice:
  - **Site event / eveniment de tip locație.** (i) La trecerea printr-un sit apare un arc de parabolă (care "la început" este degenerat) și, reciproc, apariția unui nou arc este posibilă doar la trecerea printr-un sit. (ii) În consecință, la un moment fixat, curba parabolică are maxim $(2n-1)$ arce.
  - **Circle event / eveniment de tip cerc.** (i) La întâlnirea punctului inferior al unui cerc care trece prin cel puțin trei situri și este tangent la dreapta de baleiere dispare un arc de parabolă și, reciproc, arcele de parabolă dispar doar la acest tip de evenimente. (ii) Un eveniment de tip cerc este dat de trei arce de parabolă consecutive de pe curba parabolică, deci trebuie testate toate tripletele consecutive de arce, pe măsură ce ele apar. (iii) Un astfel de eveniment este asociat unui vârf al diagramei Voronoi. (iv) Există triplete de arce consecutive (muchii ale diagramei Voronoi) pentru care muchiile nu se întâlnesc. (v) Unele evenimente de tip cerc detectate nu au loc.

Structuri de date utilizate

- **Diagrama Voronoi:** listă de muchii dublu înlănțuite $\mathcal{D}$

Structuri de date utilizate

- **Diagrama Voronoi:** listă de muchii dublu înlănțuite $\mathcal{D}$
- **Evenimente:** coadă de priorități / evenimente $\mathcal{Q}$ și arbore de căutare binar echilibrat. Sunt reținute evenimentele de tip sit, precum și evenimentele de tip cerc (este stocat "punctul inferior") - acestea sunt detectate pe parcursul algoritmului.

Structuri de date utilizate

- **Diagrama Voronoi:** listă de muchii dublu înlănțuite $\mathcal{D}$
- **Evenimente:** coadă de priorități / evenimente $\mathcal{Q}$ și arbore de căutare binar echilibrat. Sunt reținute evenimentele de tip sit, precum și evenimentele de tip cerc (este stocat "punctul inferior") - acestea sunt detectate pe parcursul algoritmului.
- **Statut:** Structura curbei parabolice (simbolic) - arbore de căutare binar echilibrat $\mathcal{T}$.

Structuri de date utilizate

- **Diagrama Voronoi:** listă de muchii dublu înlănțuite $\mathcal{D}$
- **Evenimente:** coadă de priorități / evenimente $\mathcal{Q}$ și arbore de căutare binar echilibrat. Sunt reținute evenimentele de tip sit, precum și evenimentele de tip cerc (este stocat "punctul inferior") - acestea sunt detectate pe parcursul algoritmului.
- **Statut:** Structura curbei parabolice (simbolic) - arbore de căutare binar echilibrat $\mathcal{T}$.

Structuri de date utilizate

- **Diagrama Voronoi:** listă de muchii dublu înlănțuite $\mathcal{D}$
- **Evenimente:** coadă de priorități / evenimente $\mathcal{Q}$ și arbore de căutare binar echilibrat. Sunt reținute evenimentele de tip sit, precum și evenimentele de tip cerc (este stocat "punctul inferior") - acestea sunt detectate pe parcursul algoritmului.
- **Statut:** Structura curbei parabolice (simbolic) - arbore de căutare binar echilibrat $\mathcal{T}$.
  - **pe frunze:** siturile care au arce active, la ajungerea într-un sit se inserează un arc, la ajungerea într-un eveniment de tip cerc se șterge un arc;

Structuri de date utilizate

- **Diagrama Voronoi:** listă de muchii dublu înlănțuite $\mathcal{D}$
- **Evenimente:** coadă de priorități / evenimente $\mathcal{Q}$ și arbore de căutare binar echilibrat. Sunt reținute evenimentele de tip sit, precum și evenimentele de tip cerc (este stocat "punctul inferior") - acestea sunt detectate pe parcursul algoritmului.
- **Statut:** Structura curbei parabolice (simbolic) - arbore de căutare binar echilibrat $\mathcal{T}$.
  - **pe frunze:** siturile care au arce active, la ajungerea într-un sit se inserează un arc, la ajungerea într-un eveniment de tip cerc se șterge un arc;
  - **în nodurile interne:** punctele de racord ale arcelor de parabolă (memorate simbolic) - corespund muchiilor diagramei Voronoi

Structuri de date utilizate

- **Diagrama Voronoi:** listă de muchii dublu înlănțuite $\mathcal{D}$
- **Evenimente:** coadă de priorități / evenimente $\mathcal{Q}$ și arbore de căutare binar echilibrat. Sunt reținute evenimentele de tip sit, precum și evenimentele de tip cerc (este stocat "punctul inferior") - acestea sunt detectate pe parcursul algoritmului.
- **Statut:** Structura curbei parabolice (simbolic) - arbore de căutare binar echilibrat $\mathcal{T}$.
  - **pe frunze:** siturile care au arce active, la ajungerea într-un sit se inserează un arc, la ajungerea într-un eveniment de tip cerc se șterge un arc;
  - **în nodurile interne:** punctele de racord ale arcelor de parabolă (memorate simbolic) - corespund muchiilor diagramei Voronoi
  - pointeri (eventual nuli) de la frunze către evenimentele de tip cerc; de la nodurile interne către muchiile diagramei Voronoi

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \ldots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\mathrm{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \ldots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\mathrm{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \ldots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\mathrm{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$
3. **do** elimină evenimentul cu cel mai mare y din $\mathcal{Q}$

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \ldots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\mathrm{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$
3. **do** elimină evenimentul cu cel mai mare y din $\mathcal{Q}$
4. **if** evenimentul **ev** este un eveniment de tip sit

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \ldots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\mathrm{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$
3. **do** elimină evenimentul cu cel mai mare y din $\mathcal{Q}$
4. **if** evenimentul **ev** este un eveniment de tip sit
5. **then** $\textsc{ProcessEvSit}(p_i)$, cu p_i=**ev**

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \ldots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\mathrm{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$
3. **do** elimină evenimentul cu cel mai mare y din $\mathcal{Q}$
4. **if** evenimentul **ev** este un eveniment de tip sit
5. **then** $\textsc{ProcessEvSit}(p_i)$, cu p_i=**ev**
6. **else** $\textsc{ProcessEvCerc}(\gamma)$, cu $\gamma = arc(\mathbf{ev}) \in \mathcal{T}$

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \dots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\text{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$
3.   **do** elimină evenimentul cu cel mai mare y din $\mathcal{Q}$
4.   **if** evenimentul **ev** este un eveniment de tip sit
5.   **then** $\text{ProcessEvSit}(p_i)$, cu p_i=**ev**
6.   **else** $\text{ProcessEvCerc}(\gamma)$, cu $\gamma = arc(\textbf{ev}) \in \mathcal{T}$
7. Nodurile interne încă prezente în $\mathcal{T}$ corespund semidreptelor diagramei Voronoi. Consideră un *bounding box* care conține toate vârfurile diagramei Voronoi în interiorul să și leagă semidreptele de acest *bounding box*, prin actualizarea corespunzătoare a lui $\mathcal{D}$.

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \ldots, p_n\}$ de situri în plan.
Output. Diagrama Voronoi $\text{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă de muchii dublu înlănțuite (DCEL) $\mathcal{D}$.

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înlănțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$
3. **do** elimină evenimentul cu cel mai mare y din $\mathcal{Q}$
4. **if** evenimentul **ev** este un eveniment de tip sit
5. **then** $\textsc{ProcessEvSit}(p_i)$, cu p_i=**ev**
6. **else** $\textsc{ProcessEvCerc}(\gamma)$, cu $\gamma = arc(\mathbf{ev}) \in \mathcal{T}$
7. Nodurile interne încă prezente în $\mathcal{T}$ corespund semidreptelor diagramei Voronoi. Consideră un *bounding box* care conține toate vârfurile diagramei Voronoi în interiorul să și leagă semidreptele de acest *bounding box*, prin actualizarea corespunzătoare a lui $\mathcal{D}$.
8. Traversează muchiile pentru a adăuga celulele diagramei și pointeri corespunzători.

Procedura PROCESSEVSIT (p_i)

1. Dacă $\mathcal{T}$ este vidă, inserează p_i și revine, dacă nu continuă cu 2.−5.

Procedura ProcessEvSit (p_i)

1. Dacă $\mathcal{T}$ este vidă, inserează p_i și revine, dacă nu continuă cu 2.$-$5.
2. Caută în $\mathcal{T}$ arcul α situat deasupra lui p_i. Dacă frunza reprezentând α are un pointer către un eveniment de tip cerc **ev** din $\mathcal{Q}$, atunci **ev** este o alarmă falsă și trebuie șters.

Procedura PROCESSEVSIT (p_i)

1. Dacă $\mathcal{T}$ este vidă, inserează p_i și revine, dacă nu continuă cu 2.−5.
2. Caută în $\mathcal{T}$ arcul α situat deasupra lui p_i. Dacă frunza reprezentând α are un pointer către un eveniment de tip cerc **ev** din $\mathcal{Q}$, atunci **ev** este o alarmă falsă și trebuie șters.
3. Înlocuiește frunza lui $\mathcal{T}$ care reprezintă α cu un subarbore cu trei frunze: cea din mijloc reține situl p_i și celelalte două situl p_j asociat lui α. Memorează perechile reprezentând punctele de racord în două noduri interne. Efectuează rebalansări în $\mathcal{T}$, dacă este necesar.

Procedura ProcessEvSit (p_i)

1. Dacă $\mathcal{T}$ este vidă, inserează p_i și revine, dacă nu continuă cu 2.−5.
2. Caută în $\mathcal{T}$ arcul α situat deasupra lui p_i. Dacă frunza reprezentând α are un pointer către un eveniment de tip cerc **ev** din $\mathcal{Q}$, atunci **ev** este o alarmă falsă și trebuie șters.
3. Înlocuiește frunza lui $\mathcal{T}$ care reprezintă α cu un subarbore cu trei frunze: cea din mijloc reține situl p_i și celelalte două situl p_j asociat lui α. Memorează perechile reprezentând punctele de racord în două noduri interne. Efectuează rebalansări în $\mathcal{T}$, dacă este necesar.
4. Generează noi înregistrări de tip semi-muchie în structura diagramei Voronoi ($\mathcal{D}$), pentru muchiile care separă celulele $V(p_i)$ și $V(p_j)$, corespunzând celor două noi puncte de racord.

Procedura ProcessEvSit (p_i)

1. Dacă $\mathcal{T}$ este vidă, inserează p_i și revine, dacă nu continuă cu 2.−5.
2. Caută în $\mathcal{T}$ arcul α situat deasupra lui p_i. Dacă frunza reprezentând α are un pointer către un eveniment de tip cerc **ev** din $\mathcal{Q}$, atunci **ev** este o alarmă falsă și trebuie șters.
3. Înlocuiește frunza lui $\mathcal{T}$ care reprezintă α cu un subarbore cu trei frunze: cea din mijloc reține situl p_i și celelalte două situl p_j asociat lui α. Memorează perechile reprezentând punctele de racord în două noduri interne. Efectuează rebalansări în $\mathcal{T}$, dacă este necesar.
4. Generează noi înregistrări de tip semi-muchie în structura diagramei Voronoi ($\mathcal{D}$), pentru muchiile care separă celulele $V(p_i)$ și $V(p_j)$, corespunzând celor două noi puncte de racord.
5. Verifică tripletele de arce consecutive nou create, pentru a verifica dacă muchiile corespunzătoare punctelor de racord se întâlnesc. Dacă da, inserează evenimente de tip cerc în $\mathcal{Q}$ și adaugă pointeri de la nodurile lui $\mathcal{T}$ la evenimentele corespunzătoare din $\mathcal{Q}$.

Procedura ProcessEvCerc (γ)

1. Șterge frunza $\gamma \in \mathcal{T}$ care corespunde arcului de cerc α care dispare. Actualizează în nodurile interne perechile care corespund punctelor de racord. Efectuează rebalansări în $\mathcal{T}$, dacă este necesar. Șterge toate evenimentele de tip cerc care îi corespund lui α (cu ajutorul pointerilor de la predecesorul și succesorul lui γ în $\mathcal{T}$.

Procedura ProcessEvCerc (γ)

1. Șterge frunza $\gamma \in \mathcal{T}$ care corespunde arcului de cerc α care dispare. Actualizează în nodurile interne perechile care corespund punctelor de racord. Efectuează rebalansări în $\mathcal{T}$, dacă este necesar. Șterge toate evenimentele de tip cerc care îi corespund lui α (cu ajutorul pointerilor de la predecesorul și succesorul lui γ în $\mathcal{T}$.
2. Adaugă centrul cercului care determină evenimentul ca înregistrare de tip vârf în $\mathcal{D}$. Creează înregistrări de tip semi-muchie corespunzând noului punct de racord de pe linia parabolică și asignează pointeri corespunzători.

Procedura ProcessEvCerc (γ)

1. Șterge frunza $\gamma \in \mathcal{T}$ care corespunde arcului de cerc α care dispare. Actualizează în nodurile interne perechile care corespund punctelor de racord. Efectuează rebalansări în $\mathcal{T}$, dacă este necesar. Șterge toate evenimentele de tip cerc care îi corespund lui α (cu ajutorul pointerilor de la predecesorul și succesorul lui γ în $\mathcal{T}$.
2. Adaugă centrul cercului care determină evenimentul ca înregistrare de tip vârf în $\mathcal{D}$. Creează înregistrări de tip semi-muchie corespunzând noului punct de racord de pe linia parabolică și asignează pointeri corespunzători.
3. Verifică tripletele de arce consecutive nou create (care au foștii vecini ai lui α în centru), pentru a verifica dacă muchiile corespunzătoare punctelor de racord se întâlnesc. Dacă da, inserează evenimente de tip cerc în $\mathcal{Q}$ și adaugă pointeri de la nodurile lui $\mathcal{T}$ la evenimentele corespunzătoare din $\mathcal{Q}$.

Rezultate principale

- **Teoremă**. *Diagrama Voronoi a unei mulțimi de n situri poate fi determinată cu un algoritm bazat pe paradigma dreptei de baleiere având complexitate-timp $O(n \log n)$ și complexitate-spațiu $O(n)$.*

Rezultate principale

- **Teoremă**. *Diagrama Voronoi a unei mulțimi de n situri poate fi determinată cu un algoritm bazat pe paradigma dreptei de baleiere având complexitate-timp $O(n \log n)$ și complexitate-spațiu $O(n)$.*
- **Teoremă**. *Triangularea Delaunay a unei mulțimi de n situri poate fi determinată cu un algoritm bazat pe paradigma dreptei de baleiere având complexitate-timp $O(n \log n)$ și complexitate-spațiu $O(n)$.*