UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

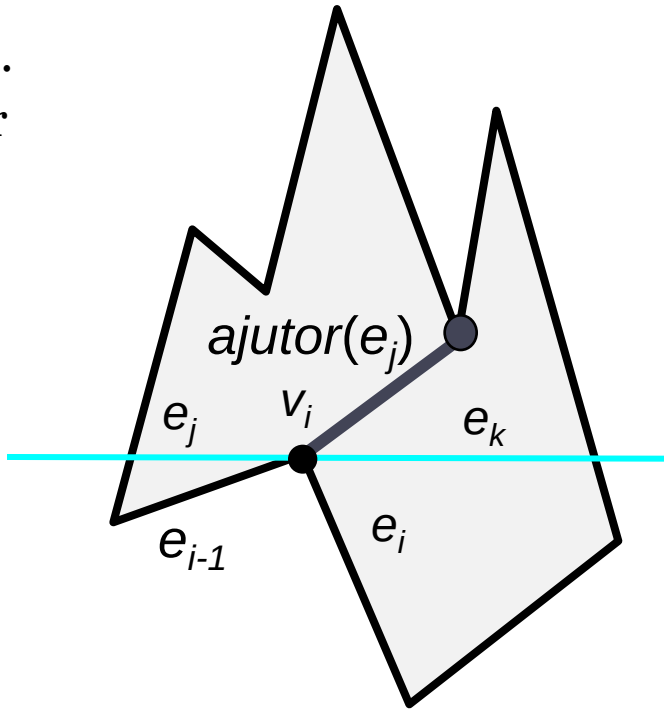
Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Geometrie computationala

8. Triangularea poligoanelor: Partitionare monotona. Alte operatii

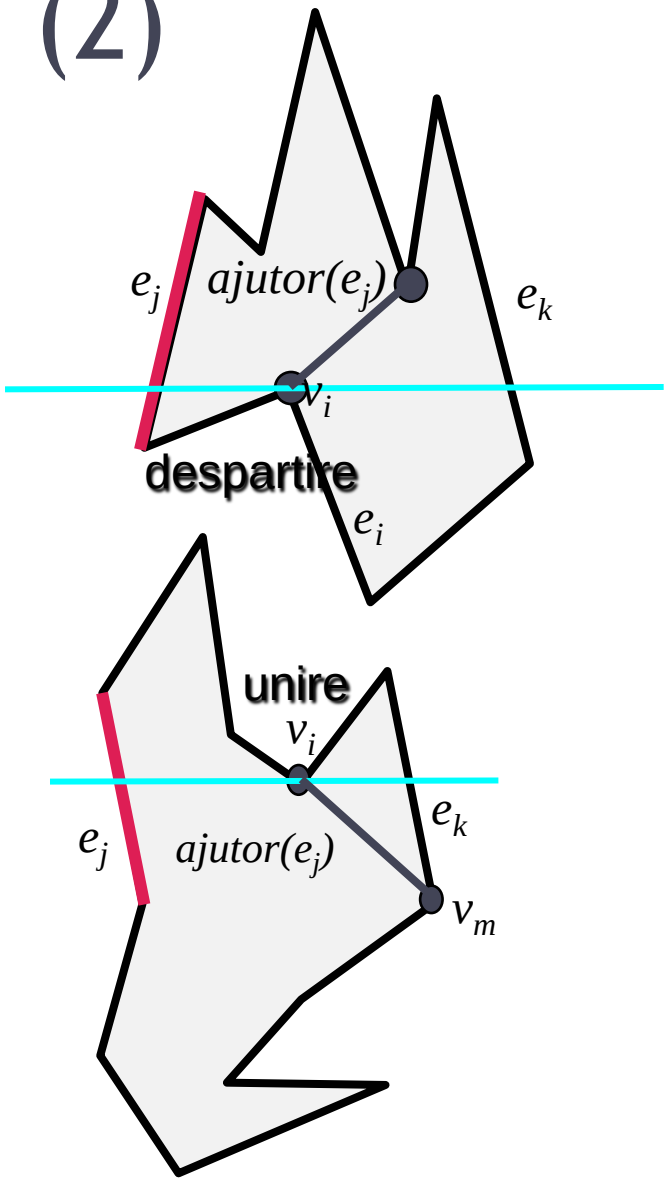
Partitionarea monotona (1)

- Se clasifica toate varfurile.
- Se foloseste o linie de cautare pe axa verticala.
- Muchiile intersectate de linia de cautare *L* vor fi sortate dupa coordonata *x* si pastrate.
- Se pastreaza evenimentele legate de varfuri intr-o coada de evenimente *Q*, sortata dupa coordonata *y*.
- Se elimina varfurile de *despartire/unire* prin legarea acestora de alte varfuri.
- Pentru fiecare muchie *e* se defineste varful *ajutor(e)* ca fiind varful cu cea mai mica coordonata *y* deasupra liniei de cautare si **vizibila** la dreapta muchiei.
- *ajutor(e)* este initializata de catre punctul final superior al lui *e*.



Partitionarea monotona (2)

- Un varf de despartire poate fi legat de varful ajutator al muchiei ce se afla imediat la stanga sa.
- Dar un varf de unire trebuie sa fie legat de un varf ce nu a fost inca procesat!
- **Idee**: Toate varfurile de unire sunt “ajutoarele” unei anumite muchii, si vor fi tratate cand acea muchie se “incheie”.



Algoritm partitionare monotona

Intrare: O lista de varfuri ordonata antiorar. Muchia e_i urmeaza imediat varful v_i.

- construieste Q folosind varfurile lui P in coordonate y.
  - daca doua sau mai multe varfuri au aceeasi coordonata y, varful avand coordonata x mai mica are prioritate
- initializeaza L ca fiind vida
- cat timp Q nu este vida
  - pop(v)
  - trateaza v

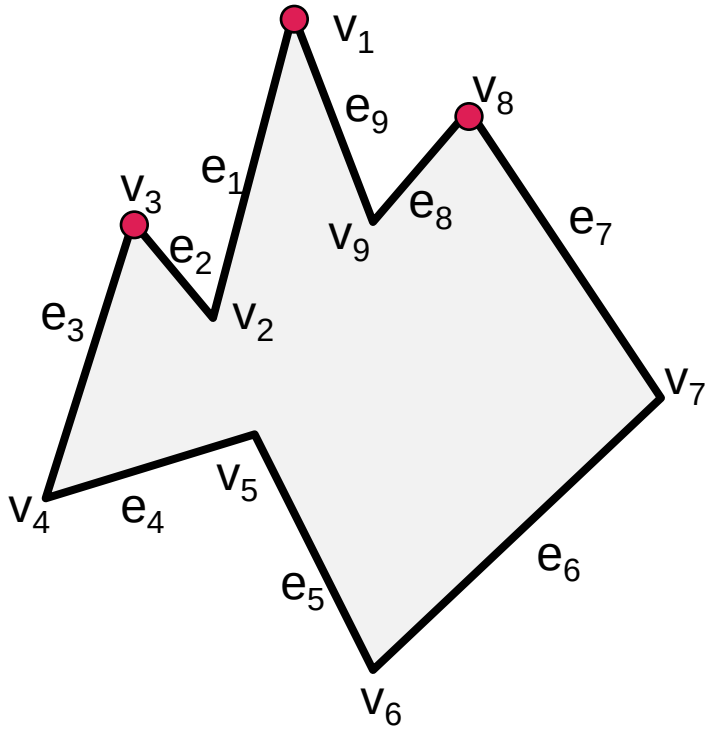
Obs:

- In timpul executiei nu se genereaza evenimente noi.
- Se trateaza toate varfurile de despartire/unire.

Partitionarea monotona: varful de start

Tratarea varfului de start v_i:

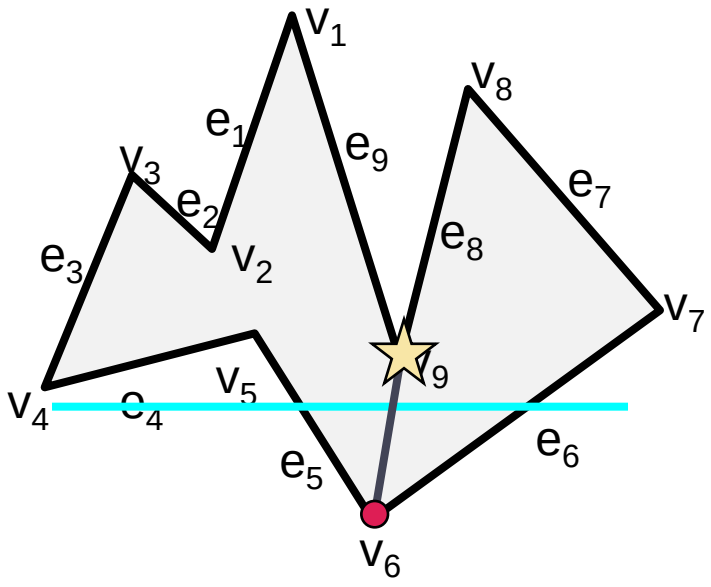
- Se adauga e_i la L
- $ajutor(e_i) \leftarrow v_i$



Partitionarea monotona: varful de final

Tratarea varfului de final v_i:

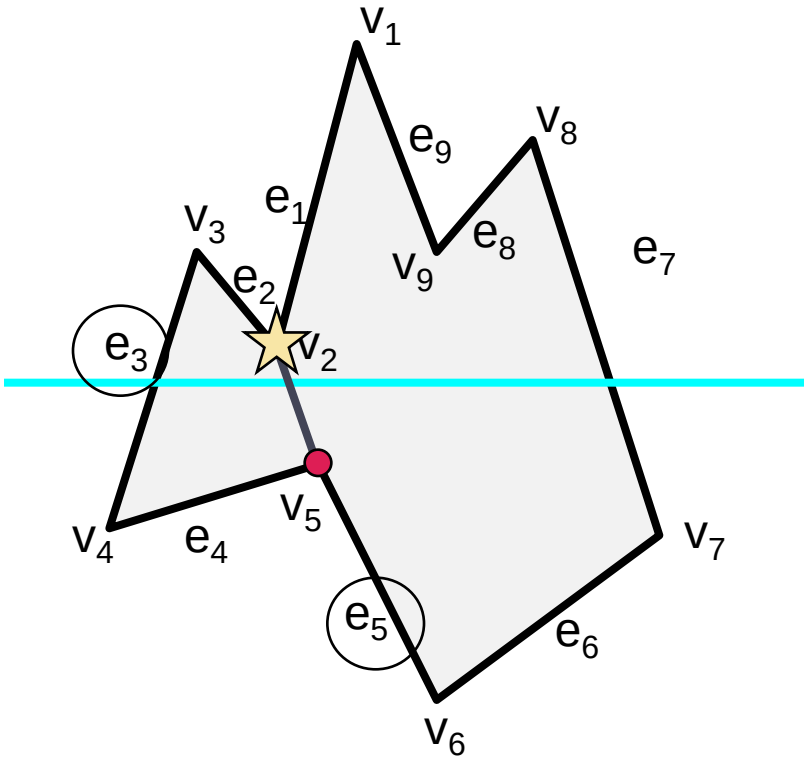
- Se elimina e_{i-1} din L
- Daca *ajutor*(e_{i-1}) este un varf de unire, atunci se conecteaza v_i cu *ajutor*(e_{i-1})



Partitionarea monotona: varful de despartire

Tratarea unui varf de despartire v_i:

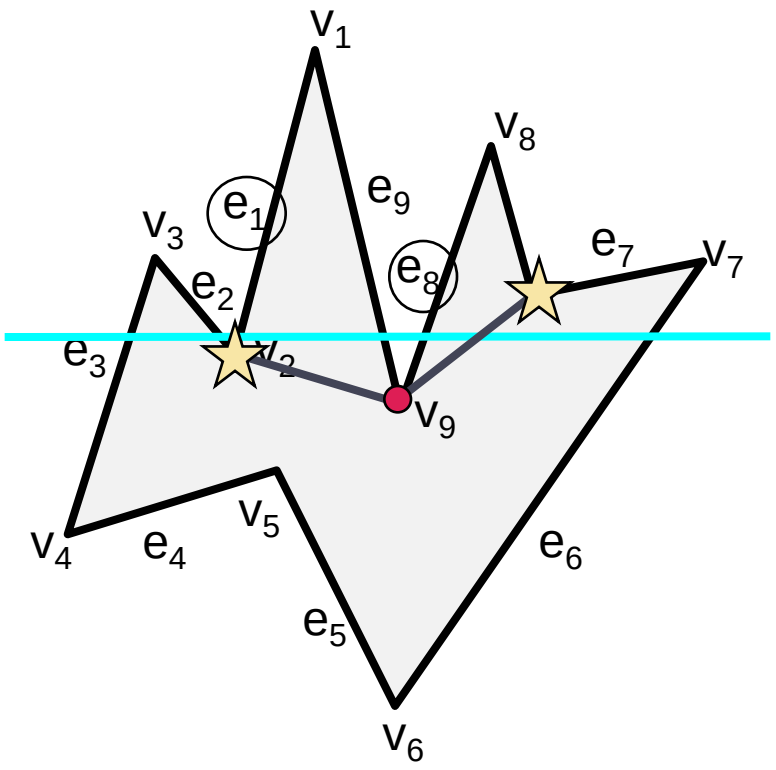
- Se insereaza e_i in L
- Se gaseste in L muchia e_j aflata direct la stanga lui v_i
- Se conecteaza v_i cu $ajutor(e_j)$
- $ajutor(e_j) \leftarrow v_i$
- $ajutor(e_i) \leftarrow v_i$



Partitionarea monotona: varful de unire

Tratarea varfului de unire v_i :

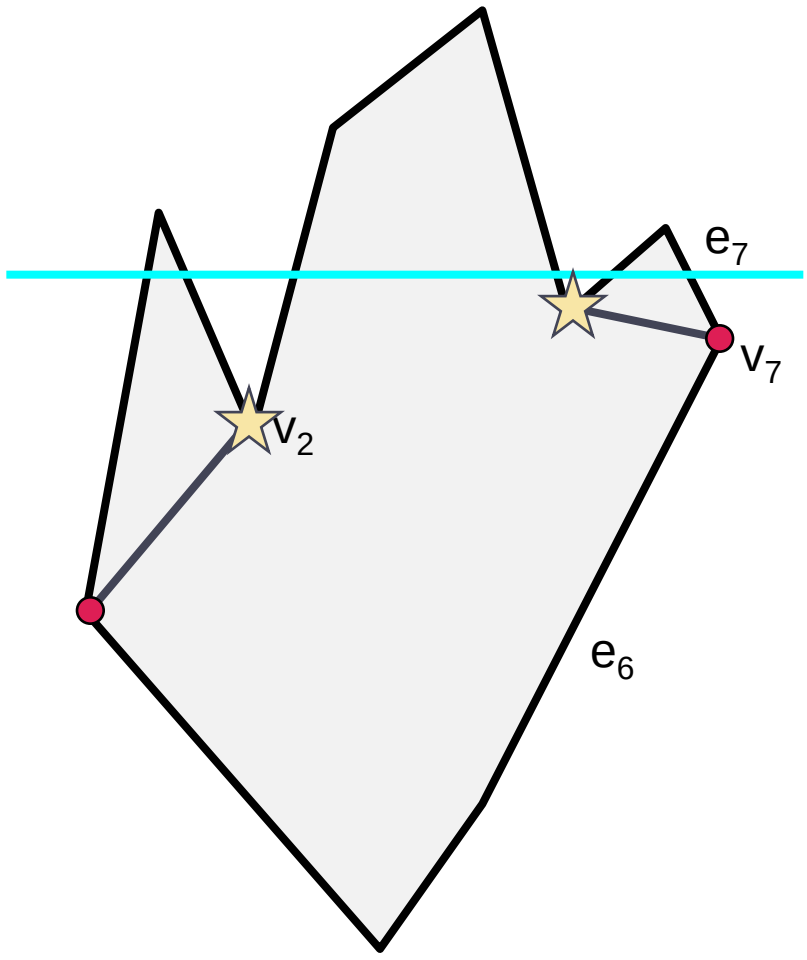
- Daca *ajutor*(e_{i-1}) este un varf de unire, atunci se leaga v_i cu *ajutor*(e_{i-1})
- Se elimina e_{i-1} din L
- Se gaseste in L muchia e_j aflata direct la stanga lui v_i
- Daca *ajutor*(e_j) este un varf de unire, atunci se leaga v_i de *ajutor*(e_j)
- *ajutor*(e_j) ← v_i



Partitionarea monotona: varful normal

Tratarea varfului normal v_i:

- **if** interiorul poligonului se afla la stanga lui v_i **then**
  - Se gaseste in L muchia e_j aflata direct la stanga lui v_i
  - Daca *ajutor*(e_j) este un varf de unire, atunci se leaga v_i de *ajutor*(e_j)
  - *ajutor*(e_j) ← v_i
- **else**
  - Daca *ajutor*(e_{i-1}) este un varf de unire, se leaga v_i de *ajutor*(e_{i-1})
  - Se elimina e_{i-1} din L
  - Se insereaza e_i in L
  - *ajutor*(e_i) ← v_i



Algorithm MAKEMONOTONE($\mathcal{P}$)

Input. A simple polygon $\mathcal{P}$ stored in a doubly-connected edge list $\mathcal{D}$.

Output. A partitioning of $\mathcal{P}$ into monotone subpolygons, stored in $\mathcal{D}$.

1. Construct a priority queue Q on the vertices of $\mathcal{P}$, using their y-coordinates as priority. If two points have the same y-coordinate, the one with smaller x-coordinate has higher priority.
2. Initialize an empty binary search tree $\mathcal{T}$.
3. **while** Q is not empty
4. **do** Remove the vertex v_i with the highest priority from Q.
5. Call the appropriate procedure to handle the vertex, depending on its type.

HANDLESTARTVERTEX(v_i)

1. Insert e_i in $\mathcal{T}$ and set $helper(e_i)$ to v_i.

HANDLEENDVERTEX(v_i)

1. **if** $helper(e_{i-1})$ is a merge vertex
2. **then** Insert the diagonal connecting v_i to $helper(e_{i-1})$ in $\mathcal{D}$.
3. Delete e_{i-1} from $\mathcal{T}$.

HANDLESPLITVERTEX(v_i)

1. Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
2. Insert the diagonal connecting v_i to $helper(e_j)$ in $\mathcal{D}$.
3. $helper(e_j) \leftarrow v_i$
4. Insert e_i in $\mathcal{T}$ and set $helper(e_i)$ to v_i.

HANDLEMERGEVERTEX(v_i)

1. **if** *helper*(e_{i-1}) is a merge vertex
2. **then** Insert the diagonal connecting v_i to *helper*(e_{i-1}) in $\mathcal{D}$.
3. Delete e_{i-1} from $\mathcal{T}$.
4. Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
5. **if** *helper*(e_j) is a merge vertex
6. **then** Insert the diagonal connecting v_i to *helper*(e_j) in $\mathcal{D}$.
7. *helper*(e_j) $\leftarrow v_i$

HANDLEREGULARVERTEX(v_i)

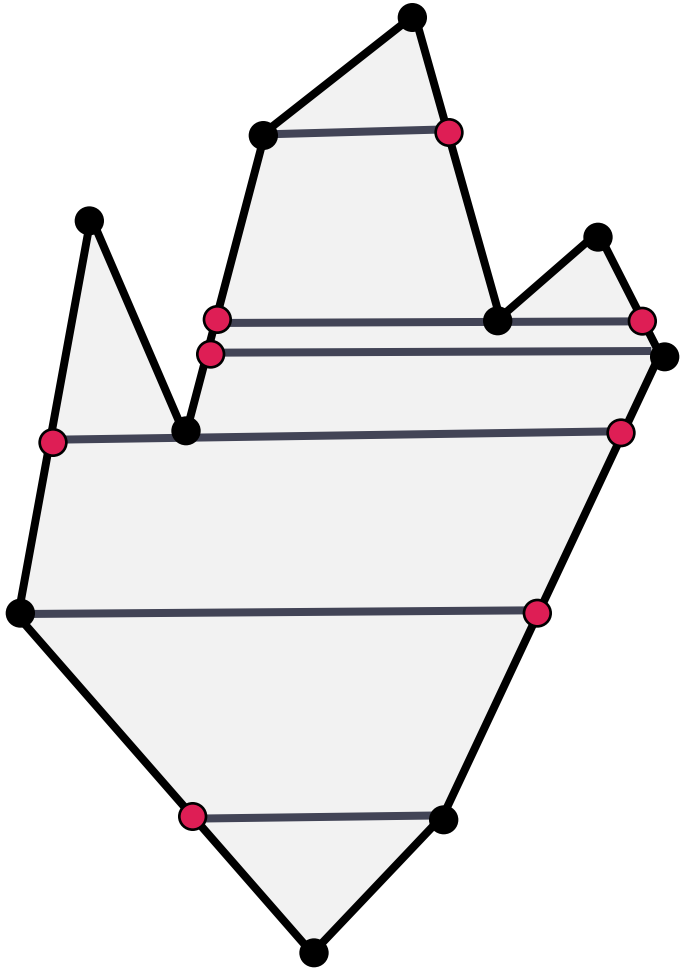
1. **if** the interior of $\mathcal{P}$ lies to the right of v_i
2. **then if** *helper*(e_{i-1}) is a merge vertex
3. **then** Insert the diagonal connecting v_i to *helper*(e_{i-1}) in $\mathcal{D}$.
4. Delete e_{i-1} from $\mathcal{T}$.
5. Insert e_i in $\mathcal{T}$ and set *helper*(e_i) to v_i.
6. **else** Search in $\mathcal{T}$ to find the edge e_j directly left of v_i.
7. **if** *helper*(e_j) is a merge vertex
8. **then** Insert the diagonal connecting v_i to *helper*(e_j) in $\mathcal{D}$.
9. *helper*(e_j) $\leftarrow v_i$

Complexitate

- Partitionarea in poligoane monotone:
  $O(n \log n)$
- Triangularea poligoanelor monotone:
  $\sum T(k_i) = T(\sum k_i) = T(n) \rightarrow O(n)$
- **Total:** ***O(n log n)***

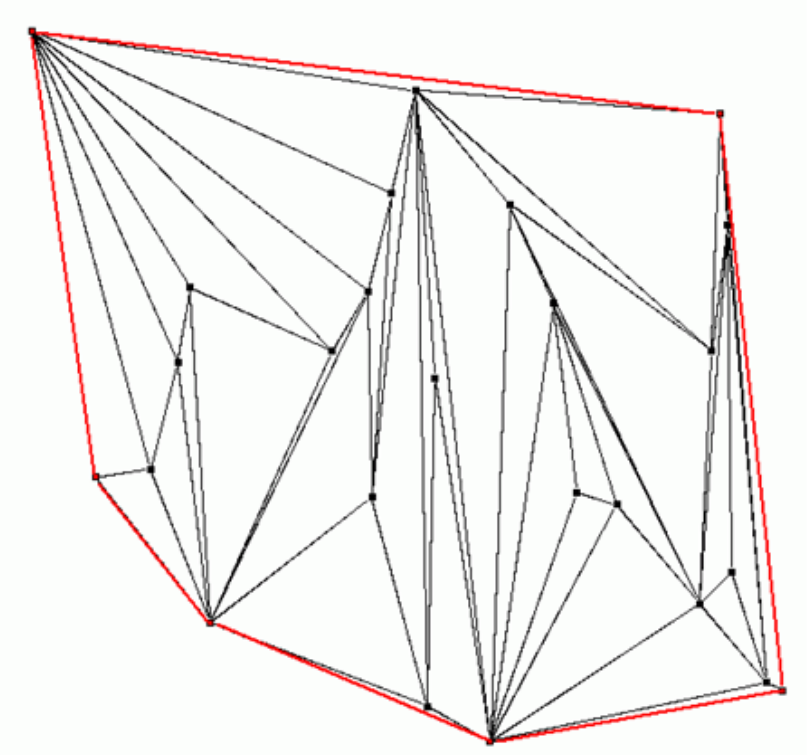
Alte operatii: Descompunerea trapezoidala

- Se descompune un poligon in trapezoide avand doua muchii perpendiculare pe o dreapta data.
- Trapezoidele sunt usor triangulabile (poligoane cu 4 laturi).
- **Sunt necesare noi puncte intermediare**
- Abordare utila pentru cautarea punctelor si alte sarcini.
- Se obtine direct folosind o linie de cautare
  - La fiecare varf, se gasesc noile puncte de suport la dreapta si la stanga
  - Se inchid trapezoidele in functie de relatiile in vecinatate.



Alte operatii: Triangularea seturilor de puncte (1)

- O triangulare *T* a unui set de puncte *V* reprezinta o multime de triunghiuri astfel incat:
  - Orice *t* din *T* atinge 3 puncte din *V* si nu contine alte puncte
  - *t1* si *t2* nu se intersecteaza, oricare ar fi *t1* si *t2* din *T*
  - *T* este o decompozitie a infasuratorii convexe a lui *V*



Alte operatii:
Triangularea seturilor de puncte (2)

- Triangularea poate fi efectuata punct cu punct.
- Exista 3 cazuri
  - Un nou punct aflat la exteriorul infasuratorii convexe curente
  - Un nou punct aflat in interiorul unui triunghi
  - Un nou punct aflat pe muchia triangularii curente

Alte operatii: Partitionarea convexa

Problema: partitionarea poligonului P in cel mai mic numar de componente convexe

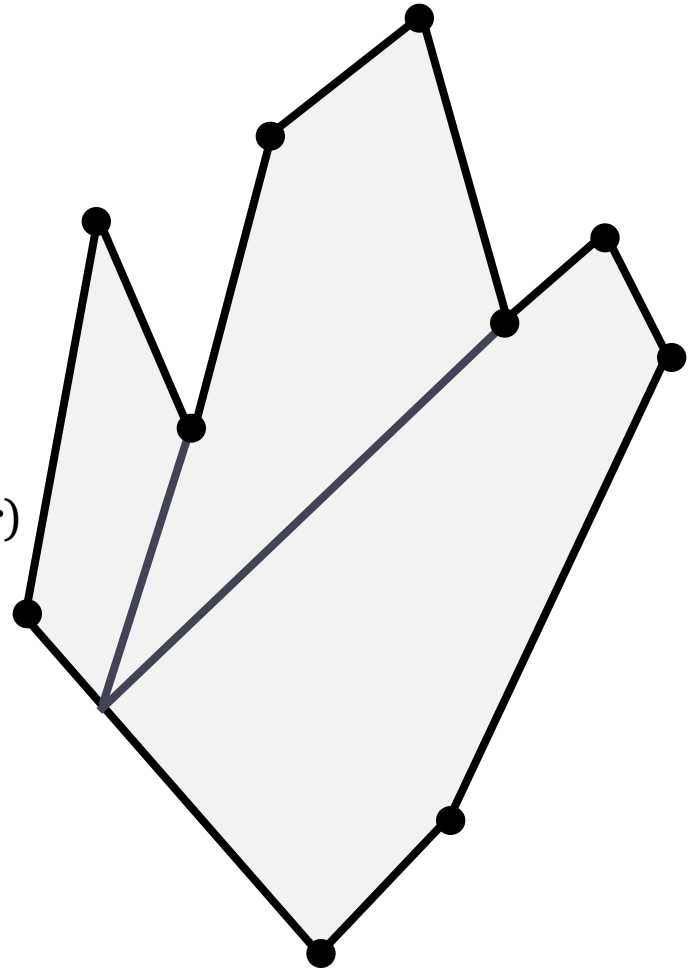
Posibilitati:

- Folosind numai varfuri din P
- Folosind varfuri aflate pe muchiile din P
- Folosind puncte noi interne lui P (puncte Steiner)

Teorema: Fie λ numarul cel mai mic de componente convexe in care se poate partitiona P. Atunci:

$$\lceil r/2 \rceil + 1 \leq \lambda \leq r + 1$$

unde r este numarul de varfuri concave din P



Algoritmul Hertl-Melhorn pentru partitionare convexa

Idee: se trianguleaza poligonul, apoi se inlatura diagonalele “neesentiale”. O diagonala este neesentiala daca inlaturarea sa nu creaza non-convexitati.

Nota: doar varfurile concave pot avea diagonale esentiale!

Lema: un varf concav poate fi capatul a maxim doua diagonale esentiale

Complexitate: $O(n \log n)$ pentru triangulare

Nota: strategia nu este optima!

Teorema: Numarul de componente nu este niciodata mai mare de 4 ori cat pentru partitionarea optima.