Triangulation

Christian Nguyen

Département d'informatique
Université de Toulon

Introduction

Décomposition d'une région polygonale du plan en triangles dont les sommets sont ceux du bord de la région.

Une triangulation permet souvent de résoudre plus facilement des problèmes portant sur la région qu'elle triangule.

Le problème du gardiennage d'une galerie d'art en est un bel exemple : « Quel est le nombre de gardiens nécessaires pour surveiller une galerie d'art, et où faut-il les placer ? ».

Formalisme

Victor Klee (1973)

Le sol de la galerie d'art a la forme d'un polygone, et les gardiens (ou caméras) sont à des positions fixes et peuvent regarder dans toutes les directions.

Géométriquement, soit P un polygone du plan et $x \in \bar{P}$ un point intérieur à P. La *zone de visibilité* V_P de x dans P est l'ensemble des points intérieurs à P et visibles depuis x dans P :

$$V_P(x) = \{y \in \bar{P} / [xy] \subset \bar{P}\}, \ [xy] \text{ segment joignant } x \text{ et } y$$

Un ensemble $X \subset P$ de taille minimale *couvre* P si

$$P = \bigcup_{x \in X} V_P(x)$$

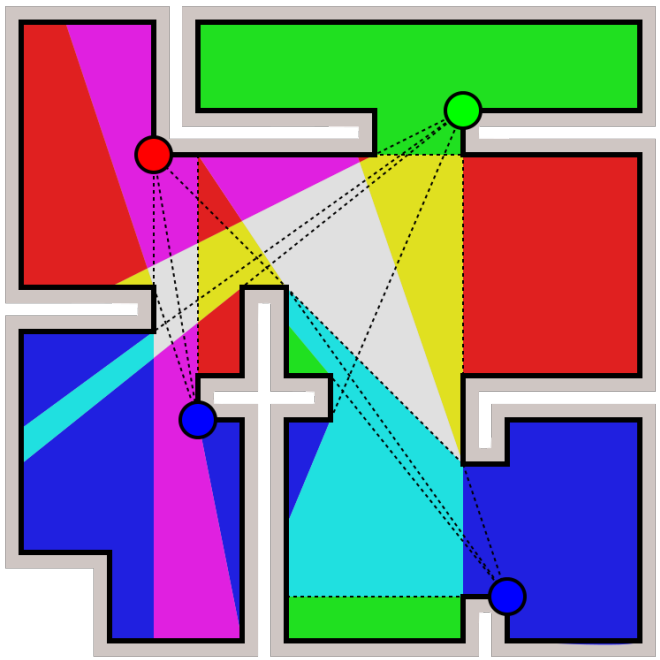
i.e. si l'union des zones de visibilités des points de X recouvre P.

Premières constatations

Le nombre de gardiens dépend de la complexité du polygone, en particulier du nombre de sommets n.

Un polygone convexe peut toujours être couvert par un seul gardien.

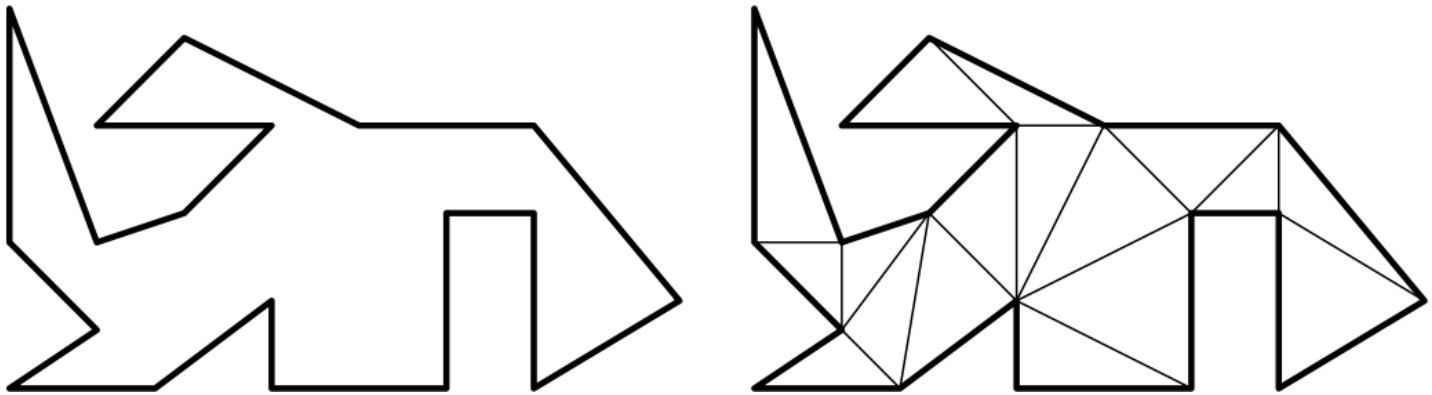
Le problème de trouver le nombre minimum de caméras pour un polygone quelconque est NP-difficile.



Décomposition en régions convexes

Soit P un polygone *simple* de n sommets, sa forme peut être complexe mais on peut le décomposer en régions convexes, en particulier en triangles.

Cela peut se faire automatiquement en créant des segments intérieurs entre paires de sommets.



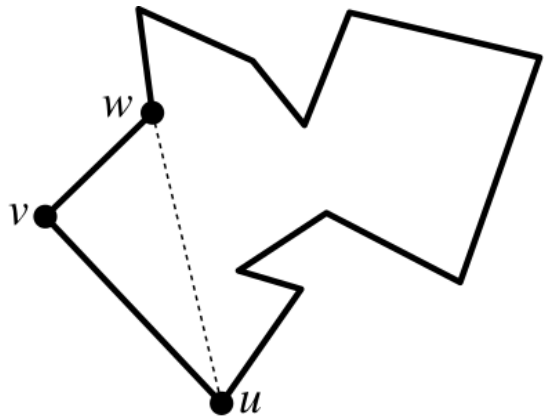
Cette triangulation n'est pas unique. Peut-elle ne pas être possible ?

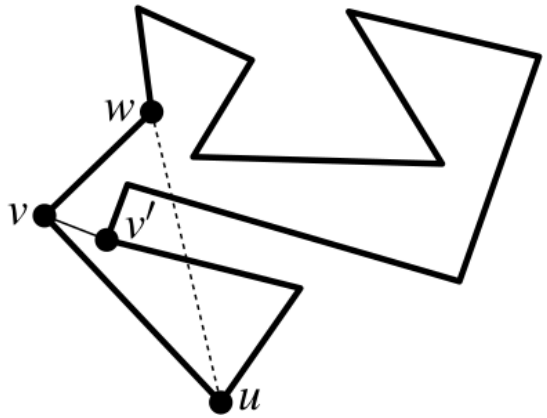
Existence de la triangulation

Tout polygone simple P à n sommets admet une triangulation $\mathcal{T}_P$ constituée de $n-2$ triangles.

Preuve : pour $n = 3$ la propriété est triviale ; pour $n > 3$, supposons la propriété vraie pour tout $m < n$ et vérifions la propriété pour n.

On doit prouver l'existence d'une diagonale dans P. Soit v le sommet le plus à gauche de P (et le plus bas en cas de non unicité). Soit u et w les deux sommets voisins de v sur la frontière de P.





Existence de la triangulation

- si le segment ouvert $\overline{uw}$ est à l'intérieur de P, on a trouvé une diagonale,
- sinon il existe un ou plusieurs sommets dans le triangle u, v, w ou sur la diagonale $\overline{uw}$, soit v' le sommet le plus éloigné de $\overline{uw}$, le segment $\overline{vv'}$ n'intersecte pas le contour de P (car sinon, ce segment aurait un sommet plus éloigné de $\overline{uw}$, ce qui est contradictoire).

Puisqu'une diagonale existe, elle partage le polygone P en deux sous-polygones P_1 et P_2, comportant respectivement m_1 et m_2 sommets, tous deux inférieurs à n. Donc, par induction, P_1 et P_2 peuvent être triangulés.

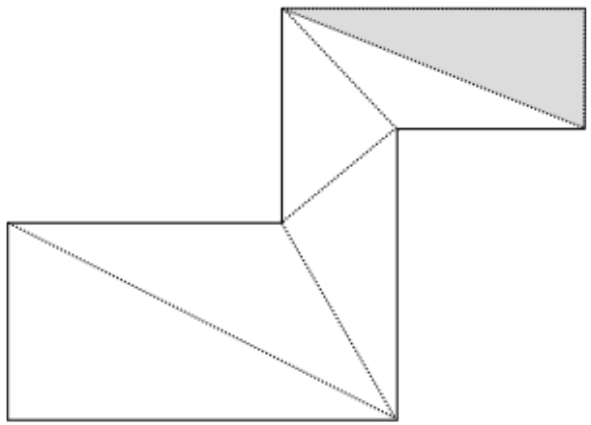
Il reste à prouver que toute triangulation $\mathcal{T}_P$ est constituée de $n-2$ triangles.

Triangulation en $n-2$ triangles

Une diagonale subdivise un polygone P en deux sous-polygones P_1 et P_2 comportant respectivement m_1 et m_2 sommets.

Chaque sommet de P appartient soit à P_1 soit à P_2, sauf les deux sommets de la diagonale qui appartiennent à la fois à P_1 et P_2.

Donc $m_1 + m_2 = n + 2$. Par induction, toute triangulation de P_i est constituée de $m_i - 2$ triangles et la triangulation de P est constituée de $(m_1 - 2) + (m_2 - 2) = n - 2$ triangles.

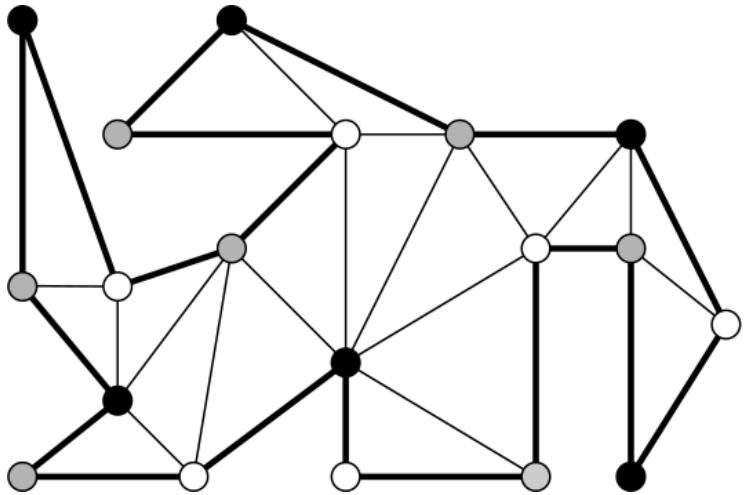


Triangulation et gardiens

Un polygone P comportant $n-2$ triangles peut être couvert par $n-2$ gardiens.

Cela n'est pas optimal. Un gardien peut être placé sur la diagonale de deux triangles, ce qui permet de réduire leur nombre à $n/2$.

Cela n'est pas optimal. On peut placer les gardiens aux sommets des triangles, car un sommet est incident à plusieurs triangles.



Triangulation et gardiens

Chaque triangle doit avoir l'un de ses sommets sélectionné.

Pour garantir cette propriété, on attribue à chaque sommet du polygone P une couleur choisie parmi trois (par exemple : noir, blanc et gris).

Cette colorisation doit être telle que toute paire de sommets reliés soit par un segment du contour ou d'une diagonale doit être de deux couleurs différentes.

C'est le *3-coloriage* d'un polygone simple triangulé.

En plaçant un gardien sur chaque sommet de la même couleur (par exemple gris), on couvre le polygone. En choisissant la couleur qui minimise le nombre de gardiens, la couverture comporte au maximum $\lfloor n/3 \rfloor$ gardiens.

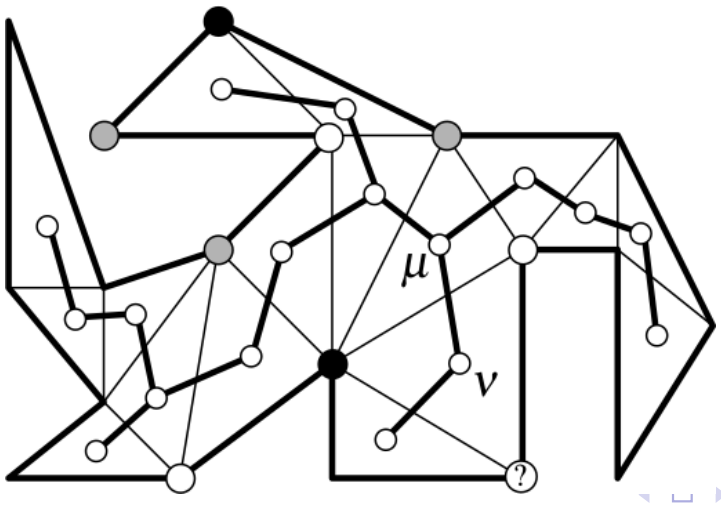
3-coloriage

Le 3-coloriage est-il toujours possible ?

Soit $\mathcal{G}(\mathcal{T}_P)$ le graphe dual de $\mathcal{T}_P$. Celui-ci comporte un nœud pour chaque triangle de $\mathcal{T}_P$ et les arcs de $\mathcal{G}(\mathcal{T}_P)$ correspondent aux diagonales de $\mathcal{T}_P$.

On a démontré qu'une diagonale subdivise P en deux, donc supprimer un arc de $\mathcal{G}(\mathcal{T}_P)$ divise le graphe en deux.

- $\mathcal{G}(\mathcal{T}_P)$ est un arbre binaire.



3-coloriage

Le 3-coloriage est basé sur l'exploration d'un arbre binaire, en profondeur d'abord par exemple.

Invariant : les sommets des triangles déjà traités sont noir, blanc ou gris et deux sommets connectés sont de couleur différente.

Depuis un nœud quelconque de $\mathcal{G}(\mathcal{T}_P)$, les trois sommets du triangle correspondant prennent les trois couleurs.

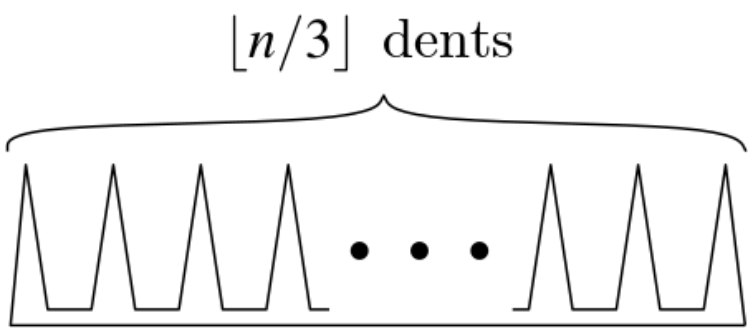
On traite un nœud ν, fils d'un nœud μ. Les triangles correspondants partage une arête donc deux sommets, ainsi seul le troisième sommet du triangle de ν doit être coloré de la couleur non choisie (ce qui est possible car les nœuds suivants n'ont pas encore été traité).

$\lfloor n/3 \rfloor$ gardiens

Un polygone simple peut être couvert par $\lfloor n/3 \rfloor$ gardiens.

Peut-on faire mieux ? Un gardien placé à un sommet peut sans doute couvrir plus que les triangles incidents.

Malheureusement, il existe des polygones à n sommets qui nécessitent au mieux $\lfloor n/3 \rfloor$ gardiens.



Il n'y a donc pas de procédé automatique aboutissant à moins de $\lfloor n/3 \rfloor$ gardiens.

- le 3-coloriage est optimal dans le cas le pire.

Triangulation

L'algorithme doit prendre en compte le fait qu'il traite un polygone simple et doit produire une structure donnée adaptée (recherche en temps constant des voisins d'un triangle par exemple).

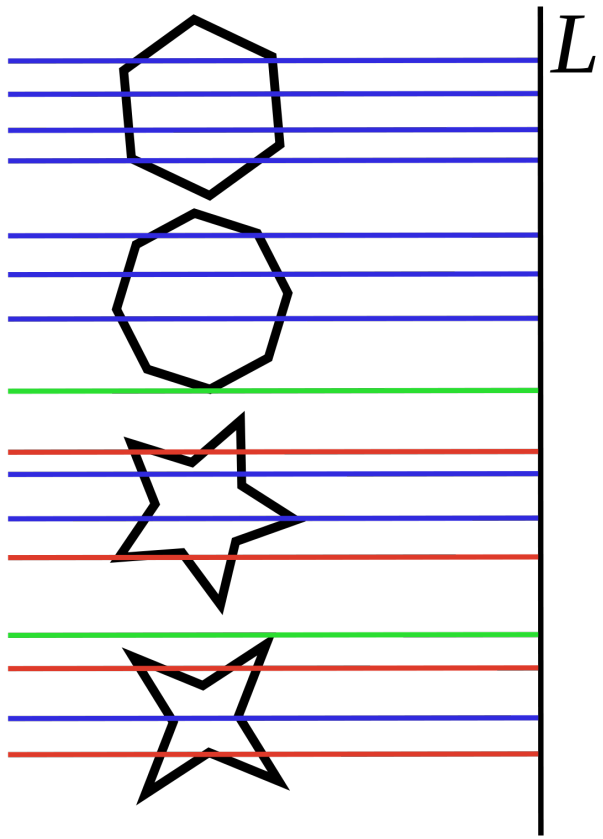
Trouver une diagonale se fait en temps linéaire. On peut considérer que l'on obtient ainsi un triangle et un sous-polygone à $n-1$ sommets. La triangulation est donc quadratique dans le pire des cas.

Pour les polygones *convexes*, l'algorithme est linéaire : il suffit de choisir un sommet et de construire des diagonales avec les autres sommets (sauf ses deux voisins).

Décomposer un polygone simple en sous-polygones convexes est difficile. Cela l'est moins s'ils sont *monotones*.

Polygone monotone

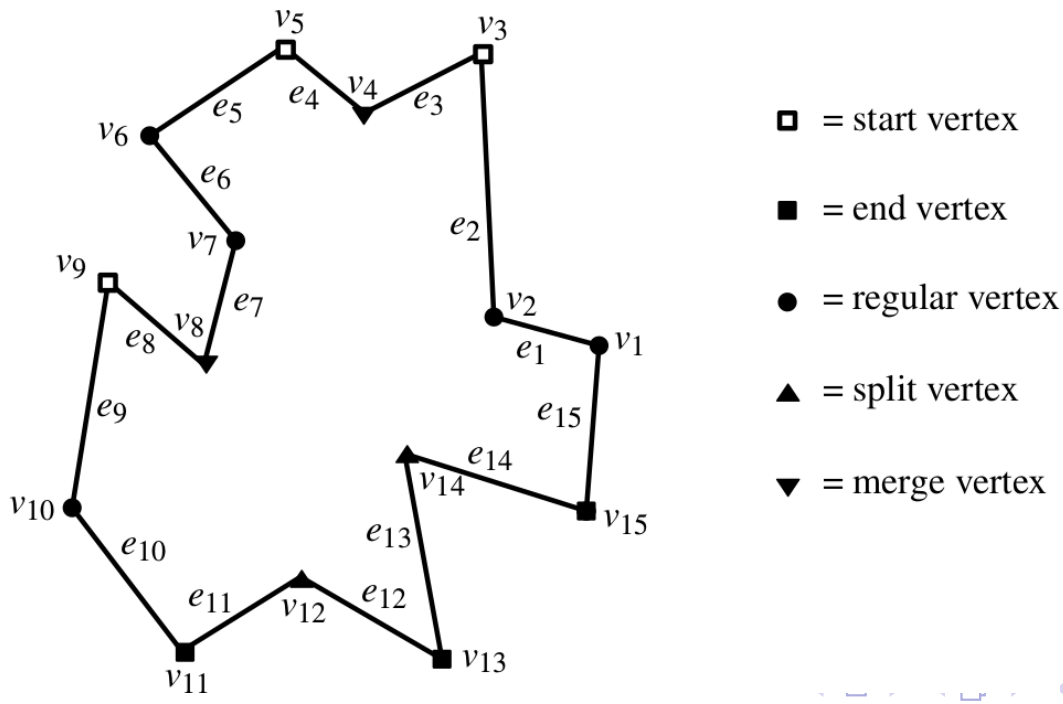
Un polygone monotone est tel que sa frontière peut être divisée en deux parties, chacune d'entre elles étant composée de points dont les coordonnées selon une dimension donnée ne font que croître.



Partitionnement en sous-polygones monotones

Parcours du contour de P d'un côté d'une droite de séparation, du sommet d'ordonnée maximale au sommet d'ordonnée minimale.

Un sommet qui change le sens de parcours (haut $\leftrightarrow$ bas) est qualifié de *tournant*. C'est à ce niveau qu'un traitement doit avoir lieu. Il existe plusieurs types de ces sommets.



Partitionnement en sous-polygones monotones

Les sommets `start`, `split`, `end` et `merge` sont des *virages* :

- □ `start` : voisins d'ordonnées inférieures et angle intérieur inférieur à π,
- ▲ `split` : idem `start` mais d'angle supérieur à π,
- ■ `end` : voisins d'ordonnées supérieures et angle intérieur inférieur à π,
- ▼ `merge` : idem `end` mais d'angle supérieur à π.

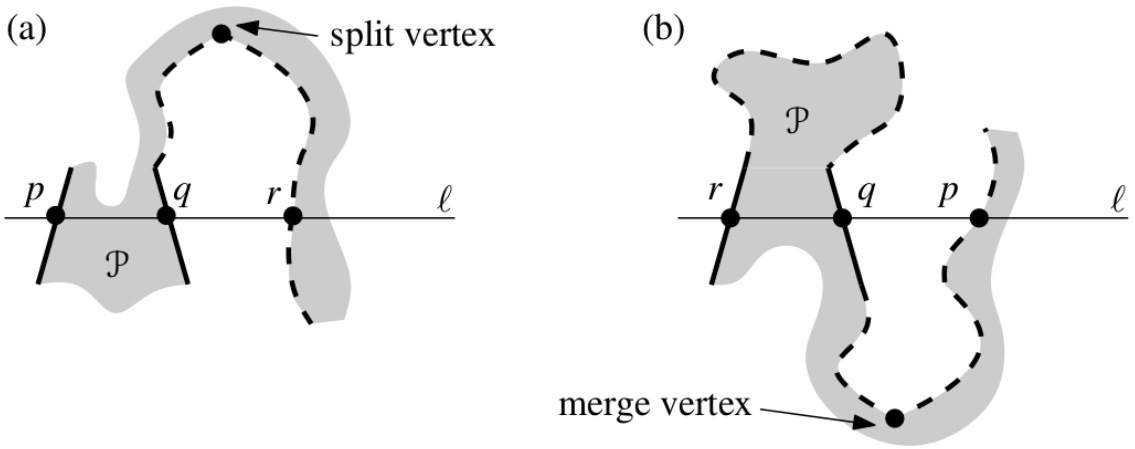
Leur nom vient de leur rôle lors de l'intersection du polygone avec une ligne de balayage.

Les autres sommets (•) sont dit *réguliers*, l'un de ses voisins étant d'ordonnée supérieure et l'autre d'ordonnée inférieure.

Partitionnement en sous-polygones monotones

Un polygone P est y-monotone s'il n'a ni sommet `split` ni sommet `merge`.

Preuve : si P n'est pas y-monotone alors il a au moins un sommet `split` ou `merge`. En effet, cela signifie que la droite de balayage comporte plus de deux intersections et cela met en évidence l'existence de l'un ou l'autre type de sommet.



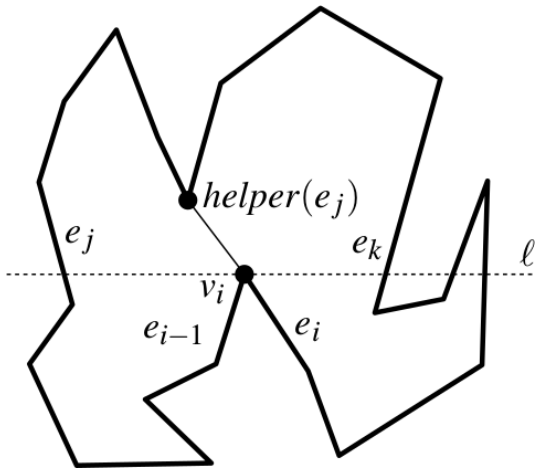
- P peut donc être partitionné en sous-polygones y-monotones en s'appuyant sur les sommets `split` et `merge`.

Partitionnement en sous-polygones monotones

L'ajout de diagonale depuis un sommet `split` se fait grâce à une ligne de balayage (avec queue de priorité).

La diagonale est créée depuis un sommet `split` v_i vers un sommet préalablement rencontré par la ligne de balayage.

Soient e_j (resp. e_k) l'arête gauche (resp. droite) de v_i, *helper*(e_j) est le bon candidat, d'ordonnée minimum entre e_j et e_k au dessus de v_i (ou, s'il n'existe pas, le sommet supérieur de e_j).

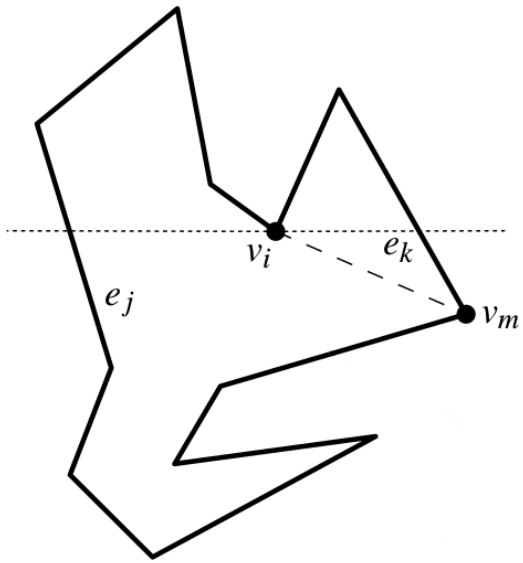


Partitionnement en sous-polygones monotones

L'ajout de diagonale depuis un sommet `merge` semble plus compliqué car il met en jeu un sommet qui n'a pas encore été traité.

Le sommet v_i est le $helper(e_j)$ lorsque la ligne de balayage l'intersecte, donc il faudrait construire la diagonale entre le sommet le plus élevé de e_j et e_k en dessous de la ligne de balayage.

Cela se produit si, quand on rencontre le nouveau $helper(e_j)$ (ici v_m sommet `split`), l'ancien $helper(e_j)$ est un sommet `merge`.



Partitionnement en sous-polygones monotones

créer_monotone(P)

Entrées : un polygone P sous la forme d'une liste doublement chaînée $\mathcal{D}$

Output : $\mathcal{D}$ une partition de P en sous-polygones monotones

construire la queue de priorité Ω des sommets de P suivant y puis x

initialiser à vide un arbre de recherche binaire τ

tant que Ω *n'est pas vide* **faire**

| déterminer le sommet $v_i \in \Omega$ et le supprimer de Ω

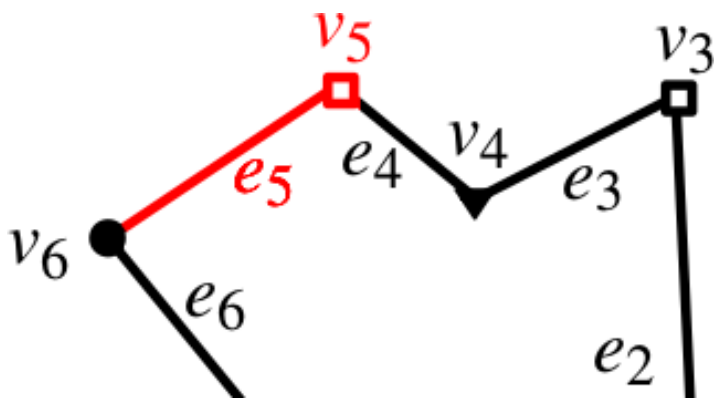
| appeler la procédure appropriée en fonction du type de v_i

fin

Partitionnement en sous-polygones monotones

`sommet_start`(v_i)

insérer e_i dans τ
$helper(e_i) \leftarrow v_i$

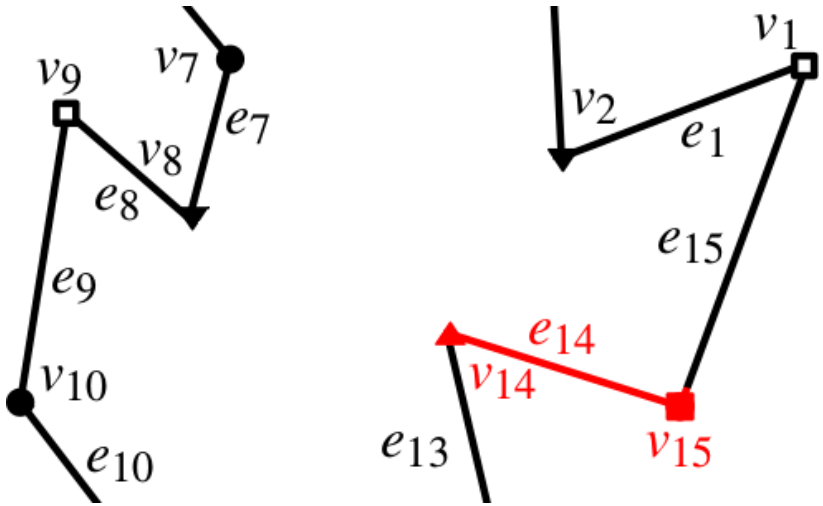


Le sommet v_5 est un sommet `start`, on insère l'arête e_5 dans l'arbre τ.

Partitionnement en sous-polygones monotones

sommet_end(v_i)

si *helper*(e_{i-1}) *est un sommet* `merge` **alors**
  insérer la diagonale (v_i, *helper*(e_{i-1})) dans $\mathcal{D}$
fin
effacer e_{i-1} de τ

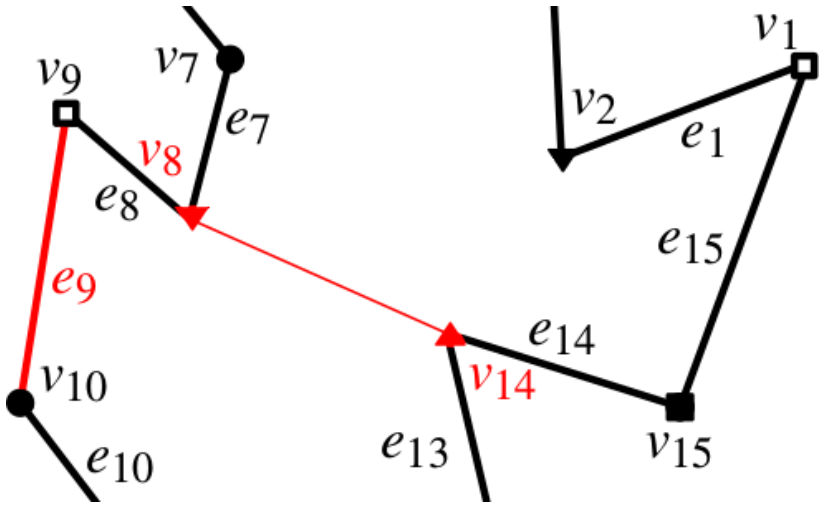


Le *helper* de l'arête e_{14} est v_{14} qui n'est pas un sommet `merge`.

Partitionnement en sous-polygones monotones

`sommet_split`(v_i)

chercher dans τ l'arête e_j immédiatement à gauche de v_i
insérer la diagonale $(v_i, helper(e_j))$ dans $\mathcal{D}$
$helper(e_i) \leftarrow helper(e_j) \leftarrow v_i$
insérer e_i dans τ

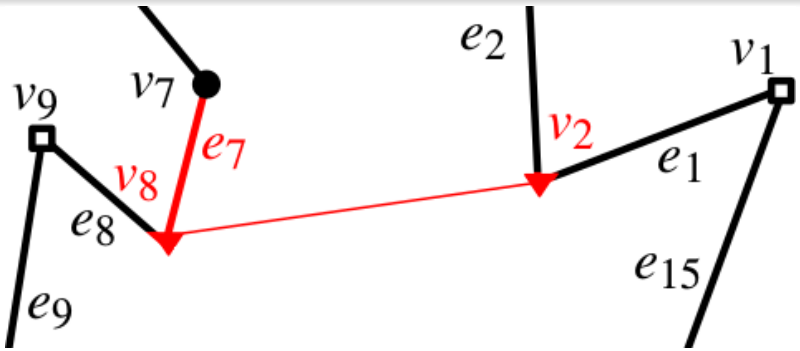


Pour le sommet `split` v_{14}, l'arête à sa gauche est e_9 dont le *helper* est v_8.

Partitionnement en sous-polygones monotones

`sommet_merge`(v_i)

si *helper*(e_{i-1}) *est un sommet* `merge` **alors**
  insérer la diagonale (v_i, *helper*(e_{i-1})) dans $\mathcal{D}$
fin
effacer e_{i-1} de τ
chercher dans τ l'arête e_j immédiatement à gauche de v_i
si *helper*(e_j) *est un sommet* `merge` **alors**
  insérer la diagonale (v_i, *helper*(e_j)) dans $\mathcal{D}$
fin
helper(e_j) $\leftarrow v_i$

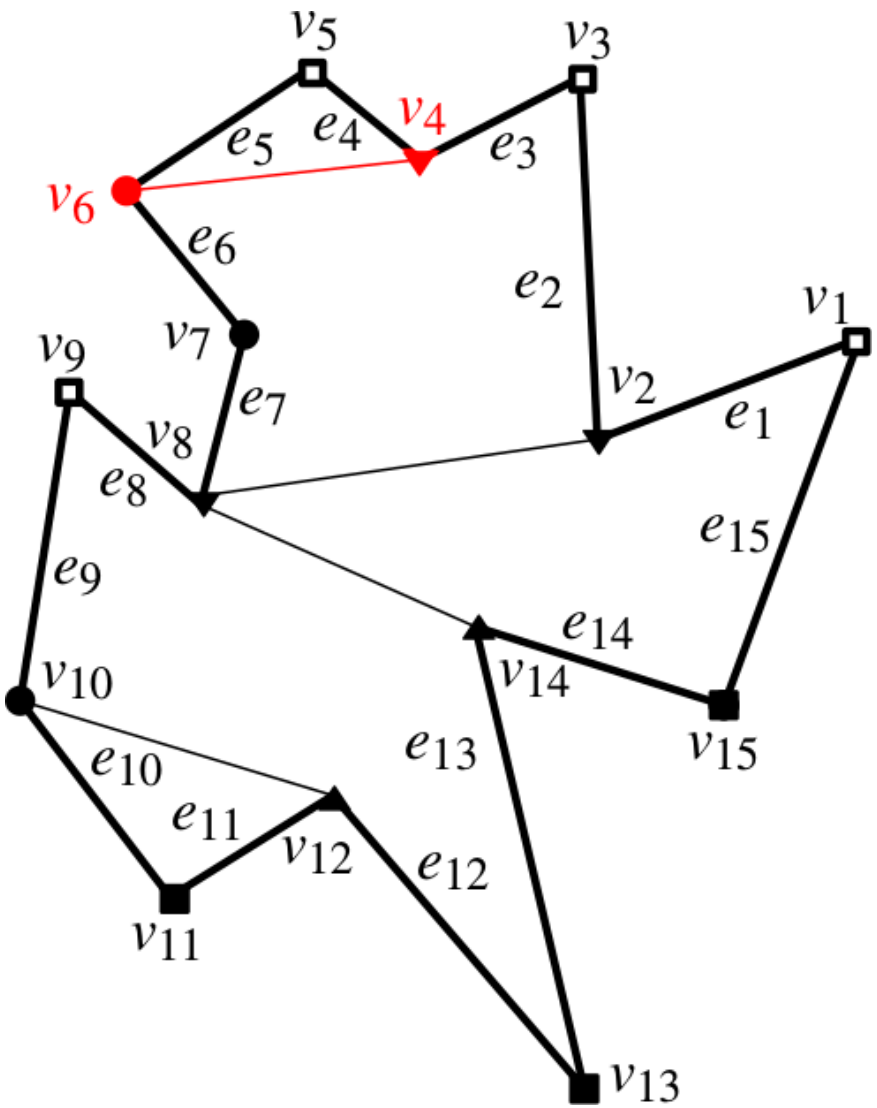


Partitionnement en sous-polygones monotones

sommet_regular(v_i)

si *l'intérieur de P est à droite de* v_i **alors**
  si *helper*(e_{i-1}) *est un sommet* merge **alors**
    insérer la diagonale (v_i, *helper*(e_{i-1})) dans $\mathcal{D}$
    effacer e_{i-1} de τ
    insérer e_i dans τ
    helper$(e_i) \leftarrow v_i$
  fin
sinon
  chercher dans τ l'arête e_j immédiatement à gauche de v_i
  si *helper*(e_j) *est un sommet* merge **alors**
    insérer la diagonale (v_i, *helper*(e_j)) dans $\mathcal{D}$
  fin
  helper$(e_j) \leftarrow v_i$
fin

Partitionnement en sous-polygones monotones



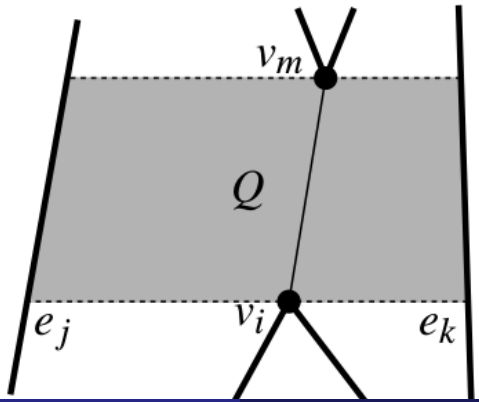
Partitionnement en sous-polygones monotones

Est-ce que l'algorithme partitionne un polygone en sous-régions monotones grâce à un ensemble de diagonales non intersectantes ?

Les sous-régions ne comportent aucun sommet `split` ou `merge`, donc elles sont bien monotones.

Il reste à prouver que les diagonales ajoutées sont valides et qu'elles ne s'intersectent pas.

Considérons l'ajout du segment $\overline{v_m v_i}$ par l'algorithme `sommet_split` (les autres algorithmes se démontrent de façon similaire), avec e_j (resp.e_k) l'arête à gauche (resp. à droite) de v_i.



Partitionnement en sous-polygones monotones

Dans ces conditions, $helper(e_j) = v_m$.

$\overline{v_m v_i}$ n'intersectent aucune arête de P car :

- dans la région Q il n'y a aucun sommet (sinon v_m ne serait pas le helper de e_j),
- aucune arête de P ne traverse la région Q en intersectant $\overline{v_m v_i}$ (sinon e_j ne serait pas l'arête gauche retenue).

Enfin, une diagonale précédemment ajoutée n'ayant aucun sommet dans Q et ayant ses deux sommets au dessus de v_i, elle ne peut intersecter $\overline{v_m v_i}$.

Triangulation d'un polygone monotone

Partitionner un polygone en sous-polygones monotones a un coût en $O(n \log n)$ et permet la triangulation en $O(n)$.

Considérons un polygone strictement y-monotone (sans arête horizontale), dont le contour peut être parcouru, du haut vers le bas, sur deux côtés.

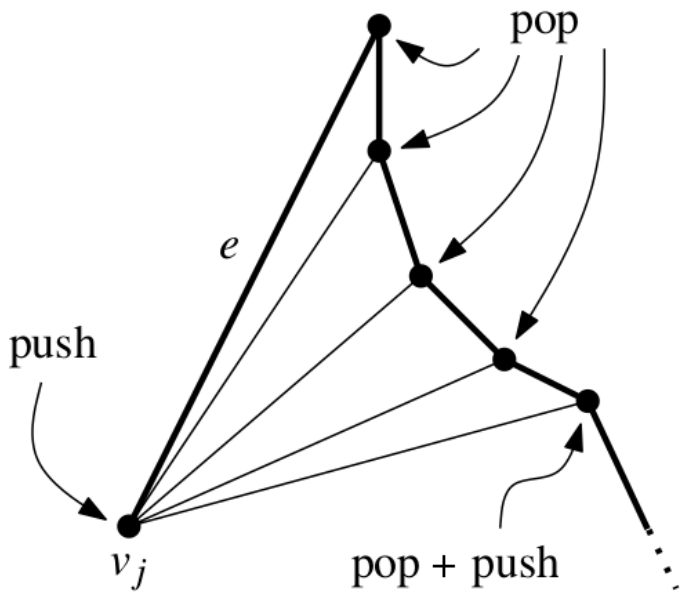
Les sommets sont traités du haut vers le bas (et de gauche à droite). On utilise une pile qui stocke les sommets non traités (i.e les sommets d'un côté qui n'ont pas été reliés à des sommets de l'autre côté). Seul le sommet en bas de la pile est convexe (invariant).

On distingue deux cas : le sommet à traiter est du même côté que les sommets de la pile (sauf un) ou il est de l'autre côté.

Triangulation d'un polygone monotone

Si le sommet v_j à traiter est du côté opposé, c'est le sommet bas du segment e. Par construction, on relie ce sommet à tous les sommets de la pile sauf le premier (il est déjà relié à v_j) et le dernier (en bas de la pile).

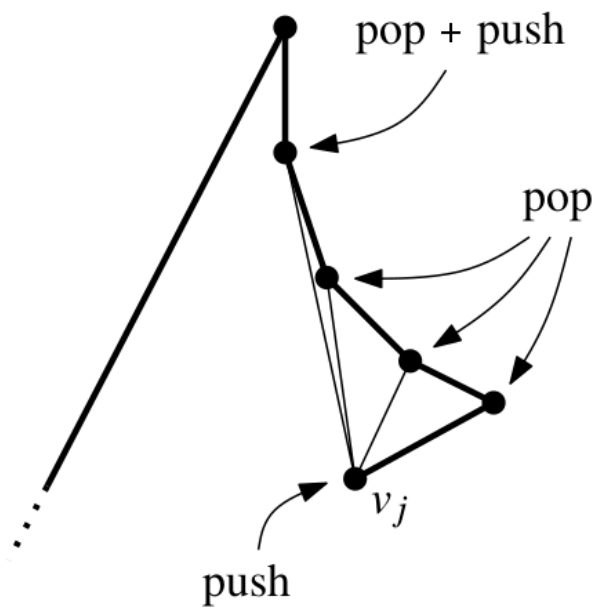
Le dernier sommet de la pile est remis dans la pile ainsi que le sommet v_j.

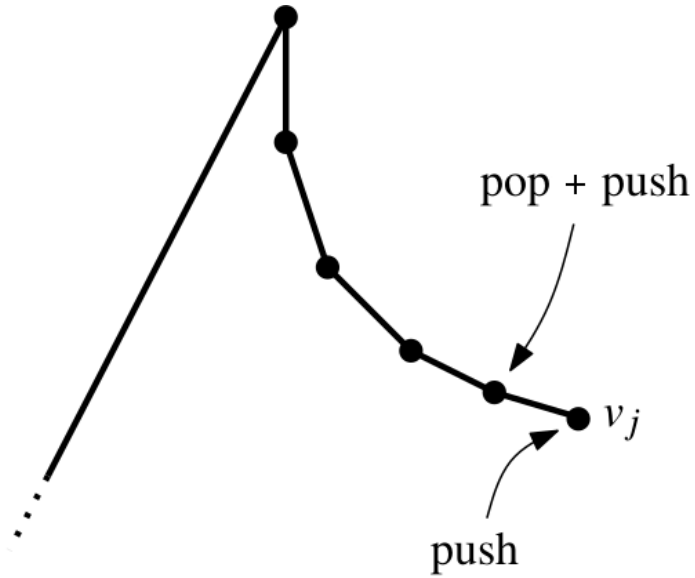


Triangulation d'un polygone monotone

Si le sommet v_j à traiter est du même côté, il ne doit pas être relié à tous les sommets de la pile mais au sous-ensemble de sommets consécutifs du sommet de la pile.

On dépile le premier sommet (il est déjà relié) puis on dépile les sommets suivants en les reliant à v_j jusqu'à rencontrer une impossibilité.





Triangulation d'un polygone monotone

`triangulation_polygone_monotone`(P)

Entrées : polygone P strictement y-monotone sous forme d'une liste doublement chaînée d'arêtes D
Output : une triangulation de P sous forme d'une liste doublement chaînée d'arêtes D
soient $u_1, u_2, \ldots, u_n$ la liste des sommets de P triés par ordonnée décroissante
initialiser une pile S et empiler u_1 et u_2
pour $j \leftarrow 3$ *à* $n-1$ **faire**
    si u_j *et sommet*(S) *sont de côté différent* **alors**
        dépiler tous les sommets de S
        insérer dans D des diagonales de u_j à chaque sommet dépilé (sauf le dernier)
        empiler u_{j-1} et u_j dans S
    sinon
        dépiler un sommet de S
        dépiler des sommets de S tant qu'ils forment avec u_j une diagonale dans P
        insérer ces diagonales dans D
        (r)empiler le dernier sommet dépiler dans S
        empiler u_j dans S
    fin
fin
insérer des diagonales de u_n aux sommets de S sauf le premier et le dernier

Triangulation d'un polygone monotone

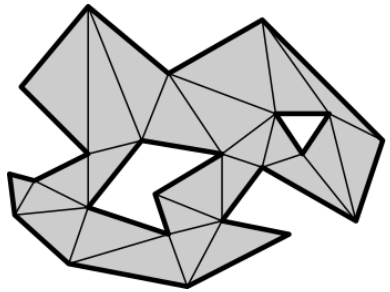
Comment traiter les polygones comportant des arêtes horizontales (ce que produit l'algorithme de décomposition en sous-polygones monotones) ?

Jusqu'ici nous avons trié les sommets ayant même ordonnée de la gauche vers la droite. Cela revient à les considérer suivant leur ordonnée dans un repère incliné dans le sens horaire.

Donc, la décomposition en sous-polygones monotones peut être considérée comme strictement monotone dans ce repère incliné et on peut appliquer l'algorithme de triangulation vu précédemment.

Triangulation

Cet algorithme est capable de trianguler un polygone troué.



Il est même capable de partitionner un ensemble de segments en triangles.