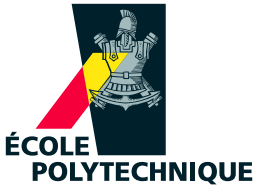
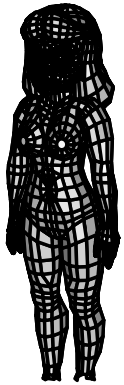
Algorithmes géométriques (INF562)

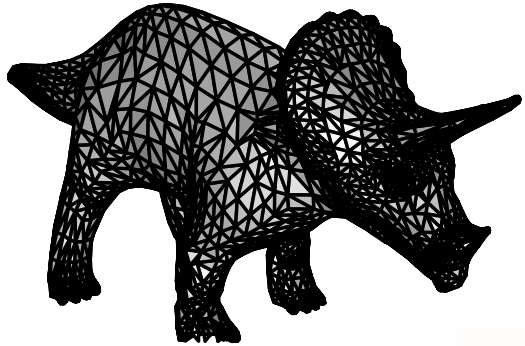
Amphi 1, mardi 3 janvier 2012

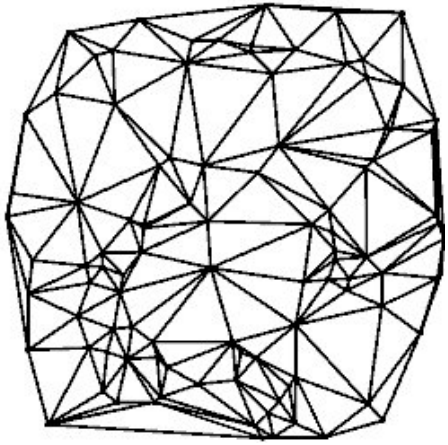
Luca Castelli Aleardi

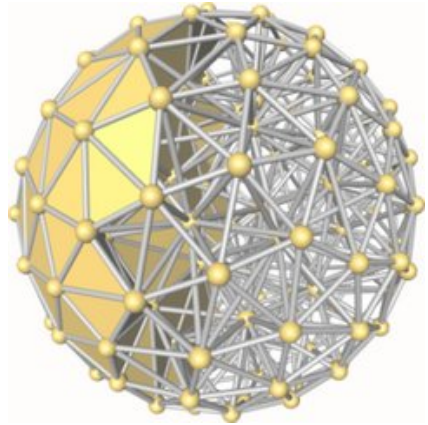


Données structurées de nature géométrique





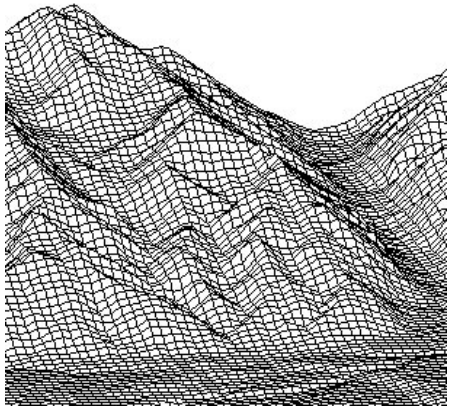




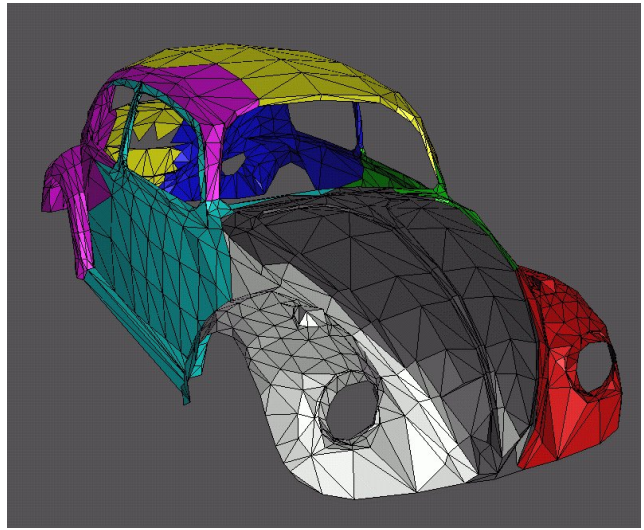
Domaines d'application

Recontruction
de surfaces

GIS Technology



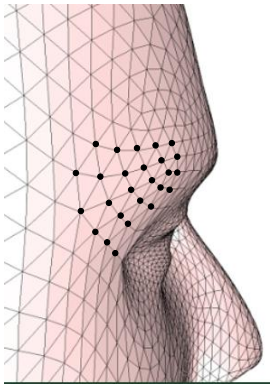
Modélisation géométrique

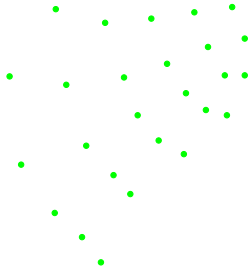


Quel type d'information?

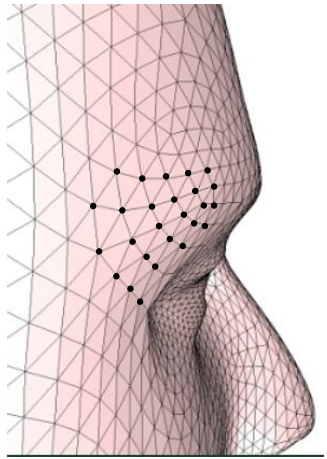
Géométrie et connectivité

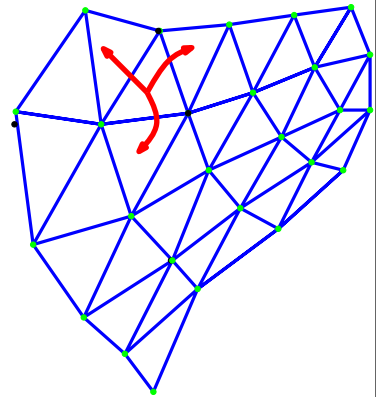
Objet géométrique



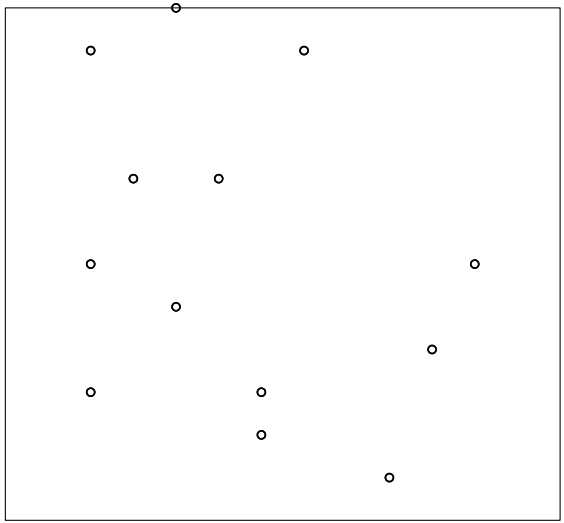


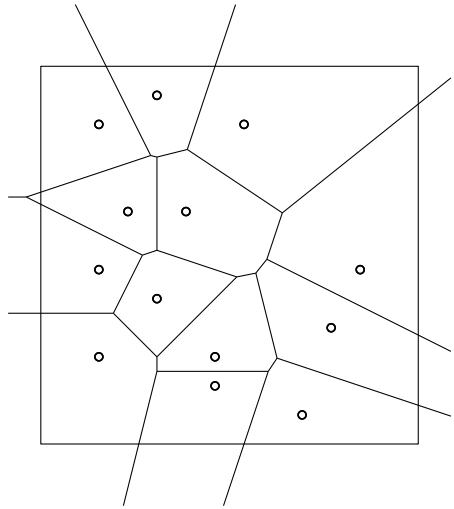
Objet combinatoire

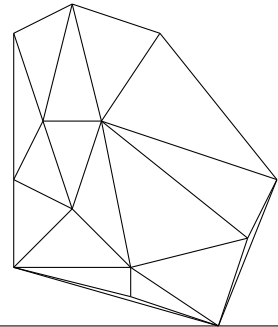




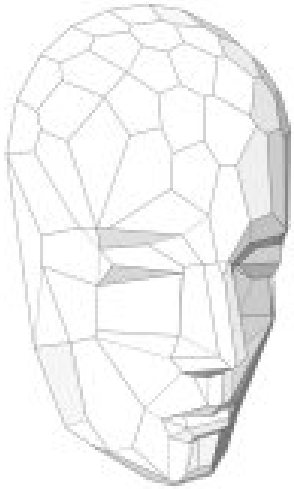
On calcule de la combinatoire

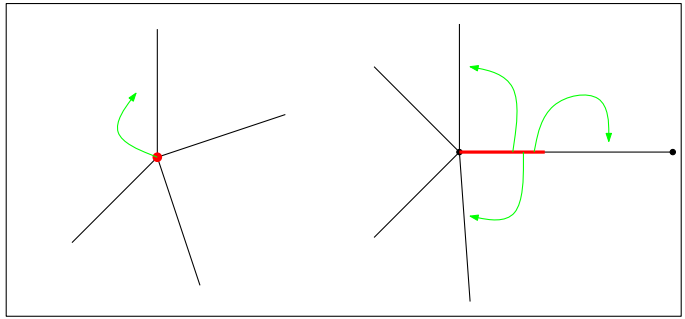


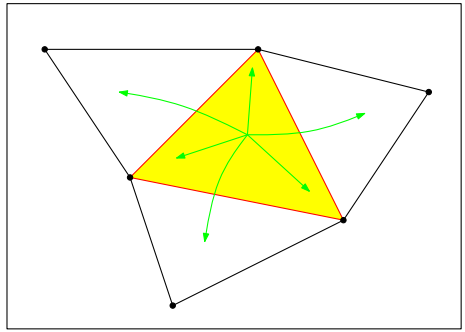




Structures de données géométriques



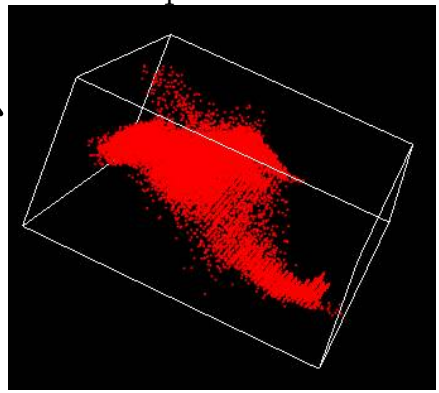




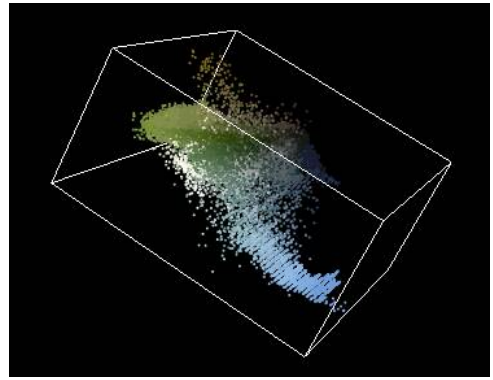
Mean-Shift clustering et segmentations d'images

image originale (50700 pixels)

50700 points en 3D
espace *Luv*

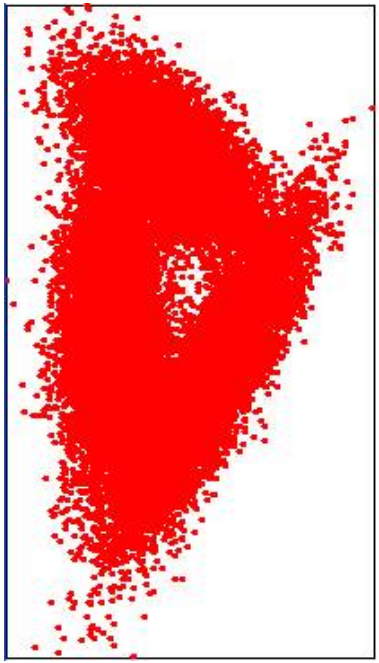


clustering



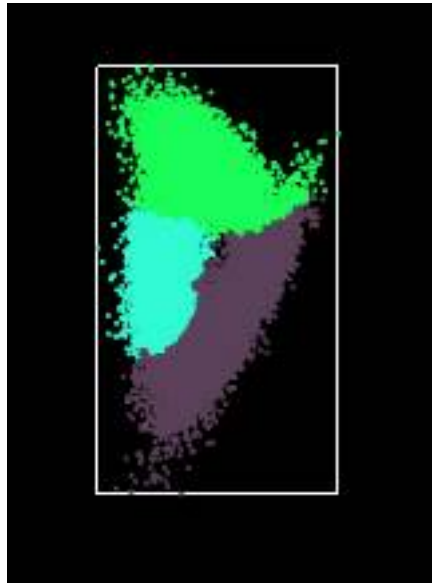
Clustering: en deux mots

n points (exemple en 2D)

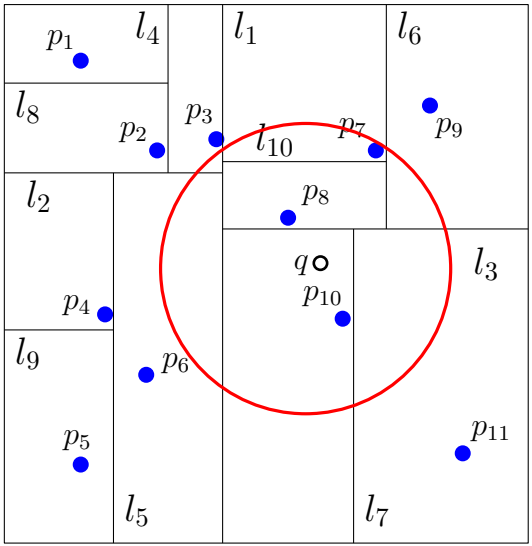


clustering

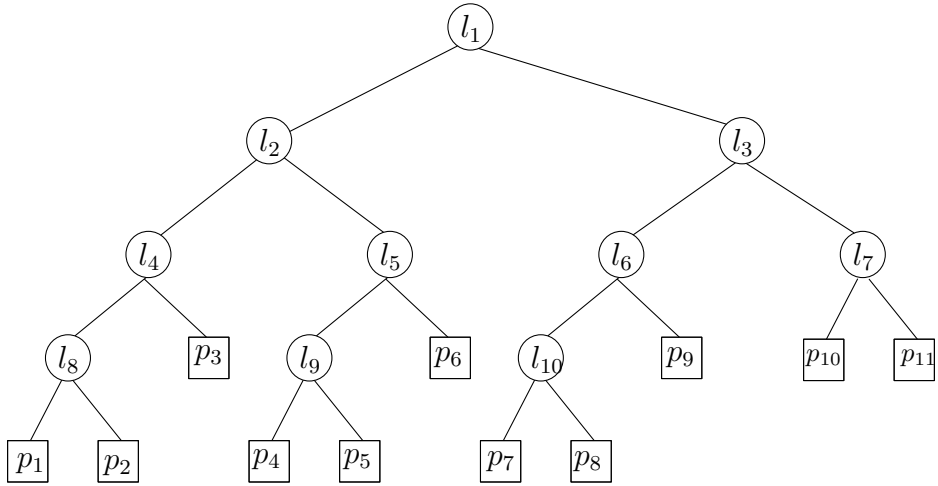
3 clusters



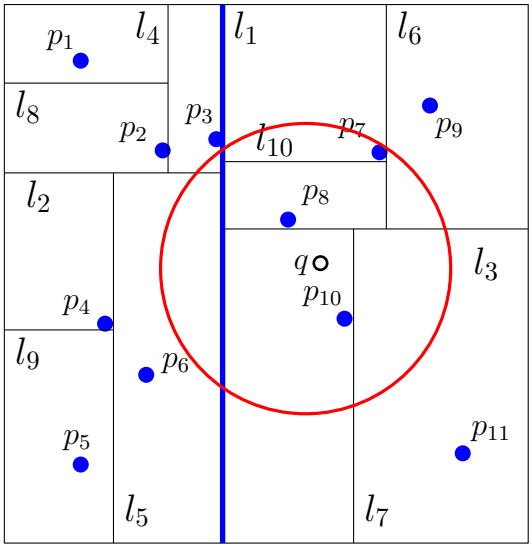
Nearest Neighbor Search

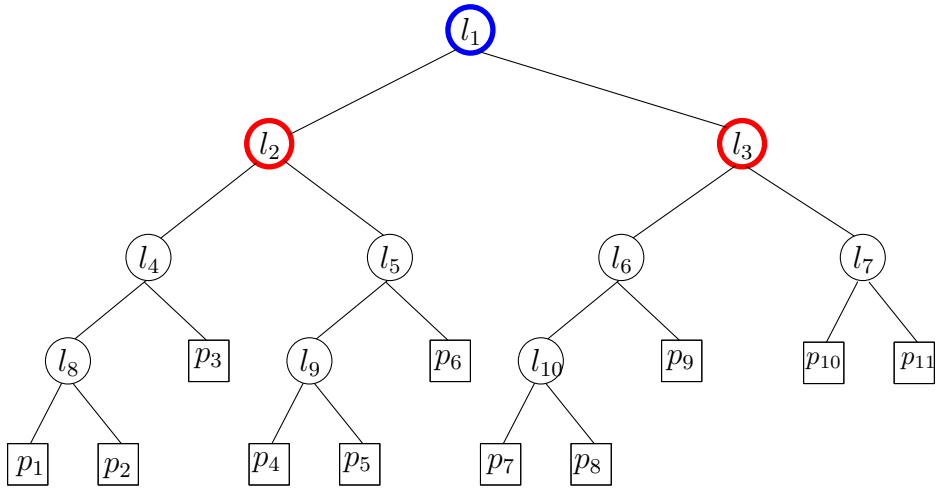


Recursive search starting at the root

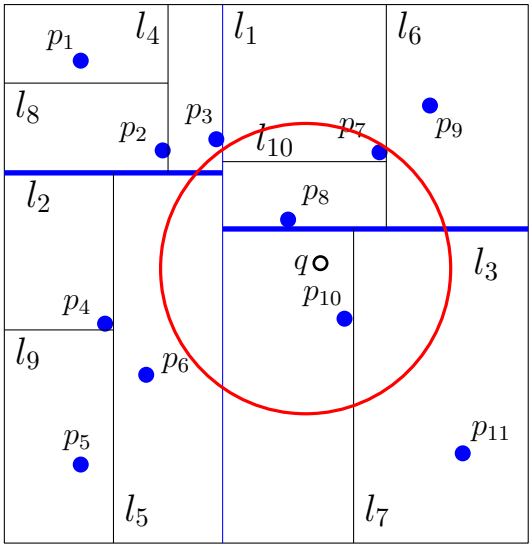


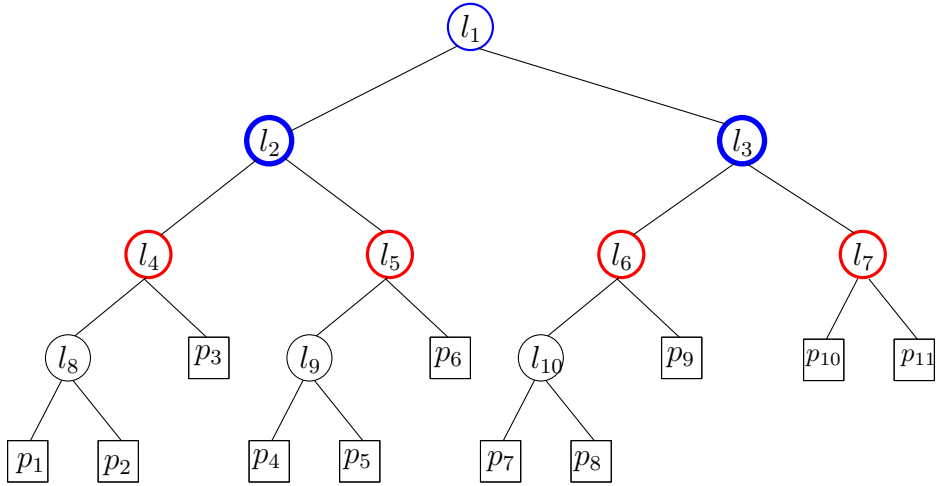
Nearest Neighbor Search



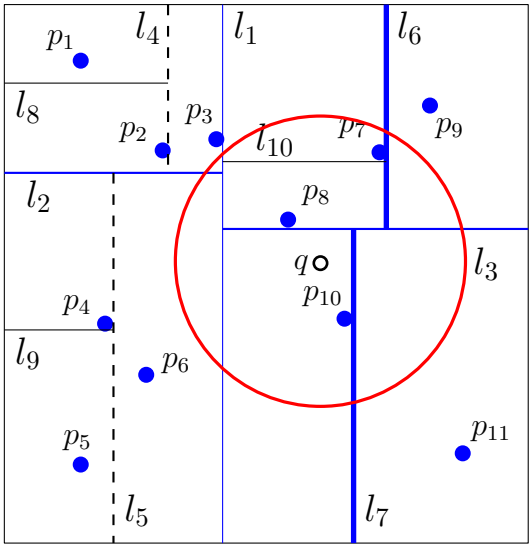


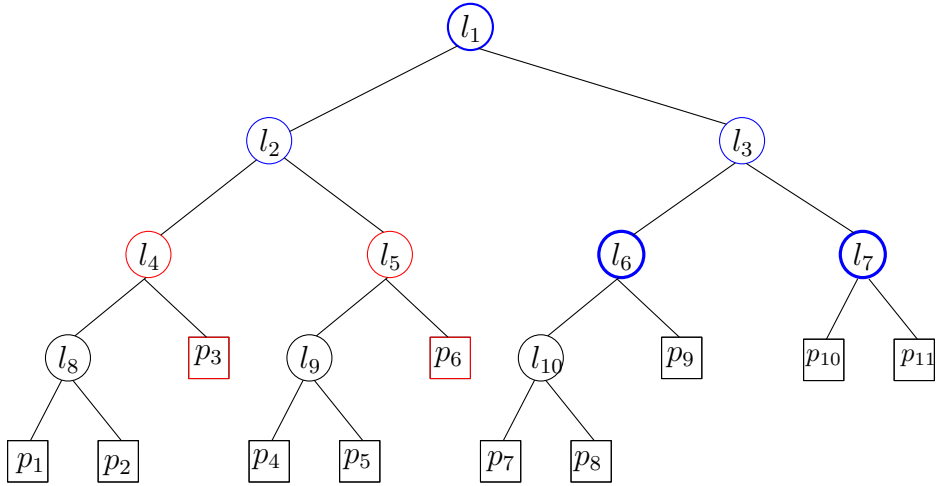
Nearest Neighbor Search



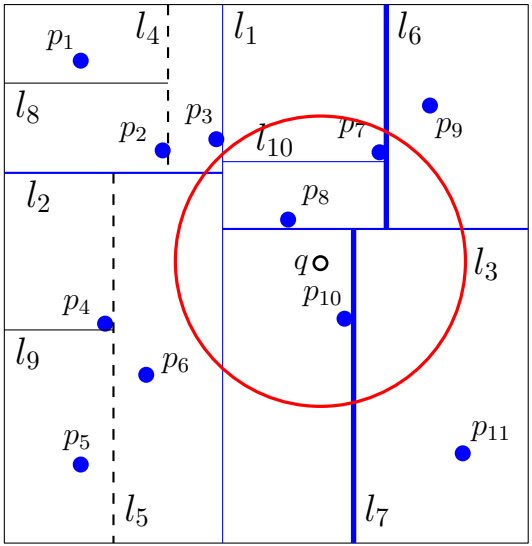


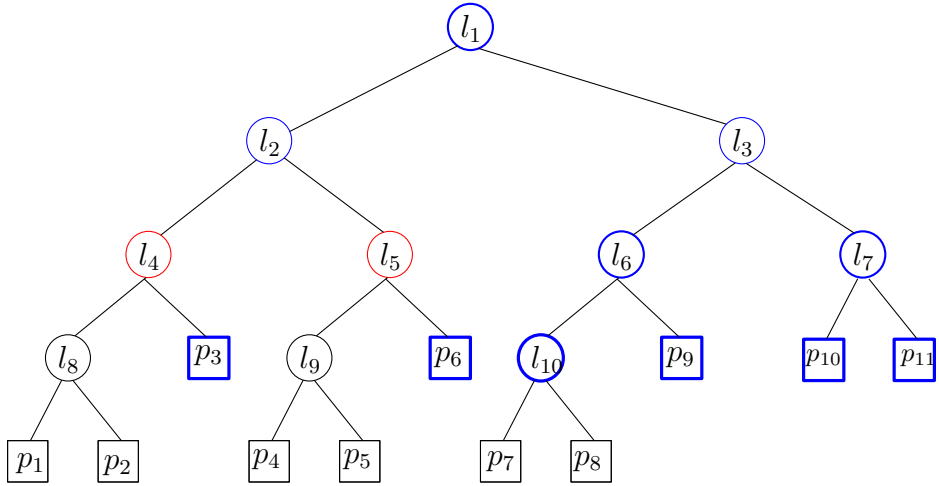
Nearest Neighbor Search





Nearest Neighbor Search

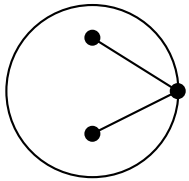


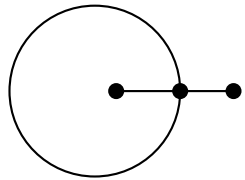


Maillages, graphes et cartes planaires

Un *graphe* $G = (V, E)$ est une paire constituee de:

- un ensemble de *sommets* $V = (v_1, \ldots, v_n)$
- une famille $E = (e_1, \ldots, e_m)$ d'elements du produit cartesien $V \times V = \{(u, v) \mid u \in V, v \in V\}$ appelés *aretes*.





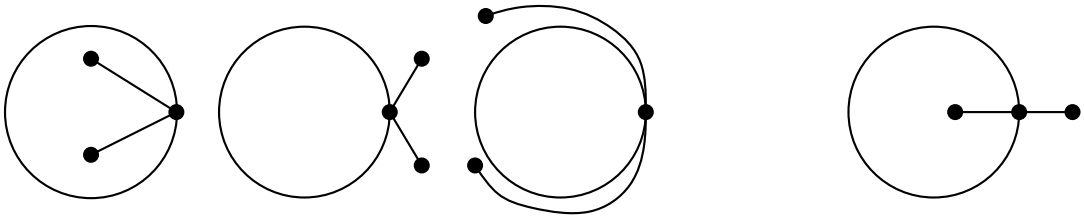
Maillages, graphes et cartes planaires

un *dessin planaire* est un plongement cellulaire de G dans R^2, qui satisfait les conditions suivantes:

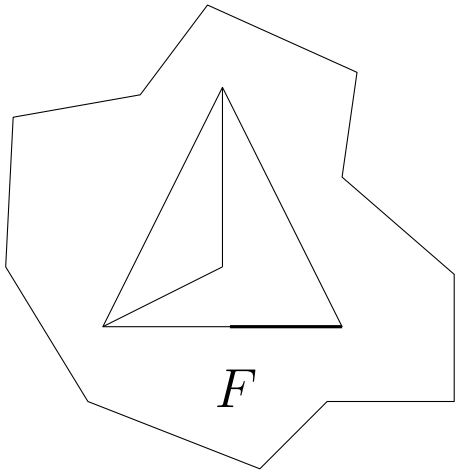
(i) les sommets du graphe sont reprśentés par des points ;

(ii) les aretes sont reprśentées par des arcs de courbes ne se coupant qu'aux sommets ;

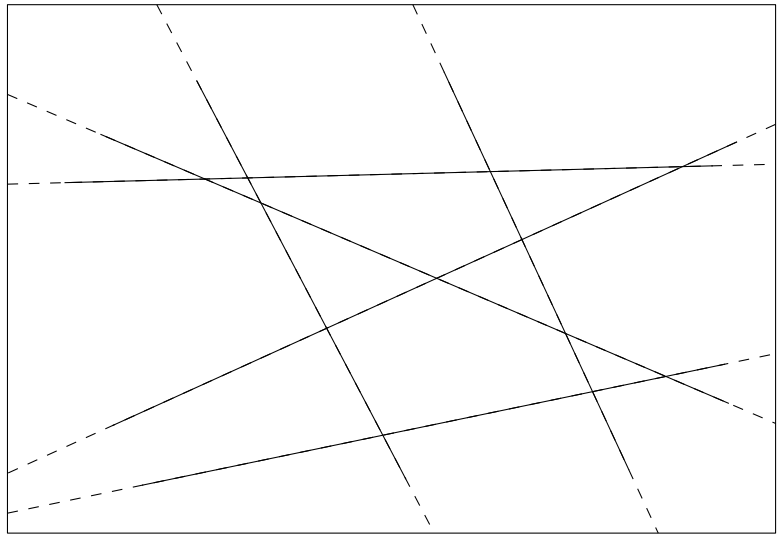
(iii) les faces sont simplement connexes.



Subdivisions planaires

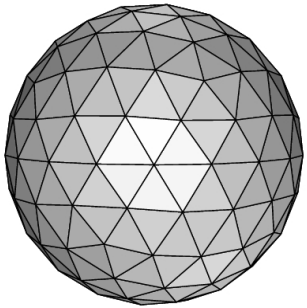


F non simplement connexe



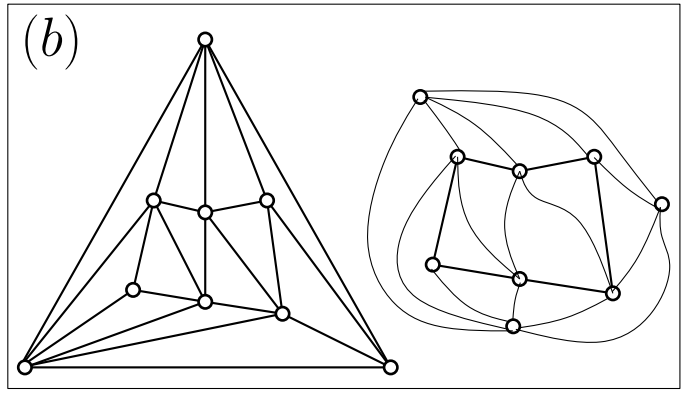
Triangulations

(a)



maillage triangulaire

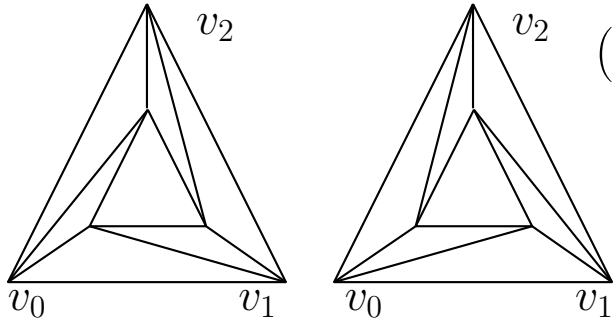
(b)



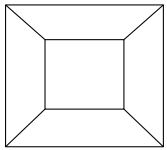
triangulation planaire

triangulations de points dans R^2

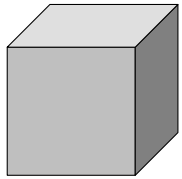
(c)



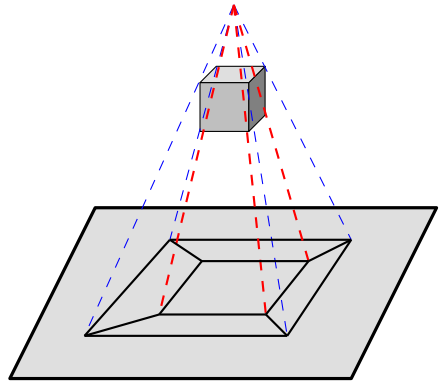
Combinatoire des maillages et graphes planaires



carte planaire



polyedre



surface combinatoire de genre 0

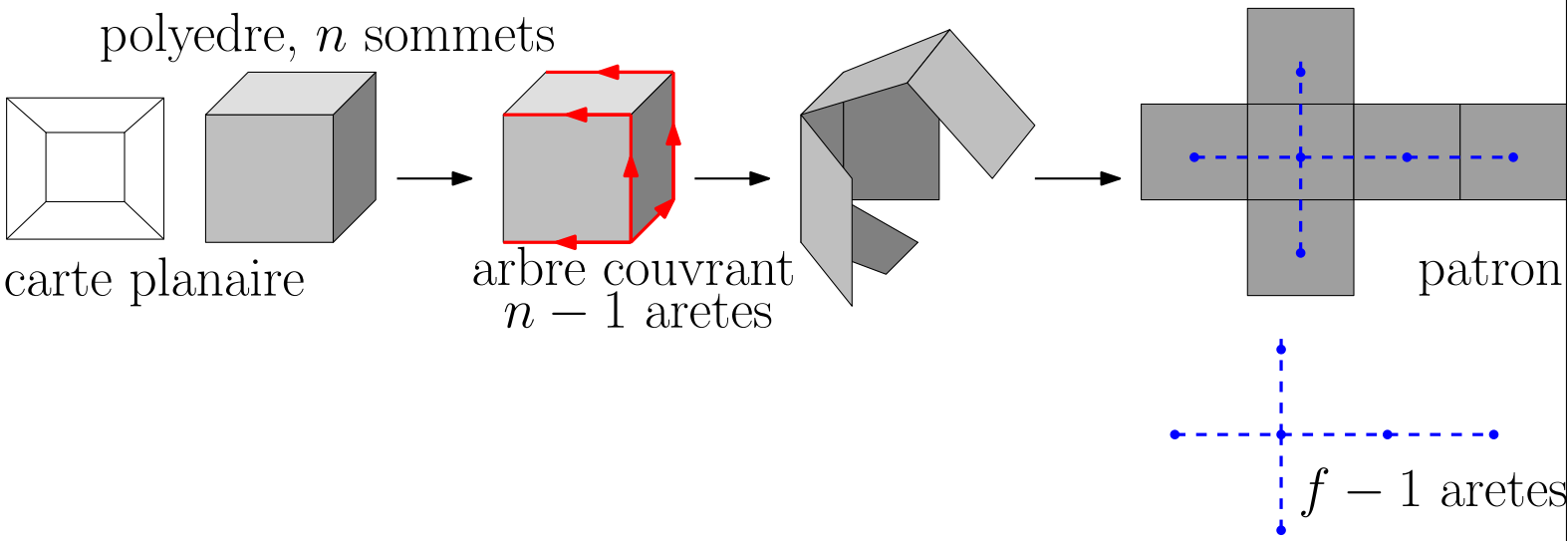
$$n - e + f = 2$$

formule d'Euler

Formule d'Euler, pour polyedres et graphes planaires

Overview de la preuve

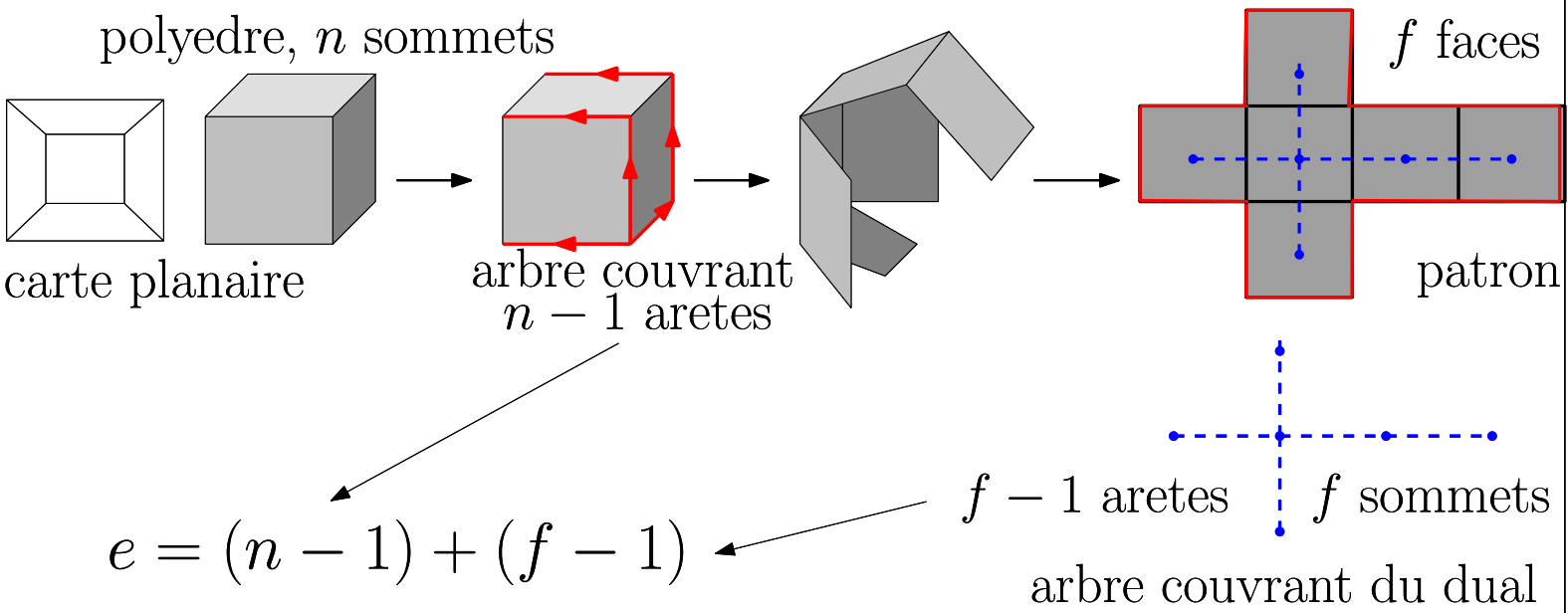
$$n - e + f = 2$$



Formule d'Euler, pour polyedres et graphes planaires

Overview de la preuve

$$n - e + f = 2$$



Conséquences de la formule d'Euler

Nombre linéaire d'aretes et faces

$f \leq 2n - 4$

$$e \leq 3n - 6$$

preuve (par double comptage)

$$f = f_1 + f_2 + f_3 + \dots$$
$$n = n_1 + n_2 + n_3 + \dots$$

toutes les faces ont degré au moins 3 ($\mathcal{G}$ est simple), on a

$$f = f_3 + f_4 + \dots$$

chaque arete apparait deux fois

$$2e = 3 \cdot f_3 + 4 \cdot f_4 + \dots$$

d'ou la relation

$$2e - 3f \geq 0$$

Conséquences de la formule d'Euler

Nombre linéaire d'aretes et faces $f \leq 2n - 4$

$$e \leq 3n - 6$$

ayant prouvé $2e - 3f \geq 0$

en appliquant Euler on trouve

$$3n - 6 = 3(e - f + 2) = 3e - 3f \geq 0$$

Eléments de géométrie convexe

Coordonnés barycentriques

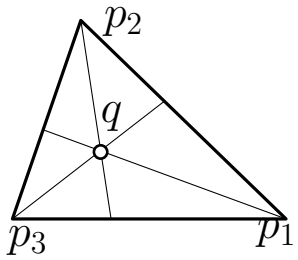
$$q = \Sigma_i^n \alpha_i p_i \text{ (avec } \Sigma_i \alpha_i = 1$$

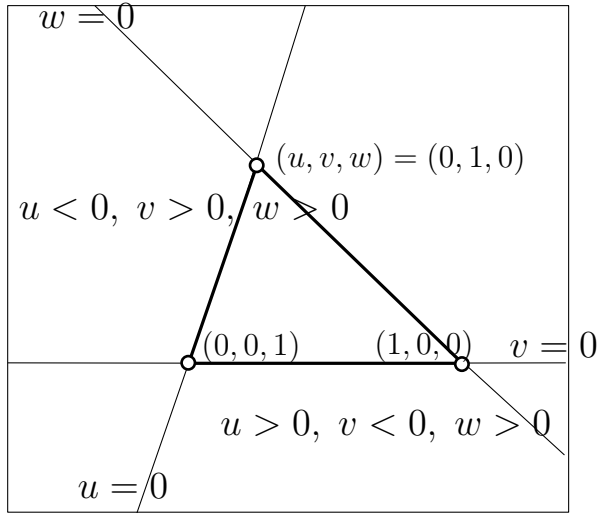
les coefficients $(\alpha_1, \ldots, \alpha_n)$ sont les *coordonnées barycentriques* de q par rapport aux points $p_1, \ldots, p_n$

Coordonnées barycentriques

$$q = \Sigma_i^n \alpha_i p_i \text{ (avec } \Sigma_i \alpha_i = 1)$$

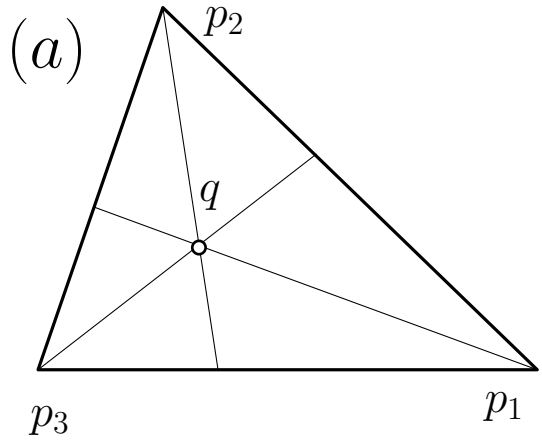
les coefficients $(\alpha_1, \ldots, \alpha_n)$ sont les *coordonnées barycentriques* de q par rapport aux points $p_1, \ldots, p_n$

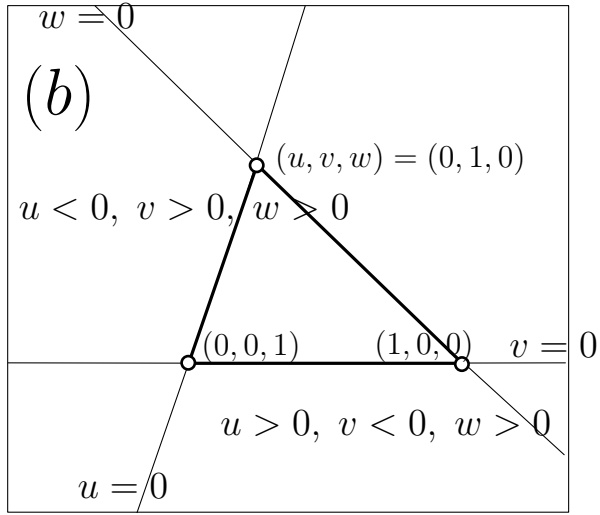




Coordonnées barycentriques

$$\mathbf{q} = u\,\mathbf{p_1} + v\,\mathbf{p_2} + w\,\mathbf{p_3} = \frac{A(q,p_2,p_3)}{A(p_1,p_2,p_3)}\mathbf{p_1} + \frac{A(p_1,q,p_3)}{A(p_1,p_2,p_3)}\mathbf{p_2} + \frac{A(p_1,p_2,q)}{A(p_1,p_2,p_3)}\mathbf{p_3}$$





Enveloppe convexe

$$aff(\mathcal{S}) = \{ \Sigma_{i=1}^{n} \alpha_i p_i \mid \Sigma_i \alpha_i = 1 \}$$

combinaison affine

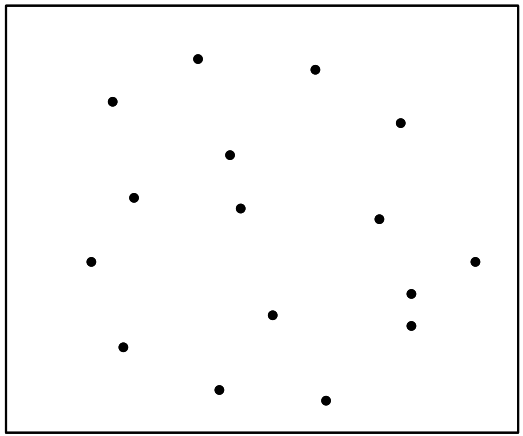
$$convex(\mathcal{S}) = \{ \Sigma_{i=1}^{n} \alpha_i p_i \mid \alpha_i \geq 0, \Sigma_i \alpha_i = 1 \}$$

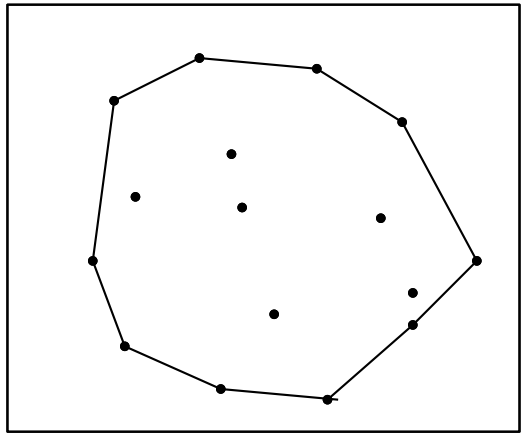
combinaison convexe

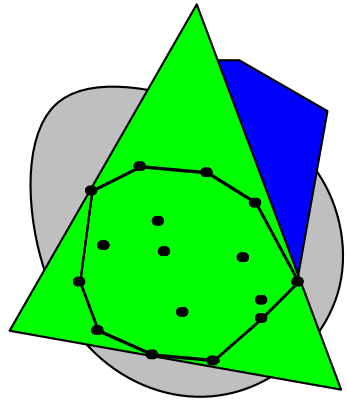
Enveloppe convexe

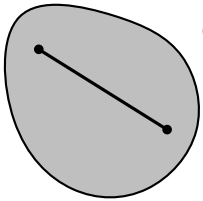
$$convex(\mathcal{S}) = \{ \Sigma_{i=1}^{n} \alpha_i p_i \mid \alpha_i \geq 0, \Sigma_i \alpha_i = 1 \}$$

combinaison convexe









ensemble convexe

intersection de tous les convexes contenant $\mathcal{S}$

Polytopes

Def: enveloppe convexe d'un ensemble fini de points en R^d

Thm de la borne supérieure

Tout polytope avec n sommets dans R^d possède au plus $O(n^{\lfloor d/2 \rfloor})$ faces au total.

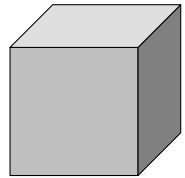
cette borne est atteinte par les polytopes cycliques.

$(t, t^2, t^3 \ldots, t^d)$

courbe des moments

en 3D

Tout 2-polytope avec n sommets possède au plus $O(n)$ faces et aretes.

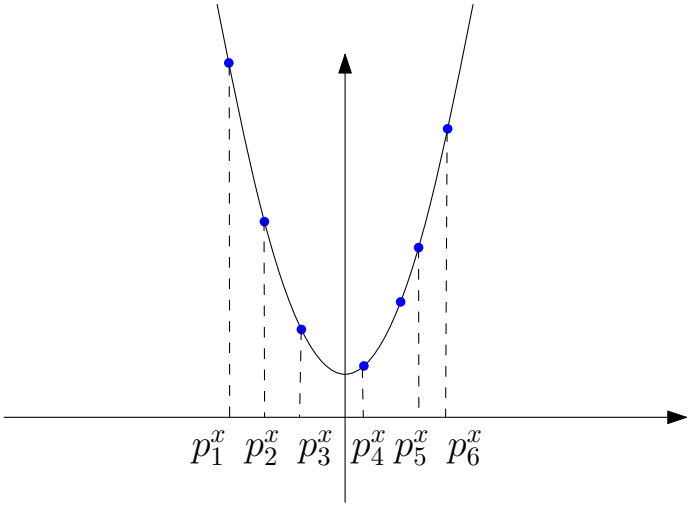


polyèdre

Complexité de calcul des enveloppes convexes

Borne inf 2D

Le calcul de l'enveloppe convexe de n points necessite temps $\Omega(n \log n)$

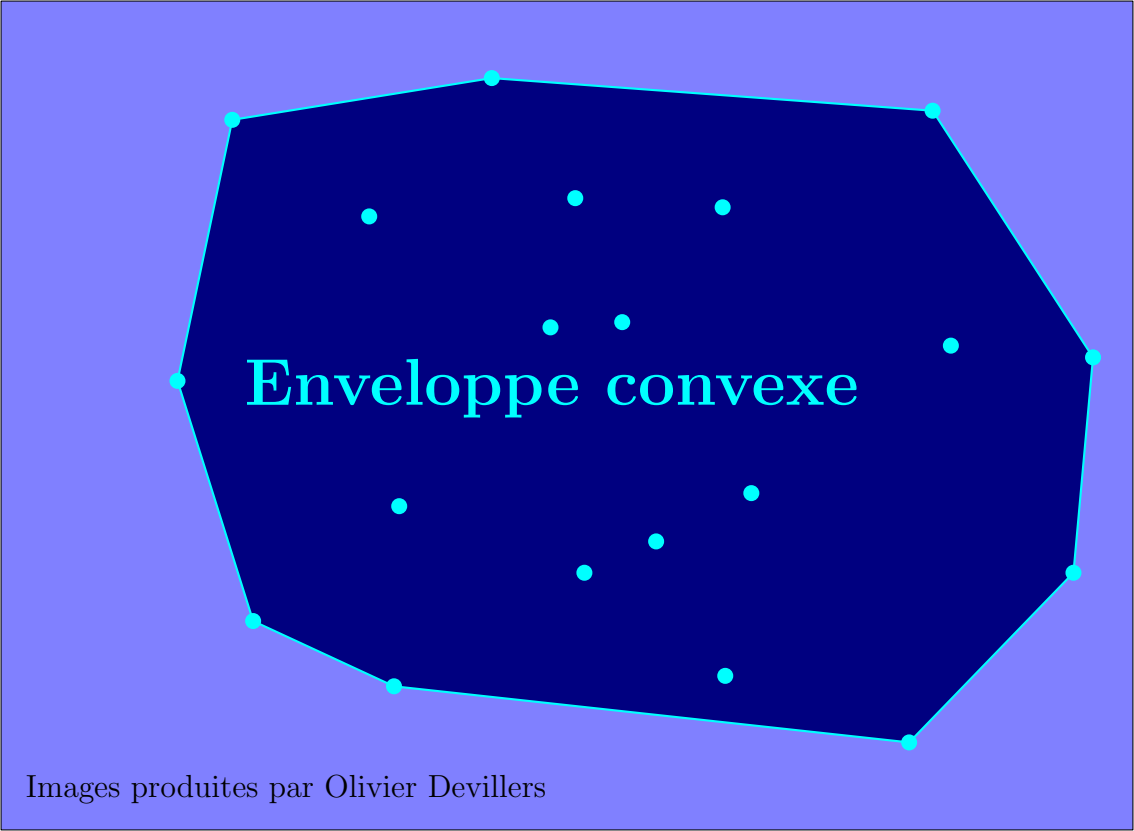


Borne inf 3D

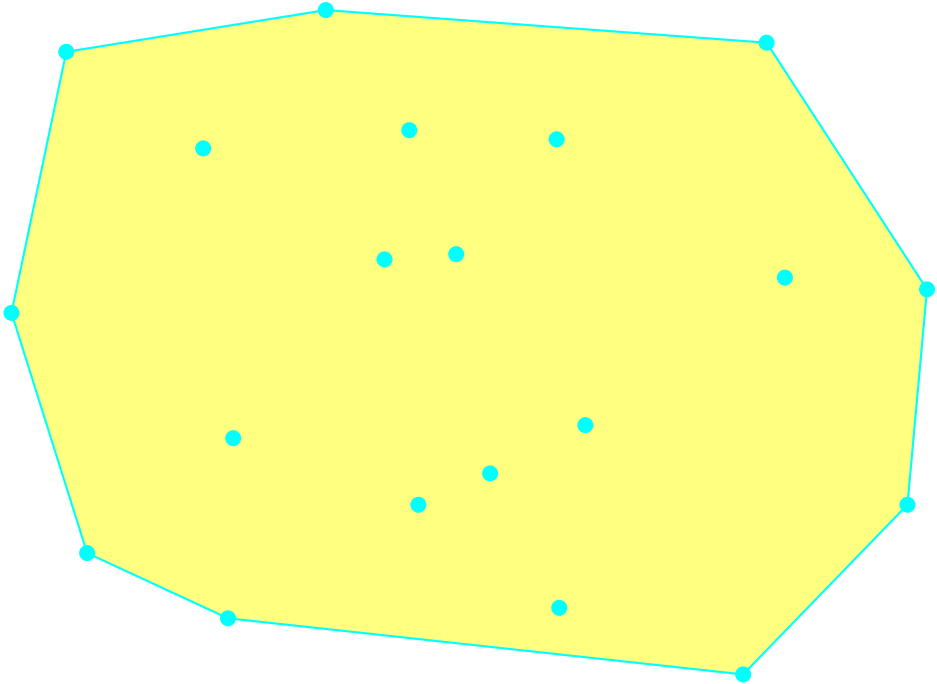
Le calcul de l'enveloppe convexe de n points necessite temps

$$O(n \log n + n^{\lfloor d/2 \rfloor})$$

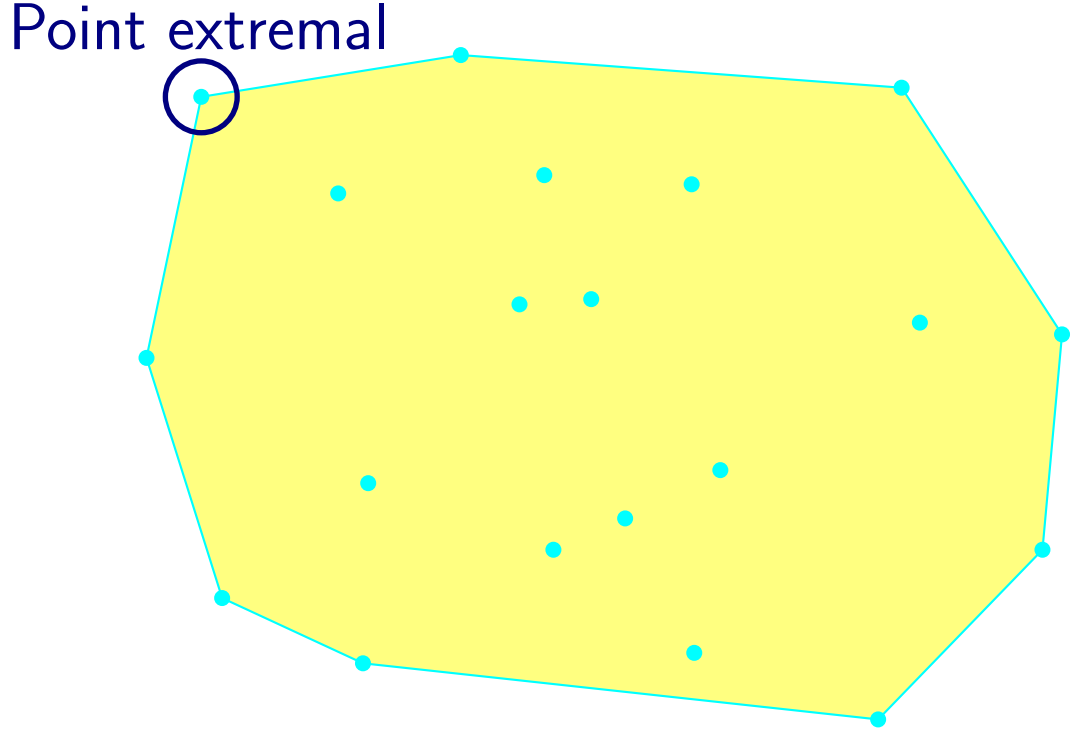
Images produites par Olivier Devillers



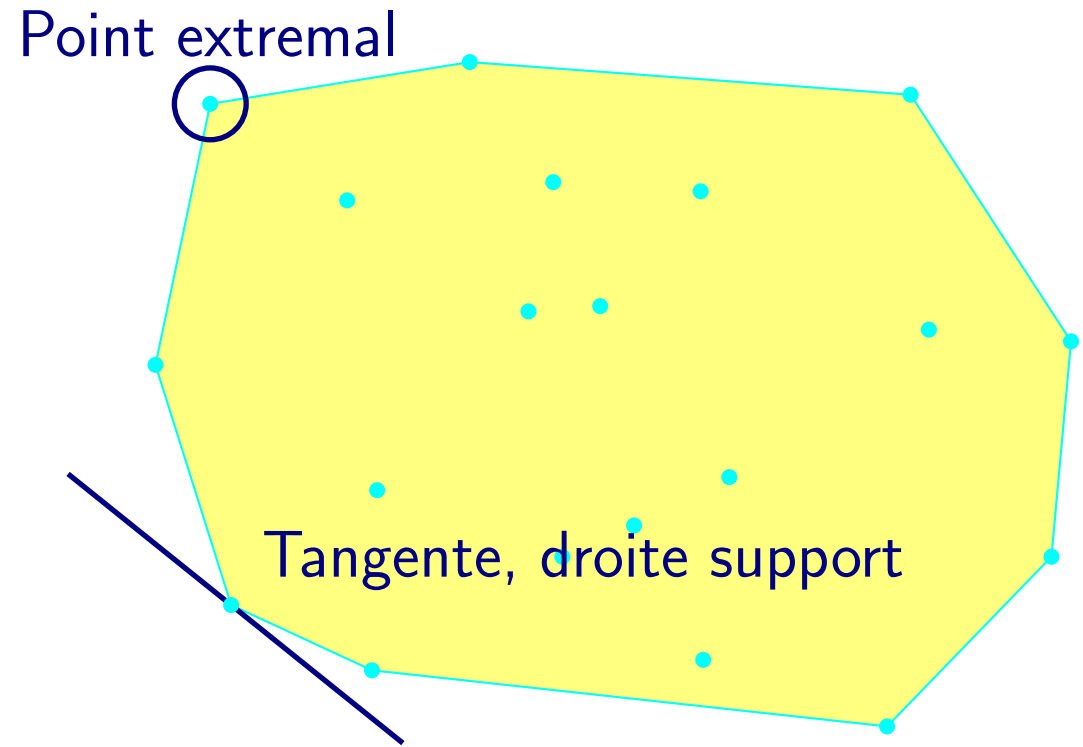
Images produites par Olivier Devillers



Images produites par Olivier Devillers

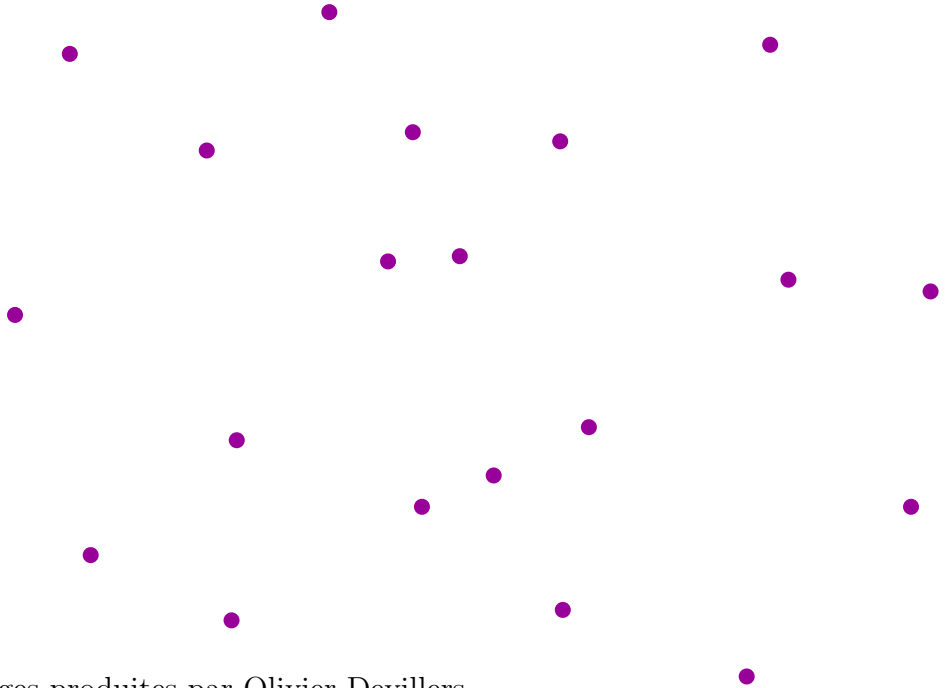


Images produites par Olivier Devillers



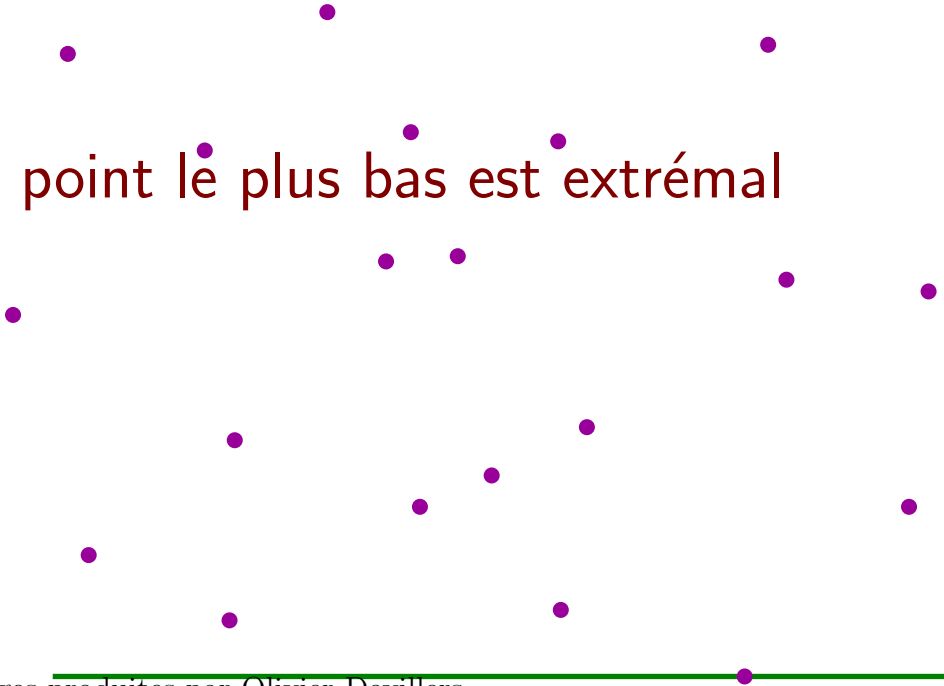
Images produites par Olivier Devillers

Premier algorithme : Jarvis



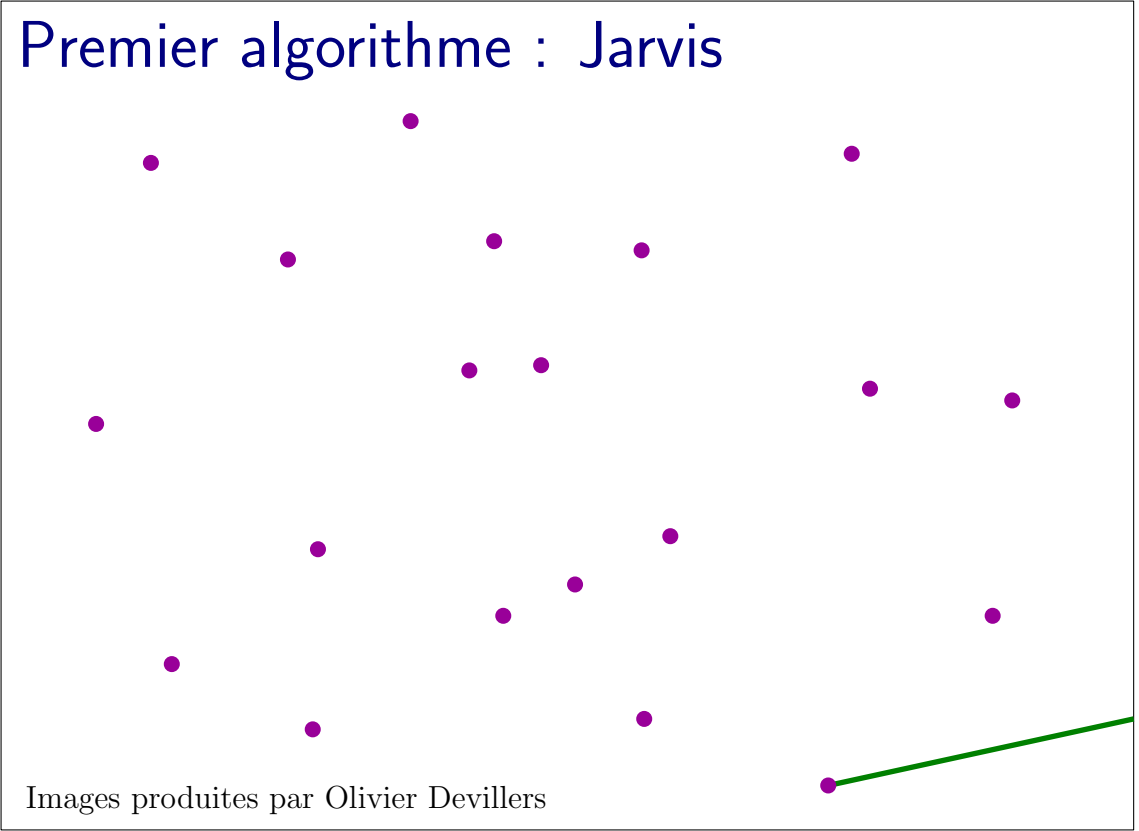
Images produites par Olivier Devillers

Premier algorithme : Jarvis



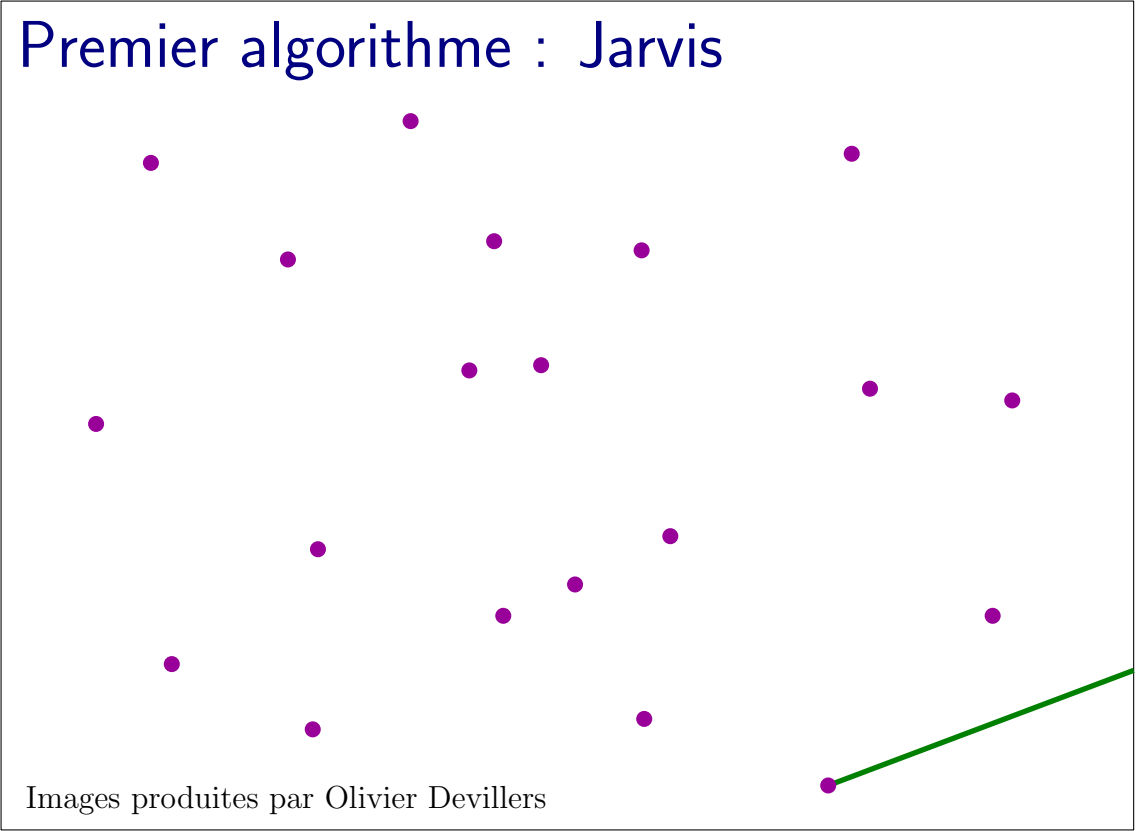
Images produites par Olivier Devillers

Premier algorithme : Jarvis



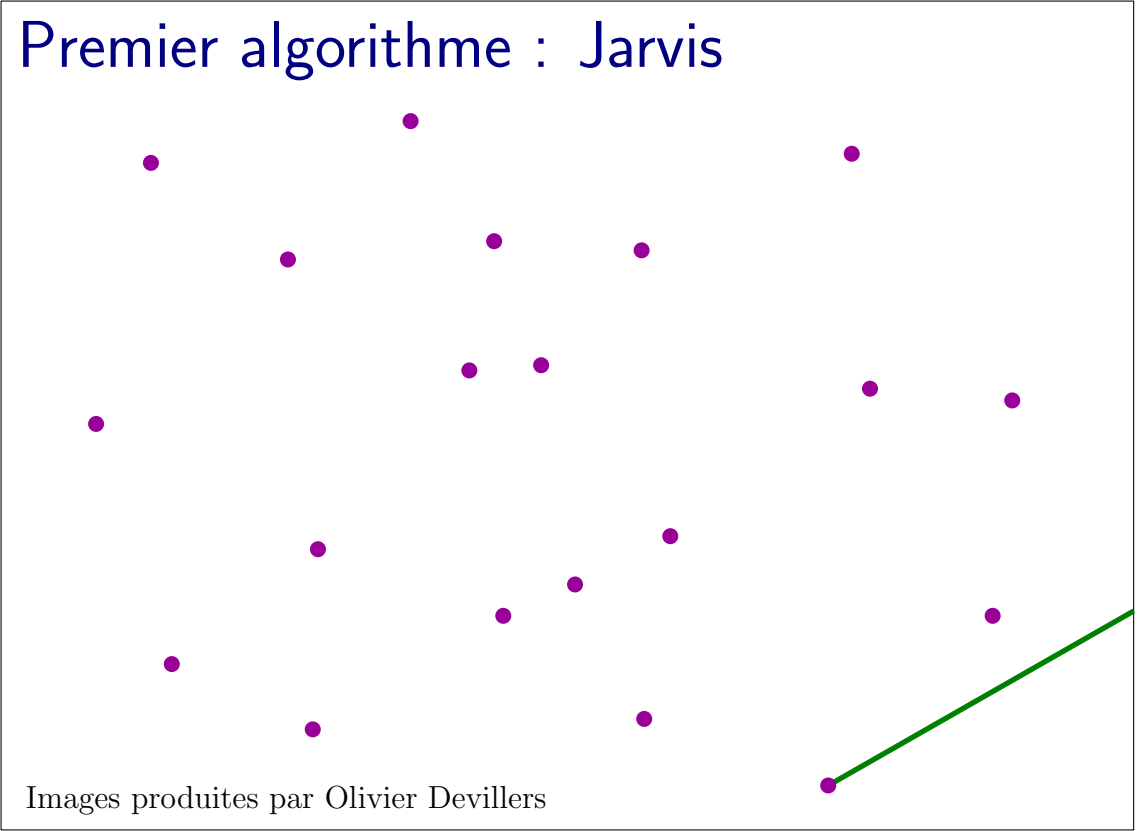
Images produites par Olivier Devillers

Premier algorithme : Jarvis



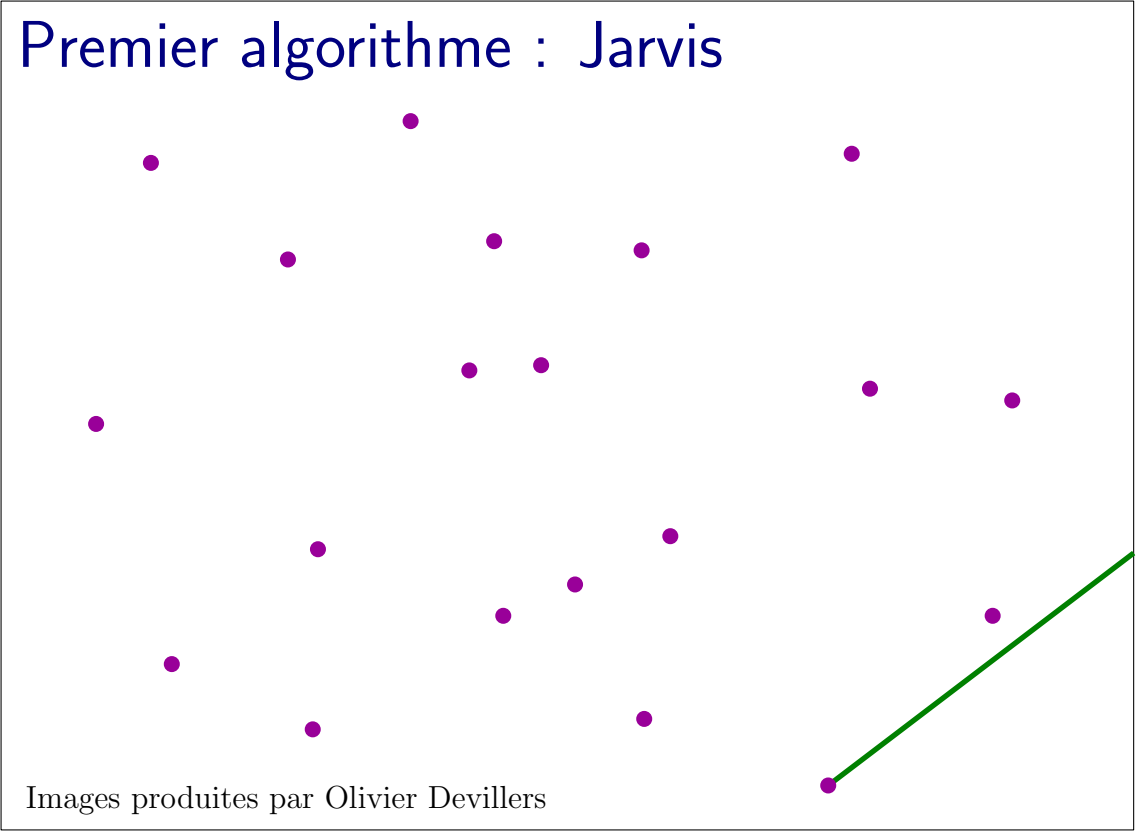
Images produites par Olivier Devillers

Premier algorithme : Jarvis



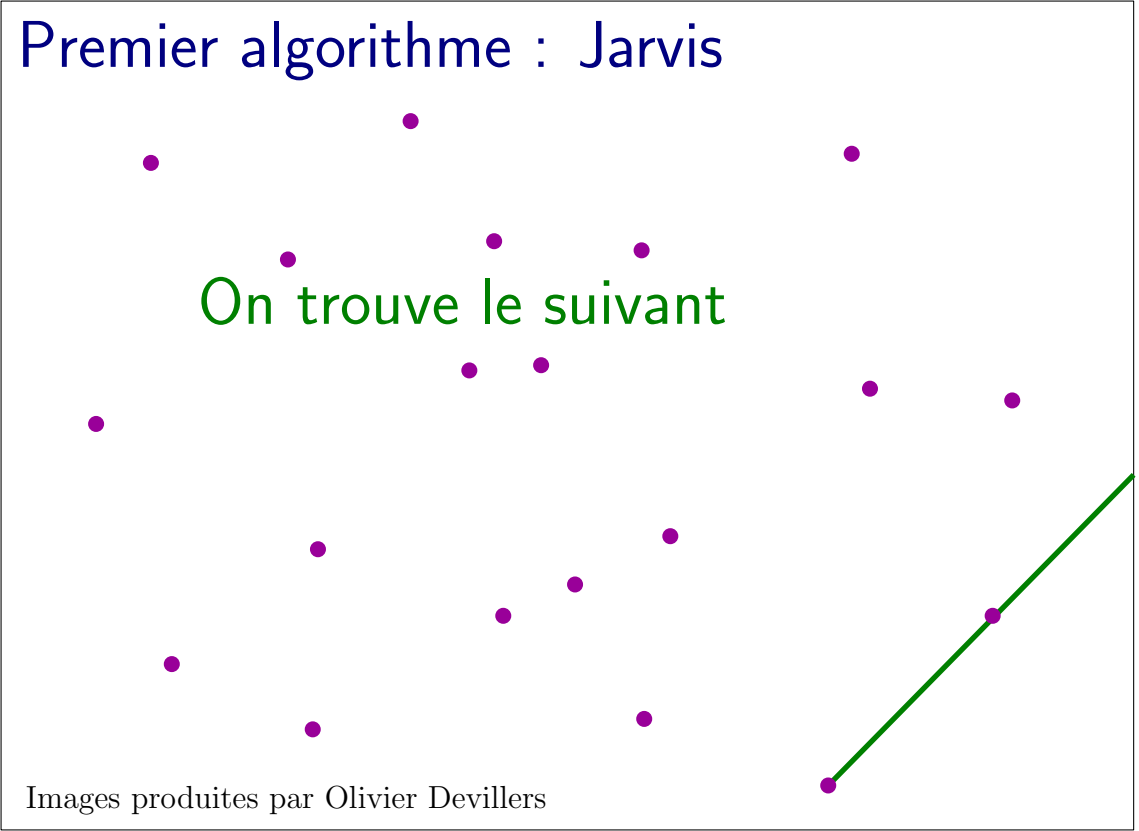
Images produites par Olivier Devillers

Premier algorithme : Jarvis



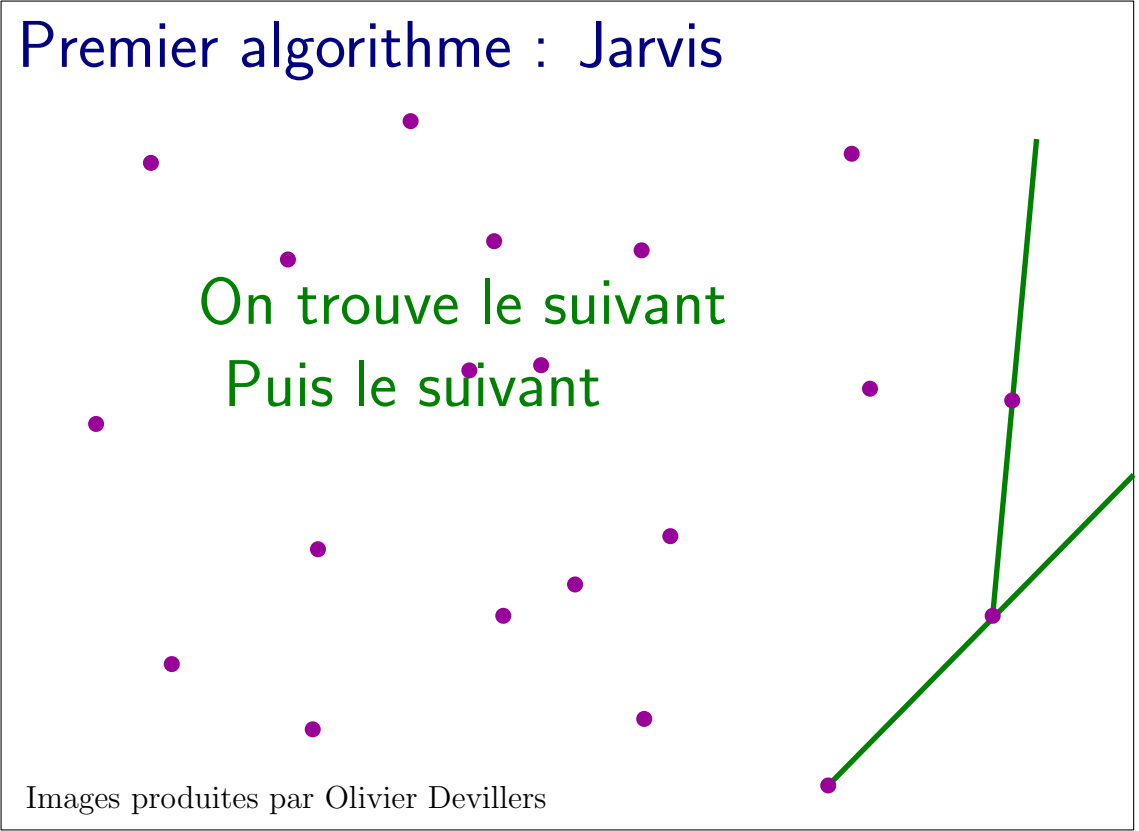
Images produites par Olivier Devillers

Premier algorithme : Jarvis



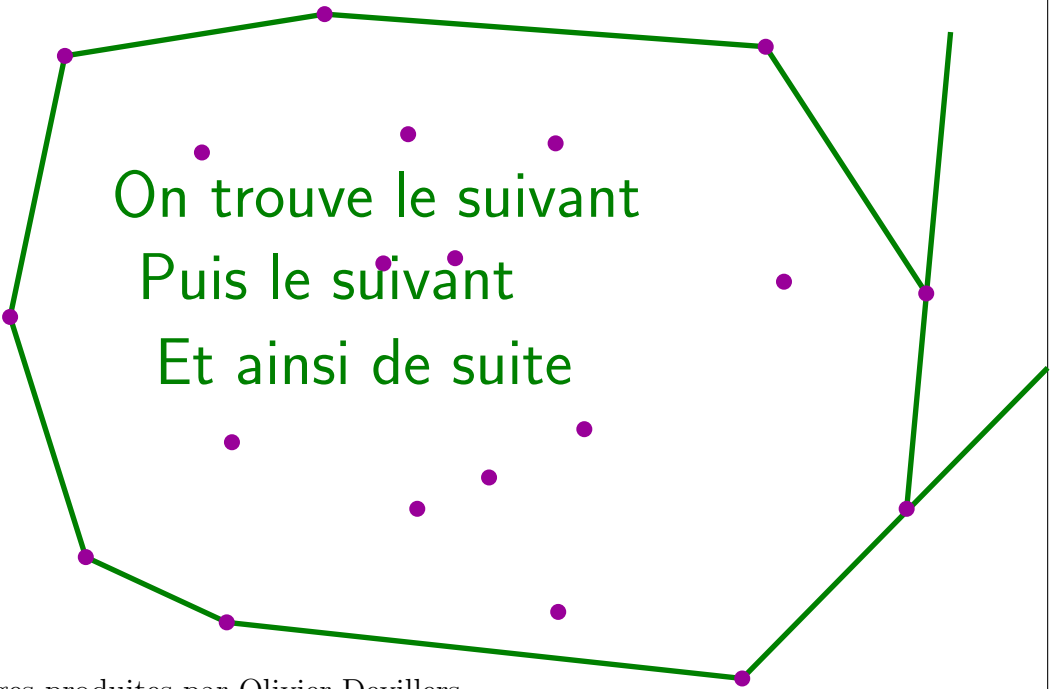
Images produites par Olivier Devillers

Premier algorithme : Jarvis



Images produites par Olivier Devillers

Premier algorithme : Jarvis



Images produites par Olivier Devillers

Premier algorithme : Jarvis

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
                $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

Images produites par Olivier Devillers

Premier algorithme : Jarvis

Complexité ?

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
                $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

Images produites par Olivier Devillers

Premier algorithme : Jarvis

Complexité ?

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
                $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

$O(n)$

Images produites par Olivier Devillers

Premier algorithme : Jarvis

Complexité ?

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
            $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

$O(n)$

Images produites par Olivier Devillers

Premier algorithme : Jarvis

Complexité ?

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
                $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

Images produites par Olivier Devillers

Premier algorithme : Jarvis

Complexité ?

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
            $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

$O(n)$

Images produites par Olivier Devillers

Premier algorithme : Jarvis

Complexité ?

$O(n^2)$

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
                $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

Images produites par Olivier Devillers

Premier algorithme : Jarvis

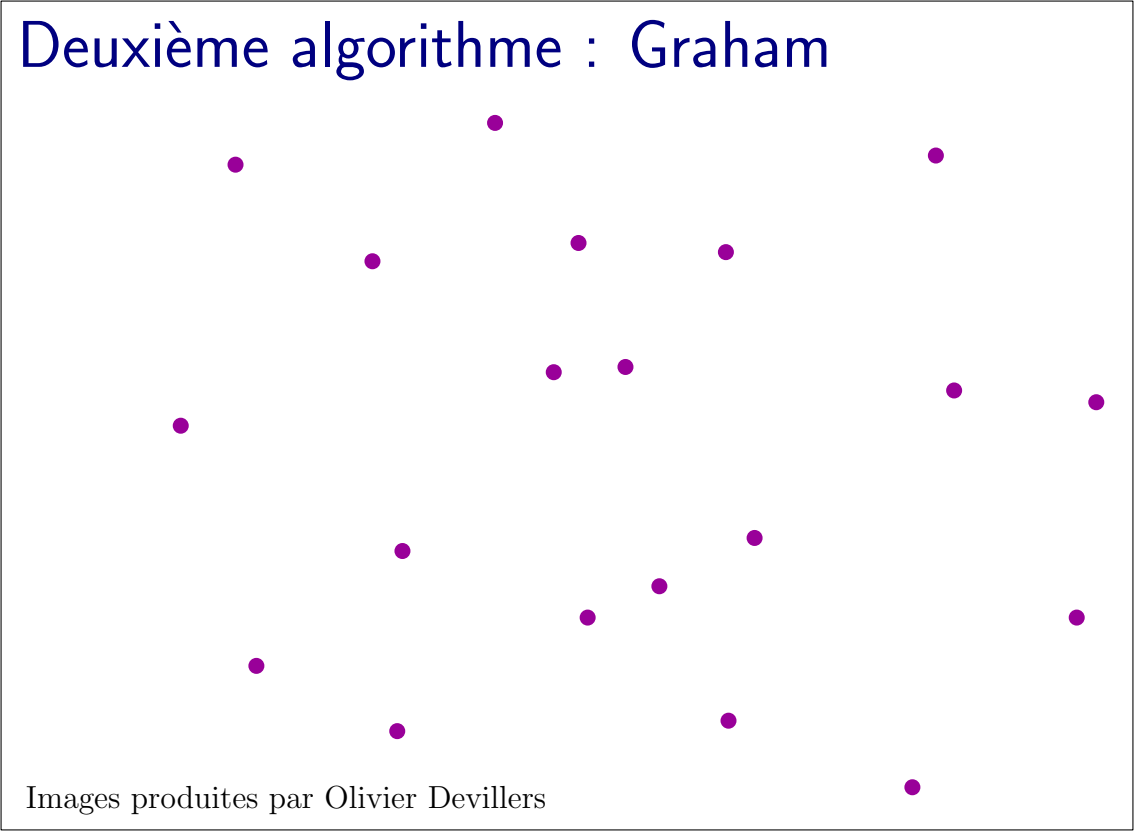
Complexité ?

$O(nh)$

entrée : S un ensemble de points.
$u =$ le point le plus bas de S;
$min = \infty$
Pour tout $w \in S \setminus \{u\}$
    si $angle(ux, uw) < min$ alors $min = angle(ux, uw)$; $v = w$;
$u.suivant = v$;
Faire
    $S = S \setminus \{v\}$
    Pour tout $w \in S$
        $min = \infty$
        si $angle(v.pred\ v, vw) < min$ alors
                $min = angle(v.pred\ v, vw)$; $v.suivant = w$;
    $v = v.suivant$;
Tant que $v \neq u$

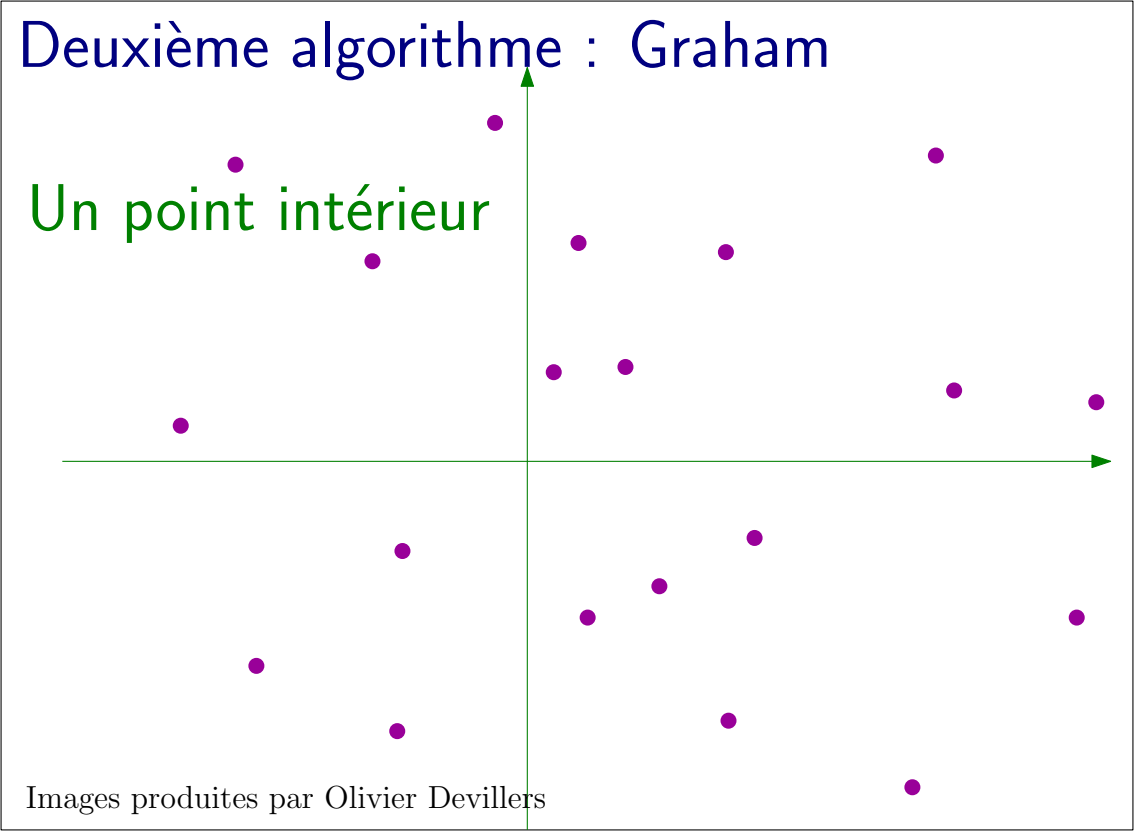
Images produites par Olivier Devillers

Deuxième algorithme : Graham



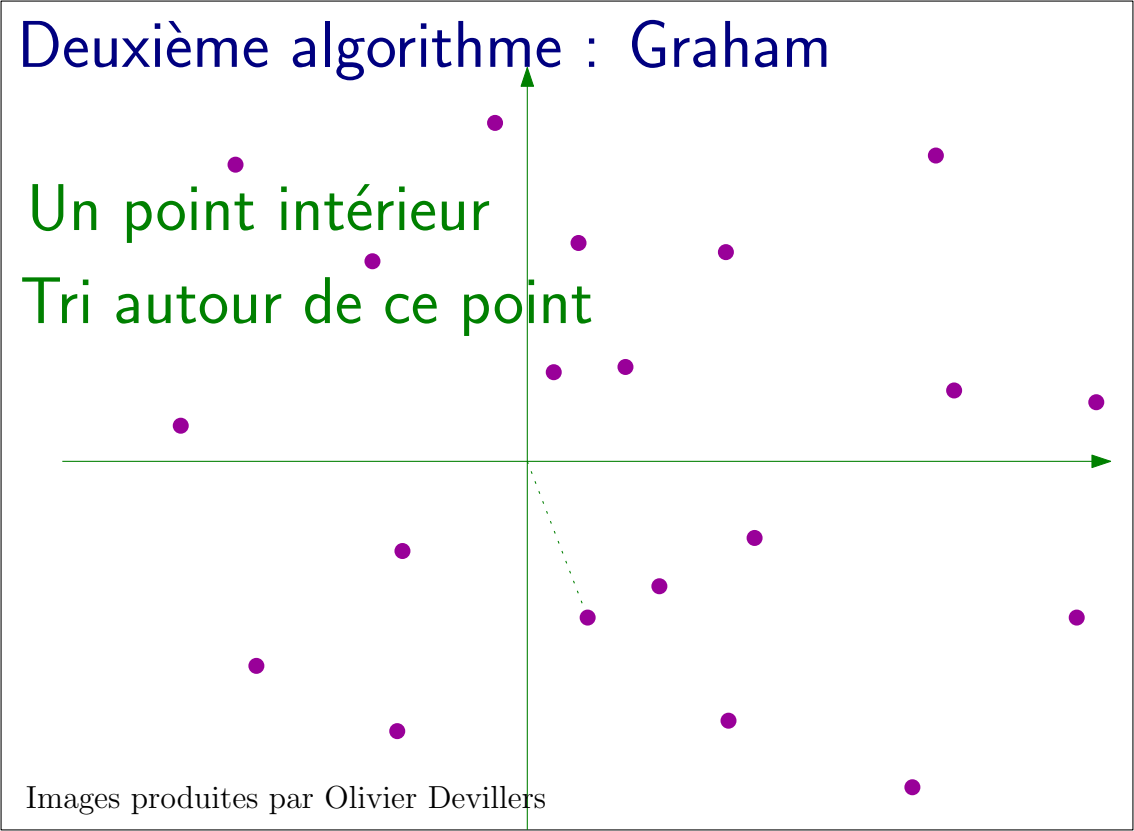
Images produites par Olivier Devillers

Deuxième algorithme : Graham



Images produites par Olivier Devillers

Deuxième algorithme : Graham

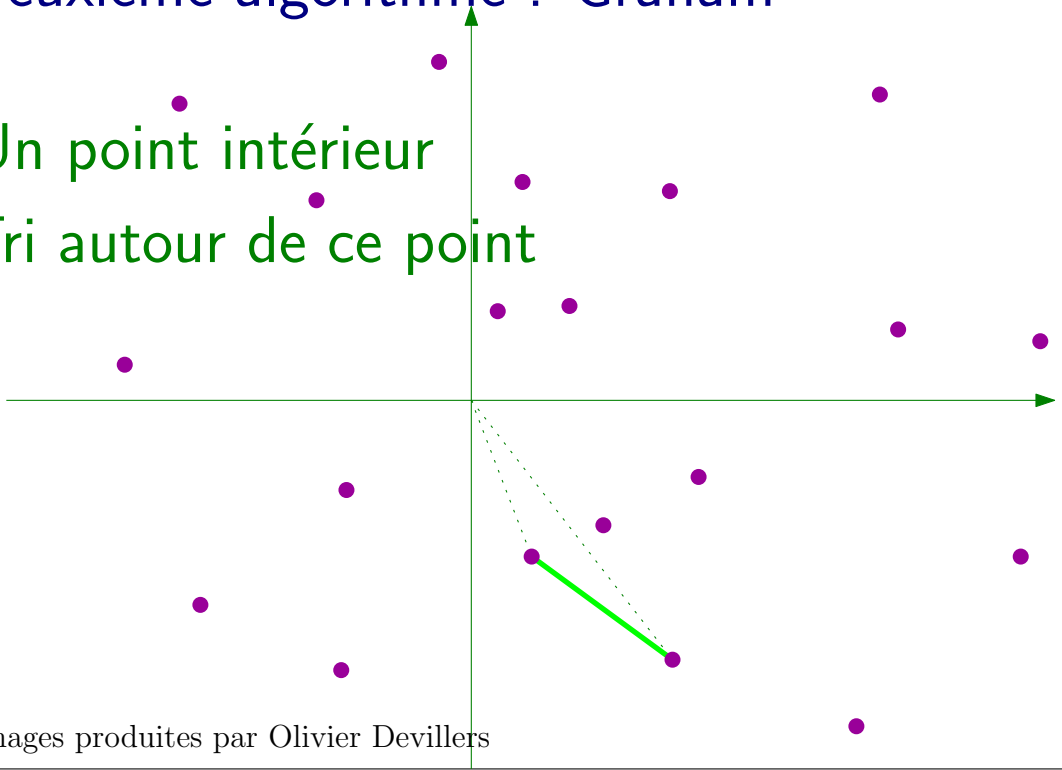


Images produites par Olivier Devillers

Deuxième algorithme : Graham

Un point intérieur

Tri autour de ce point

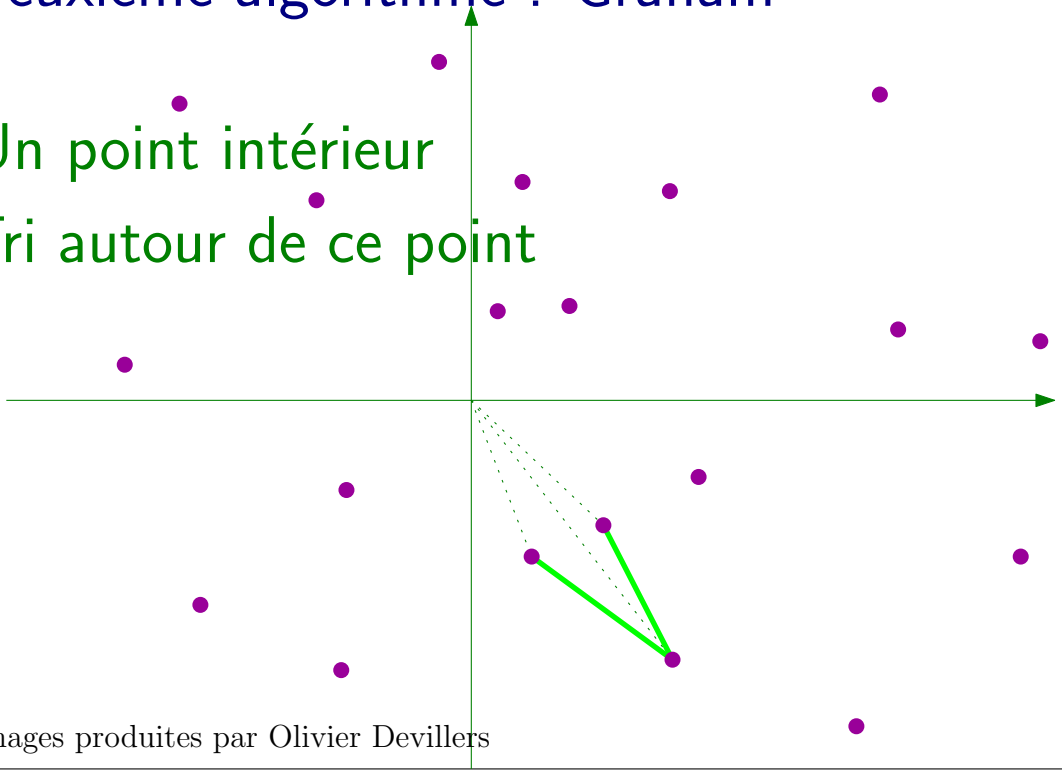


Images produites par Olivier Devillers

Deuxième algorithme : Graham

Un point intérieur

Tri autour de ce point

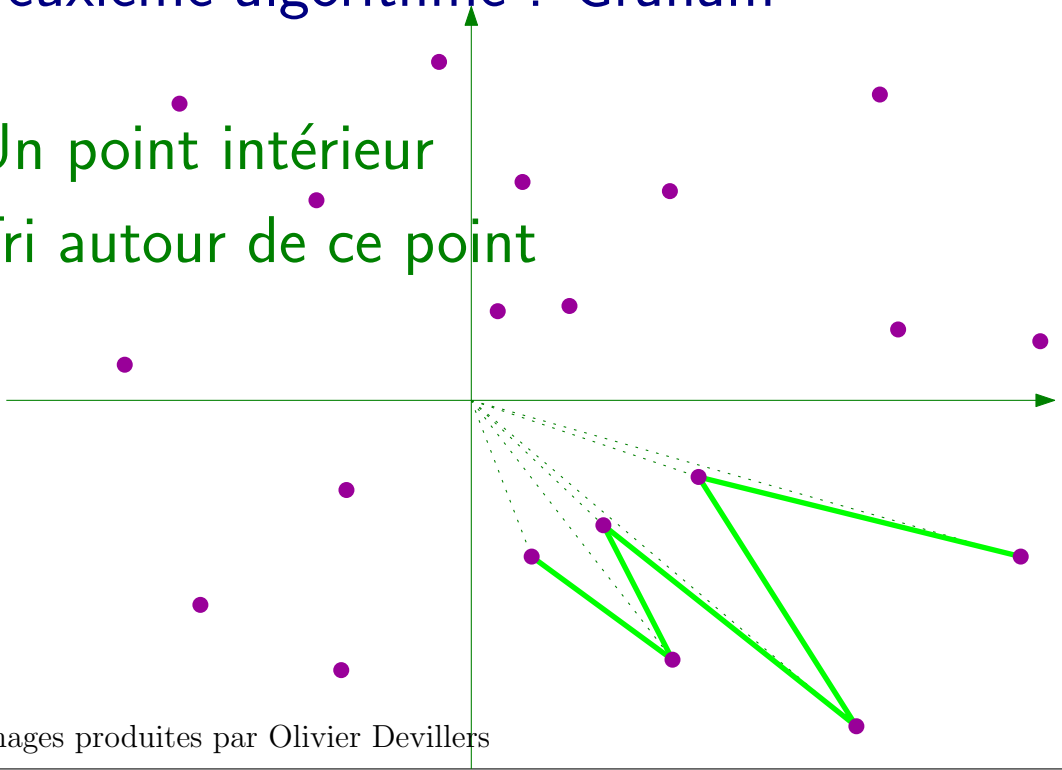


Images produites par Olivier Devillers

Deuxième algorithme : Graham

Un point intérieur

Tri autour de ce point

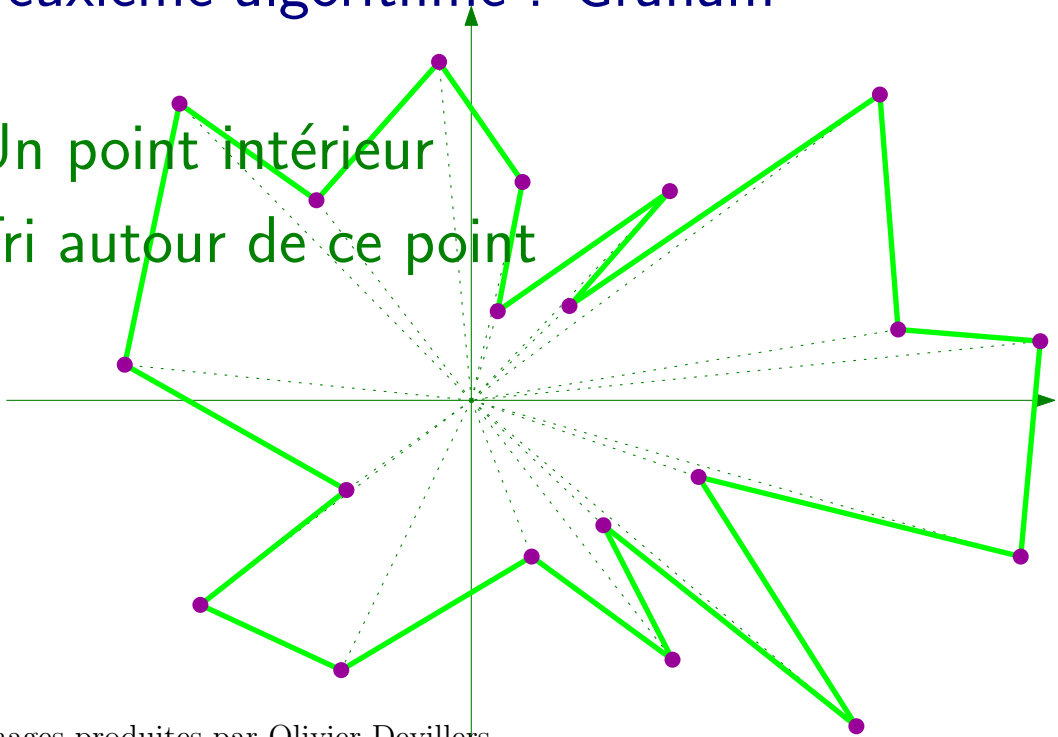


Images produites par Olivier Devillers

Deuxième algorithme : Graham

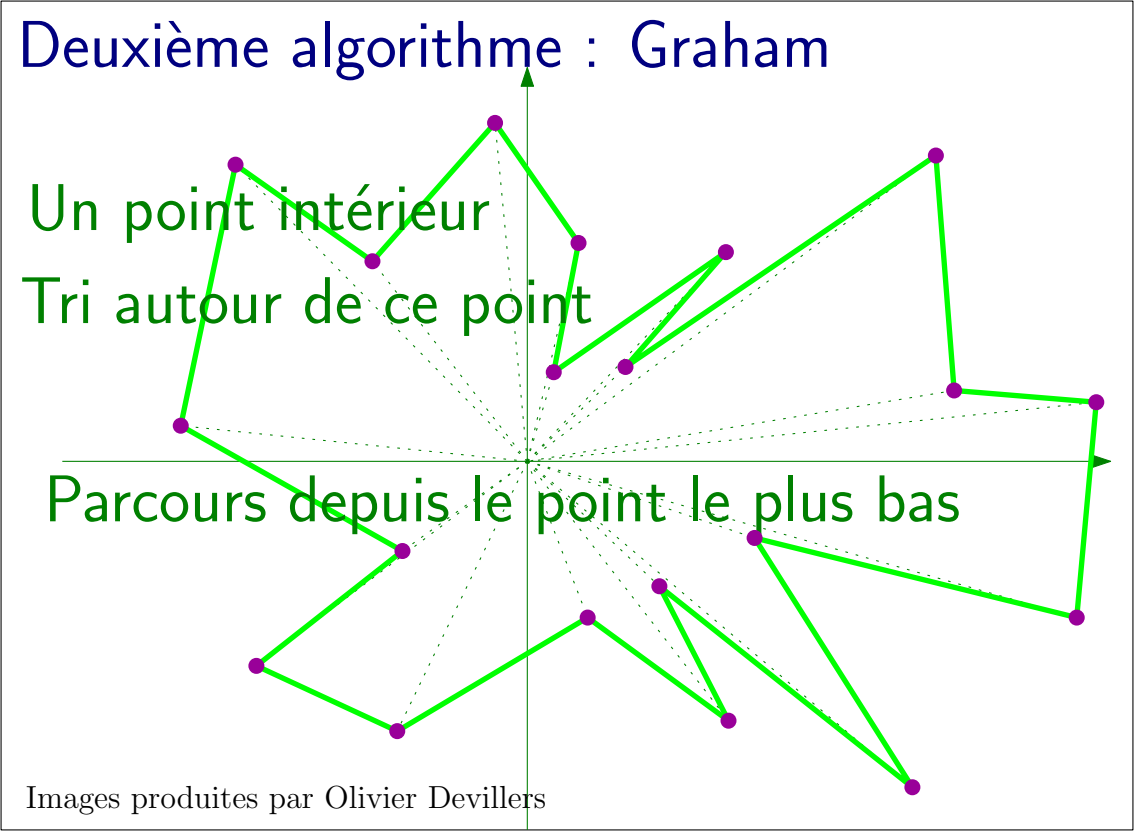
Un point intérieur

Tri autour de ce point



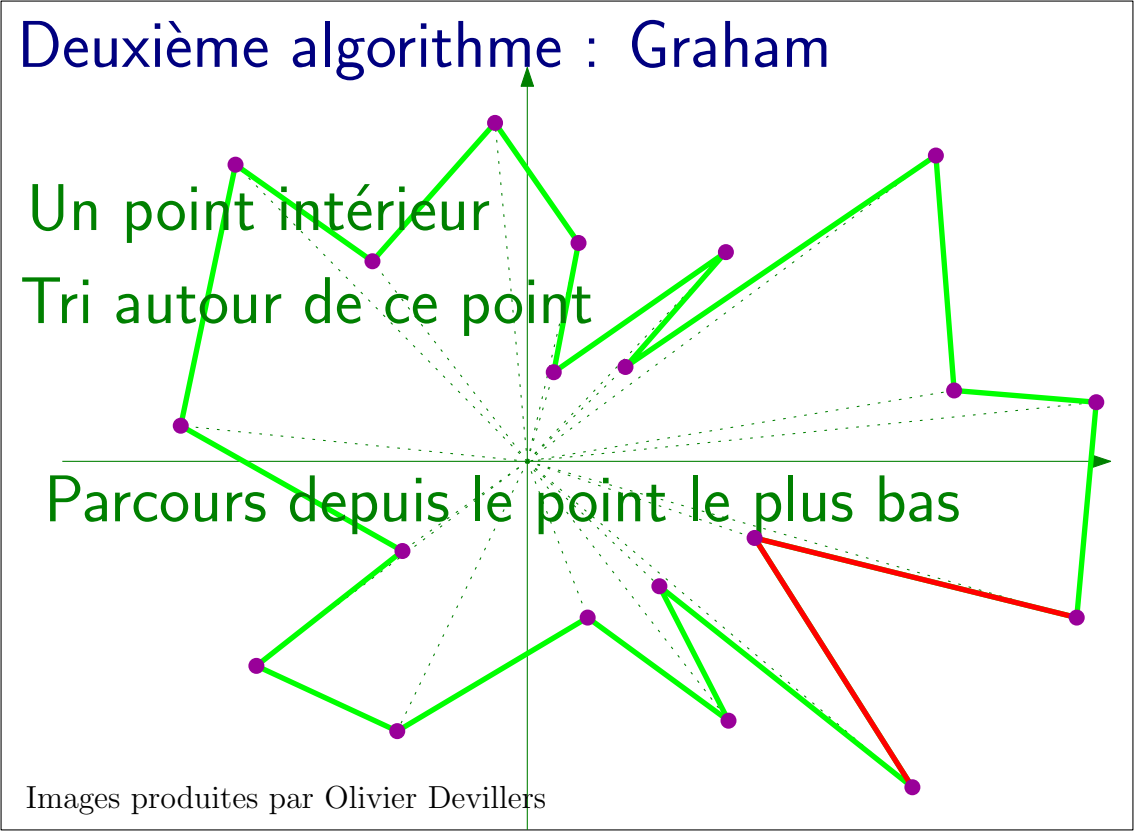
Images produites par Olivier Devillers

Deuxième algorithme : Graham



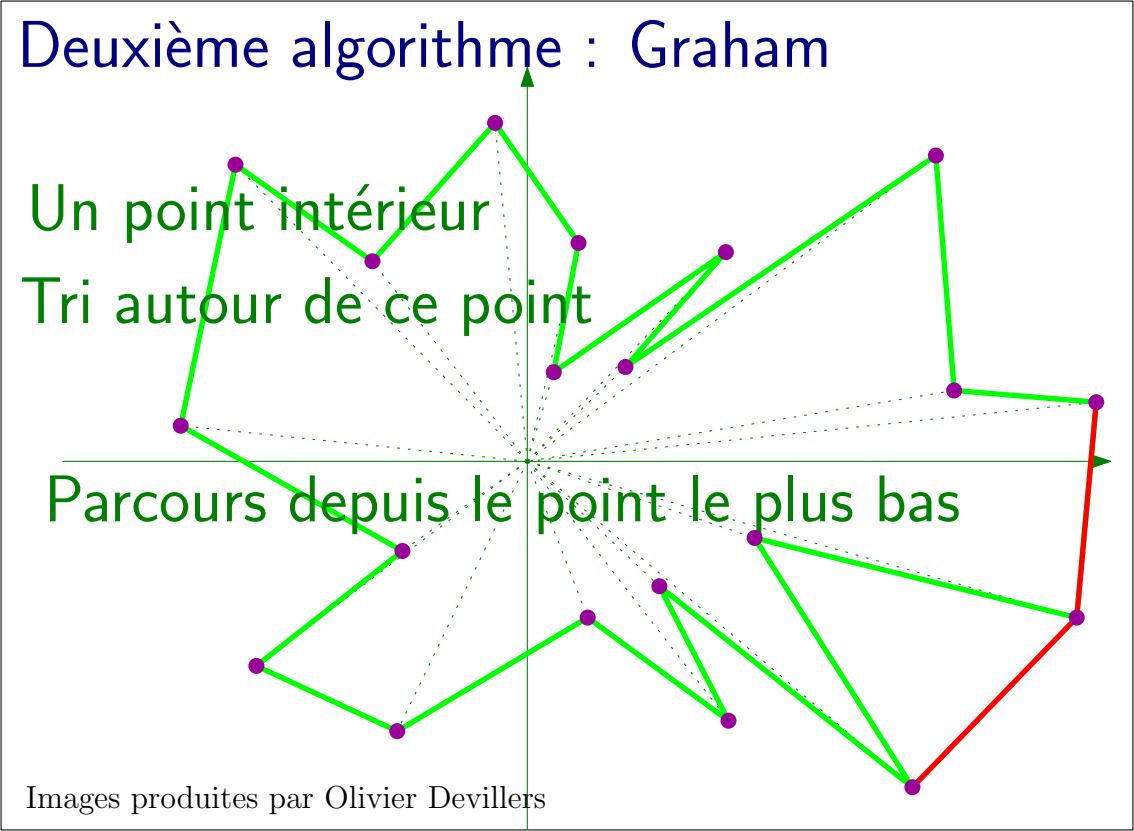
Images produites par Olivier Devillers

Deuxième algorithme : Graham



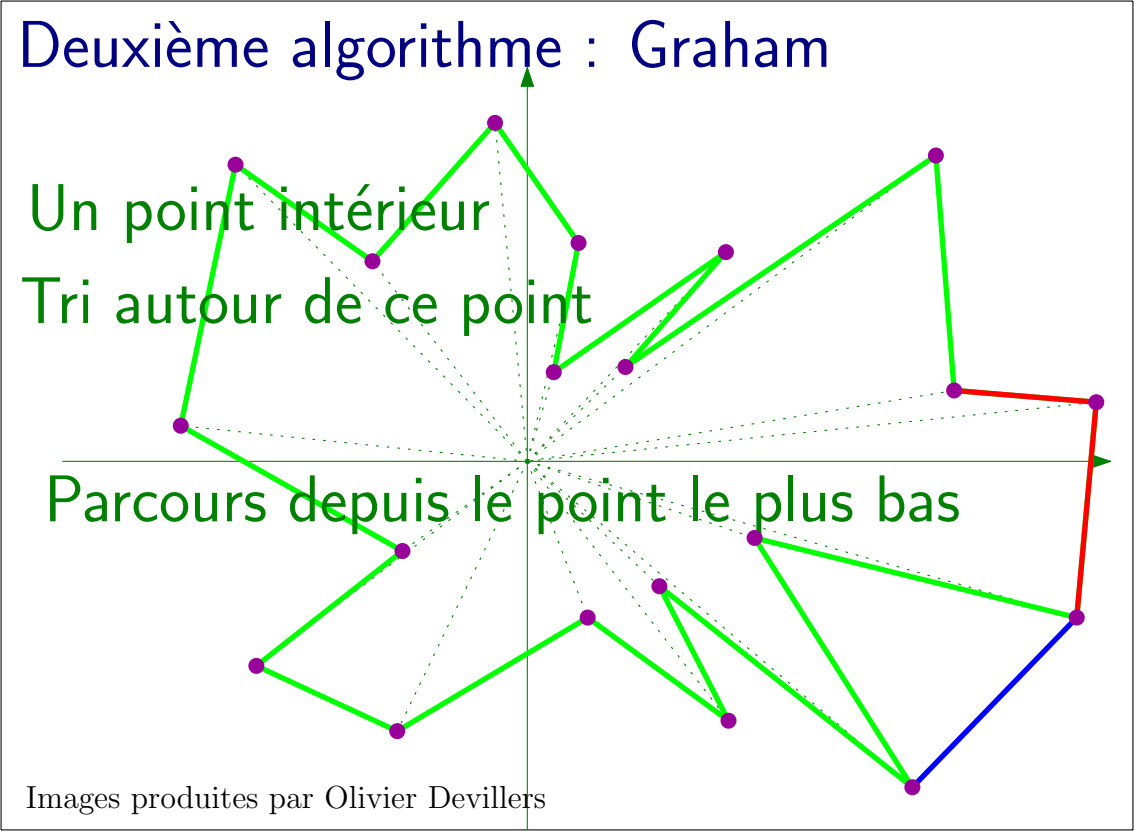
Images produites par Olivier Devillers

Deuxième algorithme : Graham



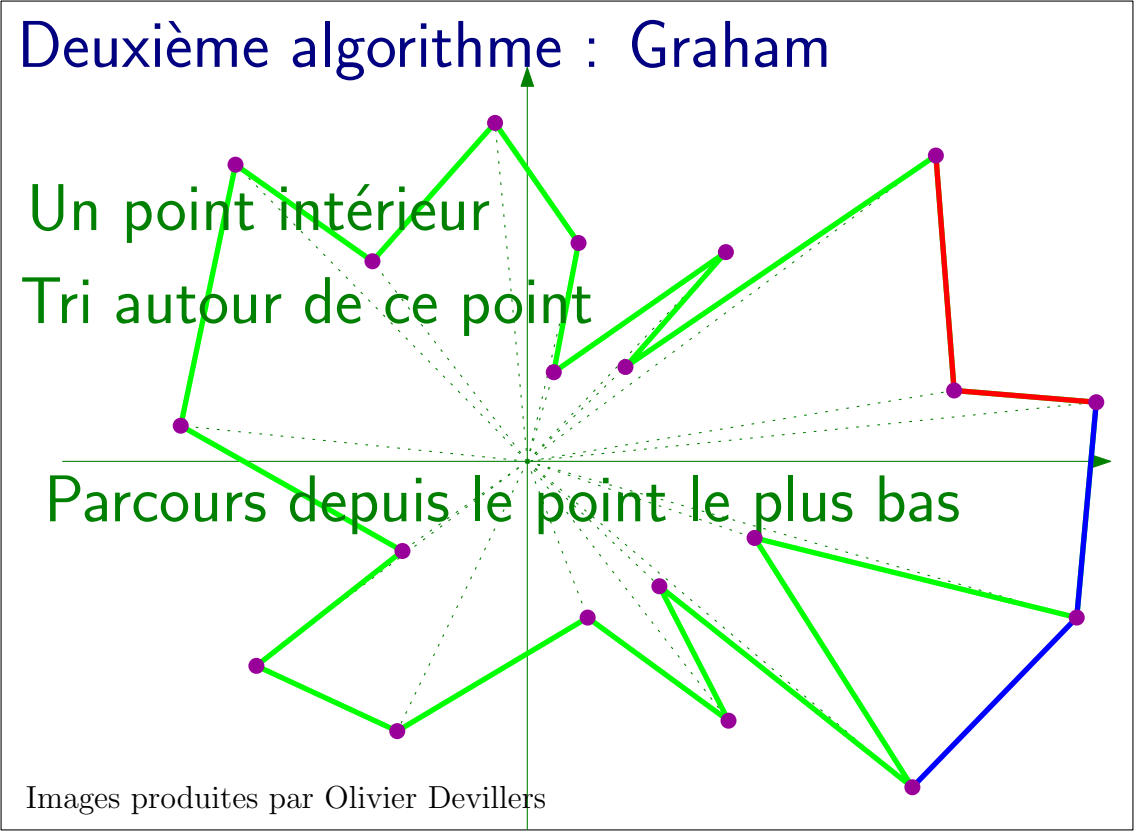
Images produites par Olivier Devillers

Deuxième algorithme : Graham



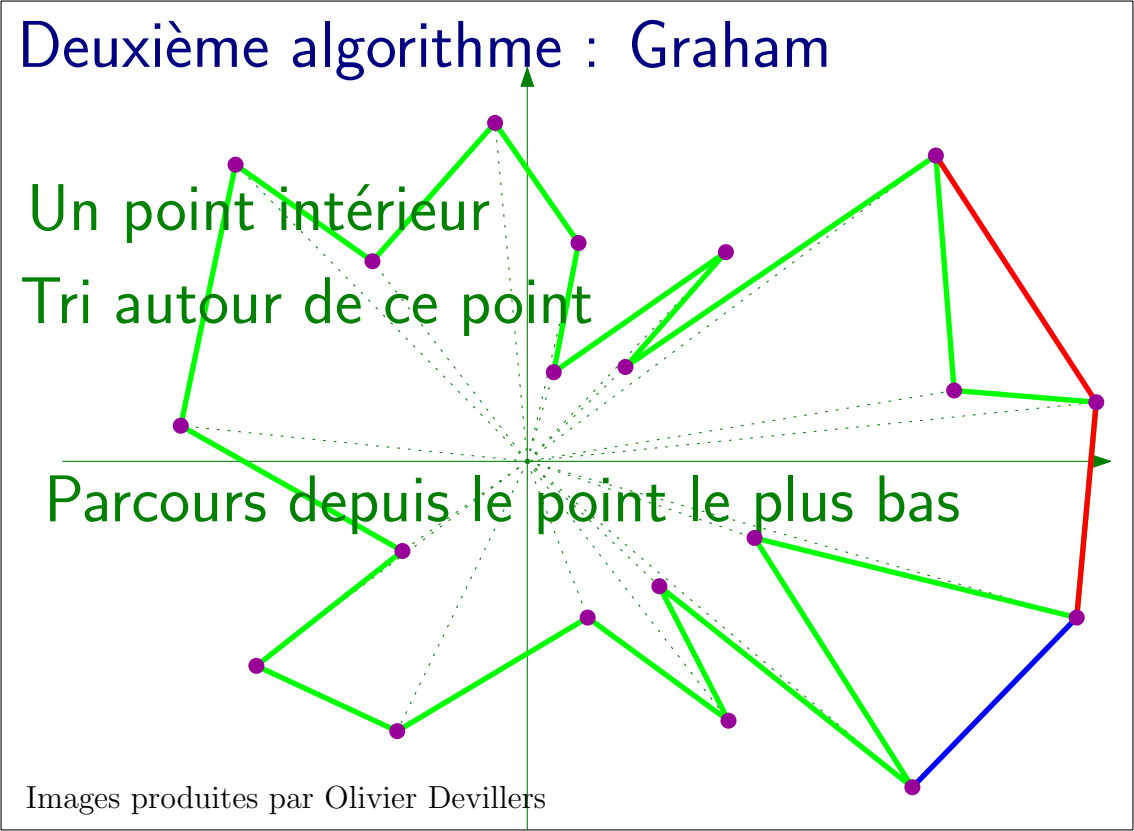
Images produites par Olivier Devillers

Deuxième algorithme : Graham



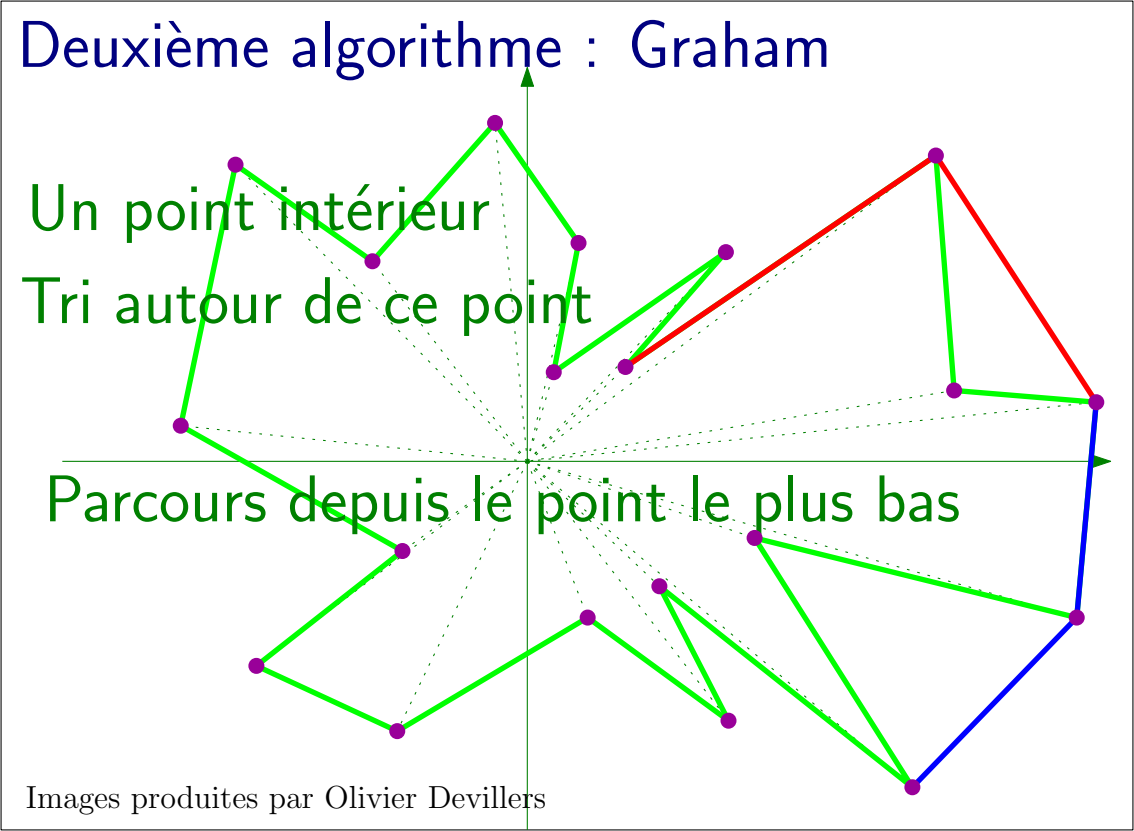
Images produites par Olivier Devillers

Deuxième algorithme : Graham



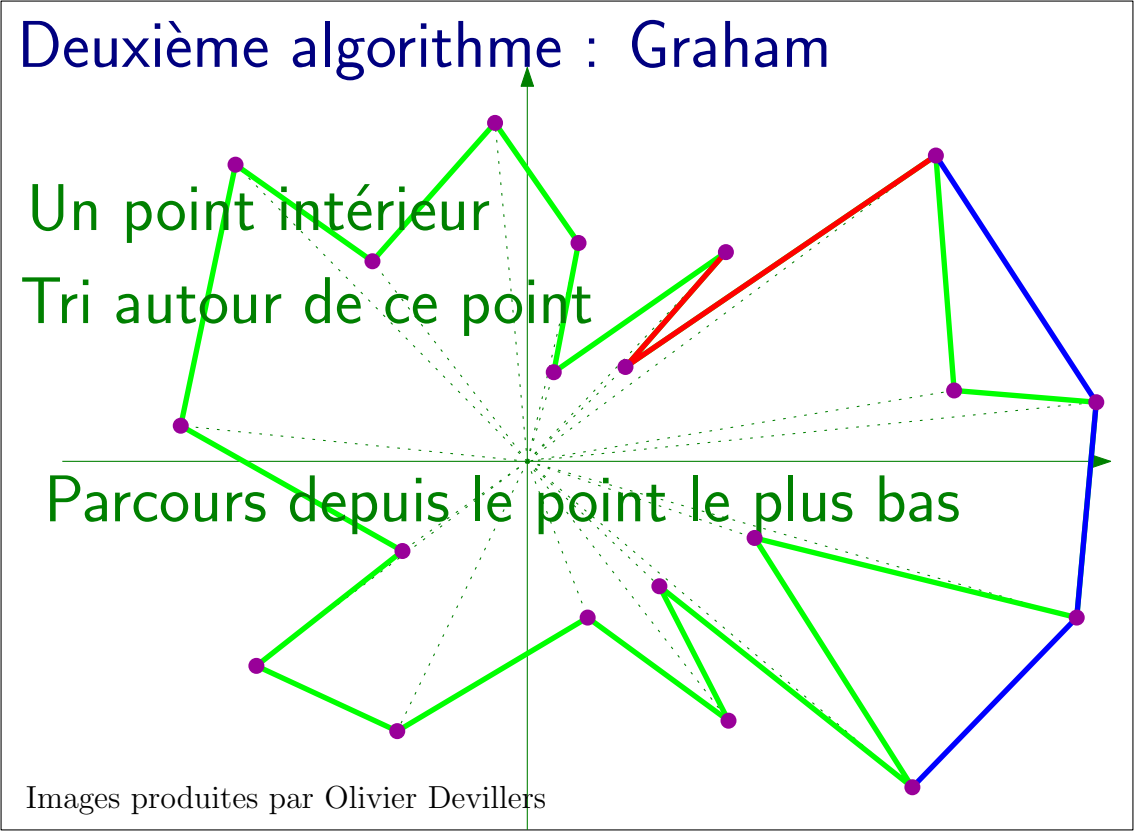
Images produites par Olivier Devillers

Deuxième algorithme : Graham



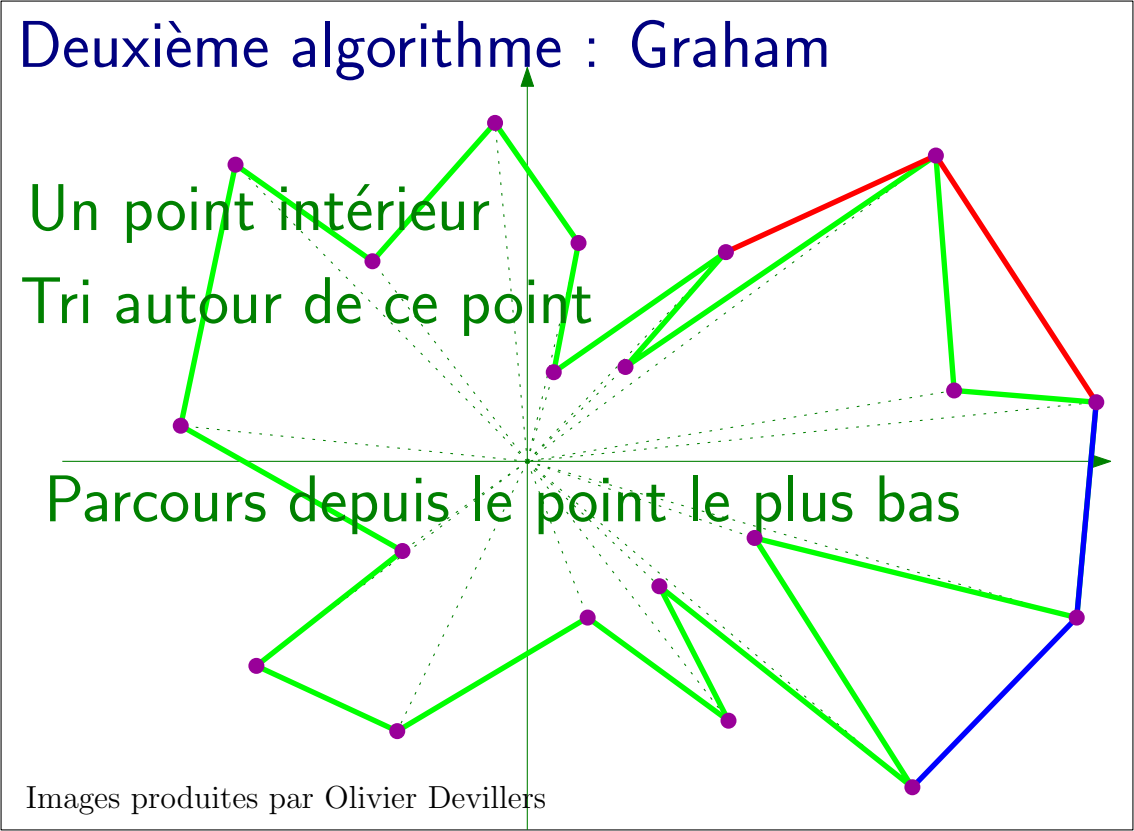
Images produites par Olivier Devillers

Deuxième algorithme : Graham



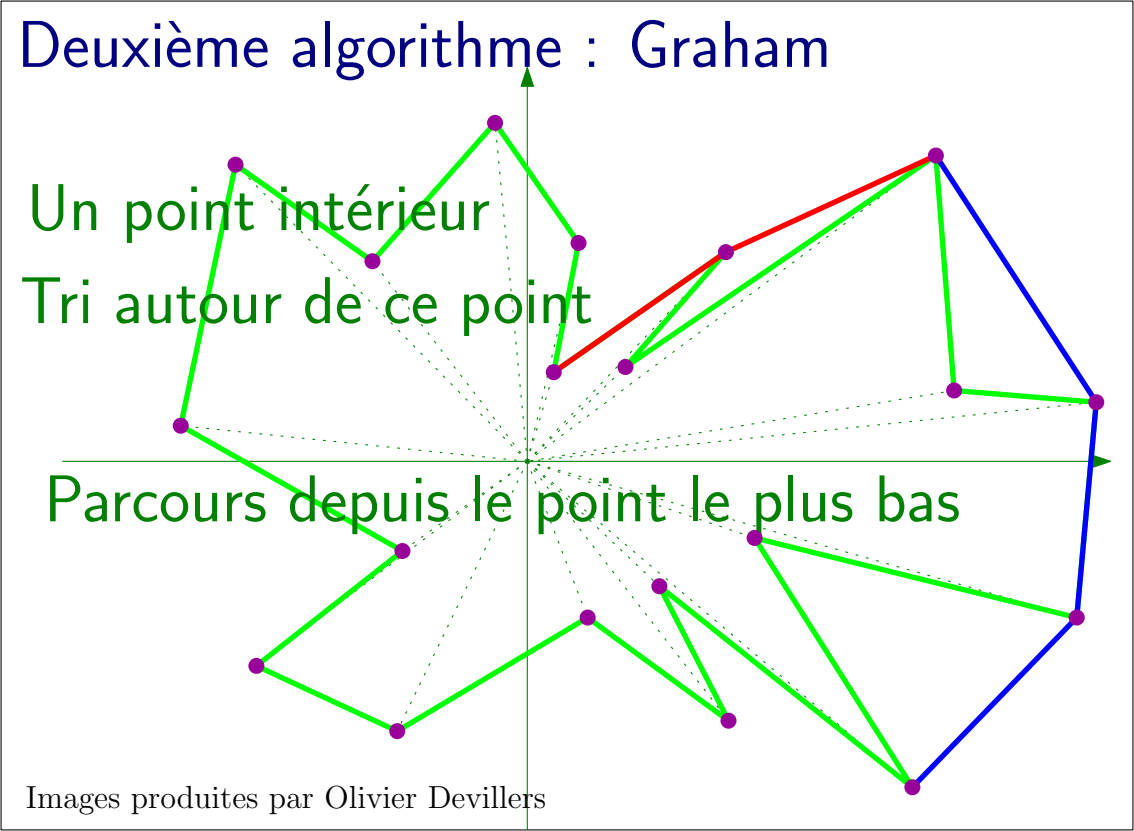
Images produites par Olivier Devillers

Deuxième algorithme : Graham



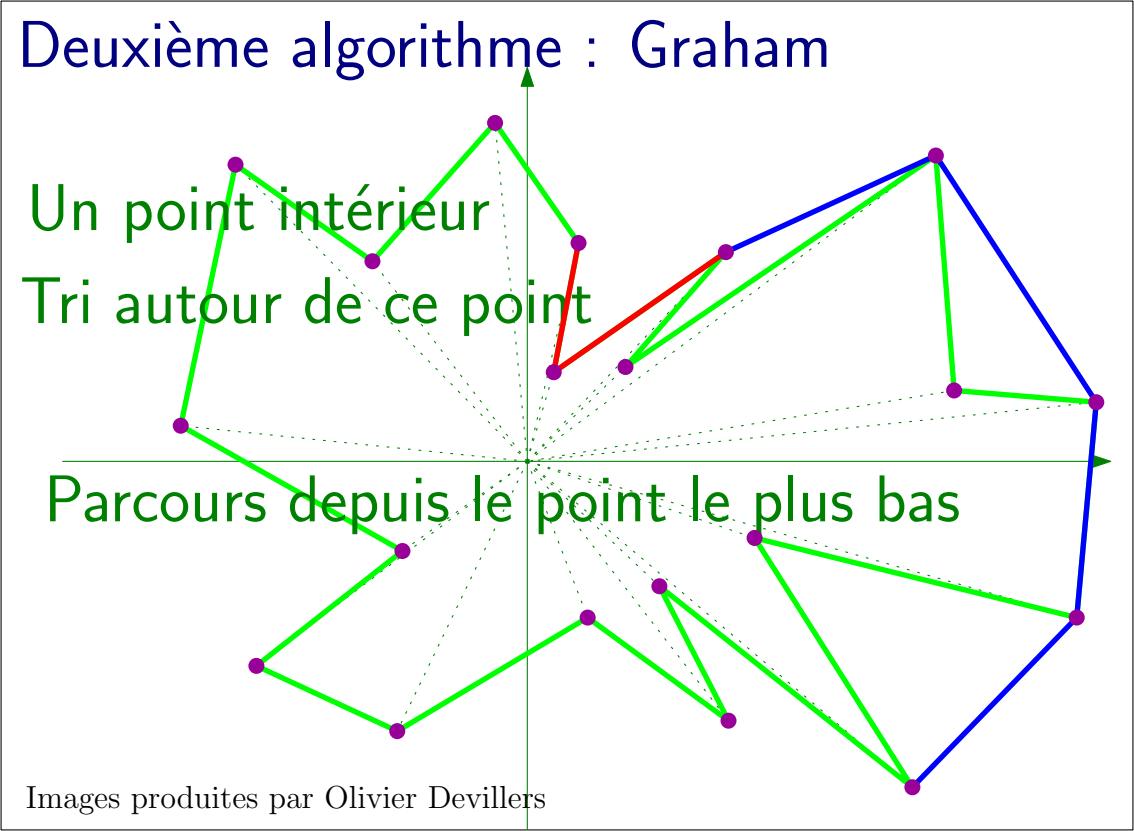
Images produites par Olivier Devillers

Deuxième algorithme : Graham



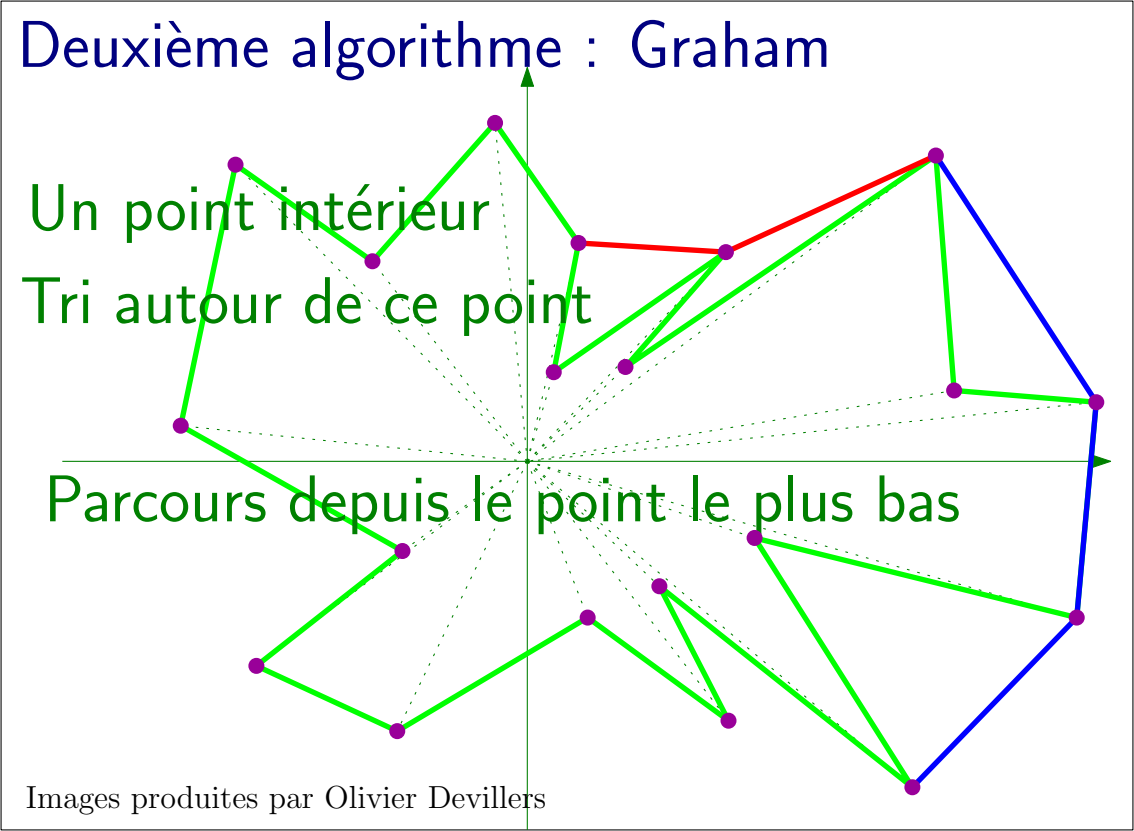
Images produites par Olivier Devillers

Deuxième algorithme : Graham



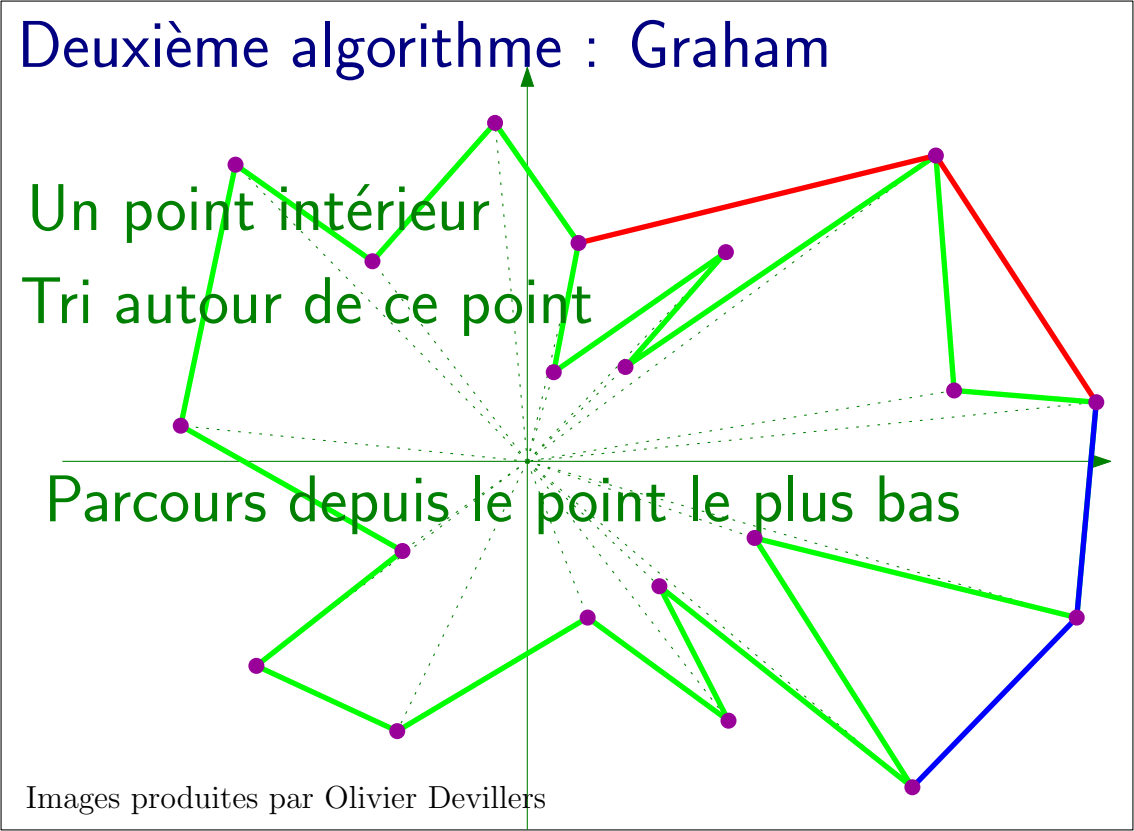
Images produites par Olivier Devillers

Deuxième algorithme : Graham



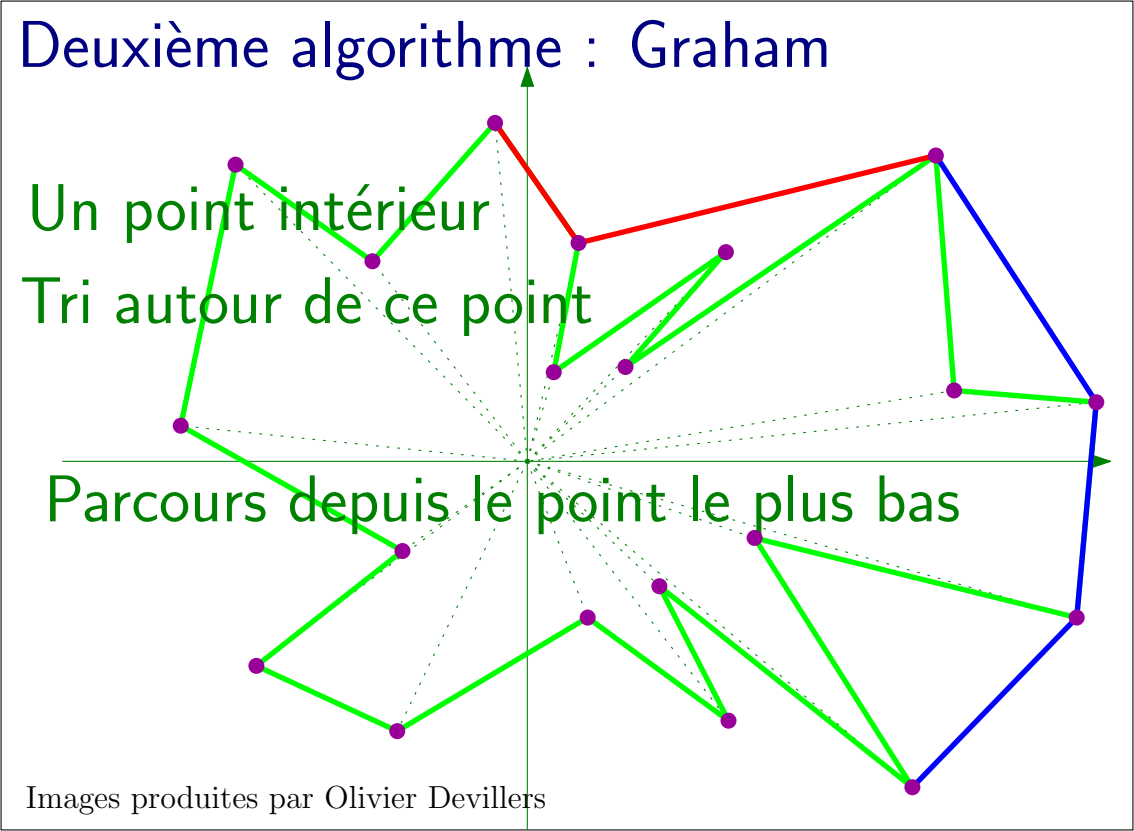
Images produites par Olivier Devillers

Deuxième algorithme : Graham



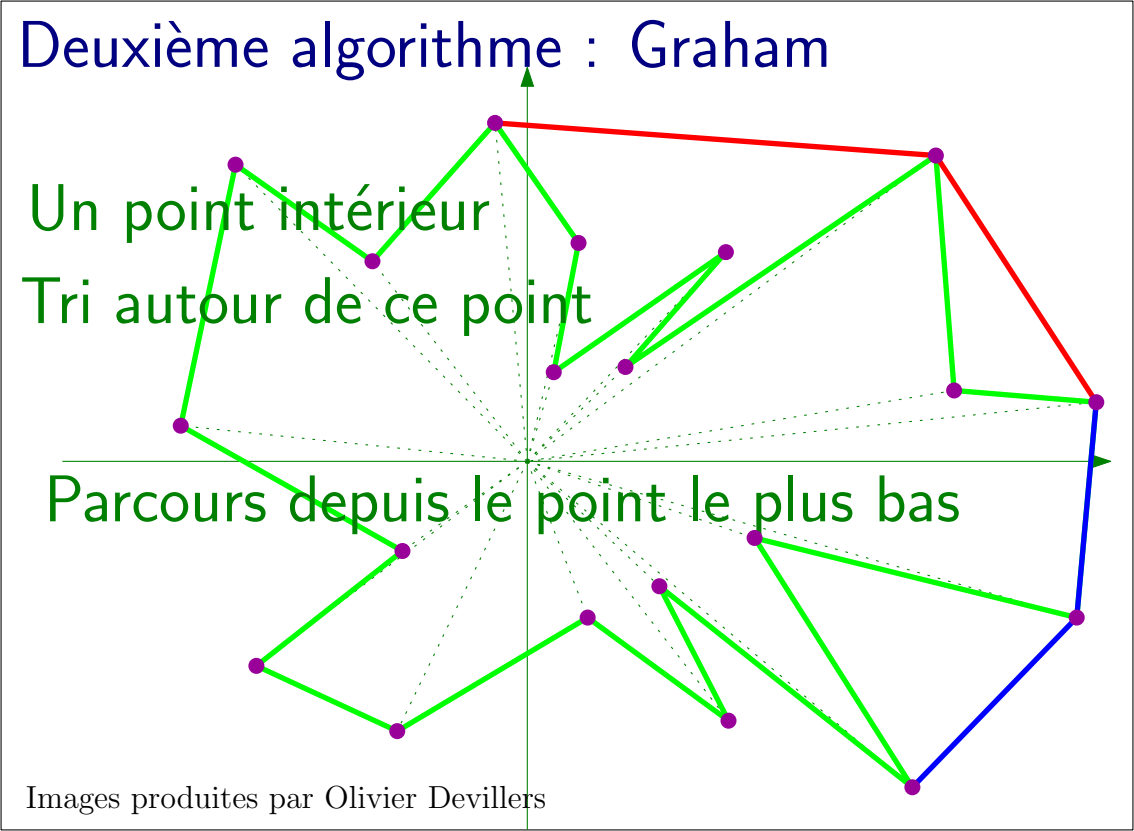
Images produites par Olivier Devillers

Deuxième algorithme : Graham



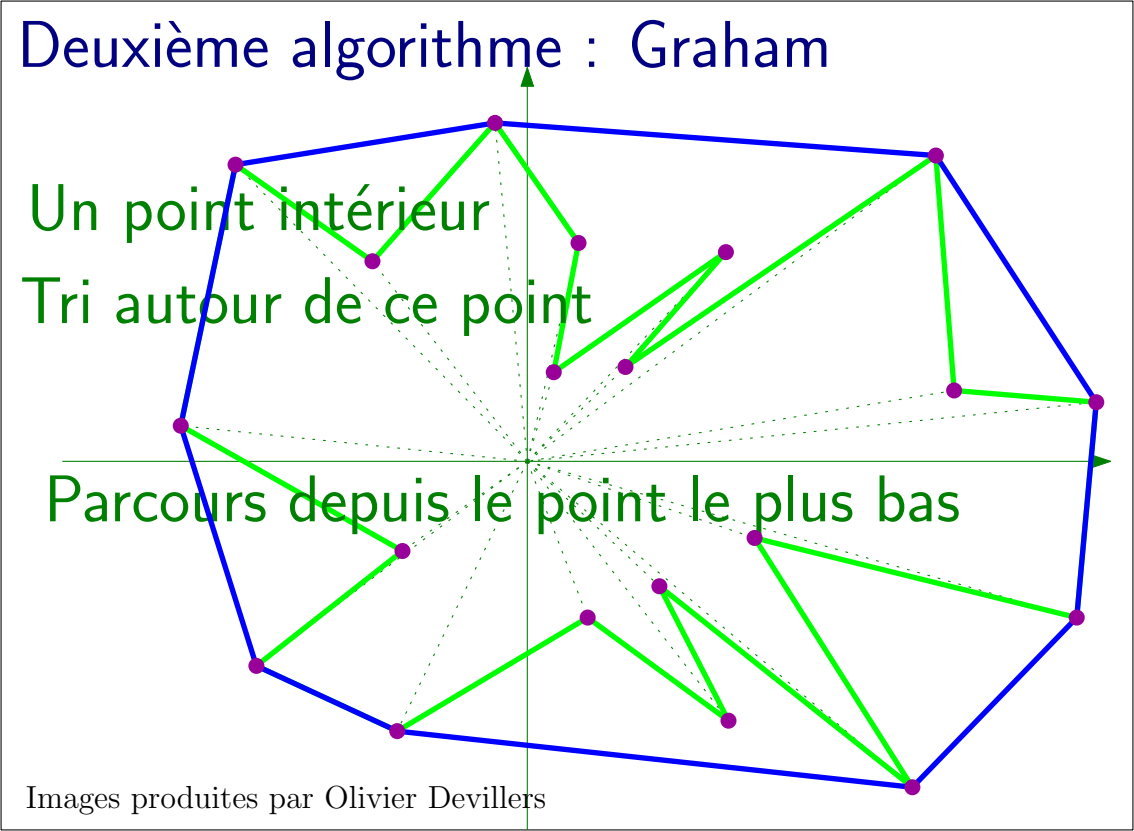
Images produites par Olivier Devillers

Deuxième algorithme : Graham



Images produites par Olivier Devillers

Deuxième algorithme : Graham



Images produites par Olivier Devillers

Deuxième algorithme : Graham

entrée : S un ensemble de points.
origine $=$ barycentre de 3 points de S;
trier S autour de l'origine;
$u =$ le point le plus bas de S;
$v = u$;
tant que $v.suivant \neq u$
    si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche
        $v = v.suivant$;
    sinon
        $v.suivant = v.suivant.suivant$;
        si $v \neq u$ $v = v.precedent$;

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.
origine $=$ barycentre de 3 points de S;
trier S autour de l'origine;
$u =$ le point le plus bas de S;
$v = u$;
tant que $v.suivant \neq u$
    si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche
        $v = v.suivant$;
    sinon
        $v.suivant = v.suivant.suivant$;
        si $v \neq u$ $v = v.precedent$;

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.
origine $=$ barycentre de 3 points de S;
trier S autour de l'origine;
$u =$ le point le plus bas de S;
$v = u$;
tant que $v.suivant \neq u$
    si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche
        $v = v.suivant$;
    sinon
        $v.suivant = v.suivant.suivant$;
        si $v \neq u$ $v = v.precedent$;

Images produites par Olivier Devillers

Deuxième algorithme : Graham
Complexité

entrée : S un ensemble de points.
origine = barycentre de 3 points de S;
trier S autour de l'origine;
u = le point le plus bas de S;
$v = u$;
tant que $v.suivant \neq u$
    si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche
        $v = v.suivant$;
    sinon
        $v.suivant = v.suivant.suivant$;
        si $v \neq u$ $v = v.precedent$;

$O(n \log n)$

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S;

$v = u$;

tant que $v.suivant \neq u$

  si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

    $v = v.suivant$;

  sinon

    $v.suivant = v.suivant.suivant$;

    si $v \neq u$ $v = v.precedent$;

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S; $O(n)$

$v = u$;

tant que $v.suivant \neq u$

  si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

    $v = v.suivant$;

  sinon

    $v.suivant = v.suivant.suivant$;

    si $v \neq u$ $v = v.precedent$;

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S; $O(n)$

$v = u$;

tant que $v.suivant \neq u$

si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

$v = v.suivant$;

sinon

$v.suivant = v.suivant.suivant$;

si $v \neq u$ $v = v.precedent$;

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S; $O(n)$

$v = u$;

tant que $v.suivant \neq u$

si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

$v = v.suivant$;

sinon

$v.suivant = v.suivant.suivant$;

si $v \neq u$ $v = v.precedent$;

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S; $O(n)$

$v = u$;

tant que $v.suivant \neq u$

  si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

    $v = v.suivant$;

  sinon

    $v.suivant = v.suivant.suivant$;

    si $v \neq u$ $v = v.precedent$;

au plus n suppressions

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S; $O(n)$

$v = u$;

tant que $v.suivant \neq u$

si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

$v = v.suivant$;

sinon

$v.suivant = v.suivant.suivant$;

si $v \neq u$ $v = v.precedent$;

au plus n suppressions

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S; $O(n)$

$v = u$;

tant que $v.suivant \neq u$

si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

$v = v.suivant$;

au plus n fois

sinon

$v.suivant = v.suivant.suivant$;

si $v \neq u$ $v = v.precedent$;

au plus n suppressions

Images produites par Olivier Devillers

Deuxième algorithme : Graham

Complexité

entrée : S un ensemble de points.

origine = barycentre de 3 points de S; $O(1)$

trier S autour de l'origine; $O(n \log n)$

u = le point le plus bas de S; $O(n)$

$v = u$;

tant que $v.suivant \neq u$

  si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche

    $v = v.suivant$;

  sinon

    $v.suivant = v.suivant.suivant$;

    si $v \neq u$ $v = v.precedent$;

$O(n)$

Images produites par Olivier Devillers

Deuxième algorithme : Graham

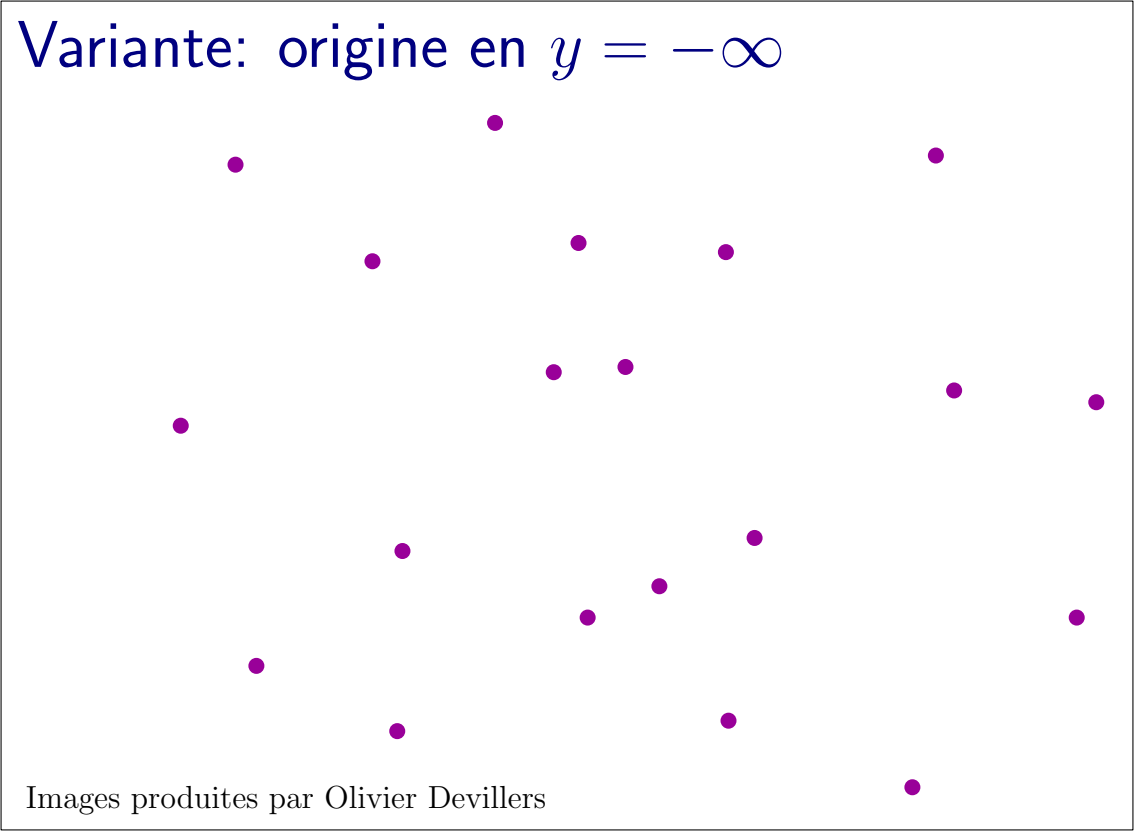
Complexité

entrée : S un ensemble de points.
origine $=$ barycentre de 3 points de S;
trier S autour de l'origine;
$u =$ le point le plus bas de S;
$v = u$;
tant que $v.suivant \neq u$
    si $(v, v.suivant, v.suivant.suivant)$ tourne à gauche
        $v = v.suivant$;
    sinon
        $v.suivant = v.suivant.suivant$;
        si $v \neq u$ $v = v.precedent$;

$O(n \log n)$

Images produites par Olivier Devillers

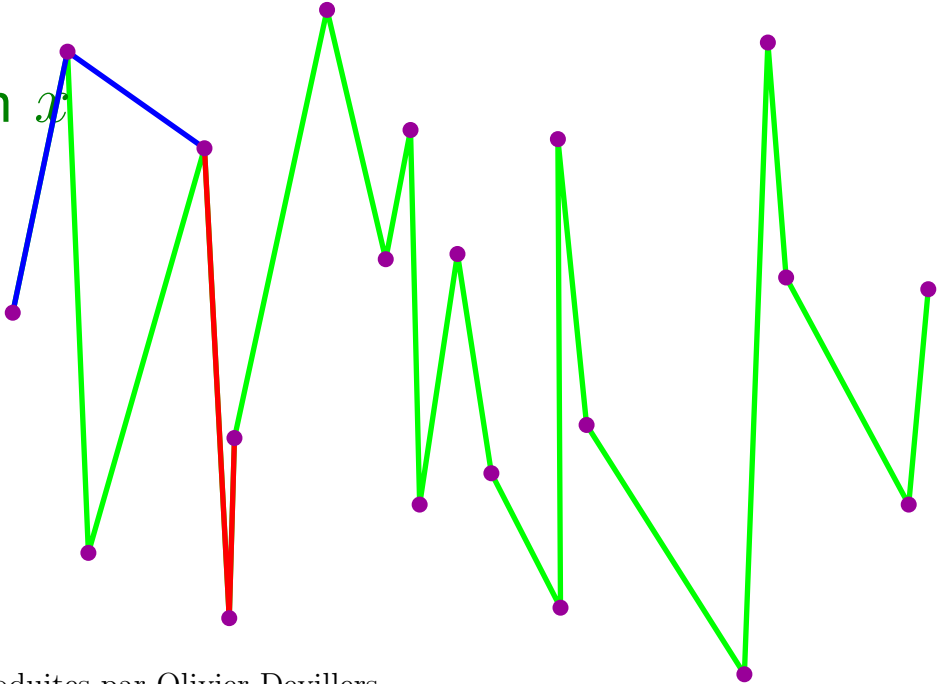
Variante: origine en $y = -\infty$



Images produites par Olivier Devillers

Variante: origine en $y = -\infty$

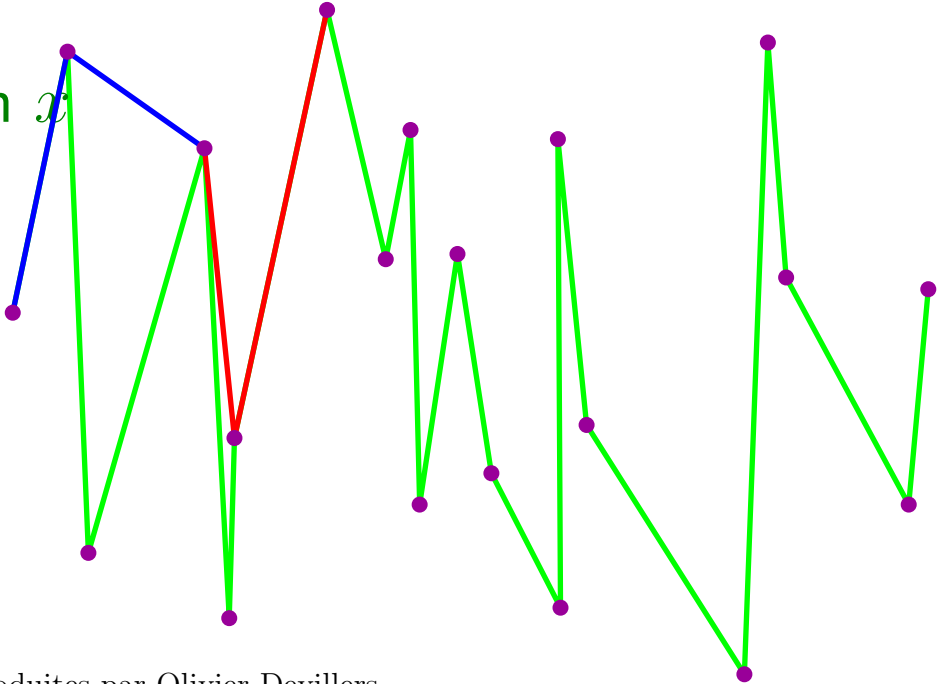
Tri en x



Images produites par Olivier Devillers

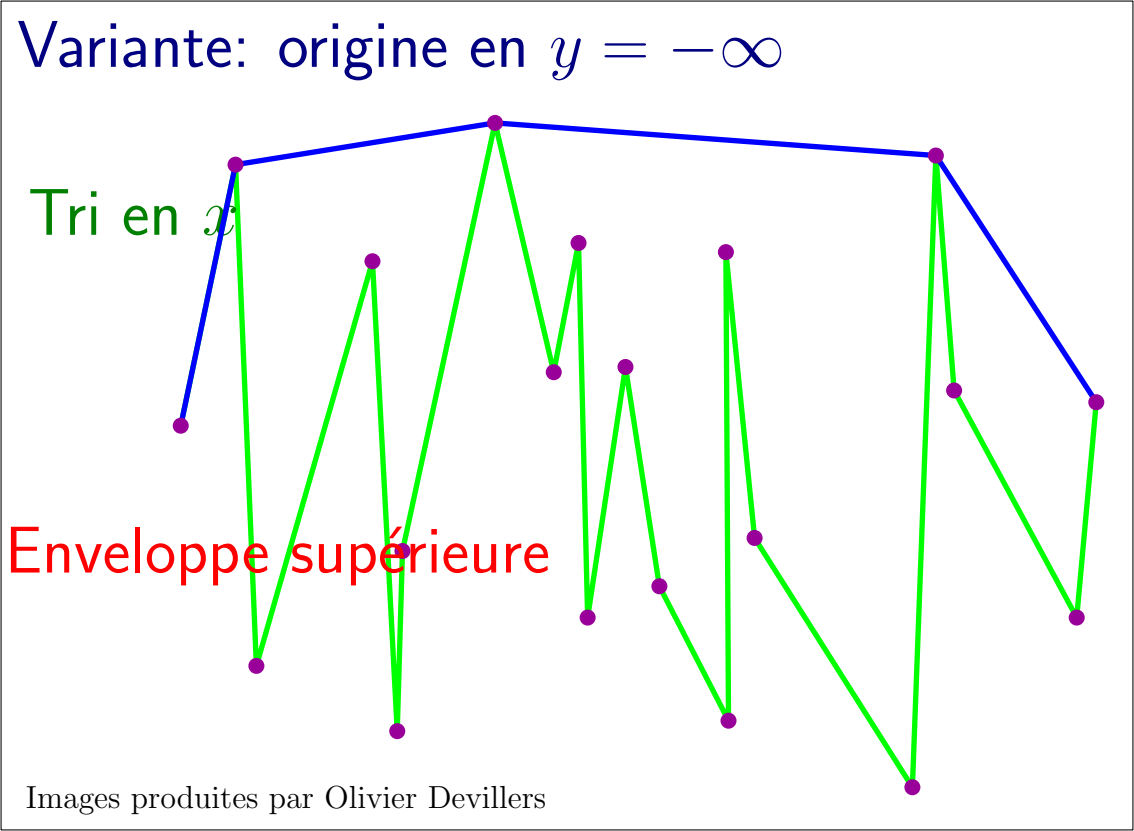
Variante: origine en $y = -\infty$

Tri en x



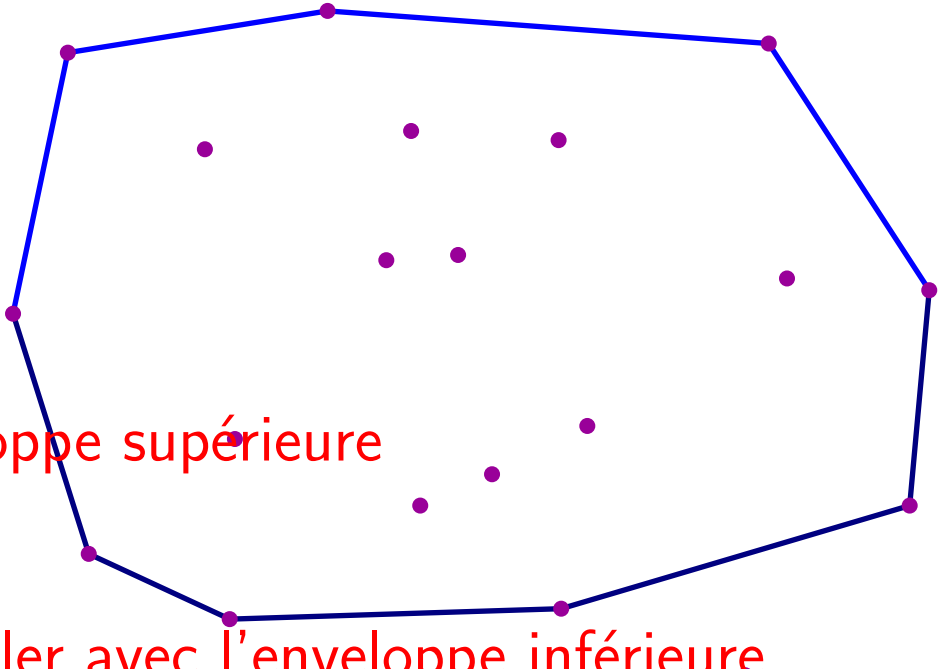
Images produites par Olivier Devillers

Variante: origine en $y = -\infty$



Images produites par Olivier Devillers

Variante: origine en $y = -\infty$

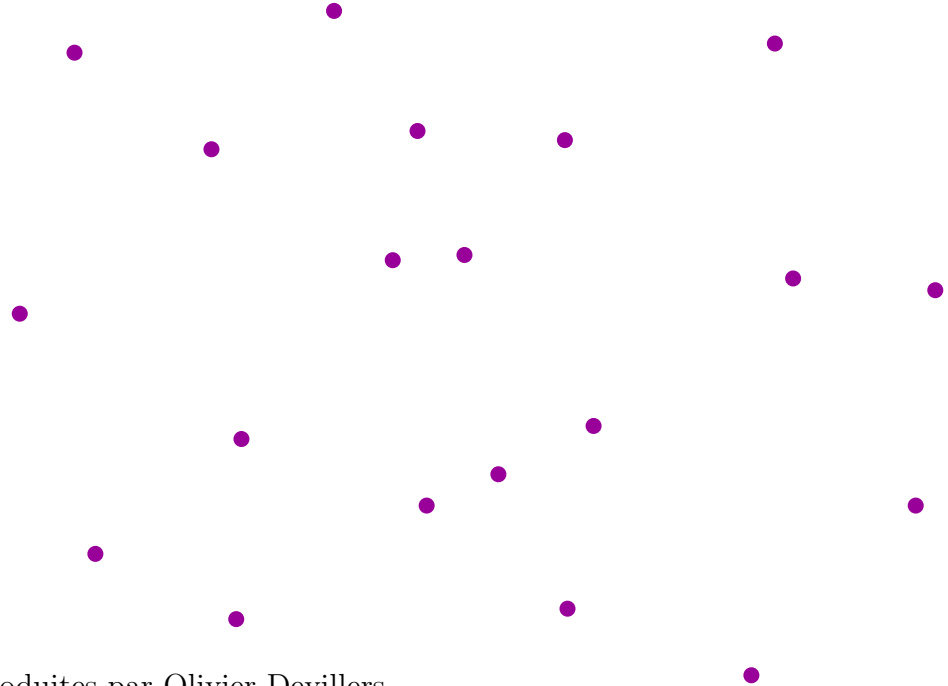


Enveloppe supérieure

À recoler avec l'enveloppe inférieure

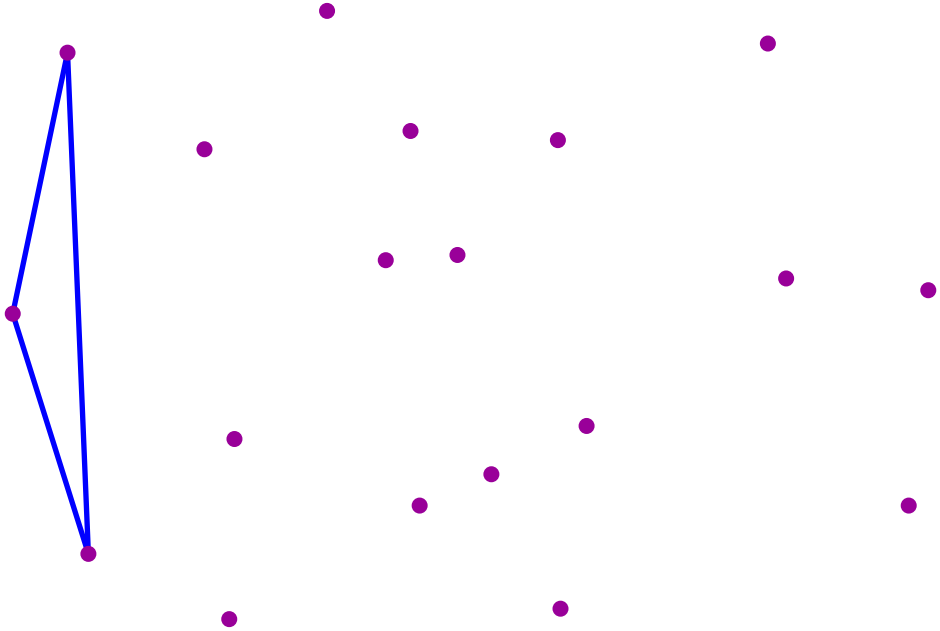
Images produites par Olivier Devillers

Autre algorithme par tri en x



Images produites par Olivier Devillers

Autre algorithme par tri en x

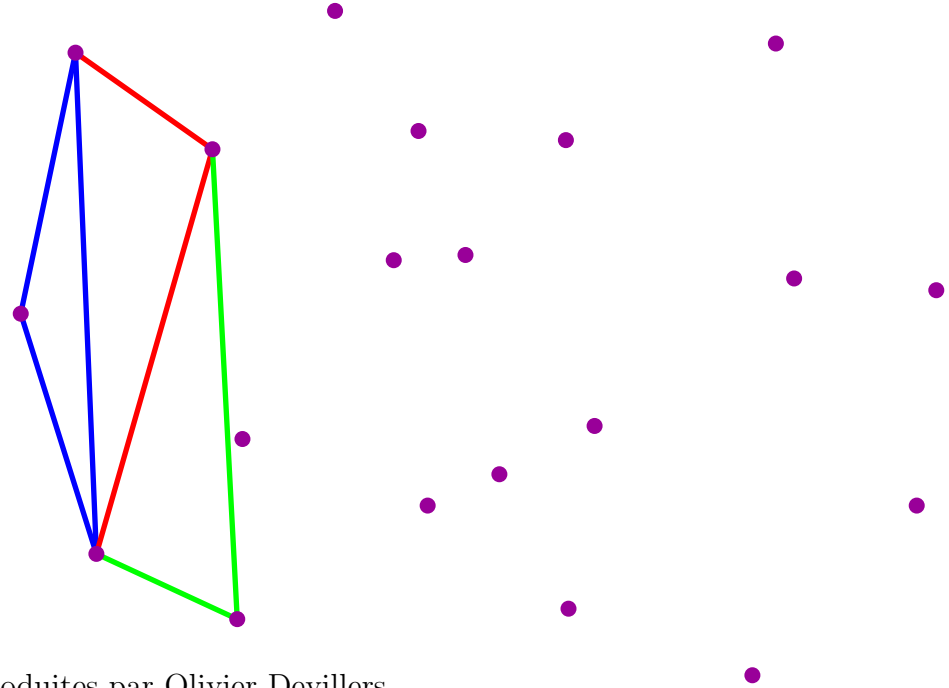


Images produites par Olivier Devillers

Autre algorithme par tri en x

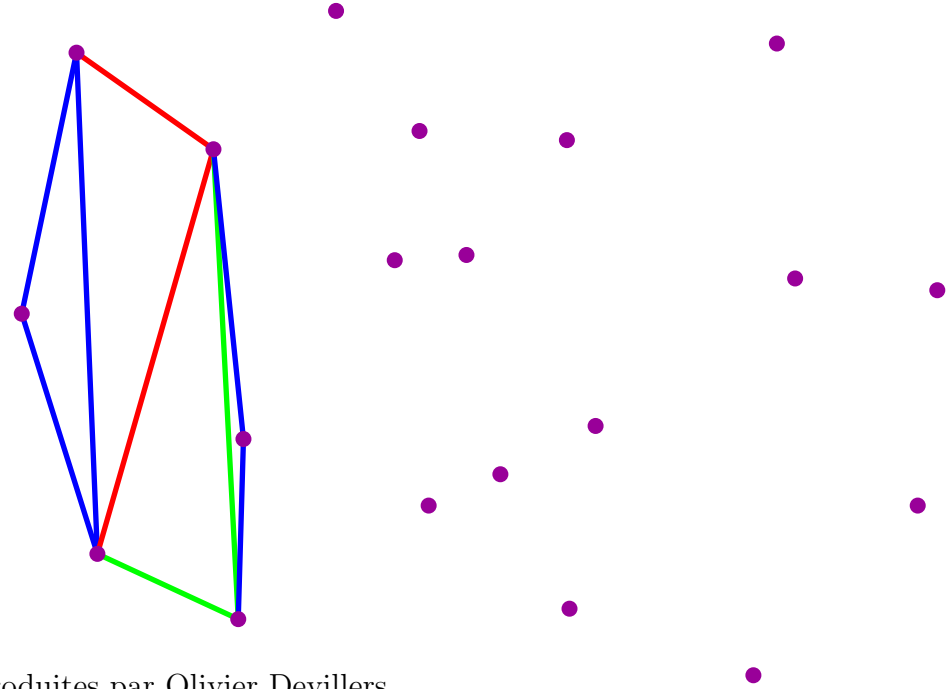
Images produites par Olivier Devillers

Autre algorithme par tri en x



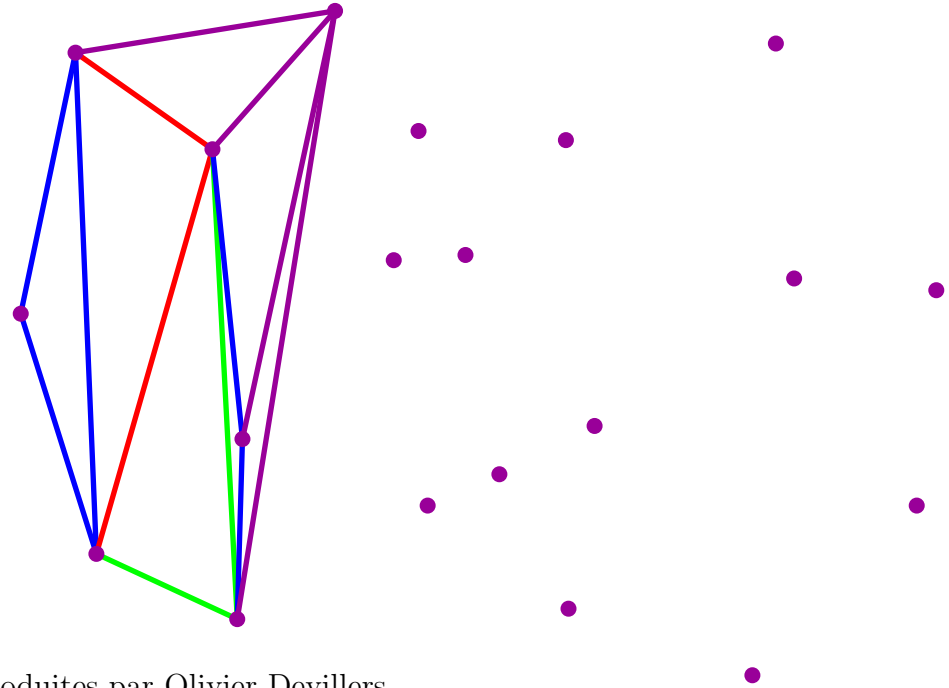
Images produites par Olivier Devillers

Autre algorithme par tri en x



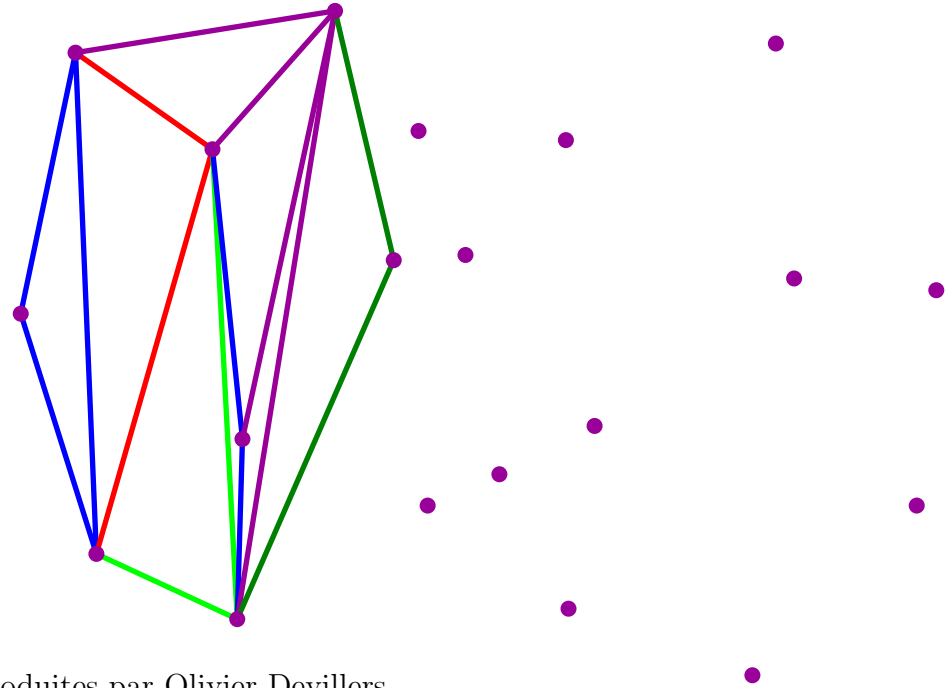
Images produites par Olivier Devillers

Autre algorithme par tri en x



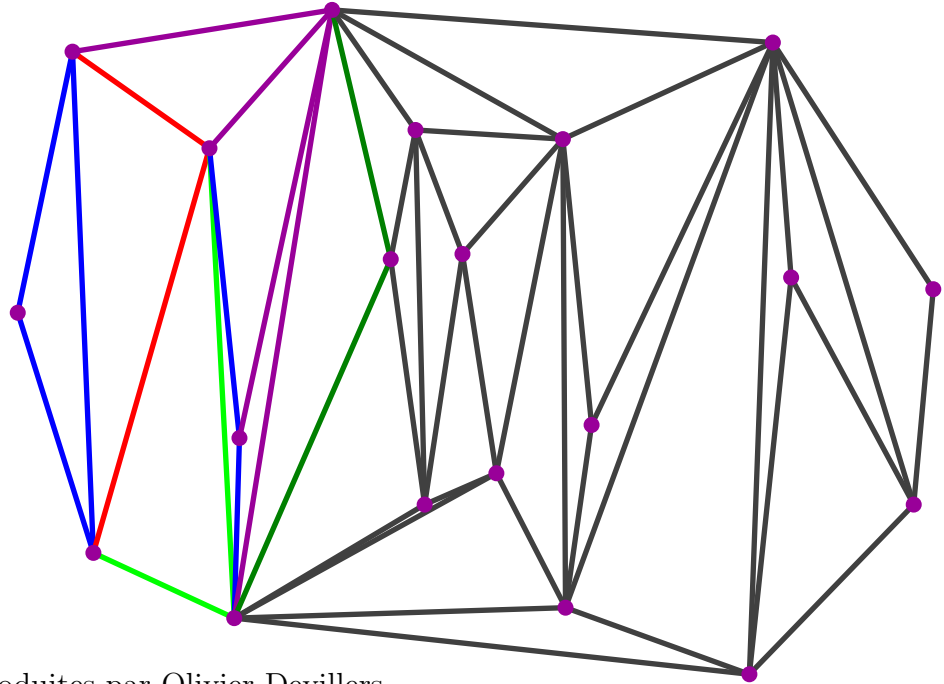
Images produites par Olivier Devillers

Autre algorithme par tri en x



Images produites par Olivier Devillers

Autre algorithme par tri en x



Images produites par Olivier Devillers

Autre algorithme par tri en x

entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

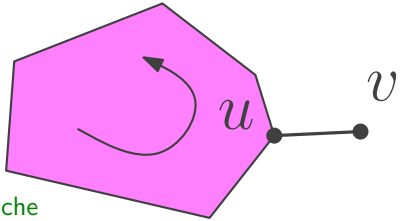
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

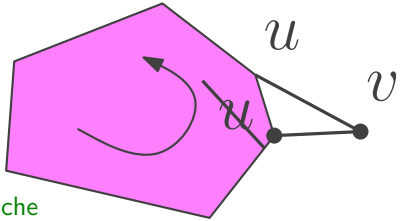
Autre algorithme par tri en x



entrée : S un ensemble de points.
trier S en x;
initier une liste circulaire avec les 3 points les plus à gauche
tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;
Pour v le prochain en x

w=u
tant que $(v, u, u.suivant)$ tourne à droite
$u = u.suivant$;
$v.suivant = u$; $u.pred = v$;
tant que $(v, w, w.pred)$ tourne à gauche
$w = w.pred$;
$v.pred = w$; $w.suivant = v$;
$u = v$;

Autre algorithme par tri en x



entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

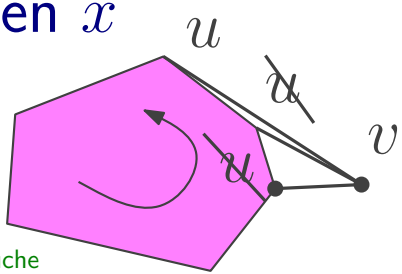
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x



entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

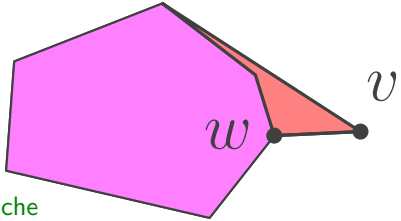
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x



entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

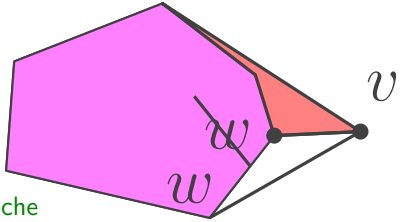
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x



entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

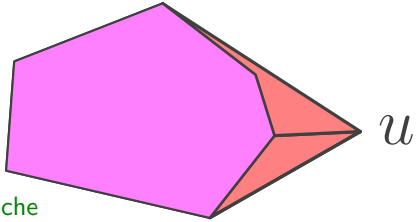
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x



entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x

Complexité

entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x

Complexité

entrée : S un ensemble de points.

trier S en x; $O(n \log n)$

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x

entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

$v.suivant = u$; $u.pred = v$;

tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$u = v$;

Autre algorithme par tri en x

Complexité

entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

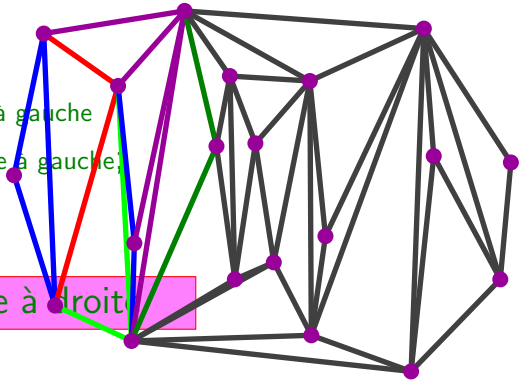
$v.suivant = u$; $u.pred = v$;

tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

w=u;



Dessiner une arête dans la triangulation

Autre algorithme par tri en x

Complexité

entrée : S un ensemble de points.

trier S en x;

initier une liste circulaire avec les 3 points les plus à gauche

tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;

Pour v le prochain en x

w=u

tant que $(v, u, u.suivant)$ tourne à droite

$u = u.suivant$;

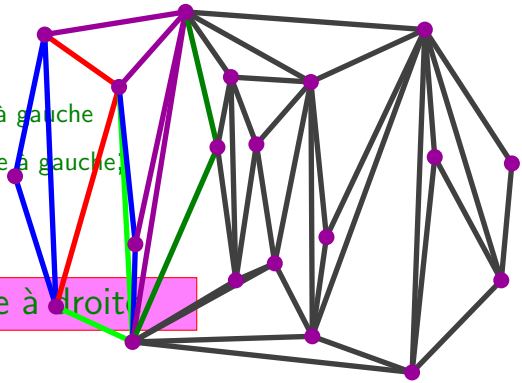
$v.suivant = u$; $u.pred = v$;

tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

$v.pred = w$; $w.suivant = v$;

$w = v$;



nb d'arêtes $\simeq 3n$

Dessiner une arête dans la triangulation

Autre algorithme par tri en x

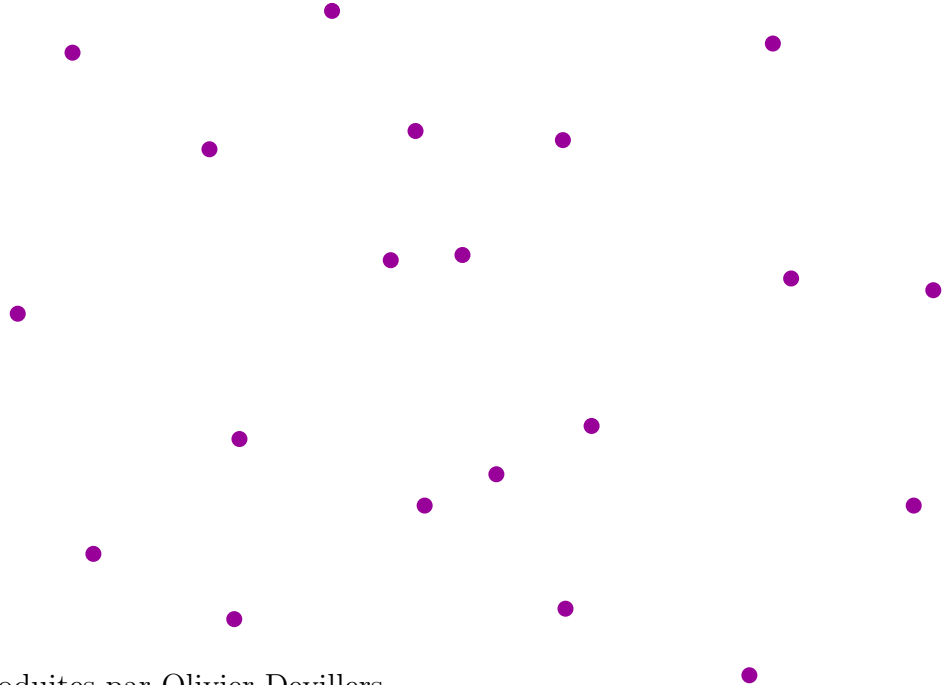
Complexité

$O(n \log n)$

entrée : S un ensemble de points.
trier S en x;
initier une liste circulaire avec les 3 points les plus à gauche
tel que u à droite et $u, u.suivant, u.suivant$ tourne à gauche;
Pour v le prochain en x

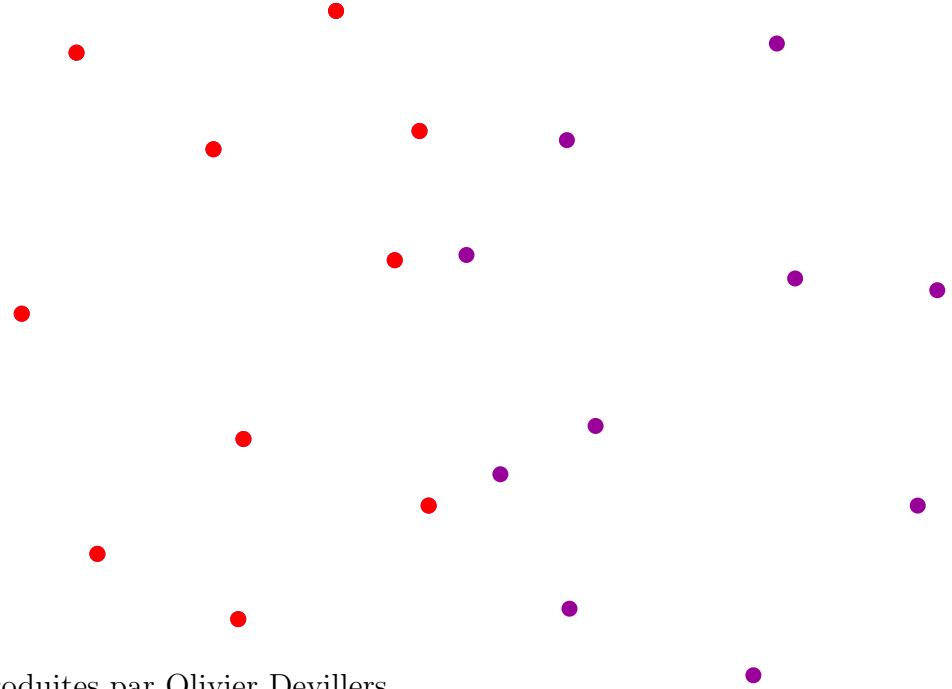
w=u
tant que $(v, u, u.suivant)$ tourne à droite
$u = u.suivant$;
$v.suivant = u$; $u.pred = v$;
tant que $(v, w, w.pred)$ tourne à gauche
$w = w.pred$;
$v.pred = w$; $w.suivant = v$;
$u = v$;

Un algorithme division fusion



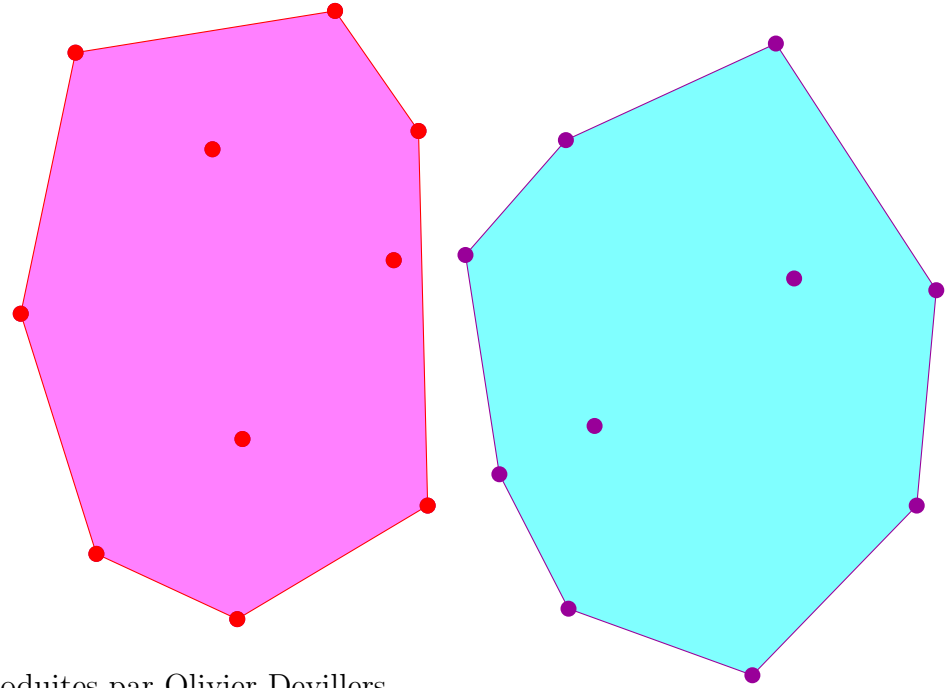
Images produites par Olivier Devillers

Un algorithme division fusion



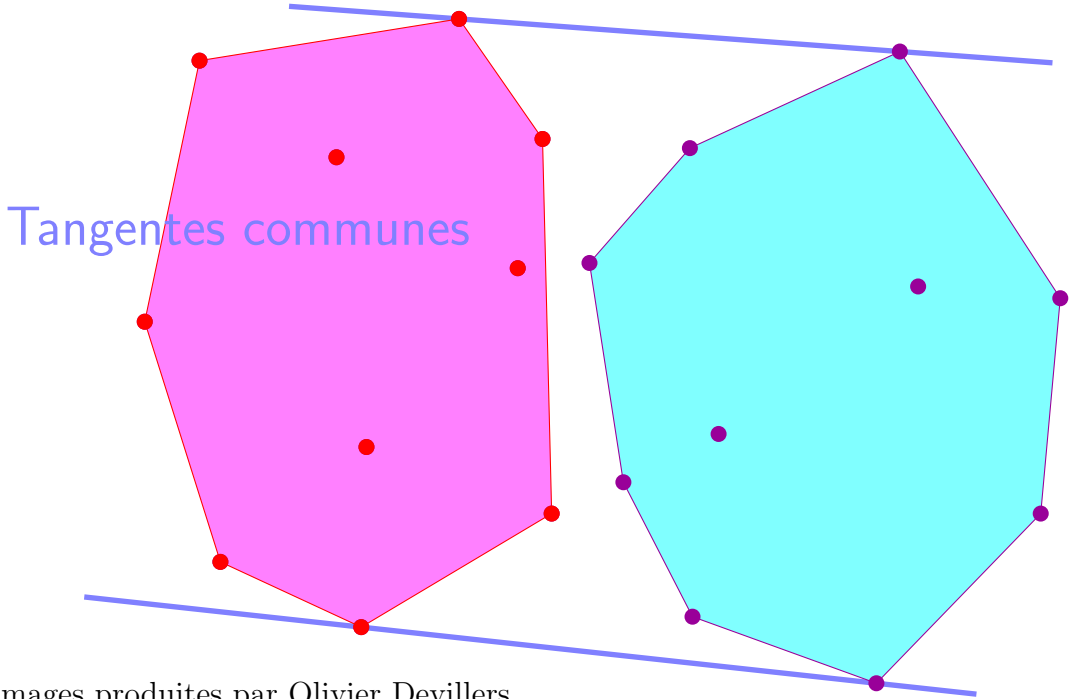
Images produites par Olivier Devillers

Un algorithme division fusion



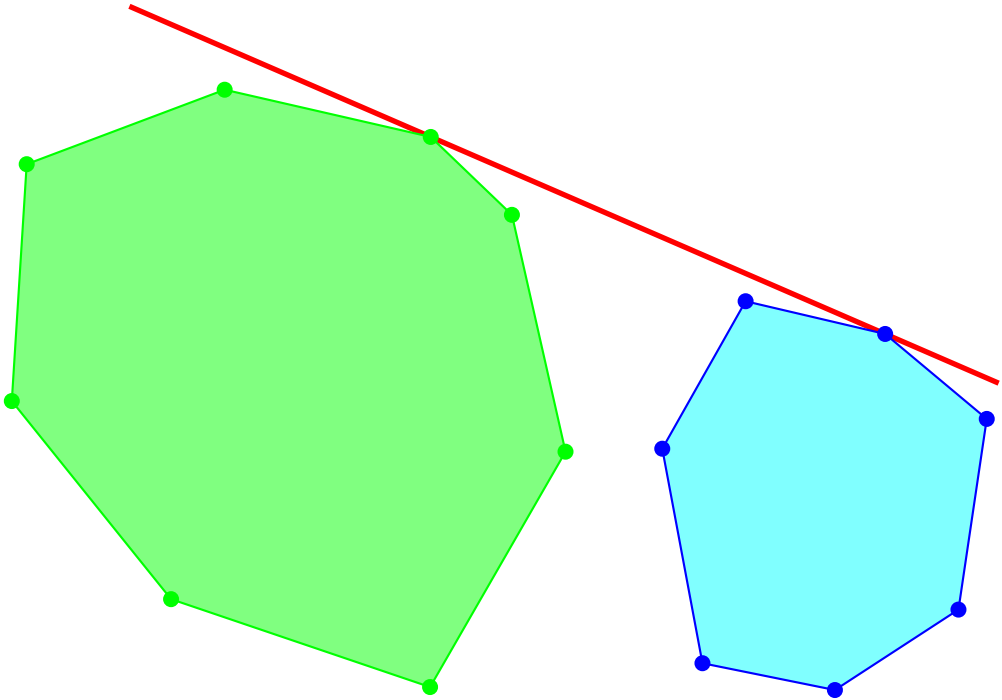
Images produites par Olivier Devillers

Un algorithme division fusion

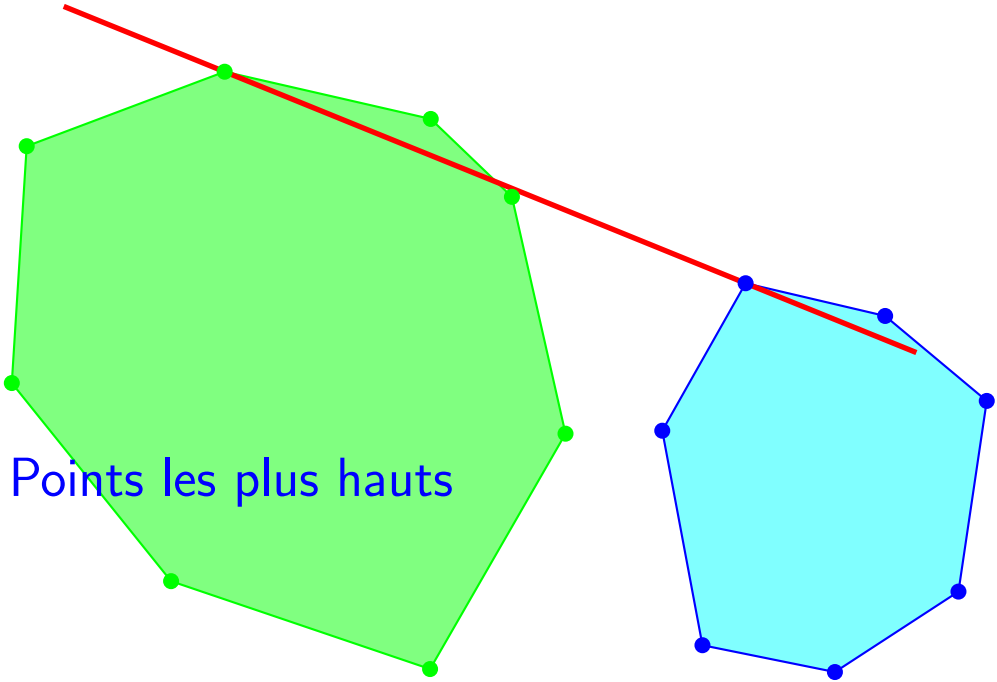


Images produites par Olivier Devillers

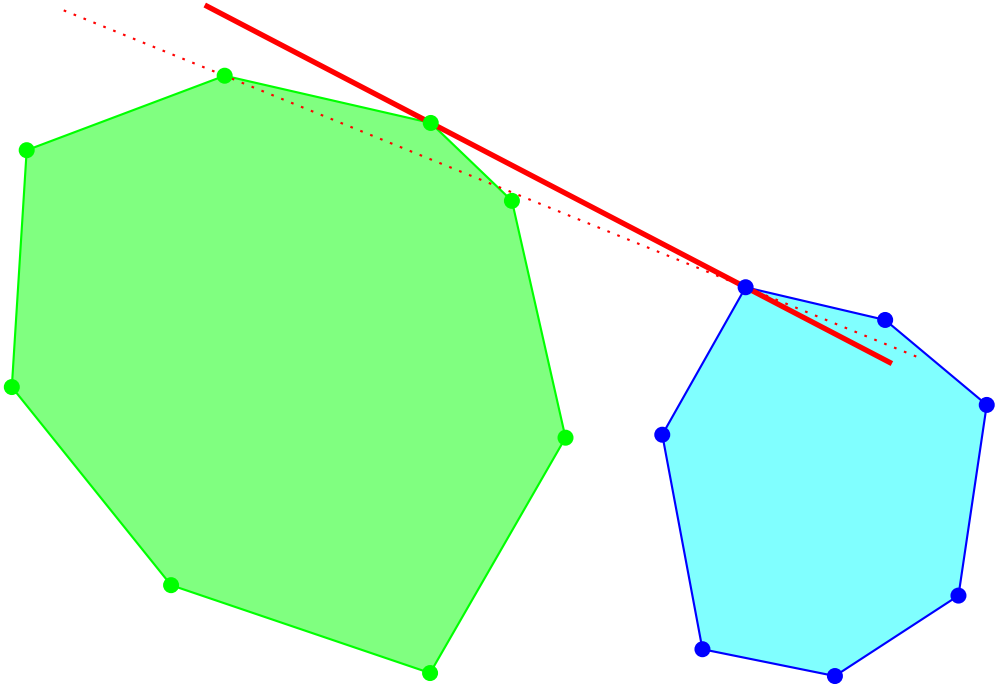
Tangente supérieure



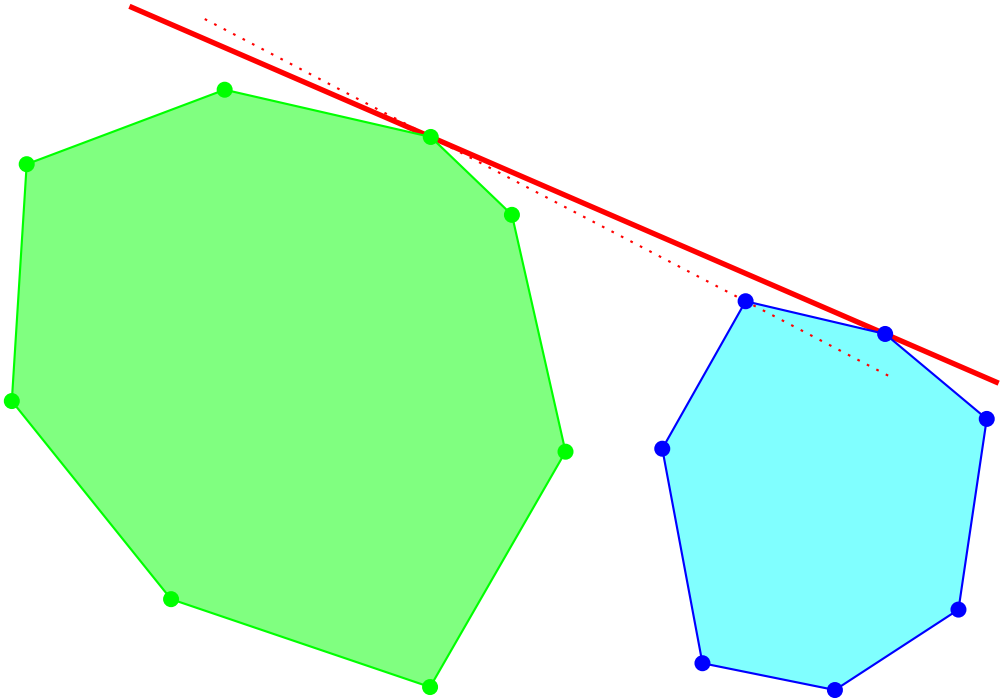
Tangente supérieure



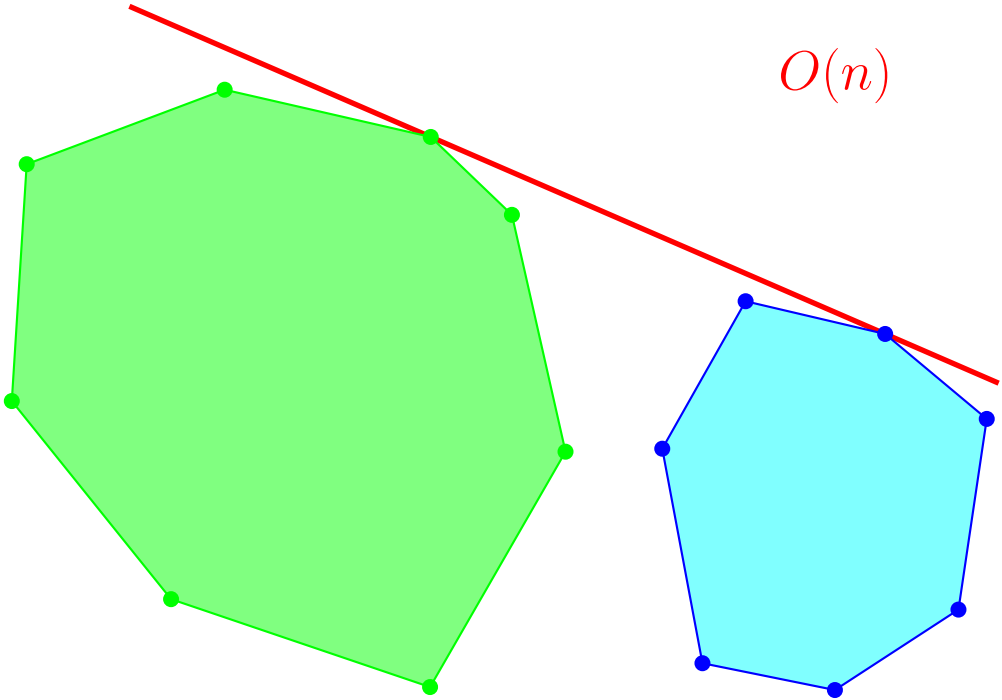
Tangente supérieure



Tangente supérieure



Tangente supérieure



Un algorithme division fusion

Complexité

$$f(n) =$$

Un algorithme division fusion

Complexité

$$f(n) = A \cdot n + f(\tfrac{n}{2}) + f(\tfrac{n}{2})$$

Un algorithme division fusion

Complexité

$$f(n) = A \cdot n + f(\tfrac{n}{2}) + f(\tfrac{n}{2})$$

$$= O(n \log n)$$

Un algorithme division fusion

Complexité

$$f(n) = A \cdot n + f(\tfrac{n}{2}) + f(\tfrac{n}{2})$$

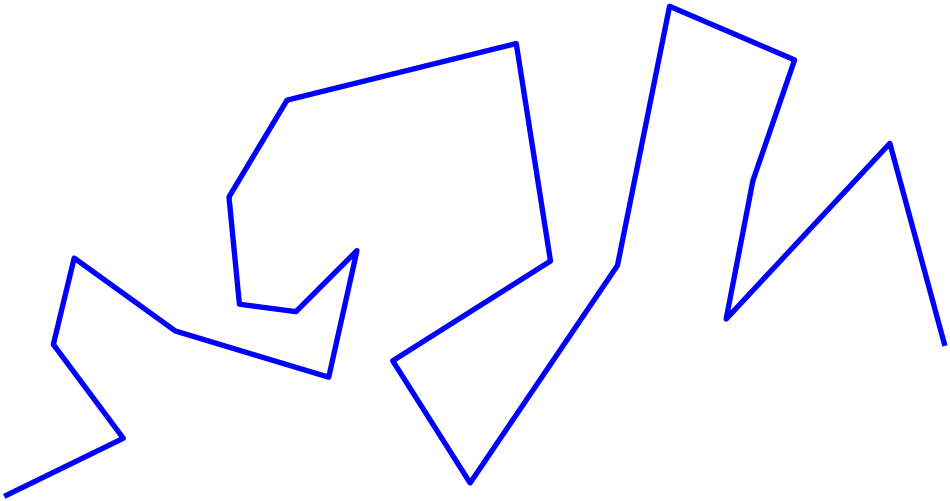
$$= O(n \log n)$$

Division et fusion en $O(n)$

Partition équilibré

(prétraitement $O(n \log n)$

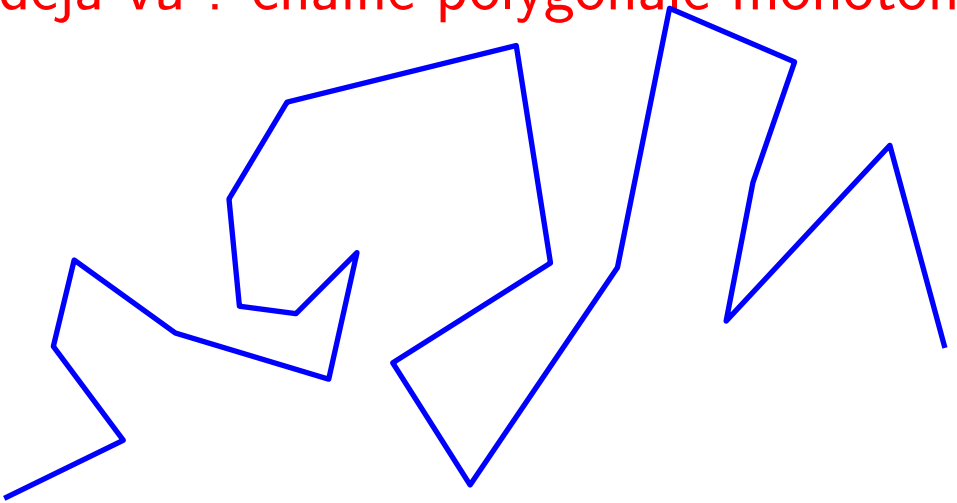
Cas particulier : polygone simple



Images produites par Olivier Devillers

Cas particulier : polygone simple

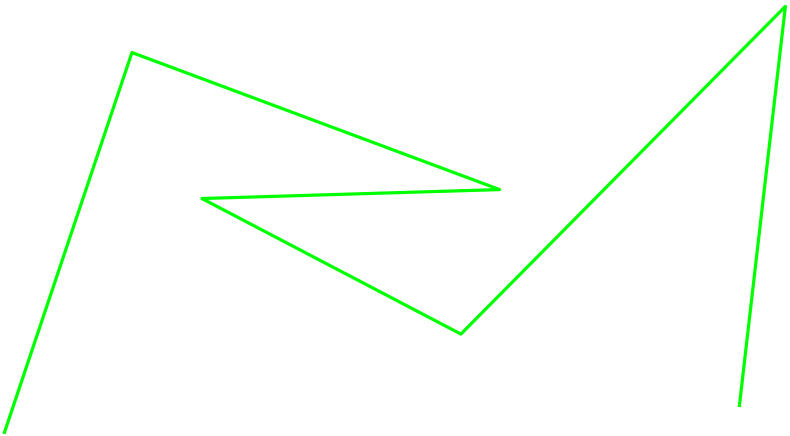
(déjà vu : chaine polygonale monotone)



Images produites par Olivier Devillers

Cas particulier : polygone simple

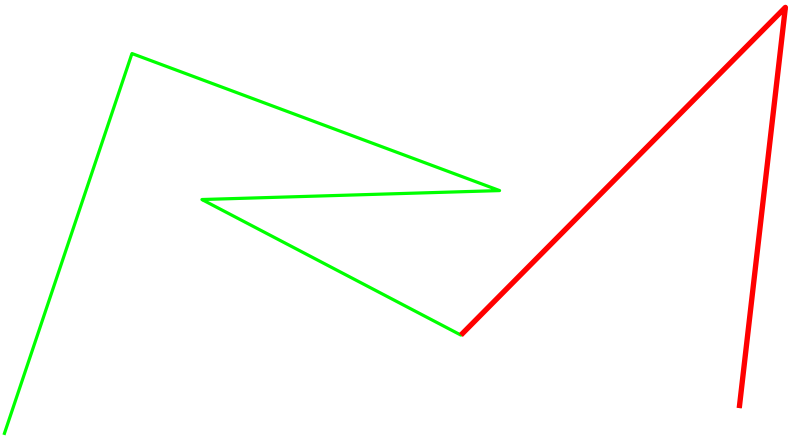
Graham marche pas



Images produites par Olivier Devillers

Cas particulier : polygone simple

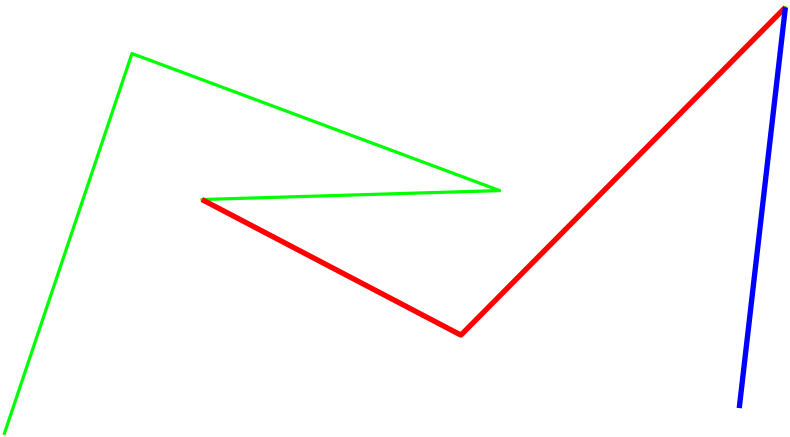
Graham marche pas



Images produites par Olivier Devillers

Cas particulier : polygone simple

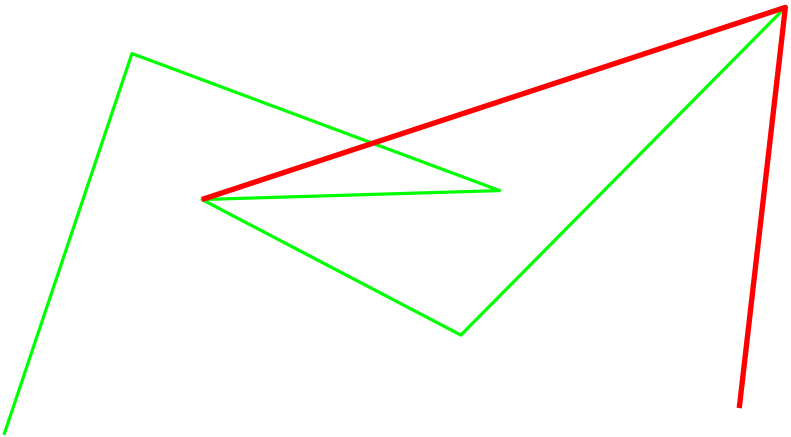
Graham marche pas



Images produites par Olivier Devillers

Cas particulier : polygone simple

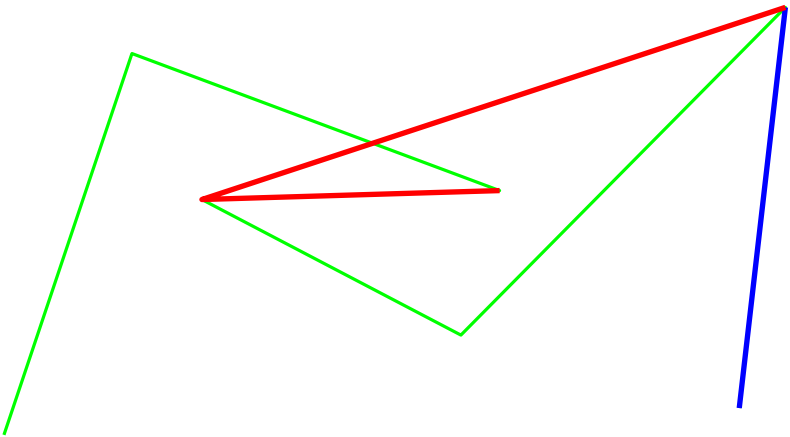
Graham marche pas



Images produites par Olivier Devillers

Cas particulier : polygone simple

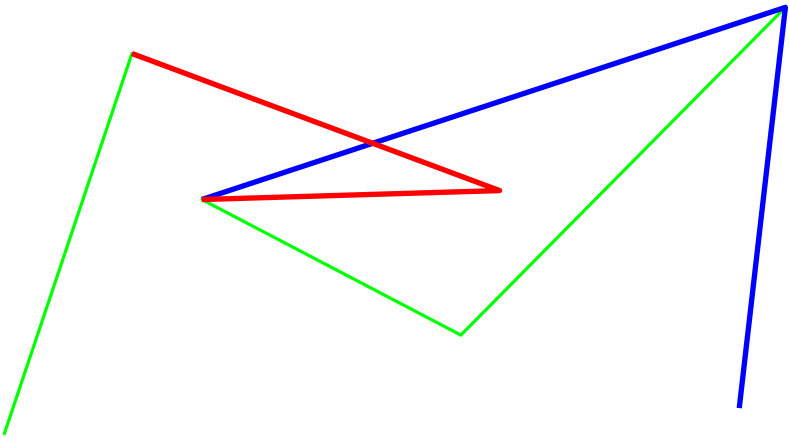
Graham marche pas



Images produites par Olivier Devillers

Cas particulier : polygone simple

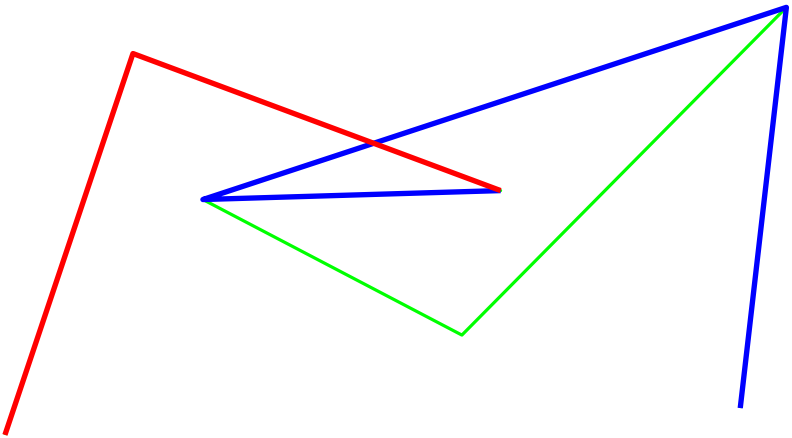
Graham marche pas



Images produites par Olivier Devillers

Cas particulier : polygone simple

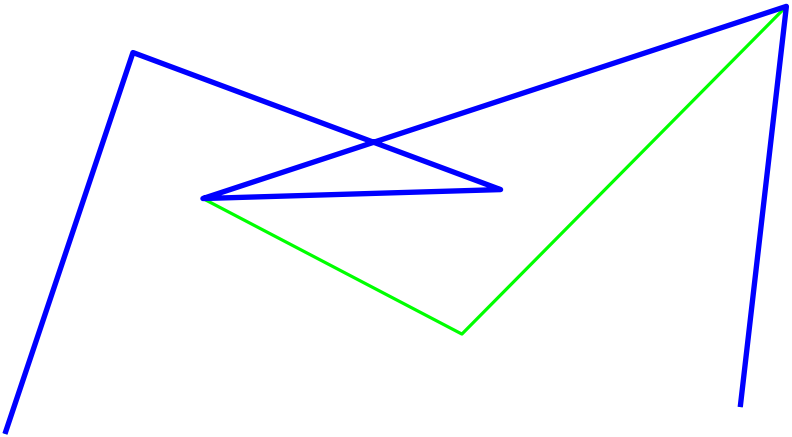
Graham marche pas



Images produites par Olivier Devillers

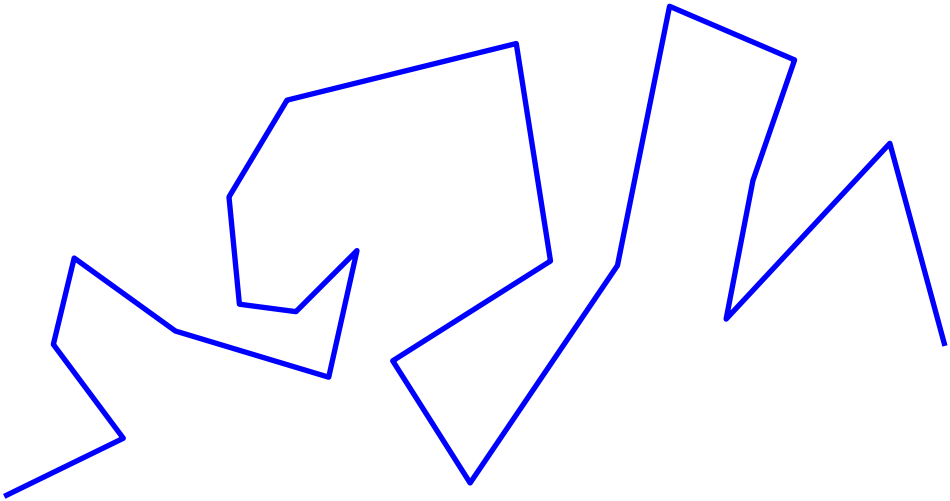
Cas particulier : polygone simple

Graham marche pas



Images produites par Olivier Devillers

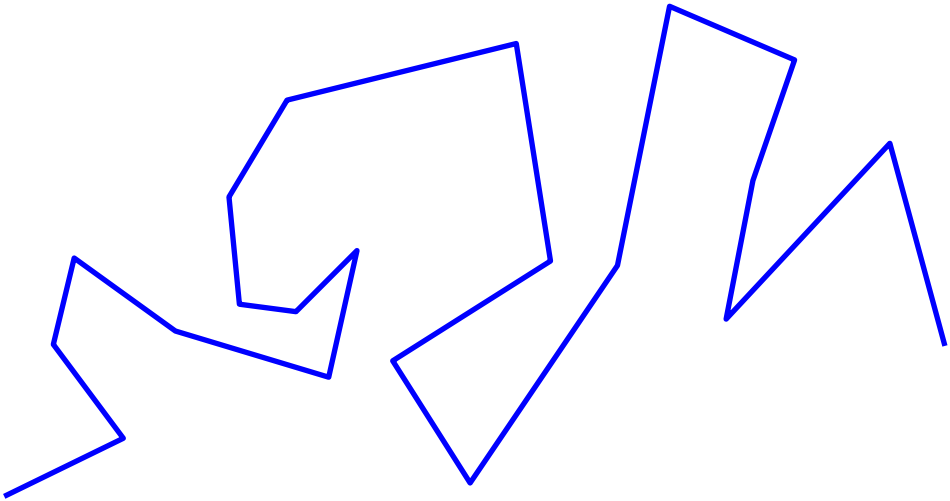
Cas particulier : polygone simple



Images produites par Olivier Devillers

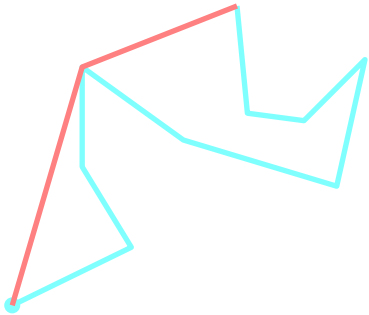
Cas particulier : polygone simple

Principe : boucher les poches



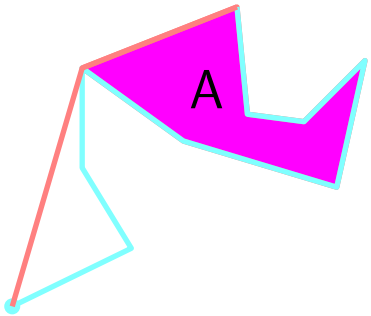
Images produites par Olivier Devillers

Cas particulier : polygone simple



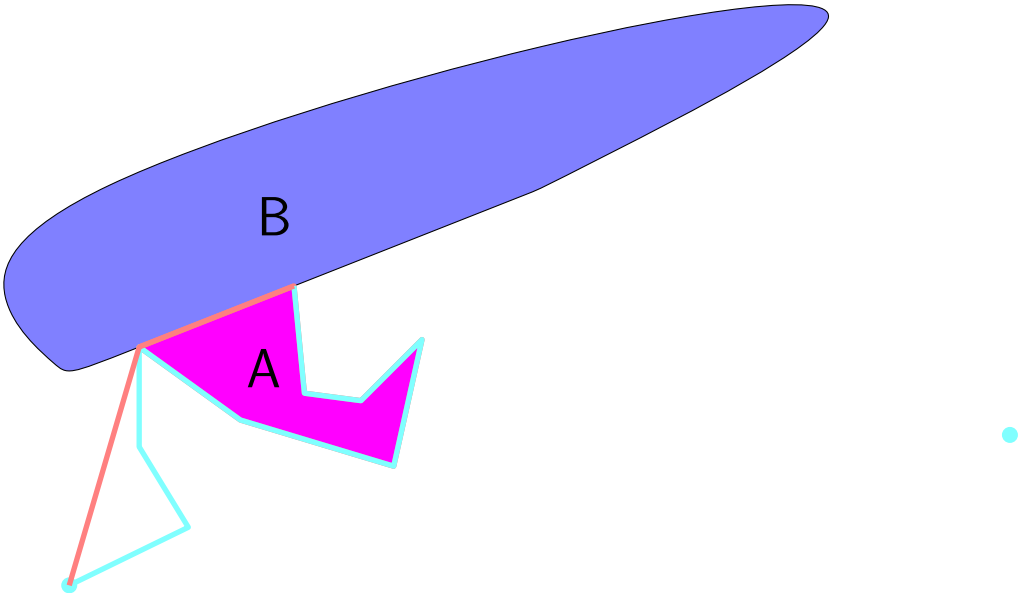
Images produites par Olivier Devillers

Cas particulier : polygone simple



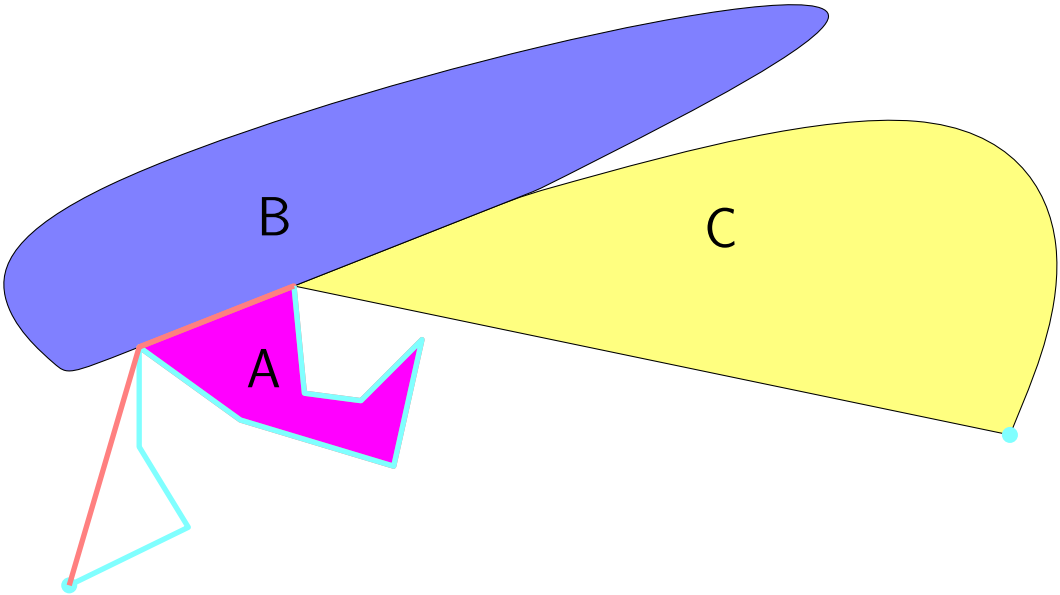
Images produites par Olivier Devillers

Cas particulier : polygone simple



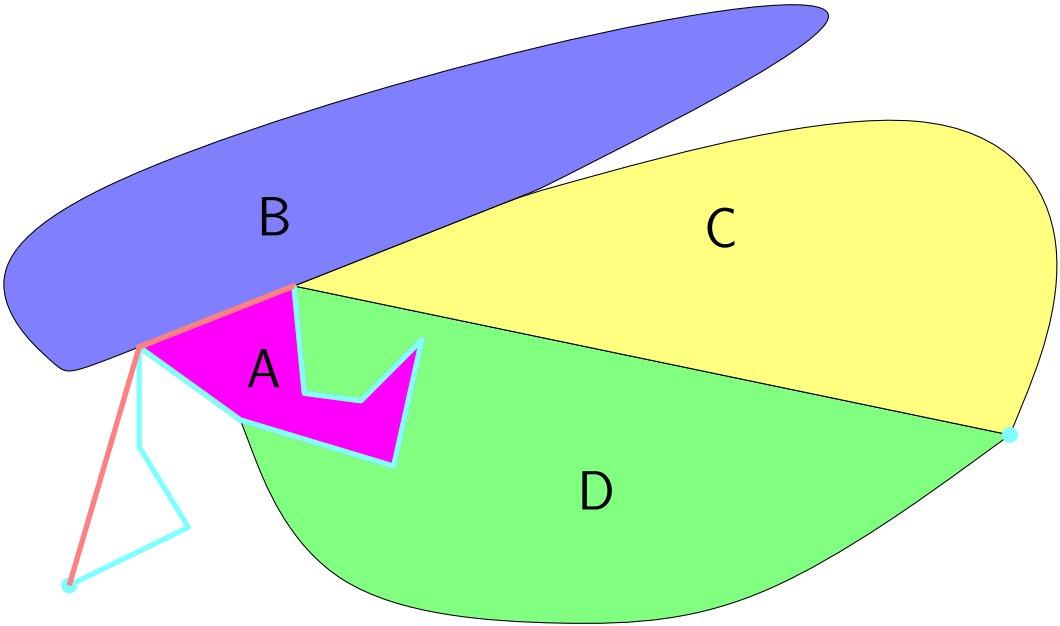
Images produites par Olivier Devillers

Cas particulier : polygone simple



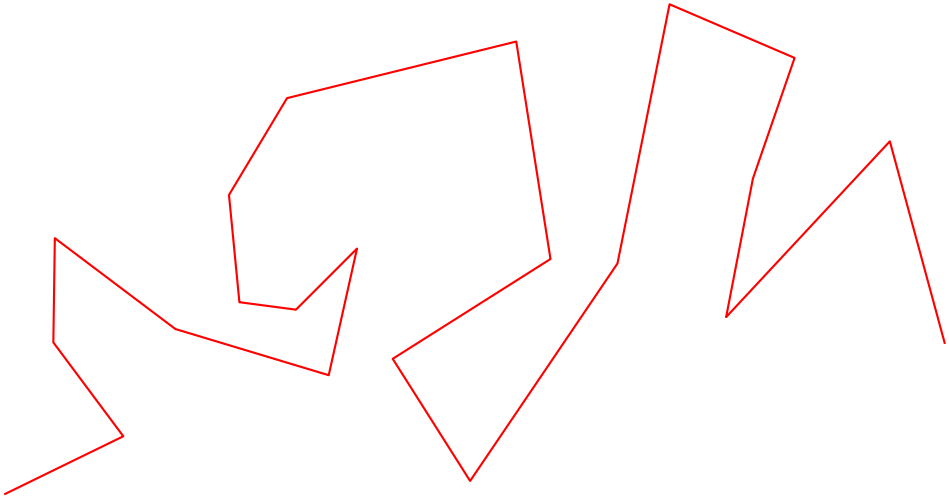
Images produites par Olivier Devillers

Cas particulier : polygone simple



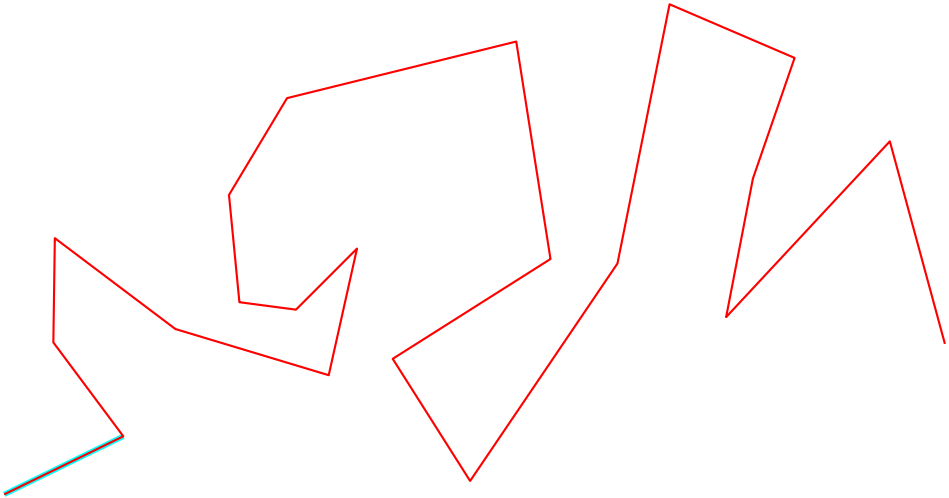
Images produites par Olivier Devillers

Cas particulier : polygone simple



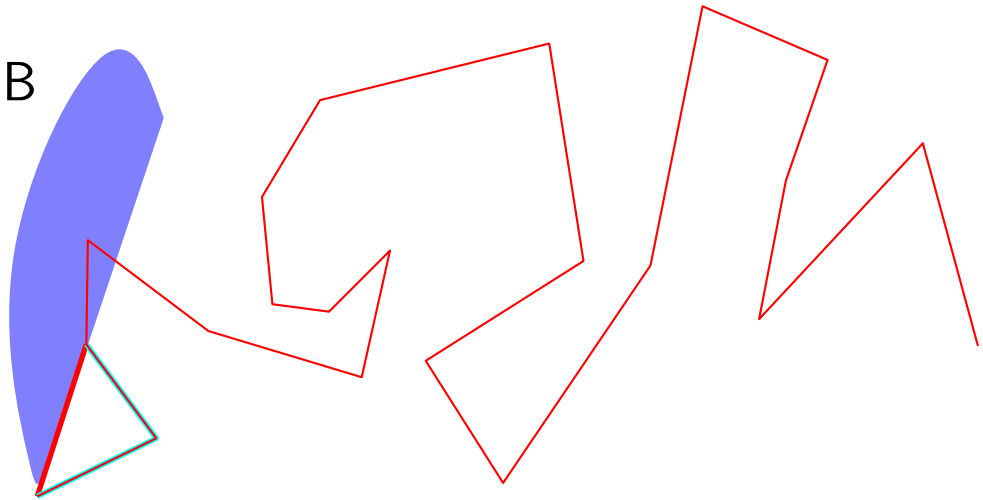
Images produites par Olivier Devillers

Cas particulier : polygone simple



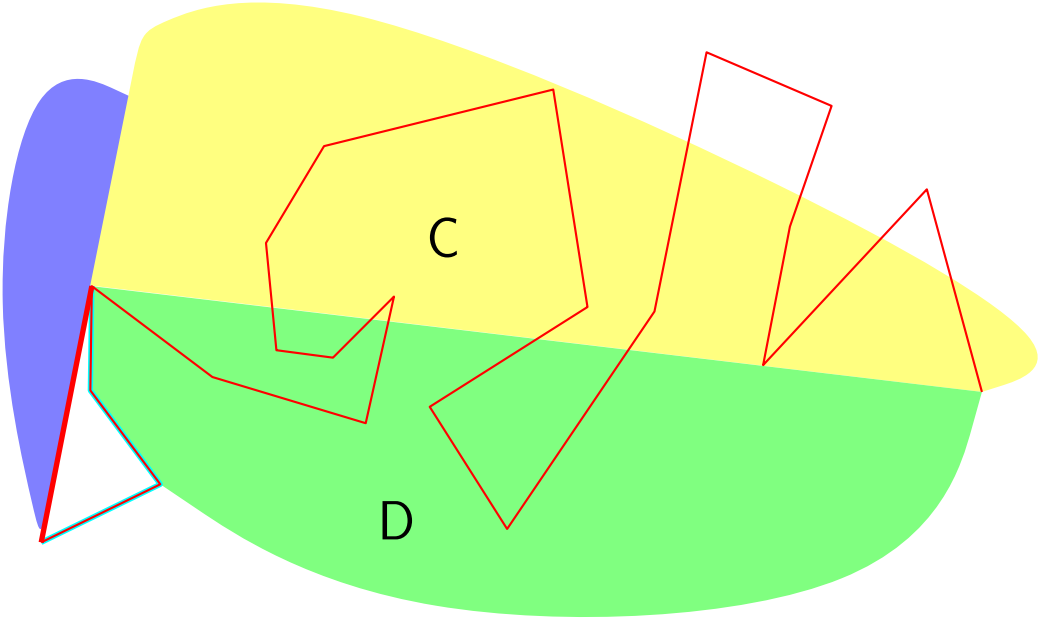
Images produites par Olivier Devillers

Cas particulier : polygone simple



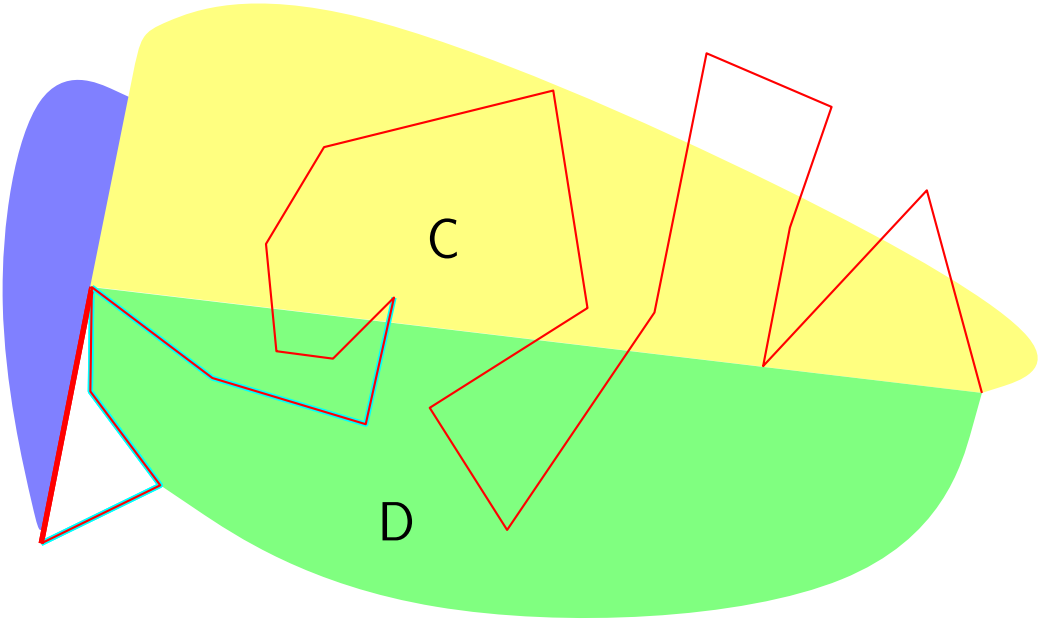
Images produites par Olivier Devillers

Cas particulier : polygone simple



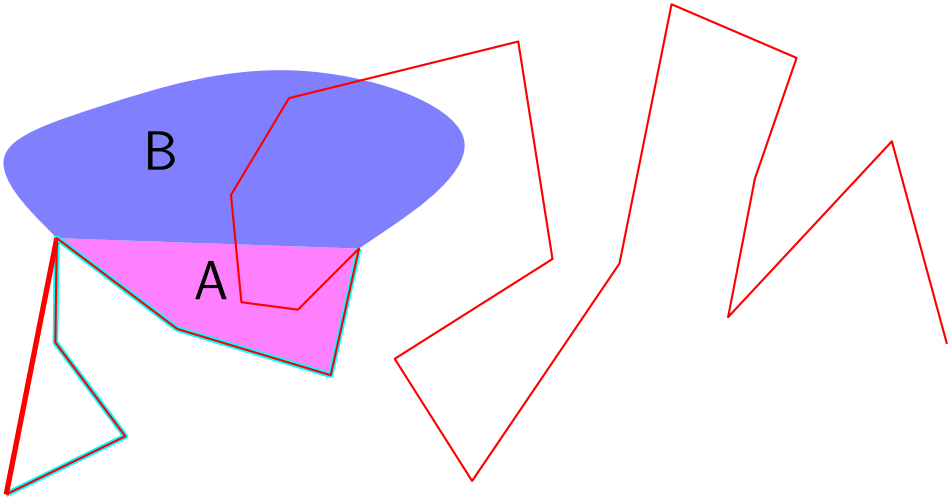
Images produites par Olivier Devillers

Cas particulier : polygone simple



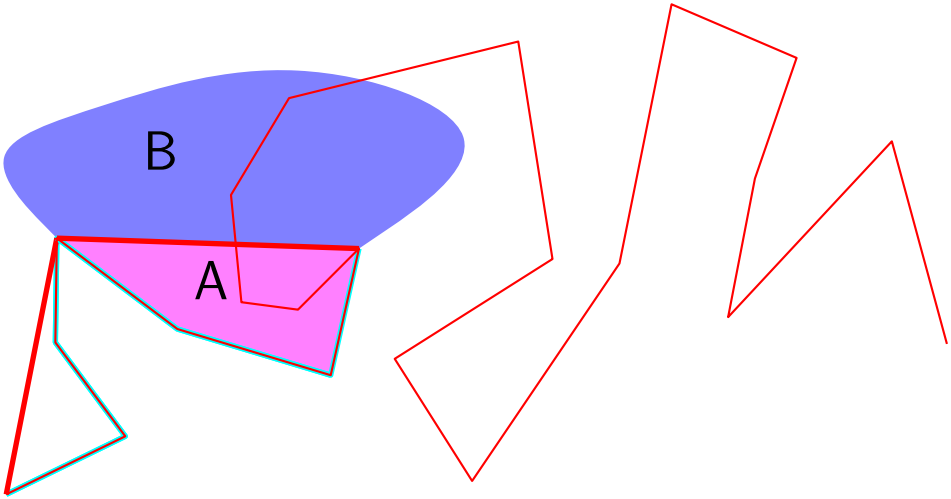
Images produites par Olivier Devillers

Cas particulier : polygone simple



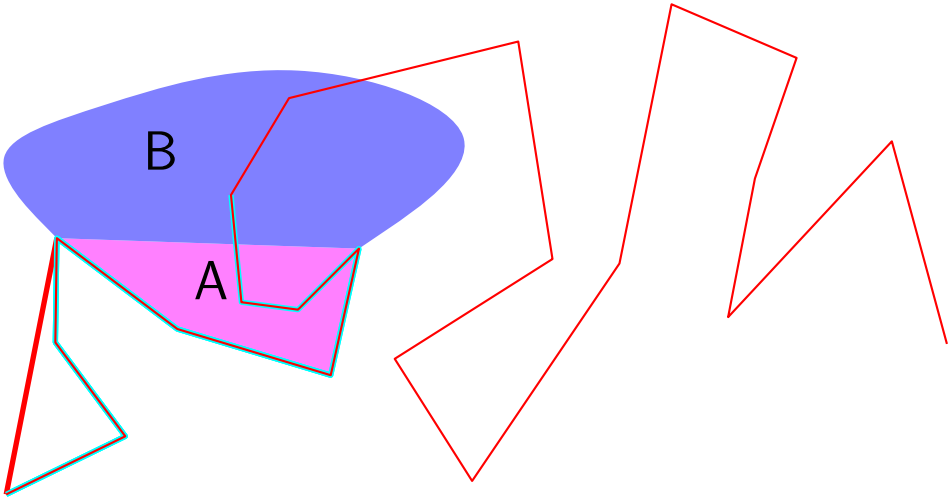
Images produites par Olivier Devillers

Cas particulier : polygone simple



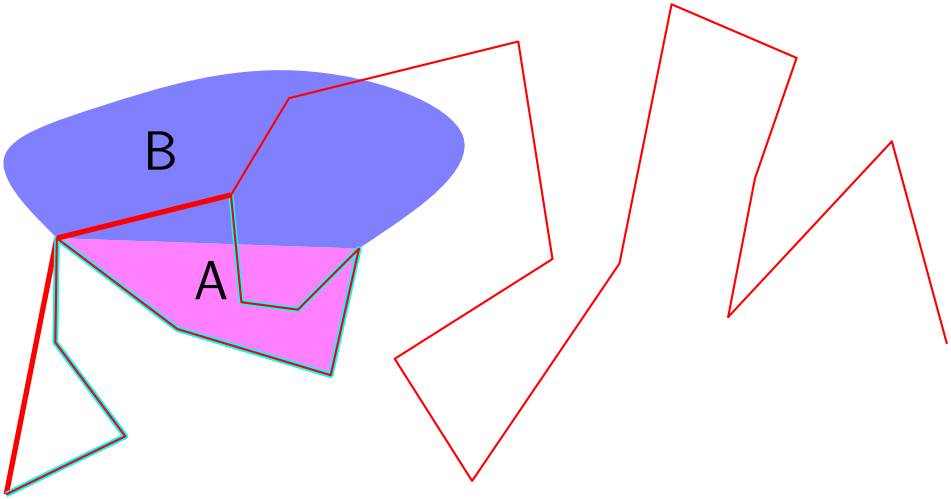
Images produites par Olivier Devillers

Cas particulier : polygone simple



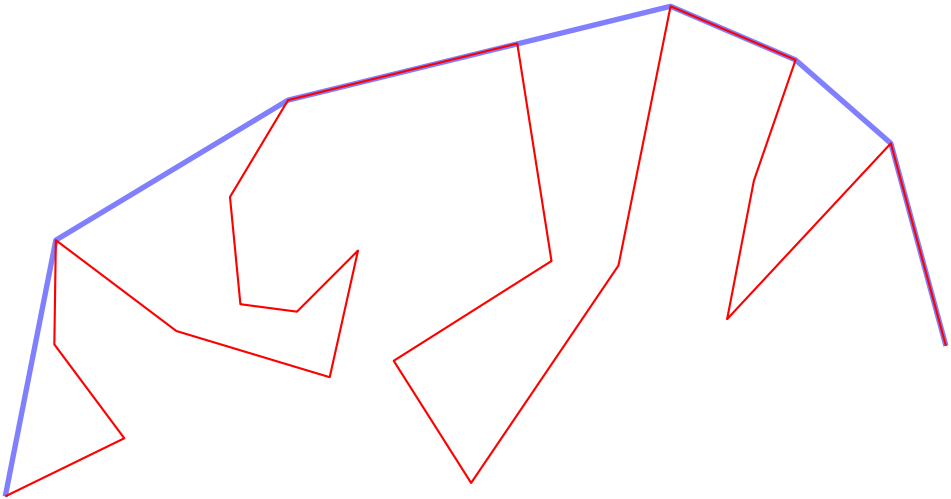
Images produites par Olivier Devillers

Cas particulier : polygone simple



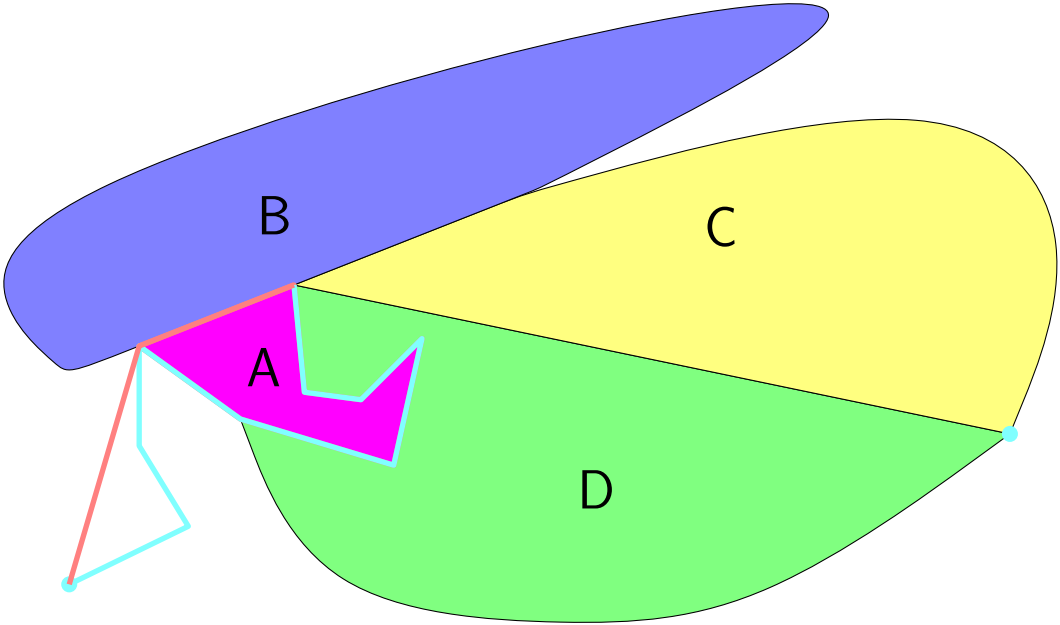
Images produites par Olivier Devillers

Cas particulier : polygone simple



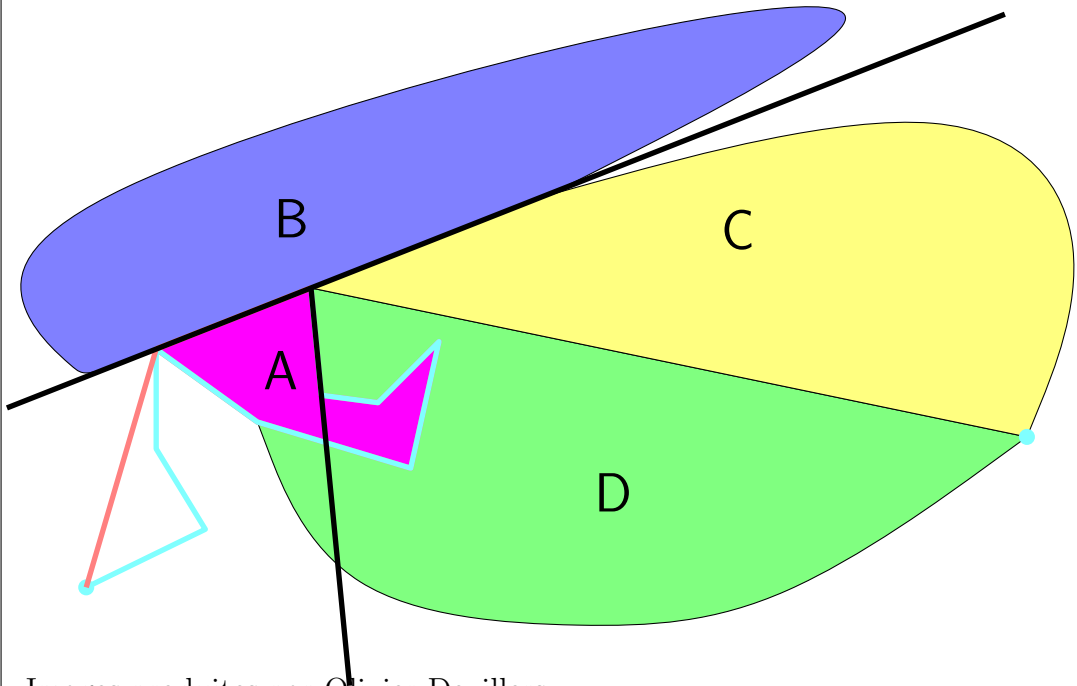
Images produites par Olivier Devillers

Cas particulier : polygone simple



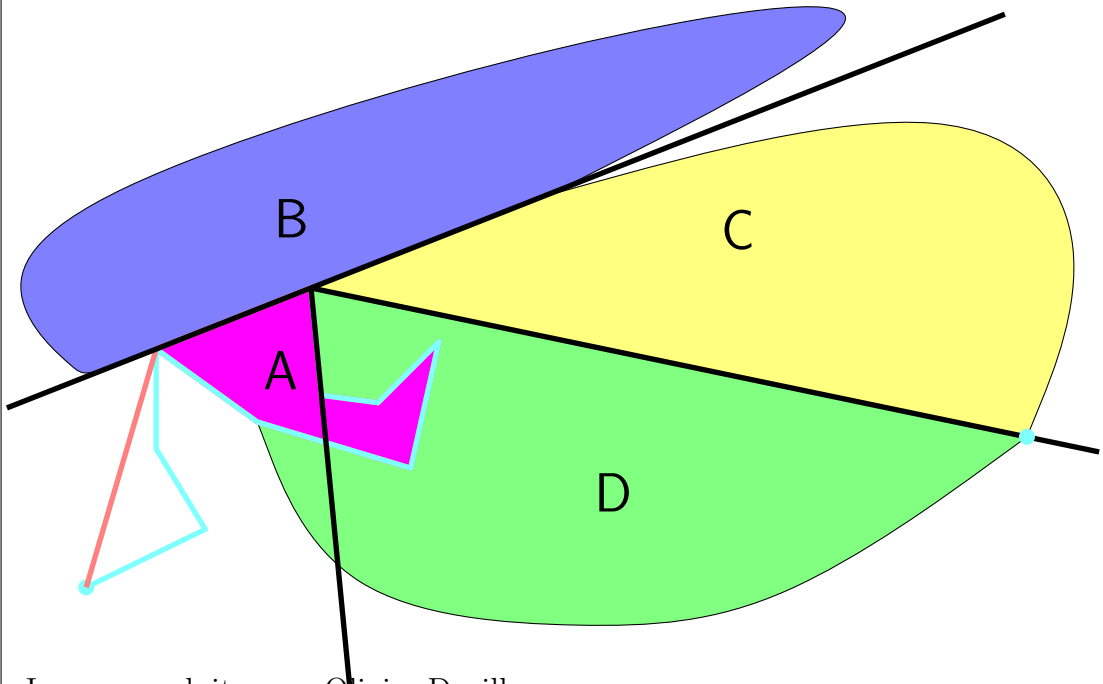
Images produites par Olivier Devillers

Cas particulier : polygone simple



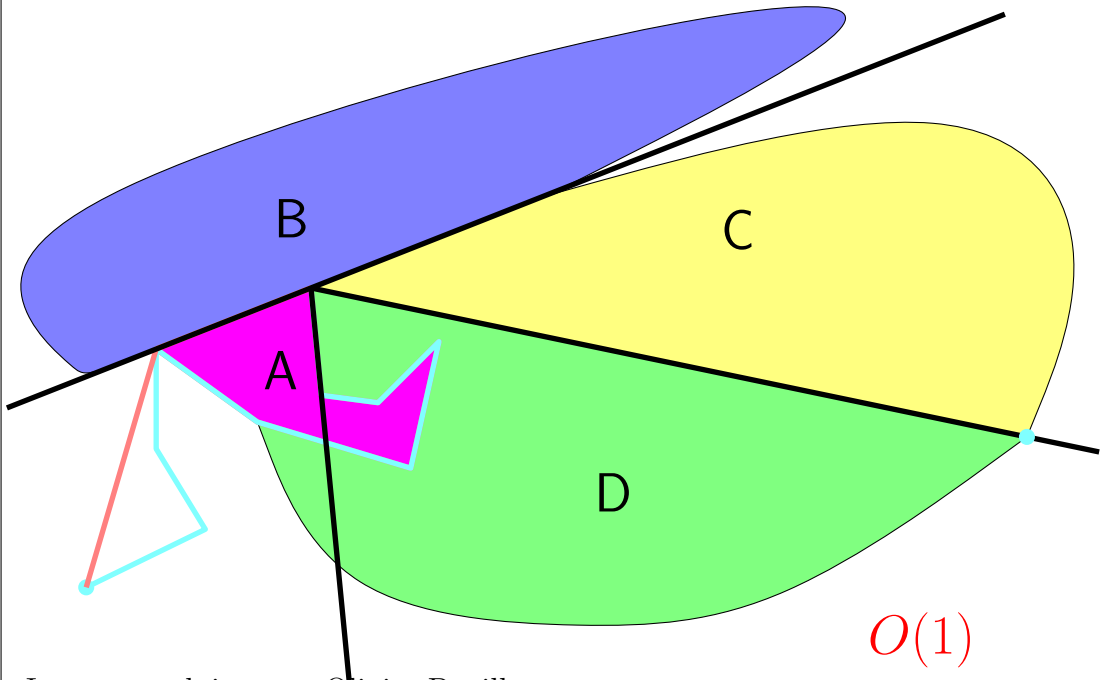
Images produites par Olivier Devillers

Cas particulier : polygone simple



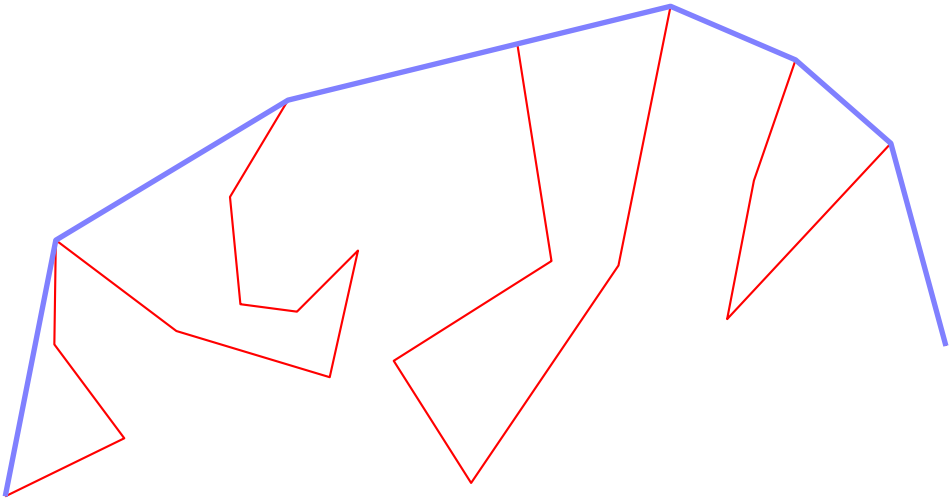
Images produites par Olivier Devillers

Cas particulier : polygone simple



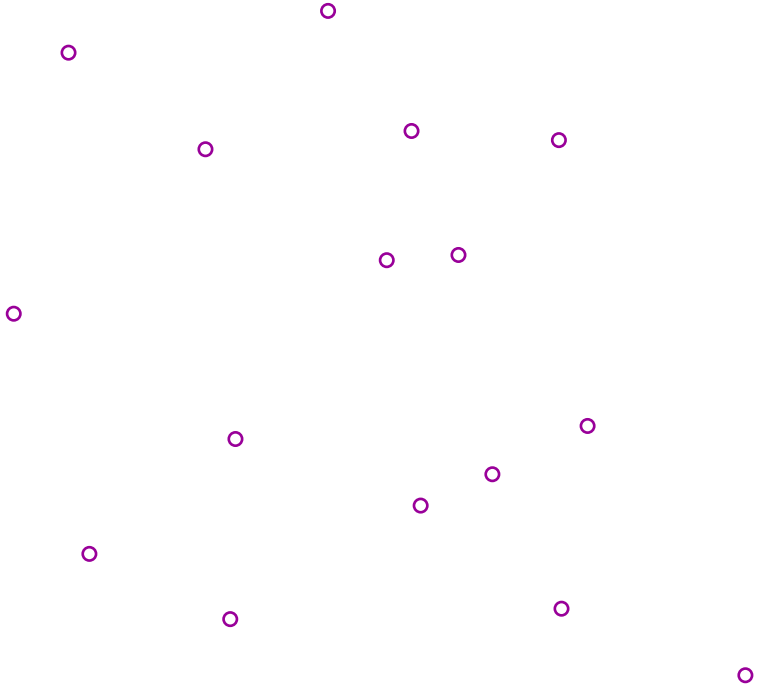
Images produites par Olivier Devillers

Cas particulier : polygone simple

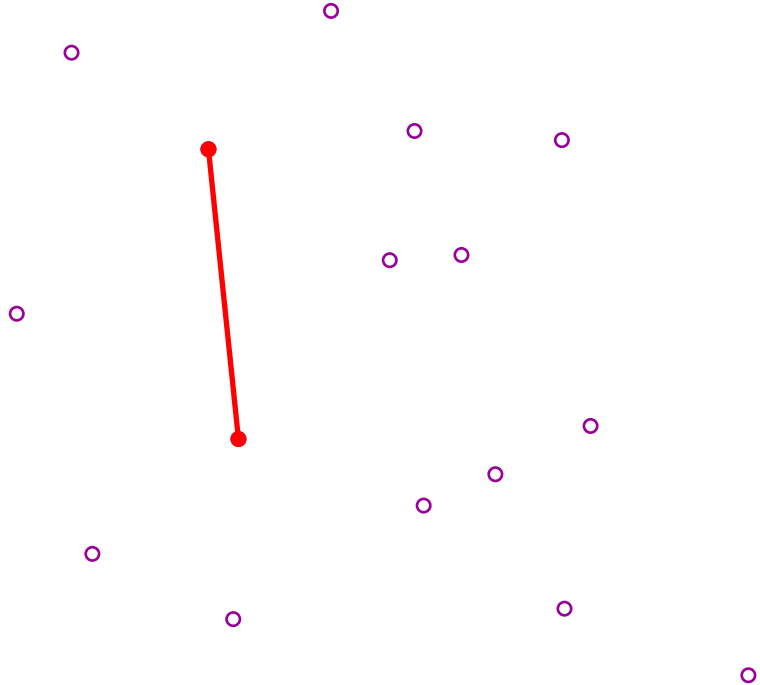


$O(n)$

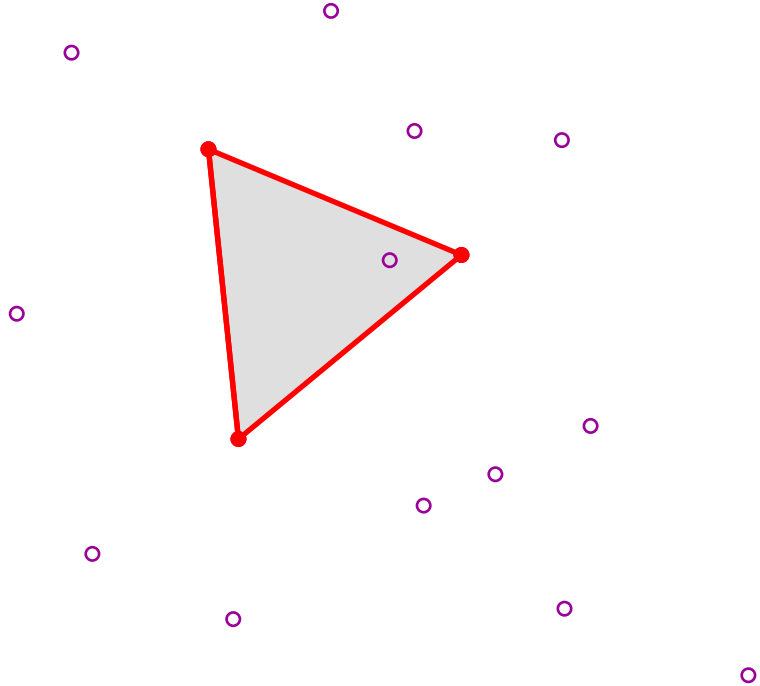
Un algorithme incrémental



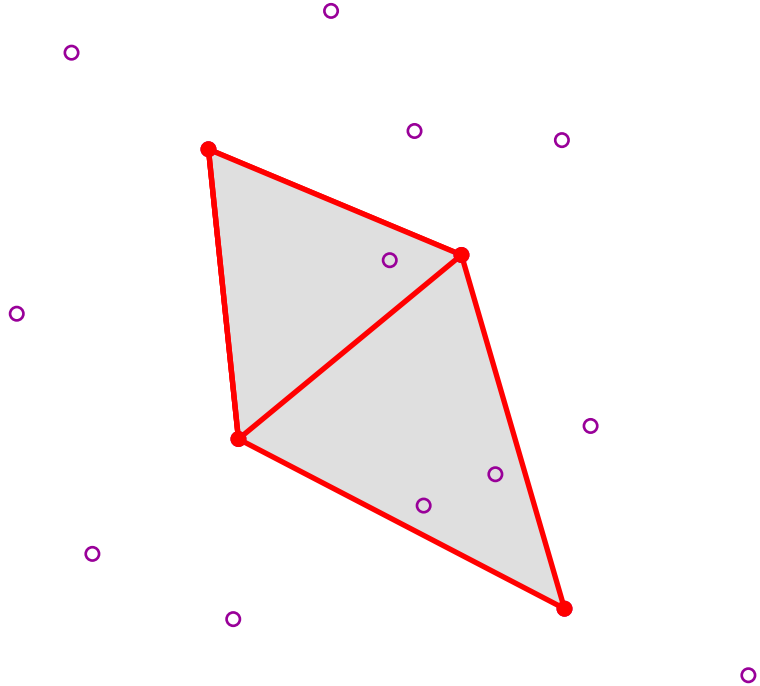
Un algorithme incrémental



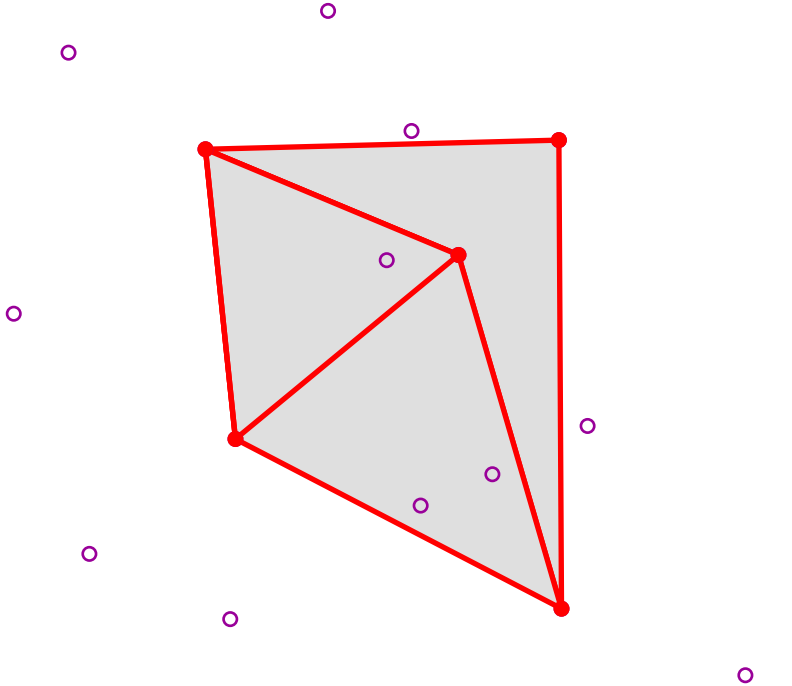
Un algorithme incrémental



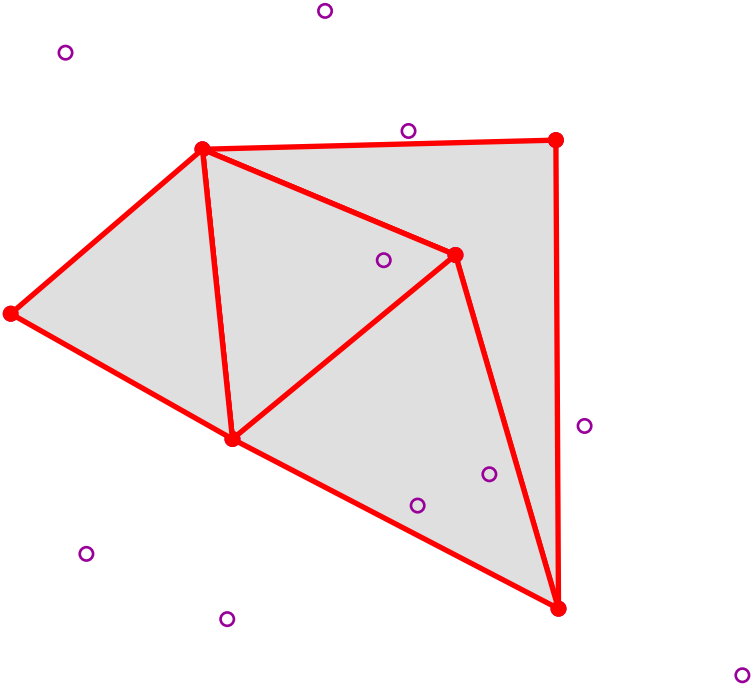
Un algorithme incrémental



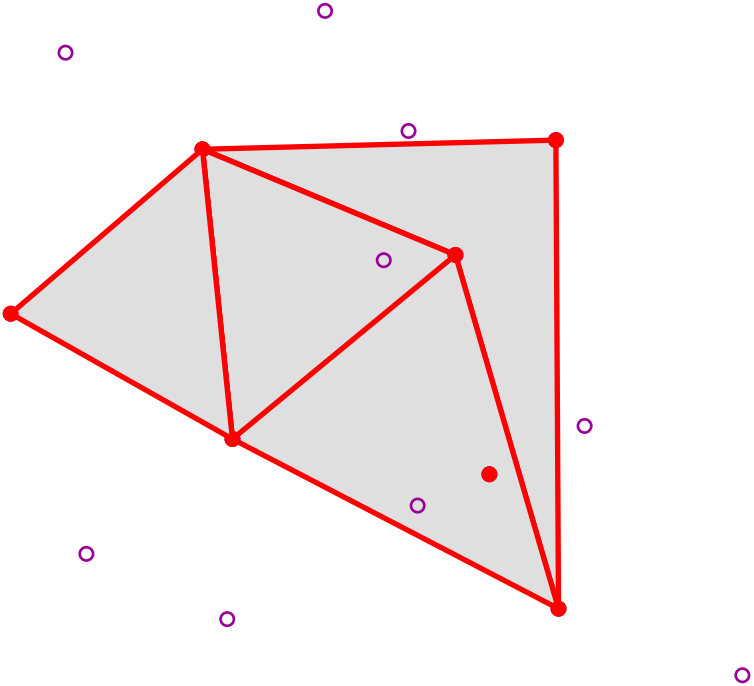
Un algorithme incrémental



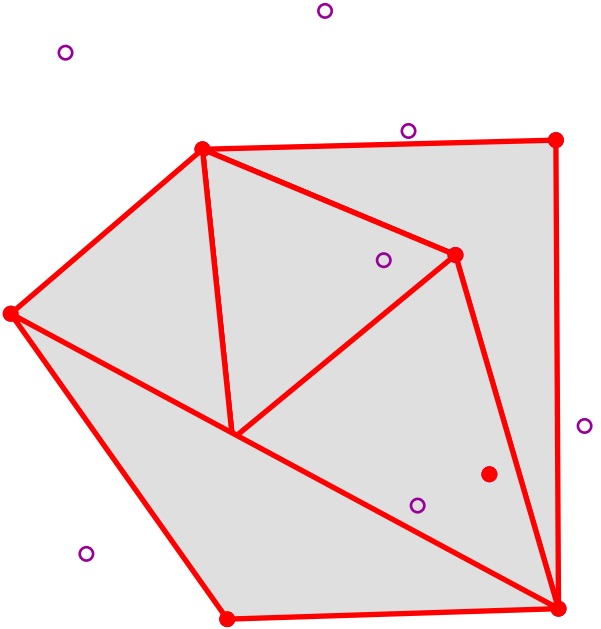
Un algorithme incrémental



Un algorithme incrémental

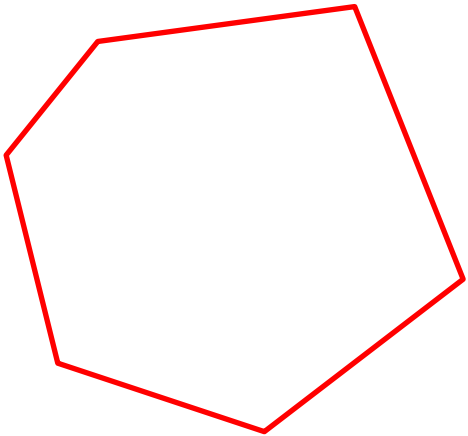


Un algorithme incrémental

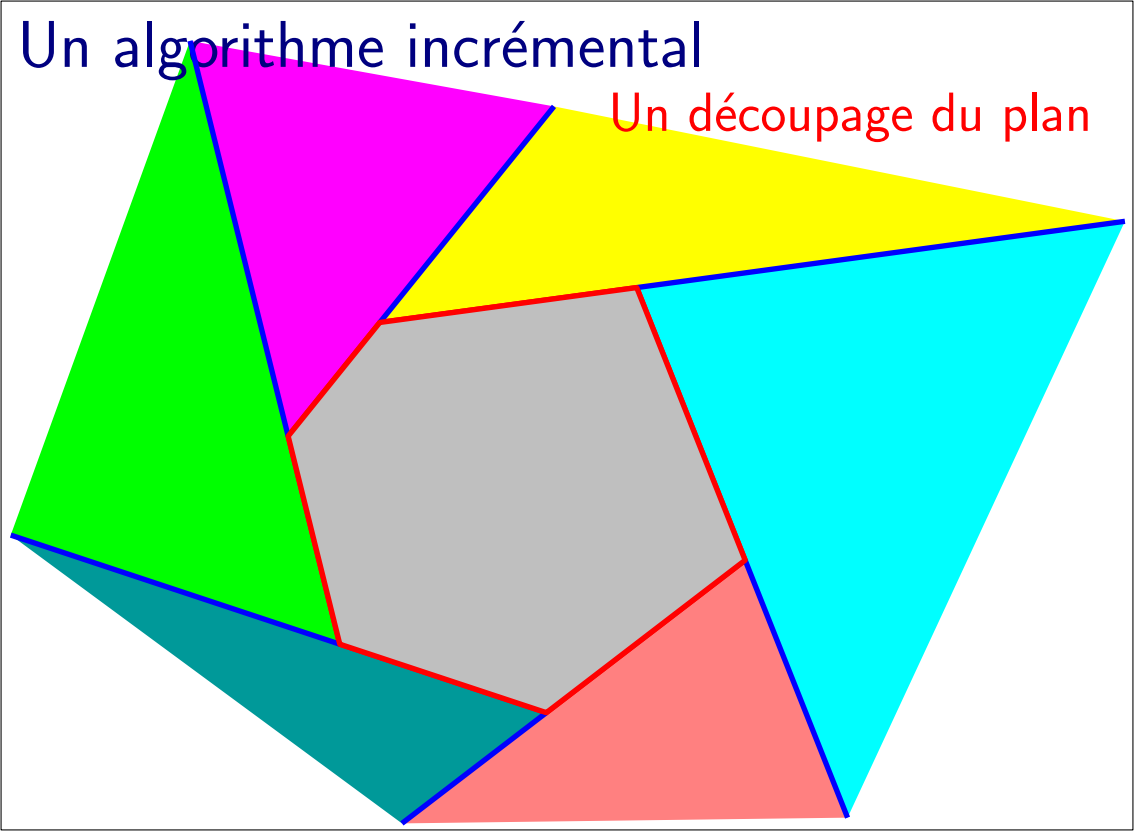


Un algorithme incrémental

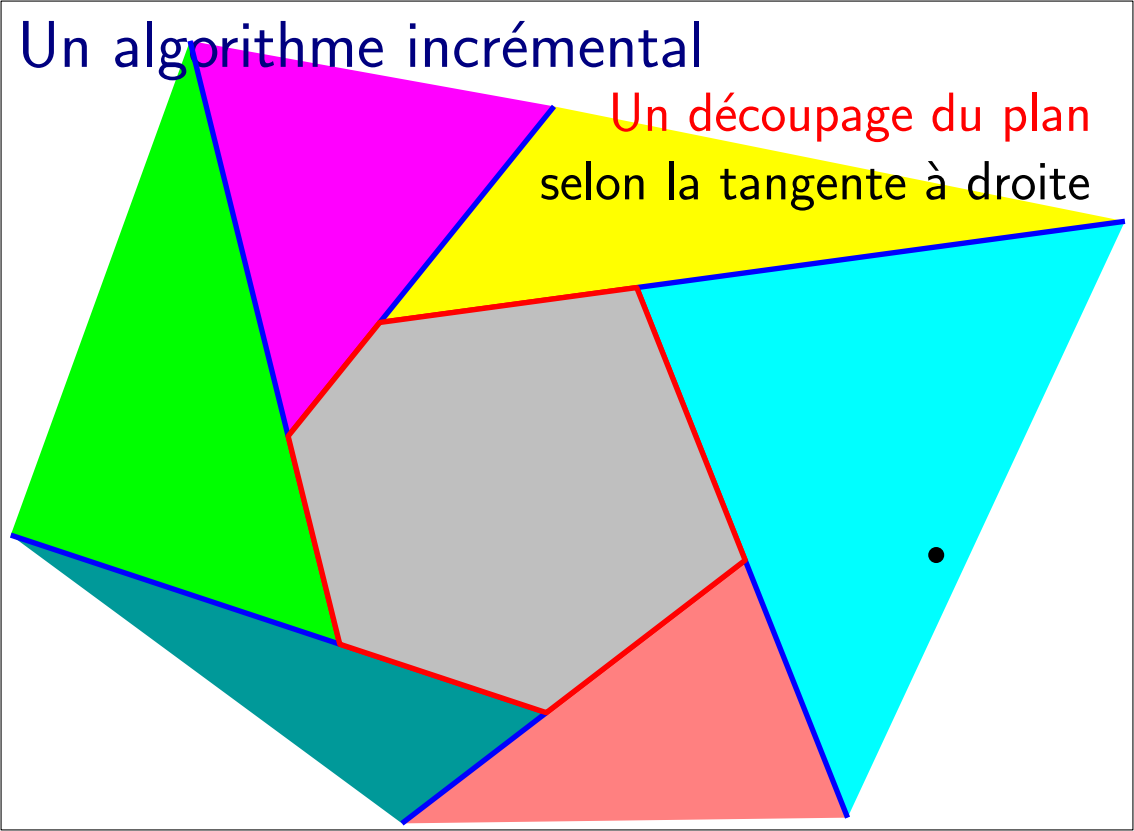
Un découpage du plan



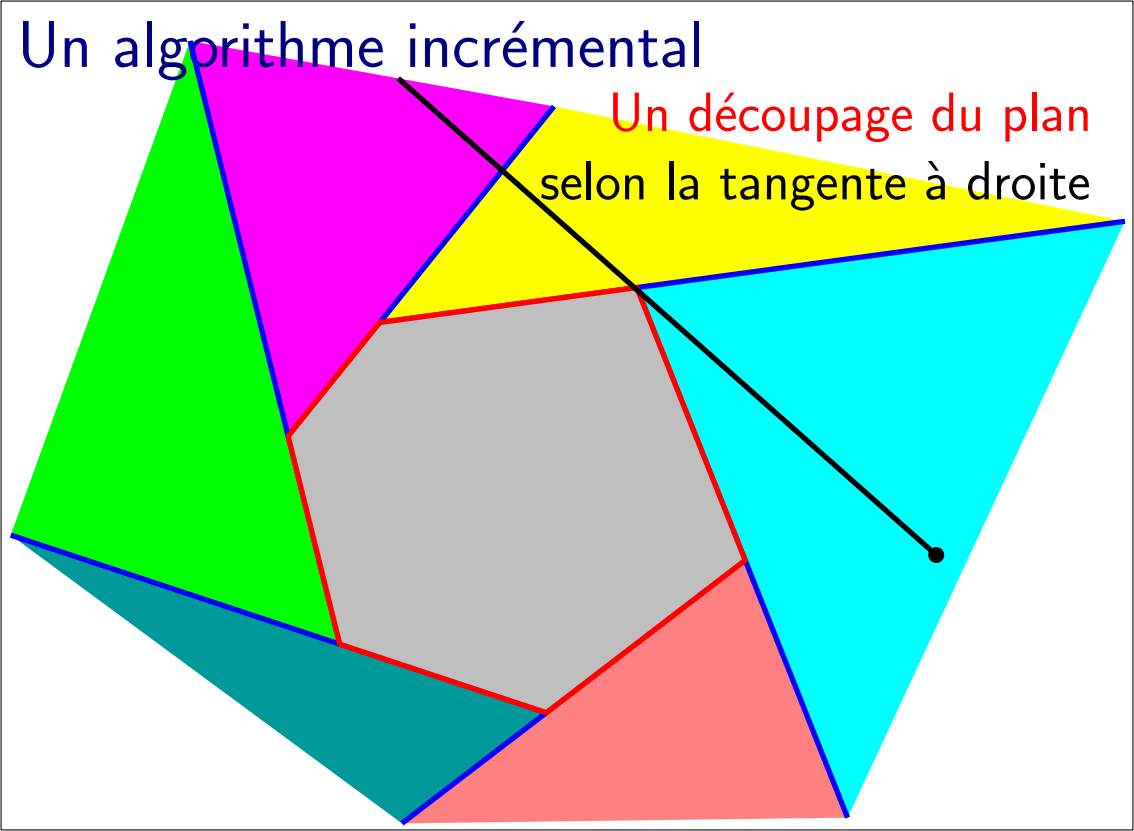
Un algorithme incrémental



Un algorithme incrémental

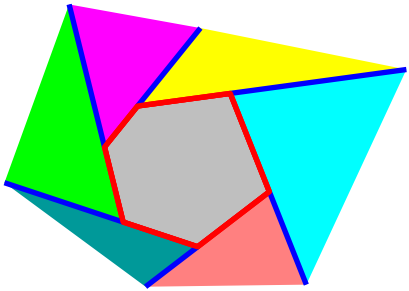


Un algorithme incrémental



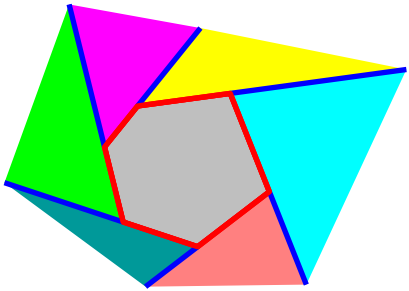
Un algorithme incrémental

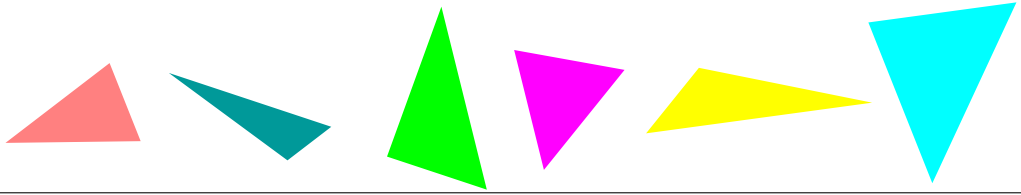
Un découpage du plan



Un algorithme incrémental

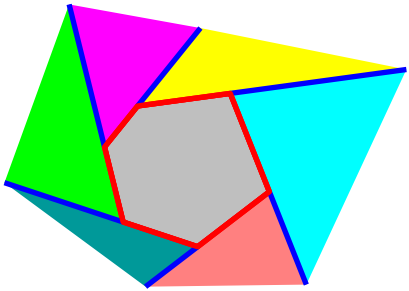
Un découpage du plan

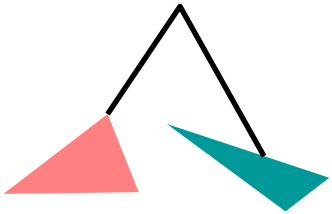


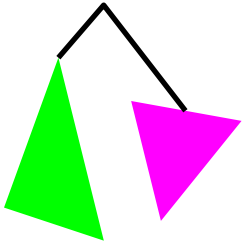


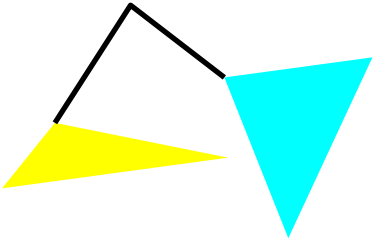
Un algorithme incrémental

Un découpage du plan



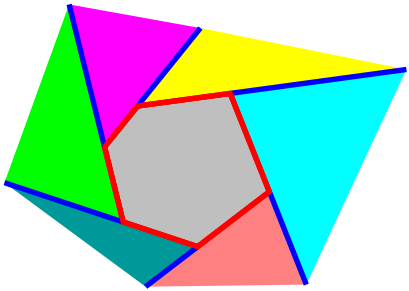


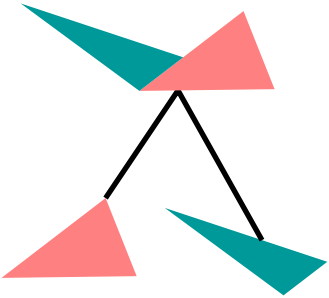


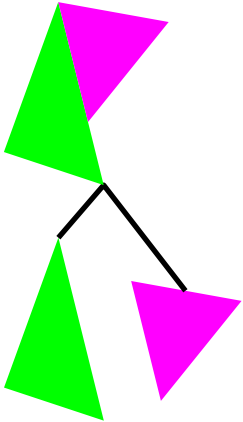


Un algorithme incrémental

Un découpage du plan

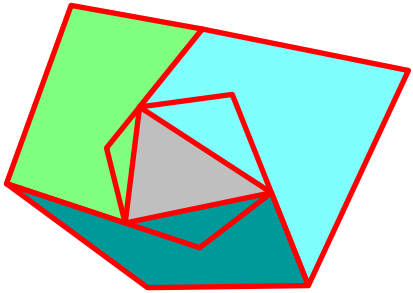


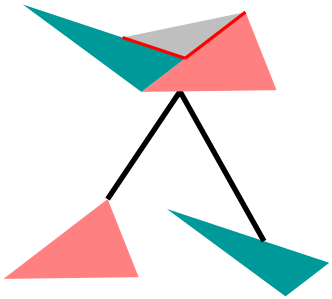


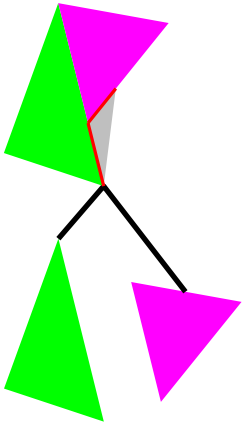


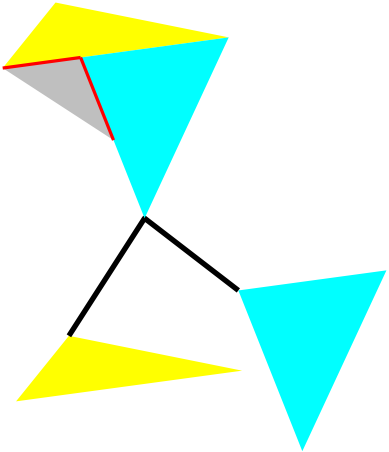
Un algorithme incrémental

Un découpage du plan



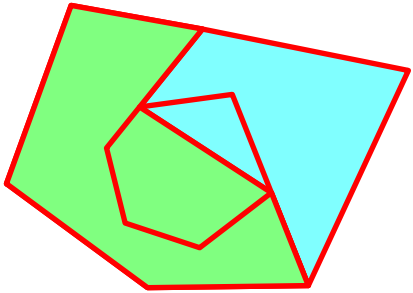


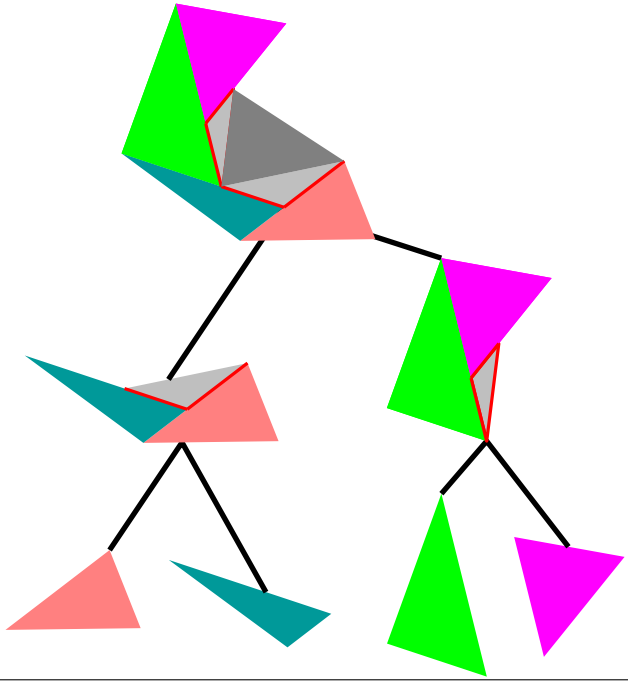


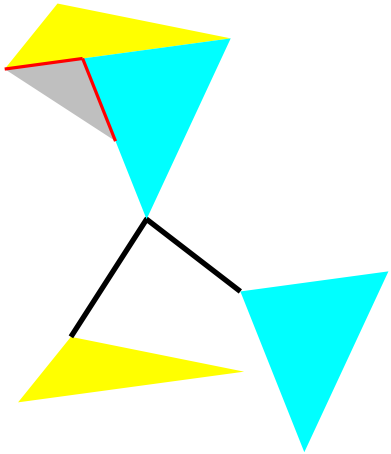


Un algorithme incrémental

Un découpage du plan

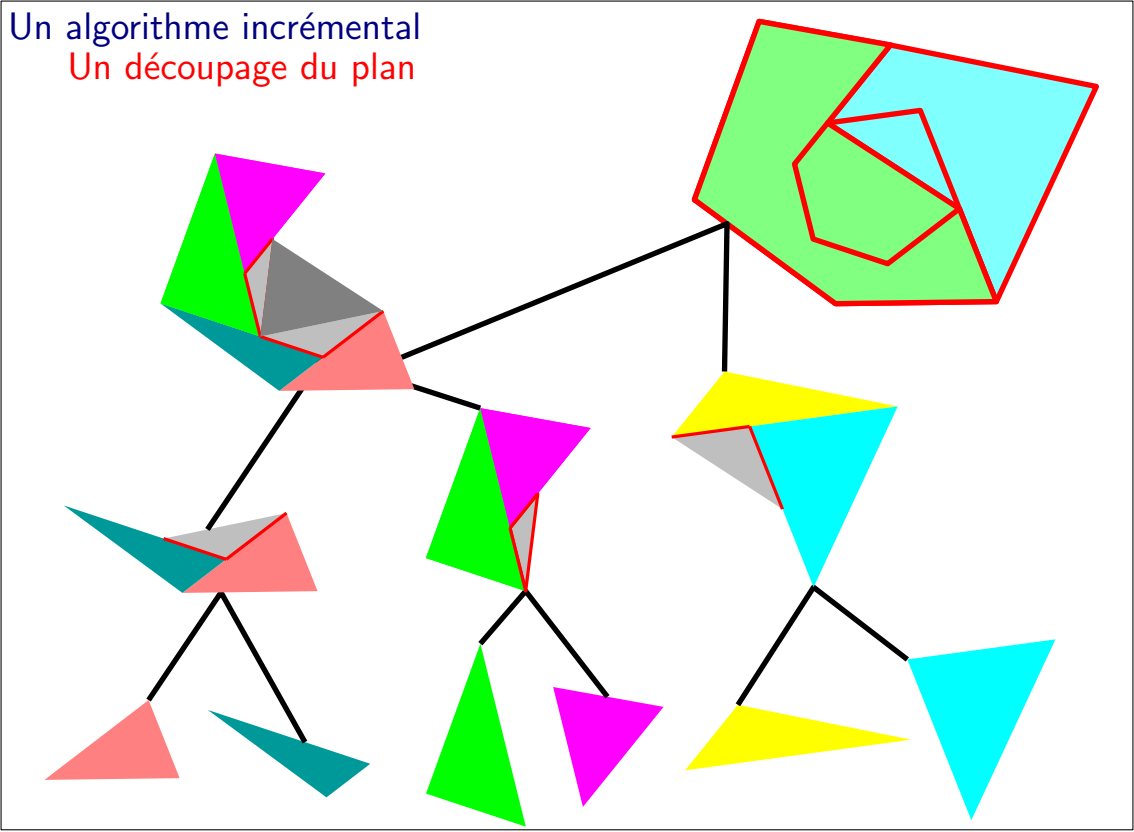




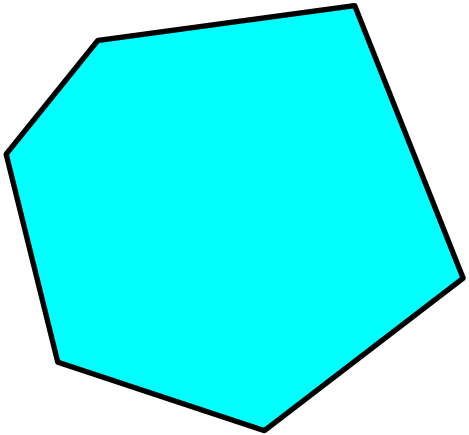


Un algorithme incrémental

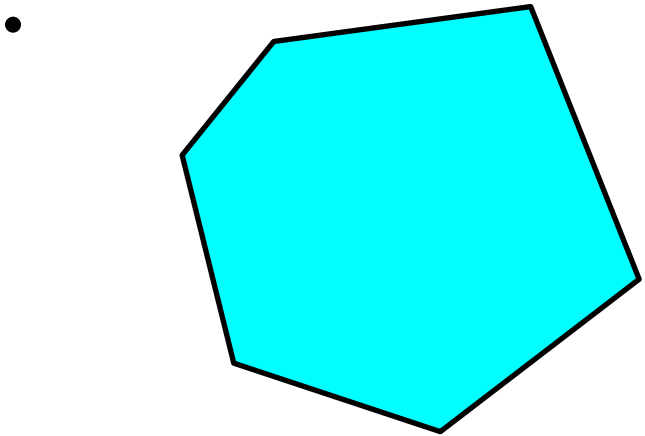
Un découpage du plan



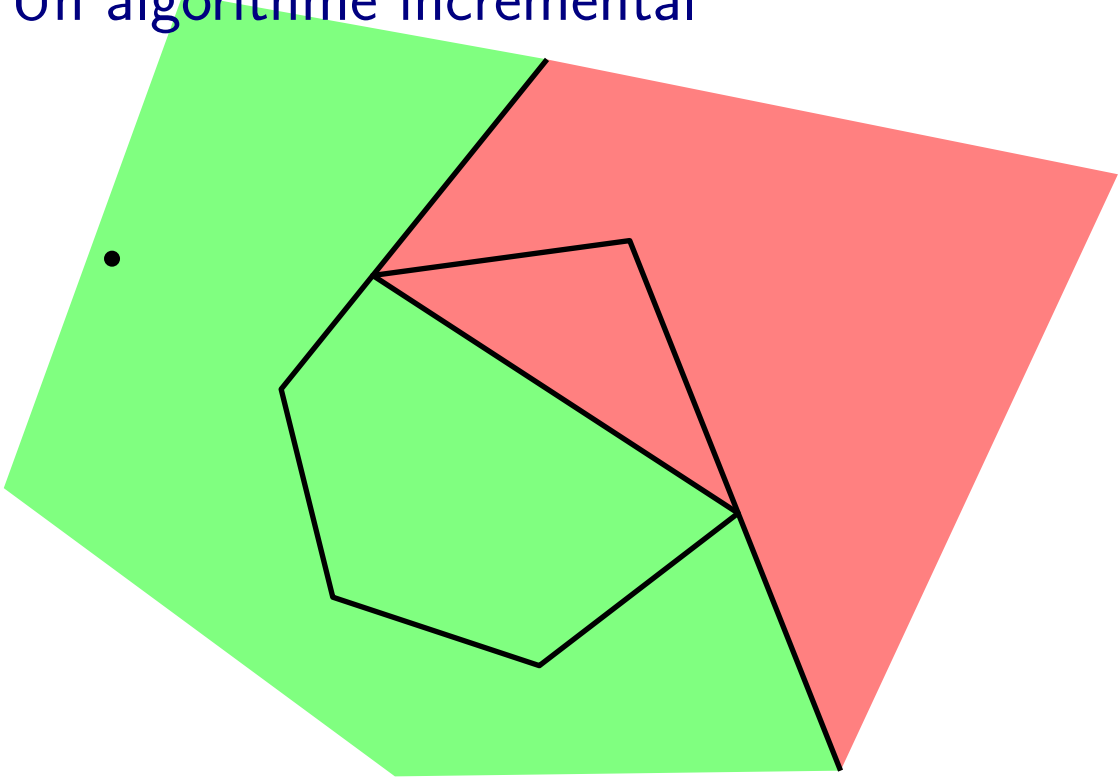
Un algorithme incrémental



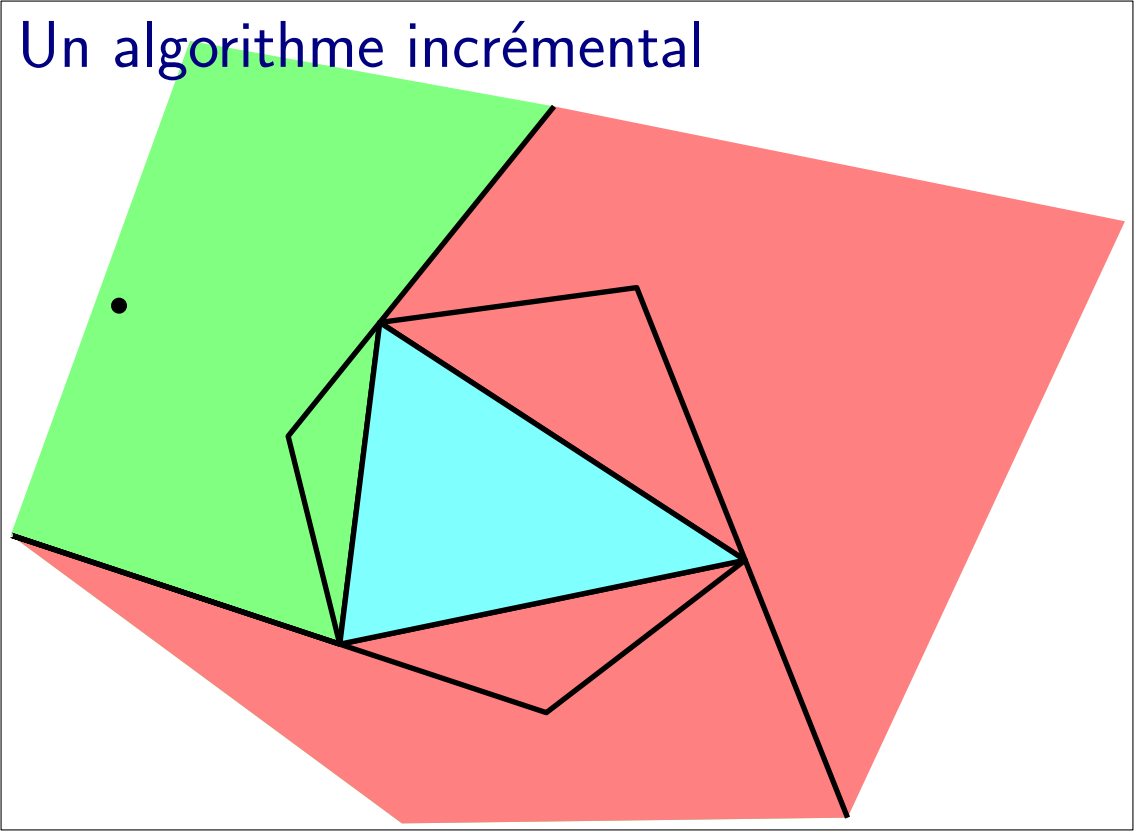
Un algorithme incrémental



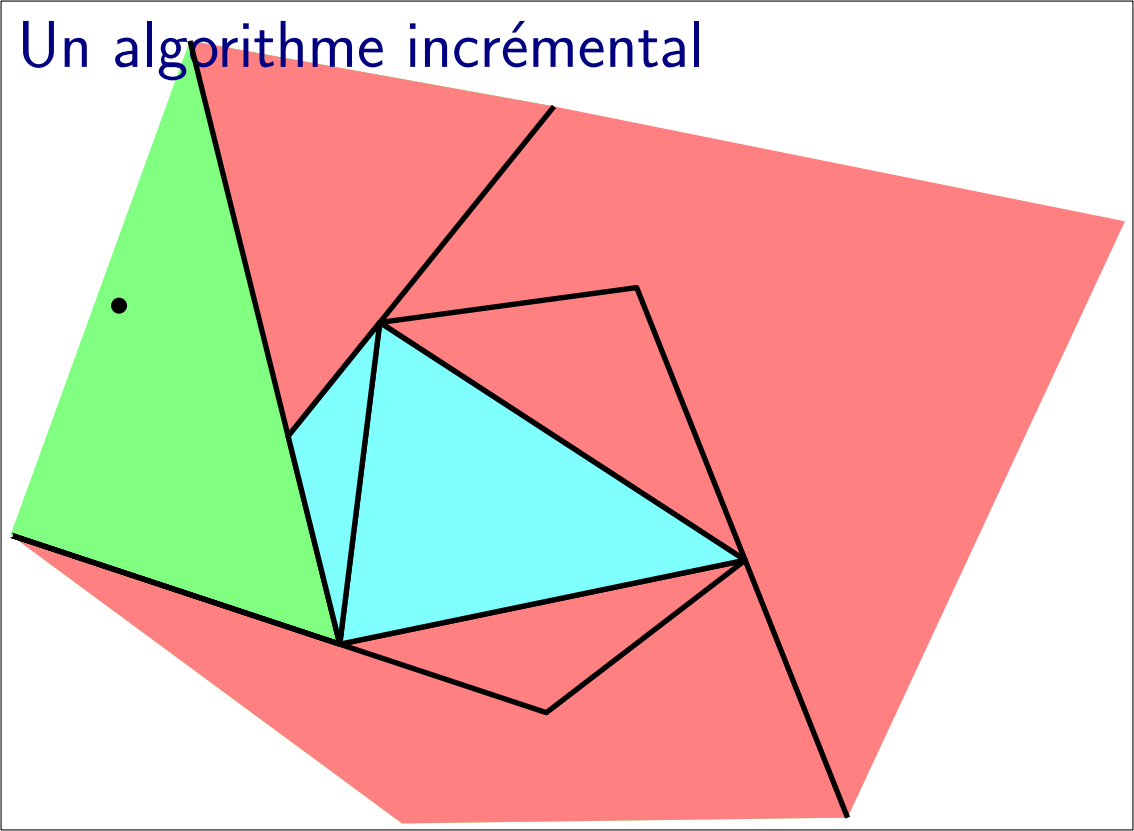
Un algorithme incrémental



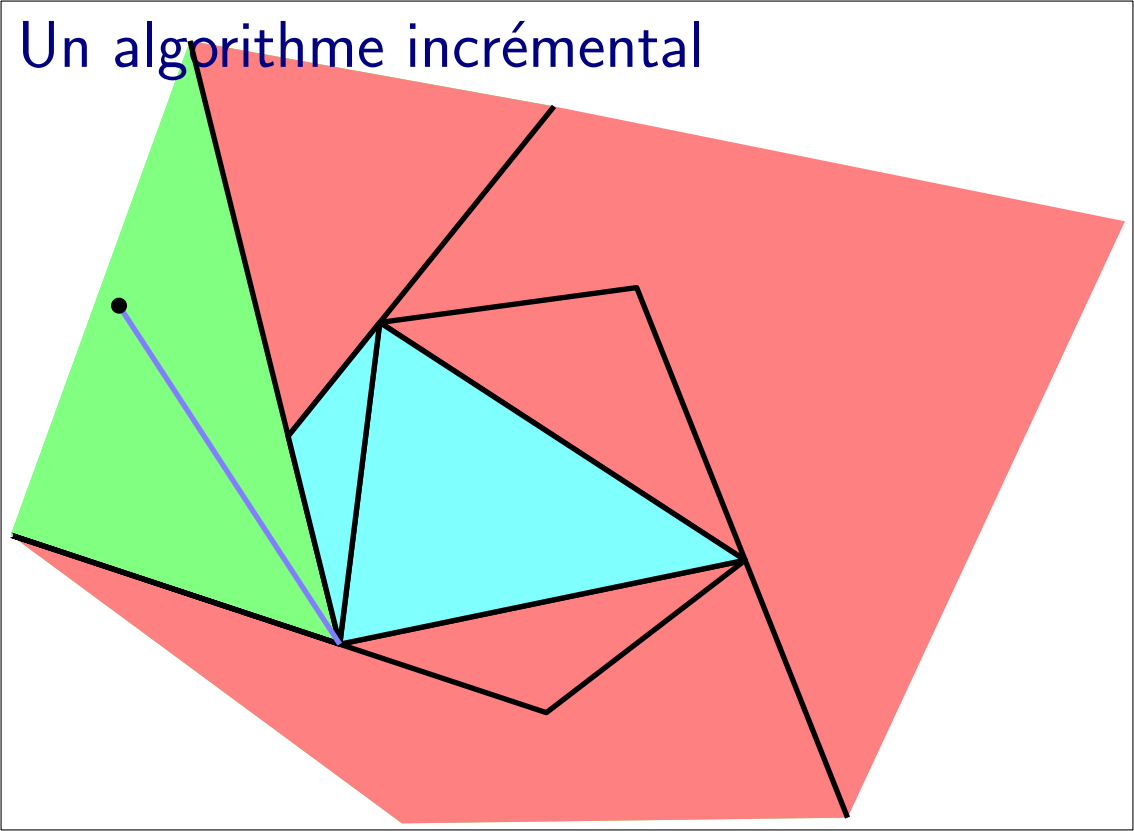
Un algorithme incrémental



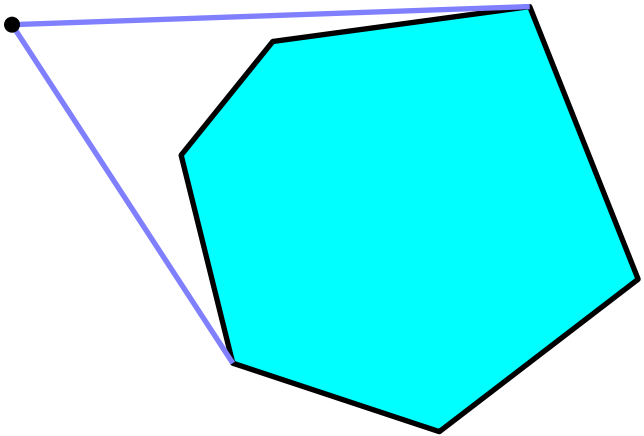
Un algorithme incrémental



Un algorithme incrémental

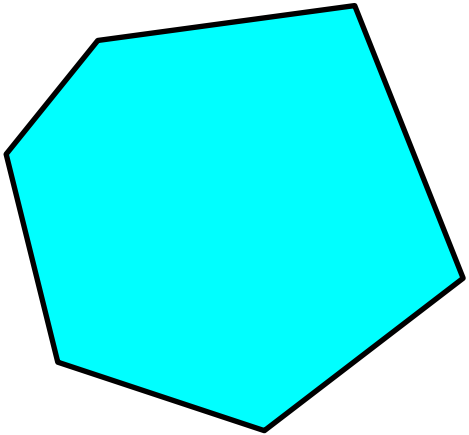


Un algorithme incrémental

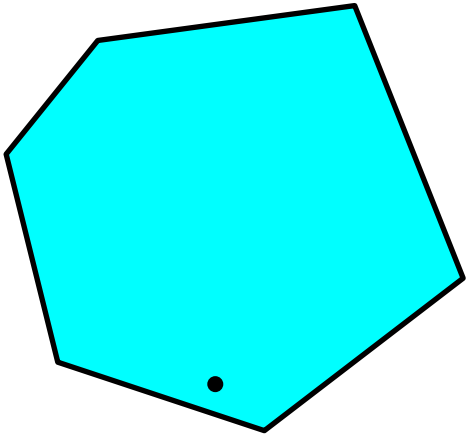


symétriquement

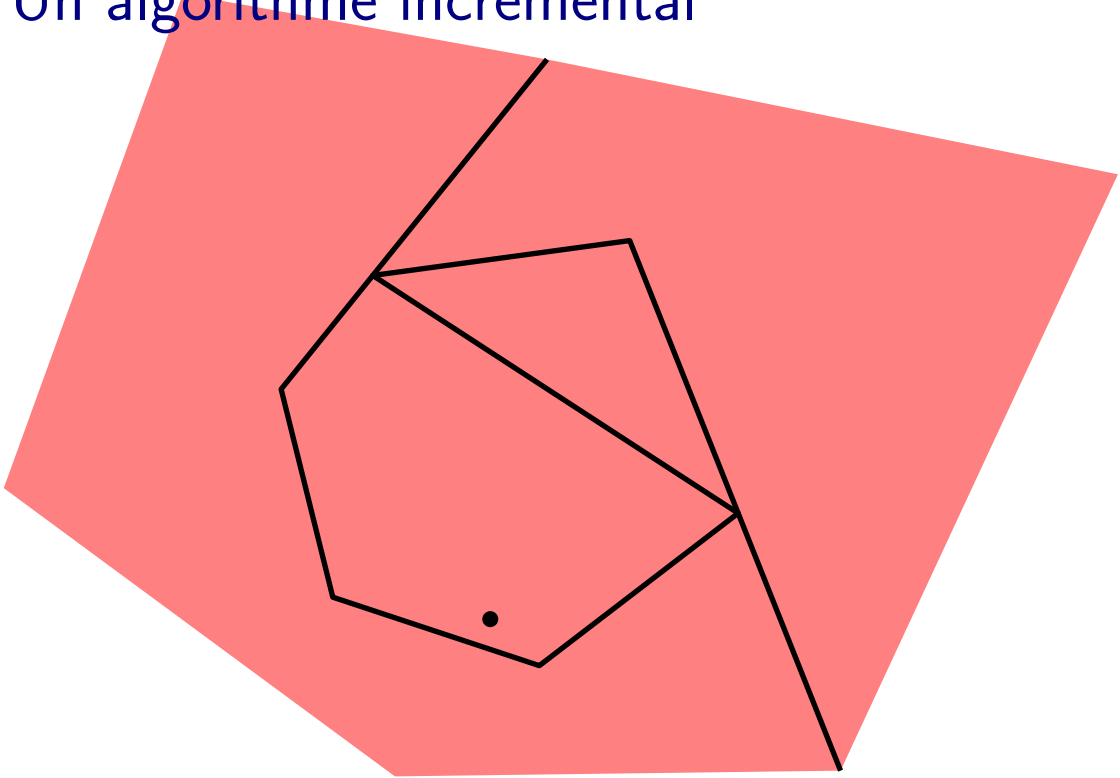
Un algorithme incrémental



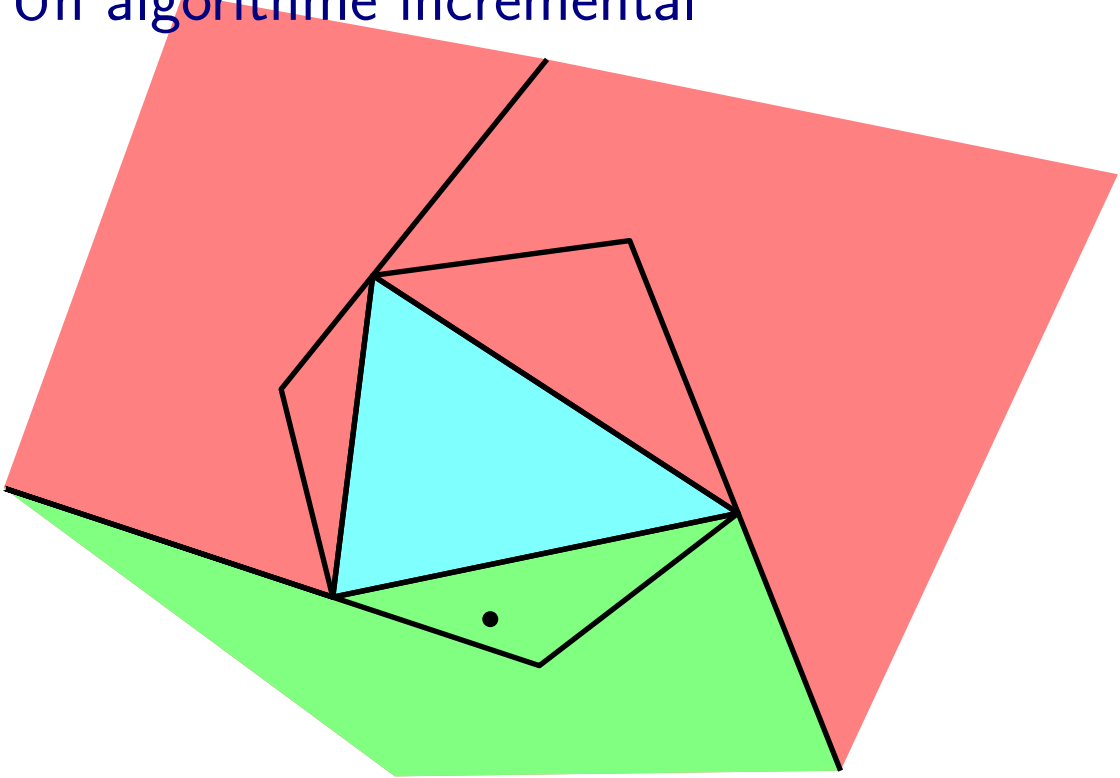
Un algorithme incrémental



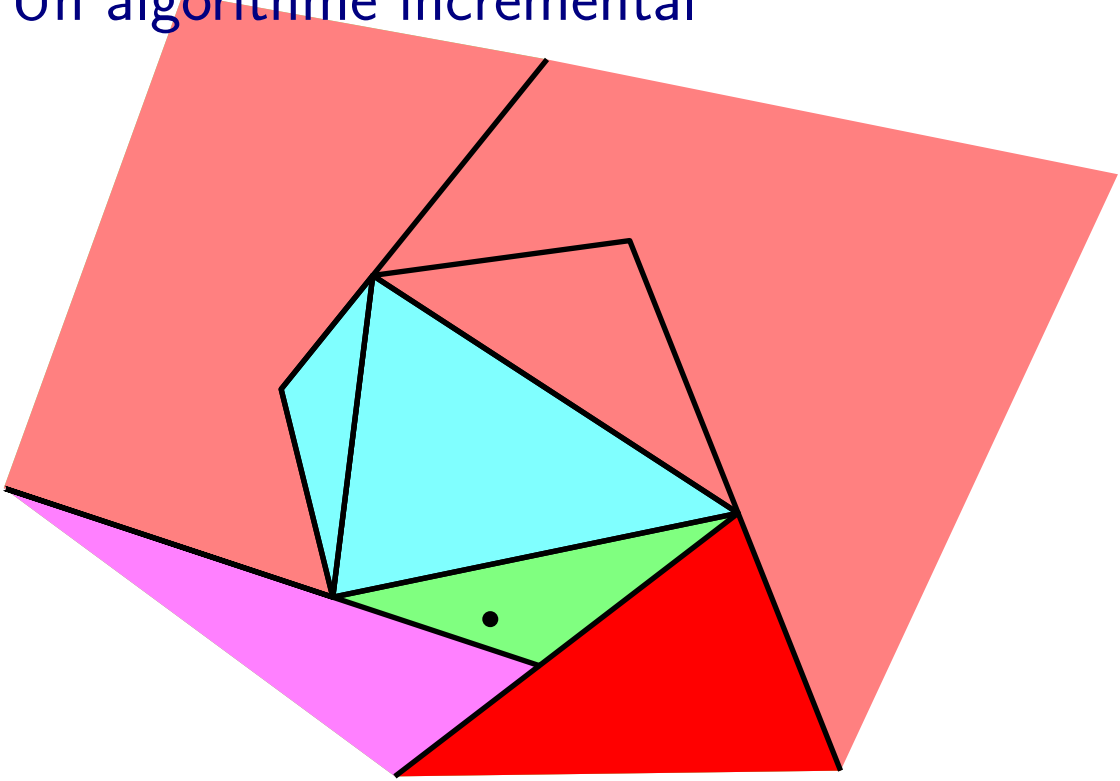
Un algorithme incrémental



Un algorithme incrémental

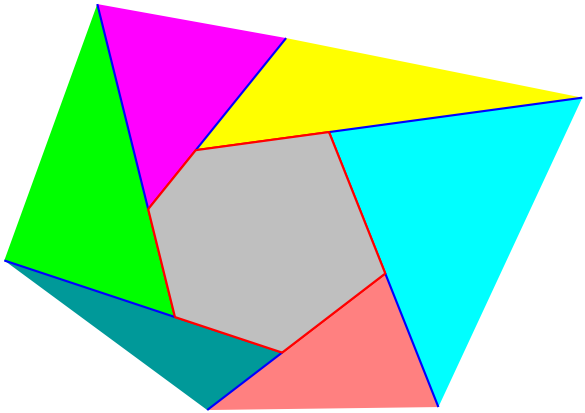


Un algorithme incrémental



Un algorithme incrémental

Complexité

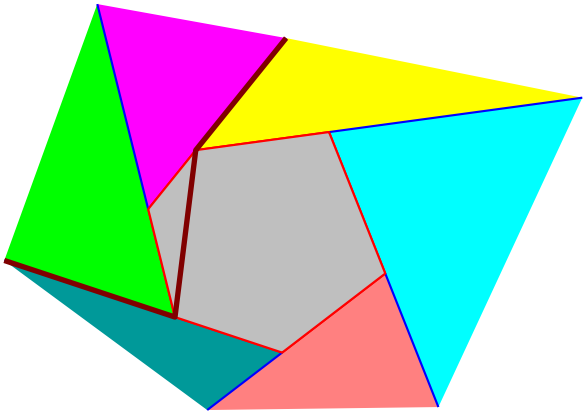


Images produites par Olivier Devillers

Un algorithme incrémental

Complexité

Complexité d'un nœud



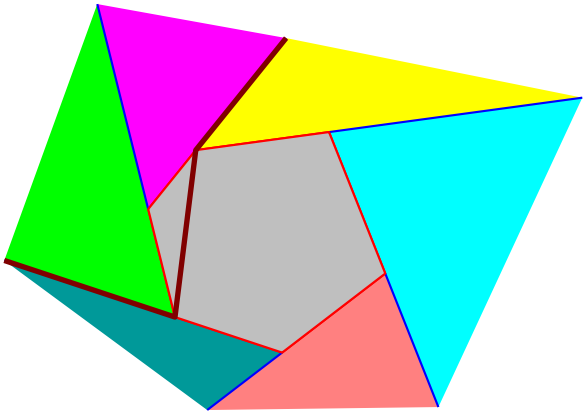
Images produites par Olivier Devillers

Un algorithme incrémental

Complexité

Complexité d'un nœud

$O(1)$



Images produites par Olivier Devillers

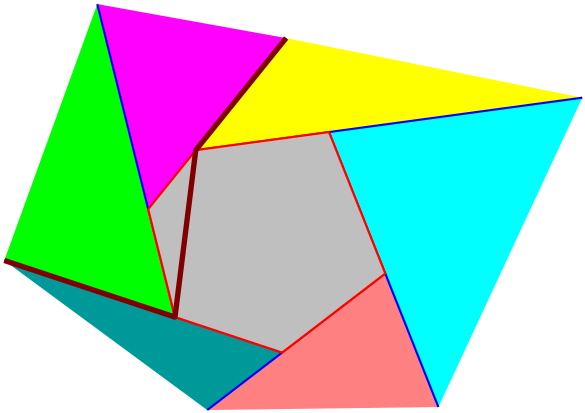
Un algorithme incrémental

Complexité

Complexité d'un nœud

$O(1)$

Arbre équilibré



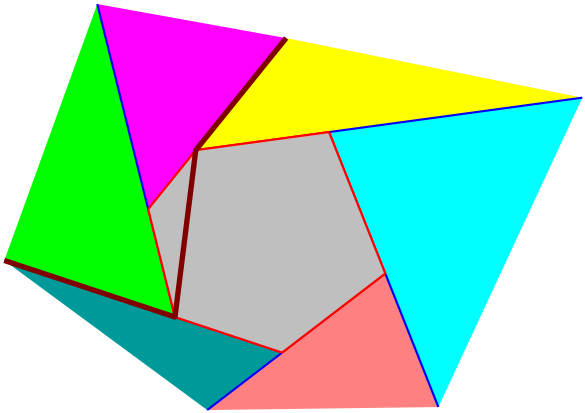
Images produites par Olivier Devillers

Un algorithme incrémental

Complexité

Complexité d'un nœud

$O(1)$

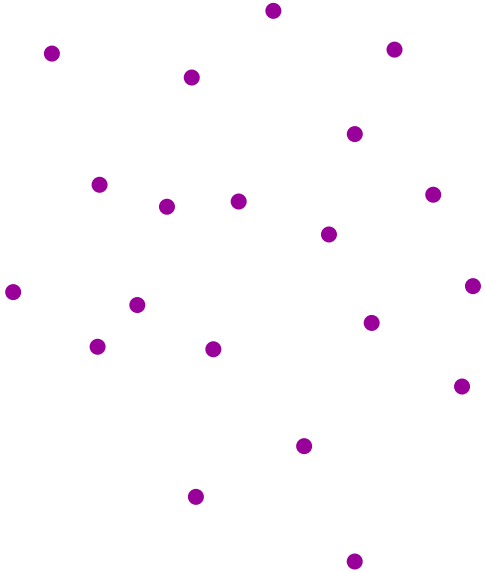


Arbre équilibré

$O(\log n)$ par insertion

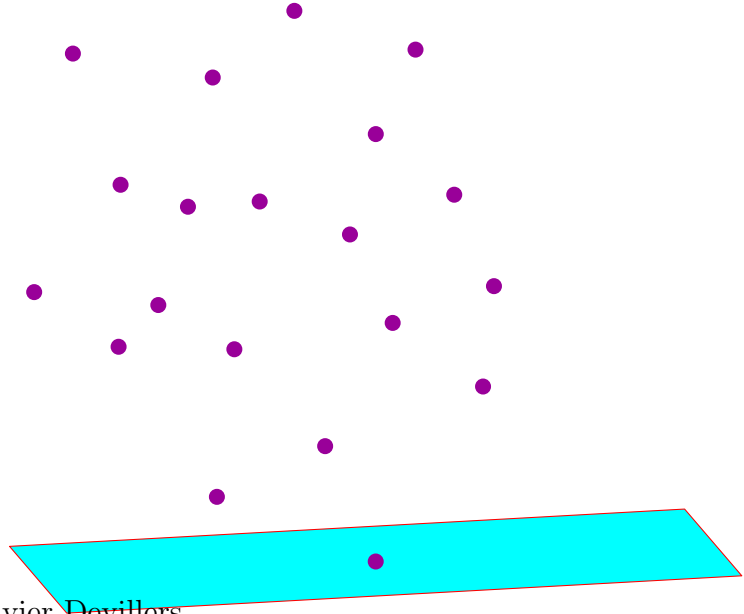
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



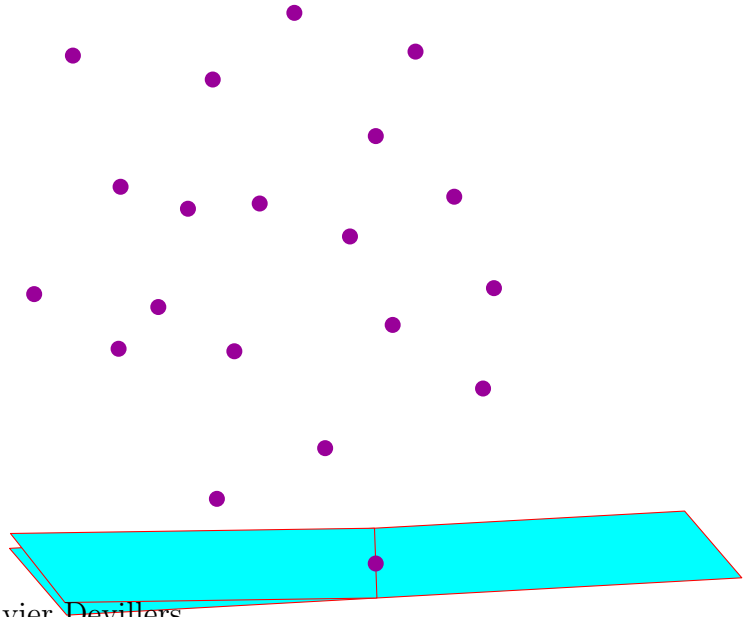
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



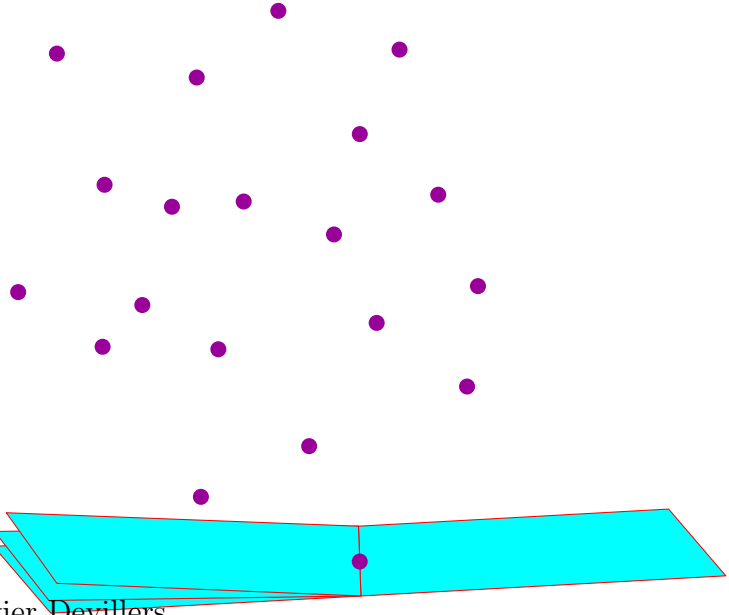
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



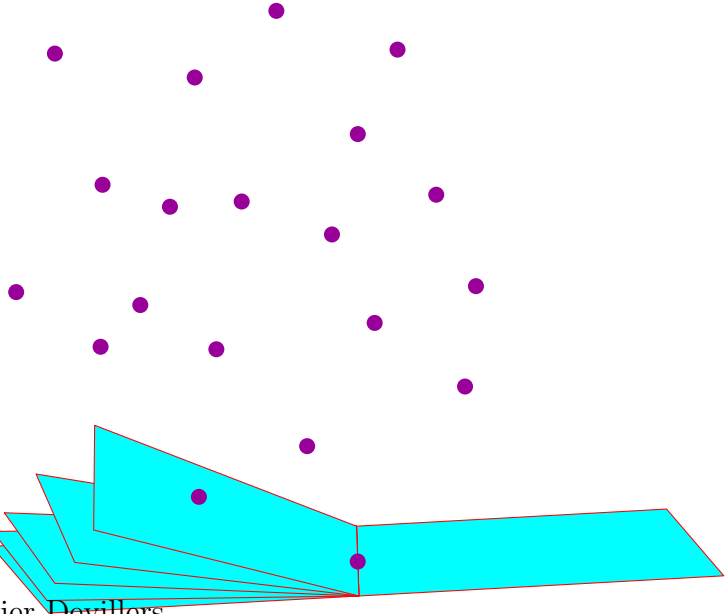
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



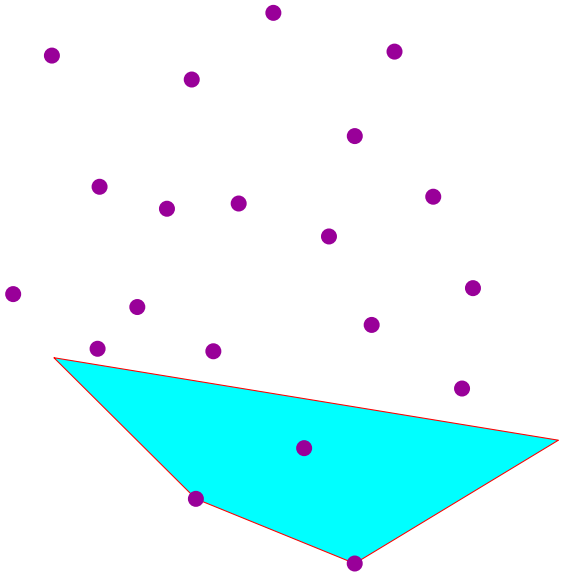
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



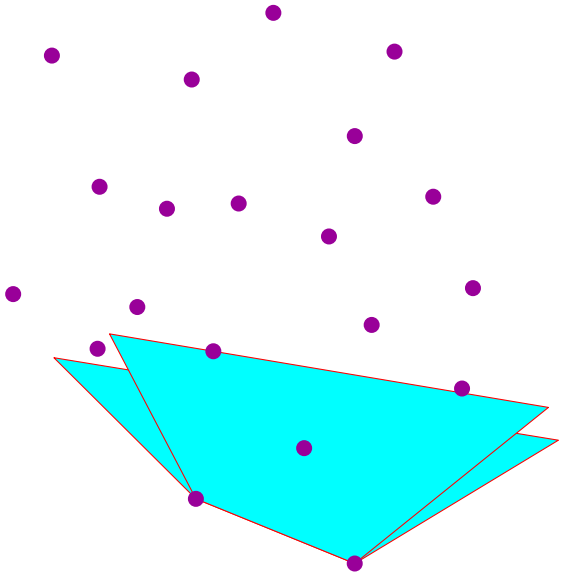
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



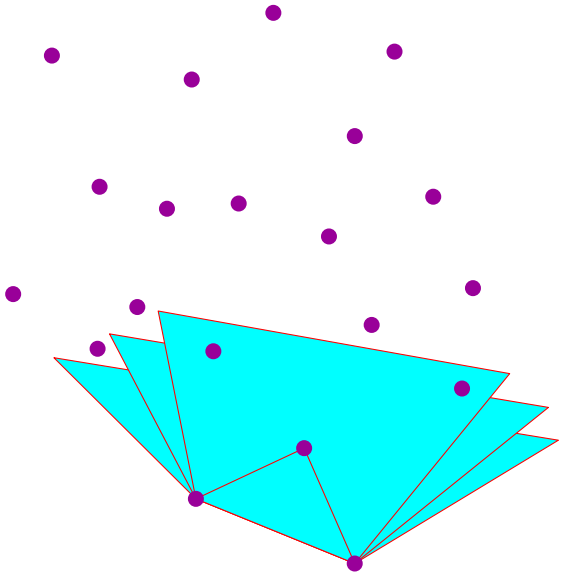
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



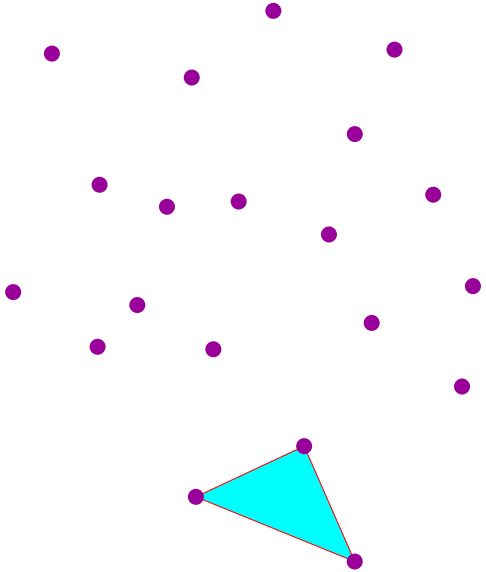
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



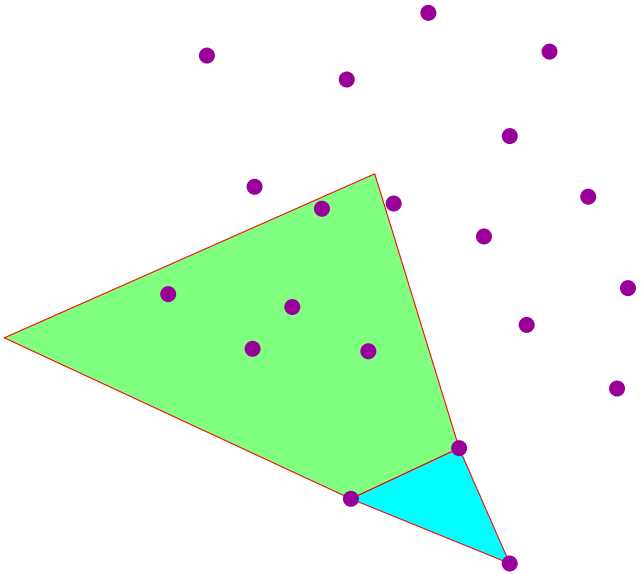
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



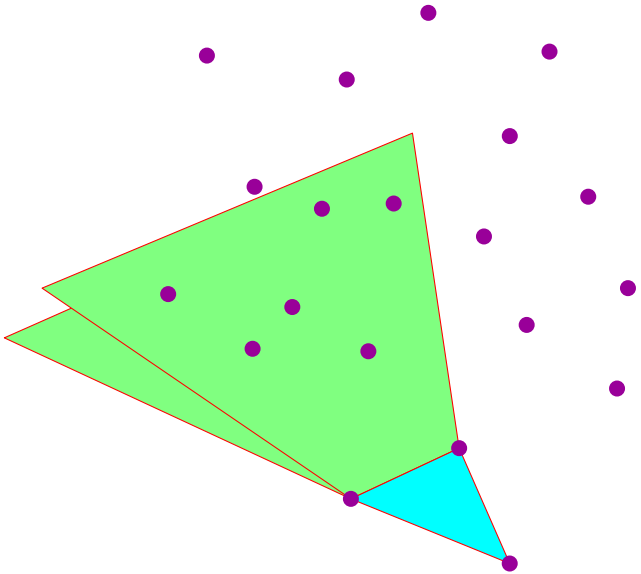
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



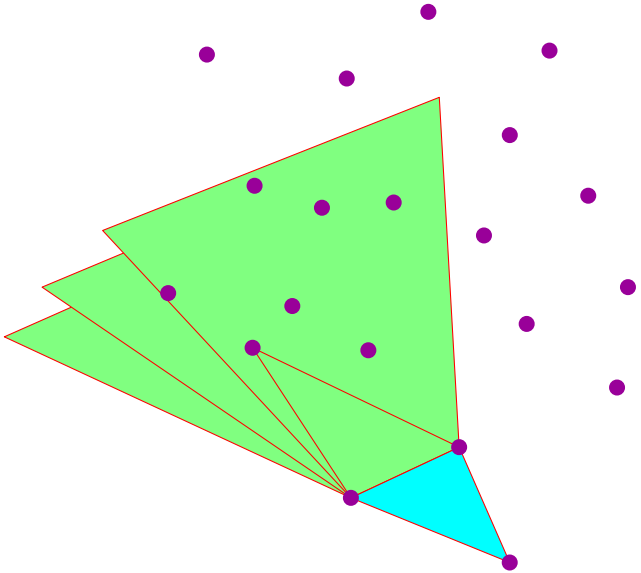
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



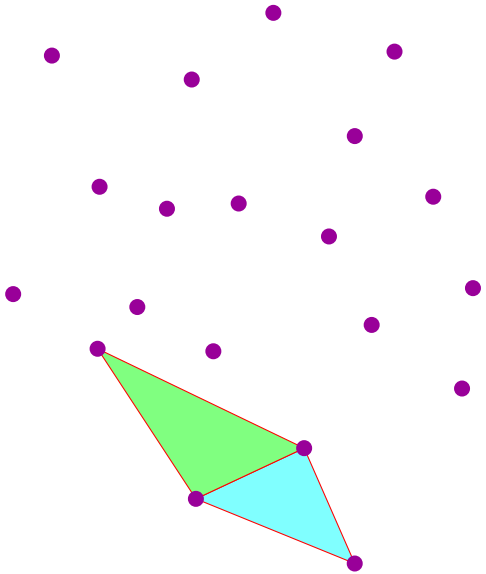
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



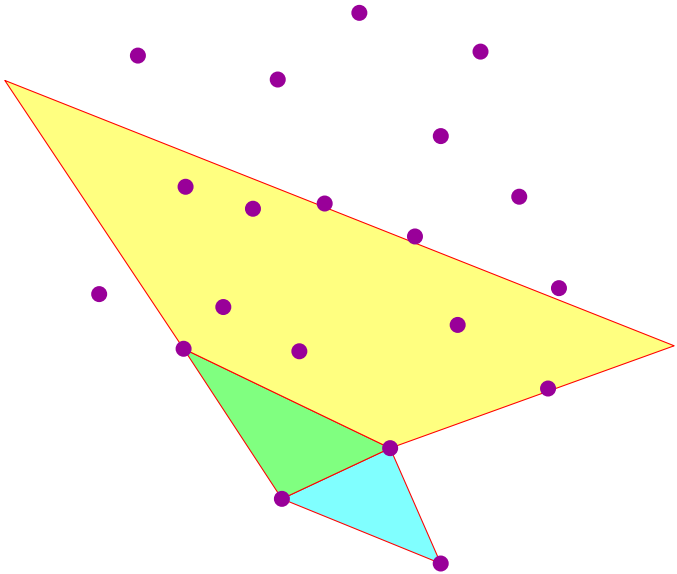
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



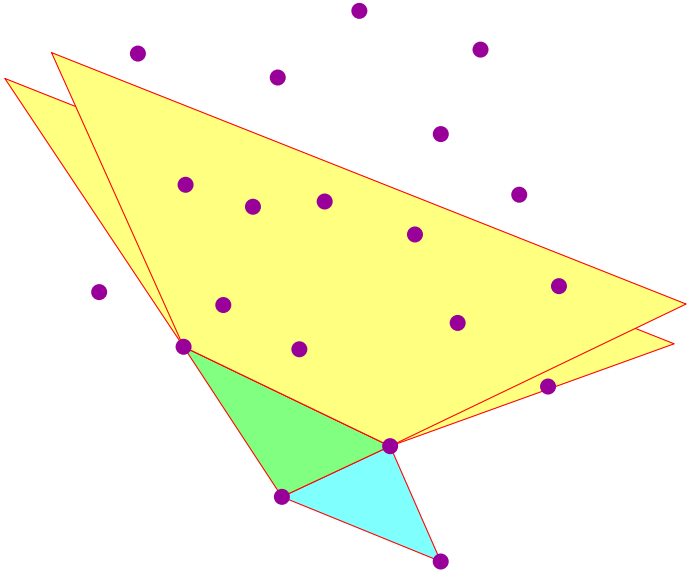
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



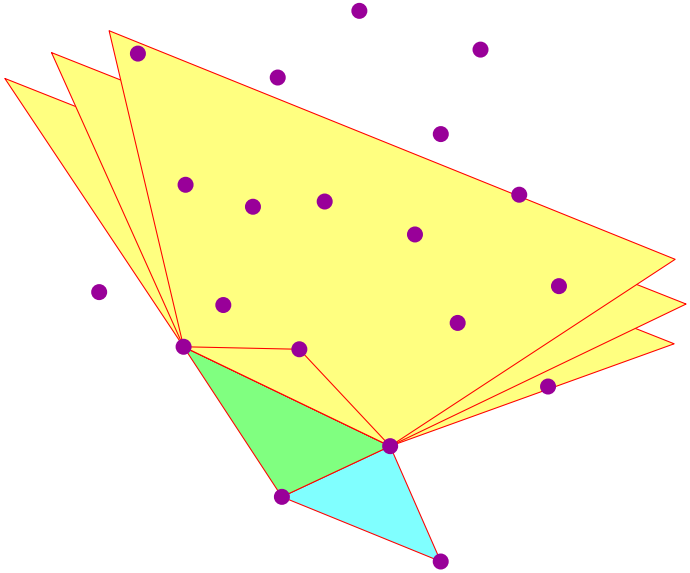
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



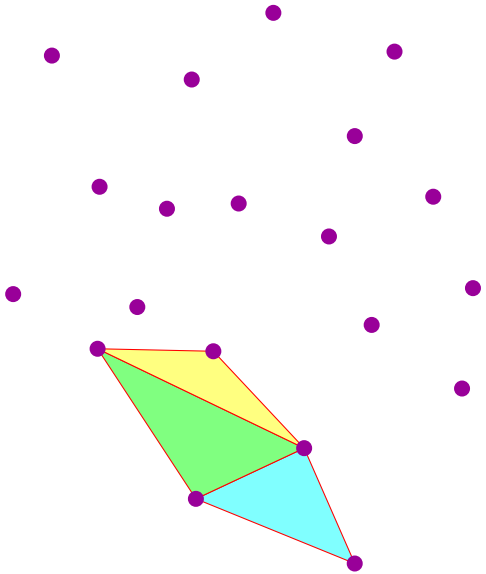
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



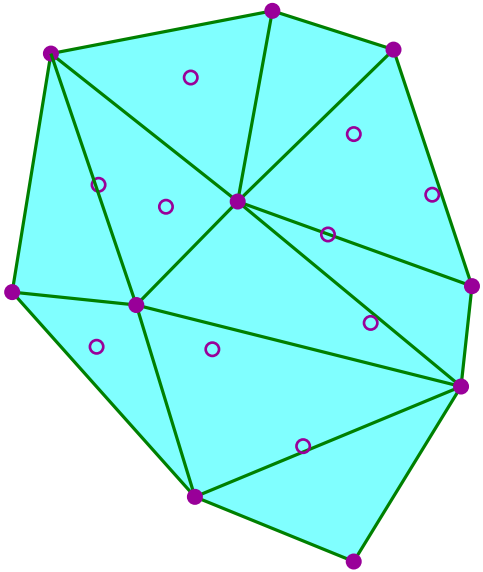
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



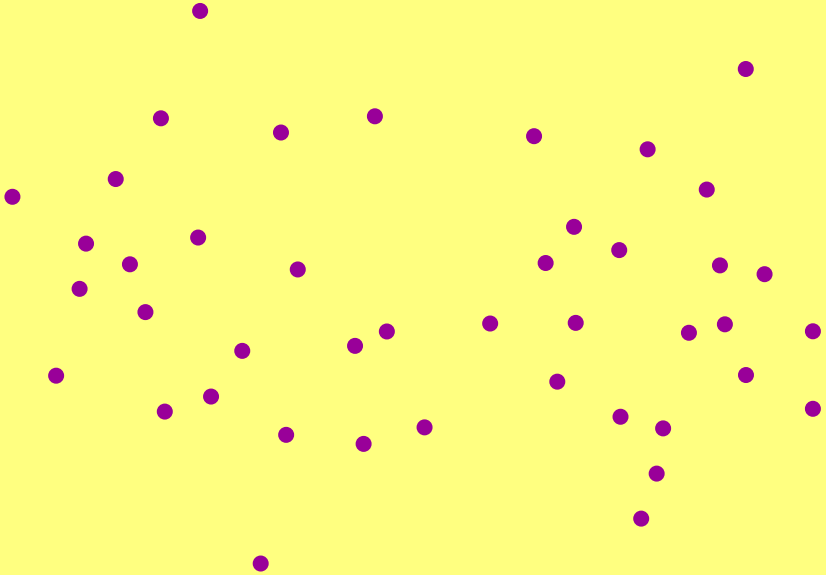
Images produites par Olivier Devillers

L'algorithme du paquet cadeau



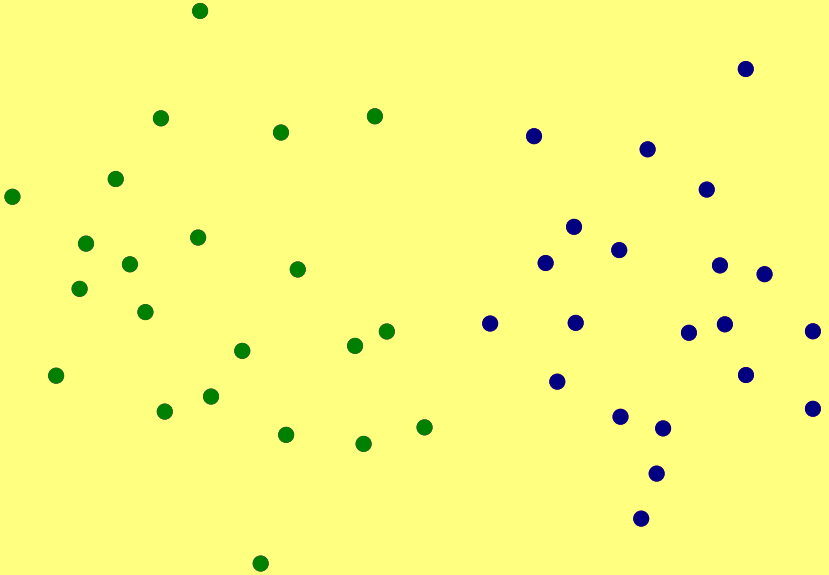
Images produites par Olivier Devillers

Algorithme division fusion



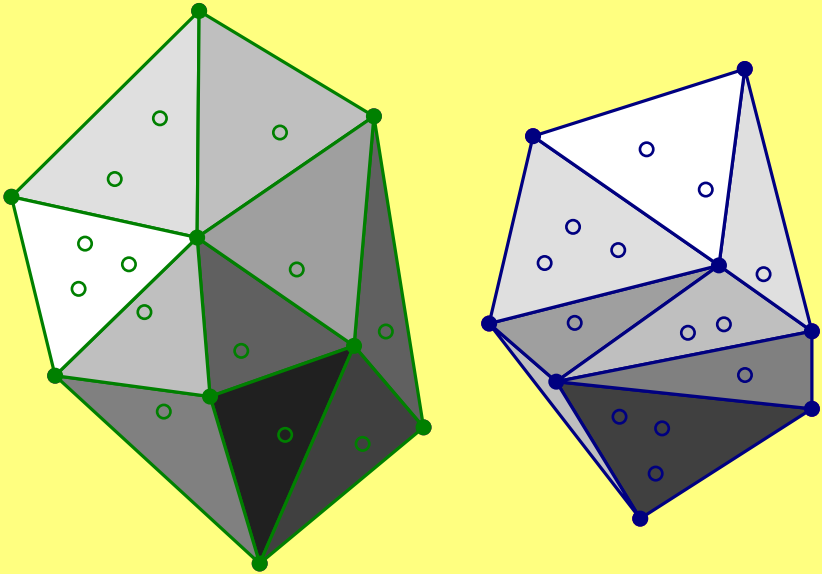
Images produites par Olivier Devillers

Algorithme division fusion



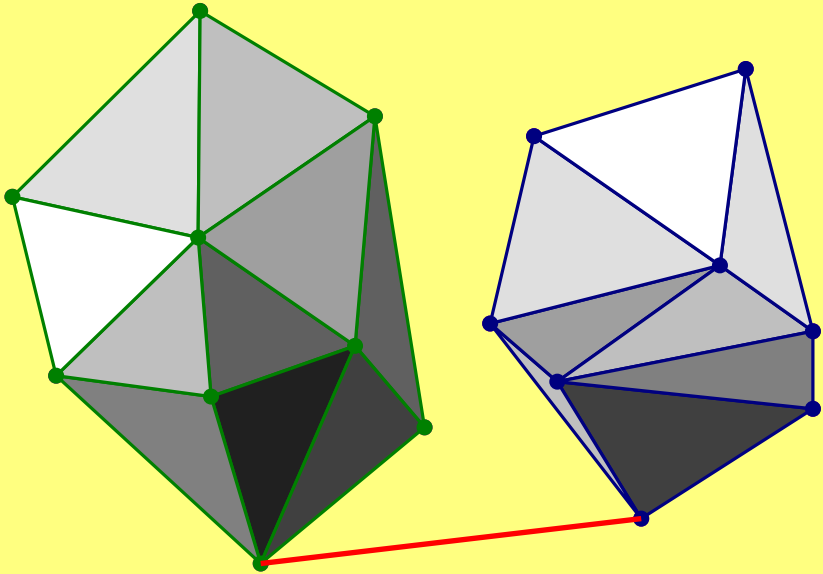
Images produites par Olivier Devillers

Algorithme division fusion



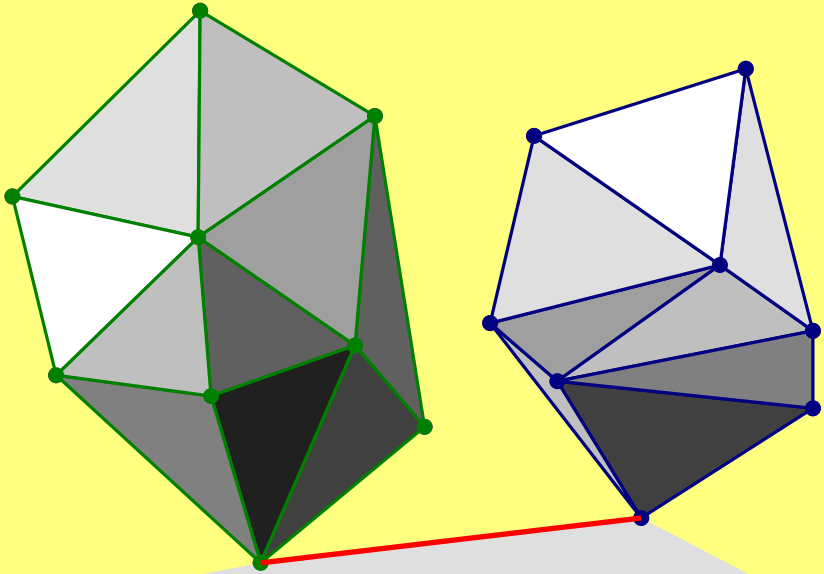
Images produites par Olivier Devillers

Algorithme division fusion



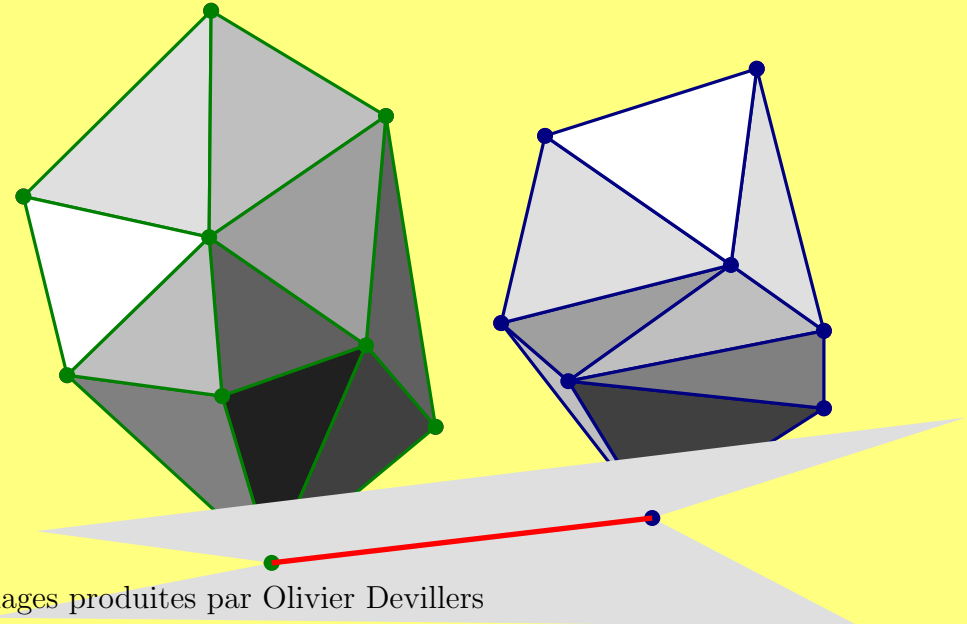
Images produites par Olivier Devillers

Algorithme division fusion



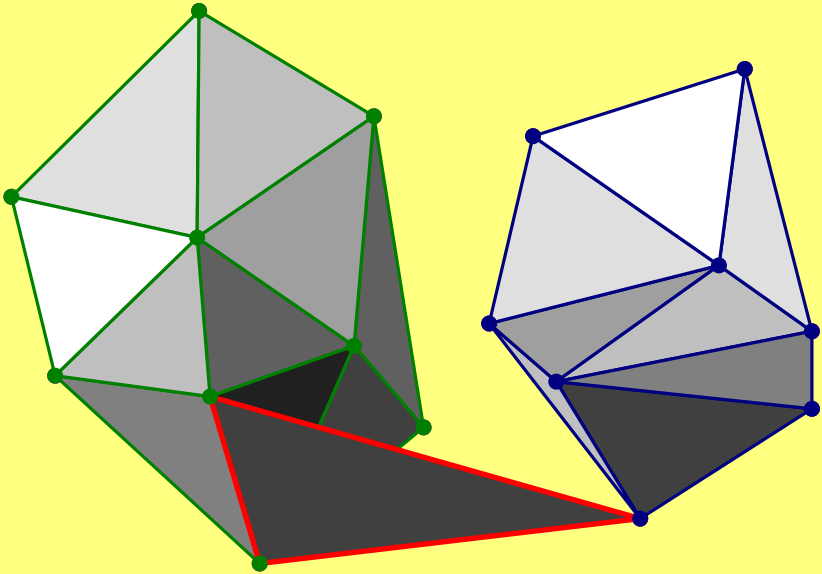
Images produites par Olivier Devillers

Algorithme division fusion



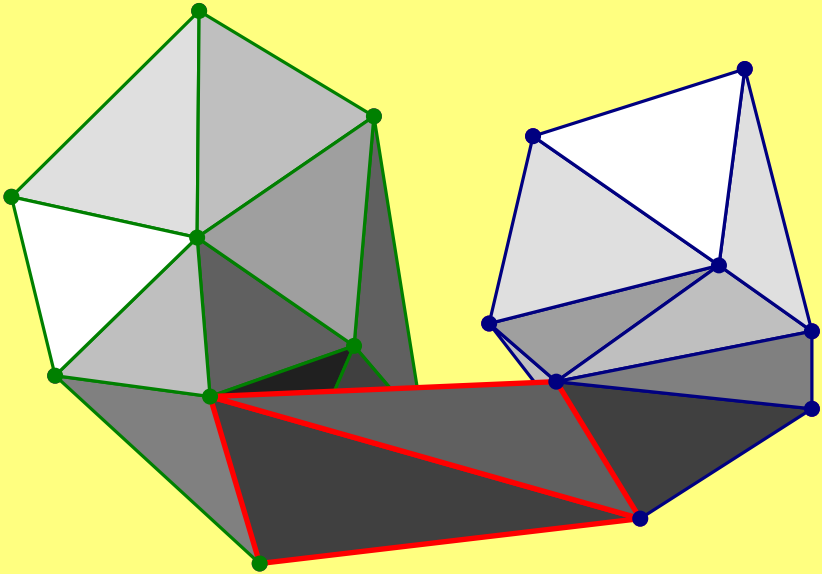
Images produites par Olivier Devillers

Algorithme division fusion



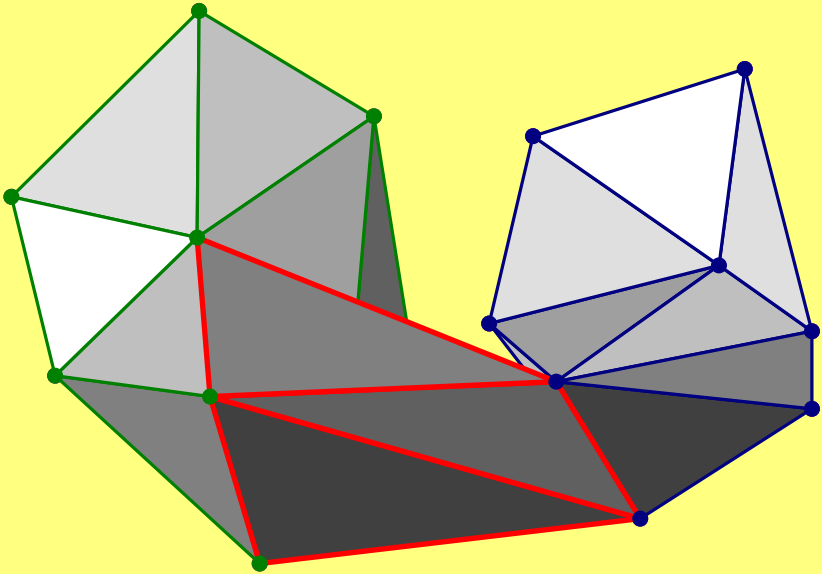
Images produites par Olivier Devillers

Algorithme division fusion



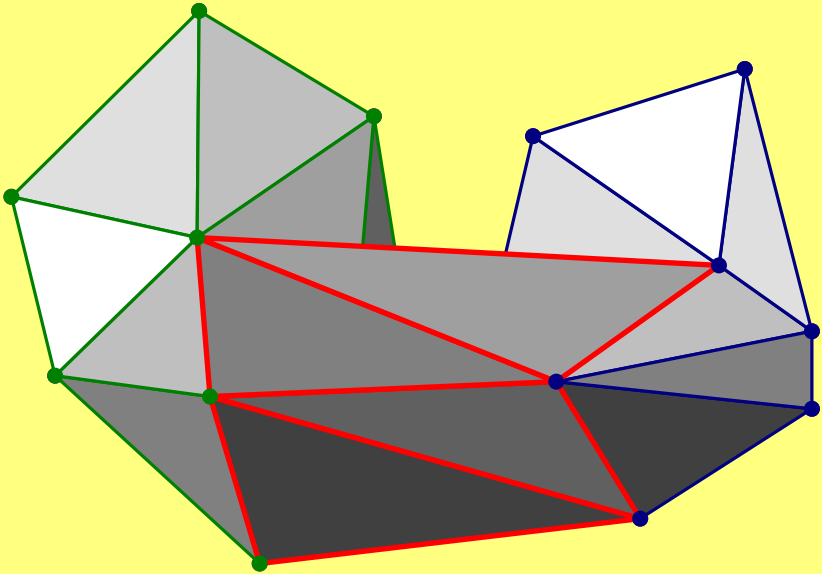
Images produites par Olivier Devillers

Algorithme division fusion



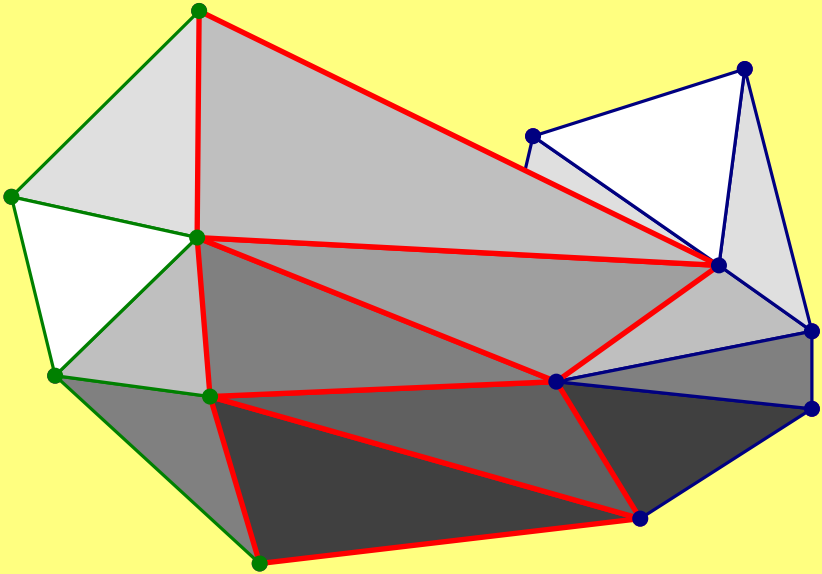
Images produites par Olivier Devillers

Algorithme division fusion



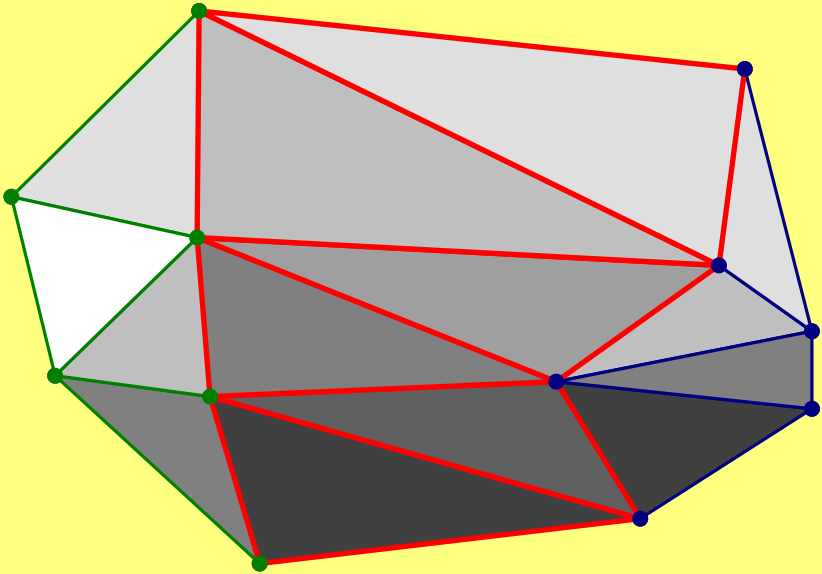
Images produites par Olivier Devillers

Algorithme division fusion



Images produites par Olivier Devillers

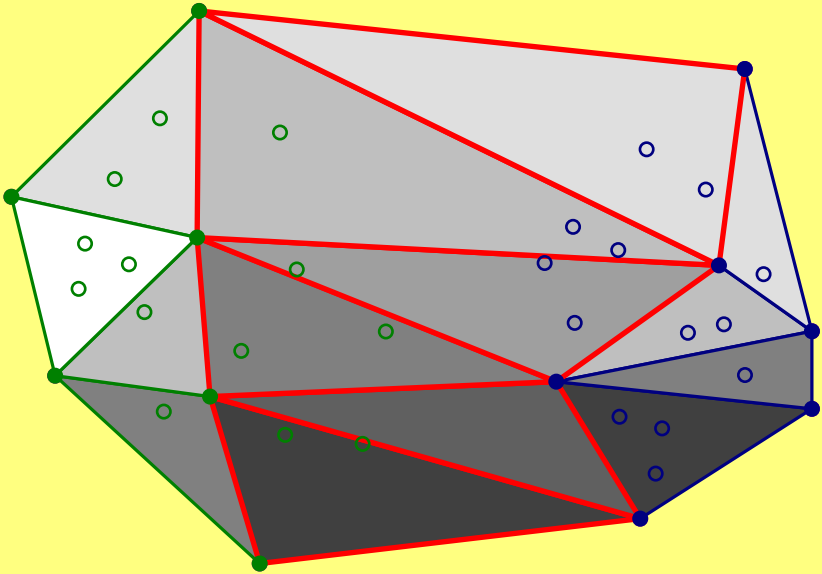
Algorithme division fusion



Images produites par Olivier Devillers

Algorithme division fusion

$O(n)$

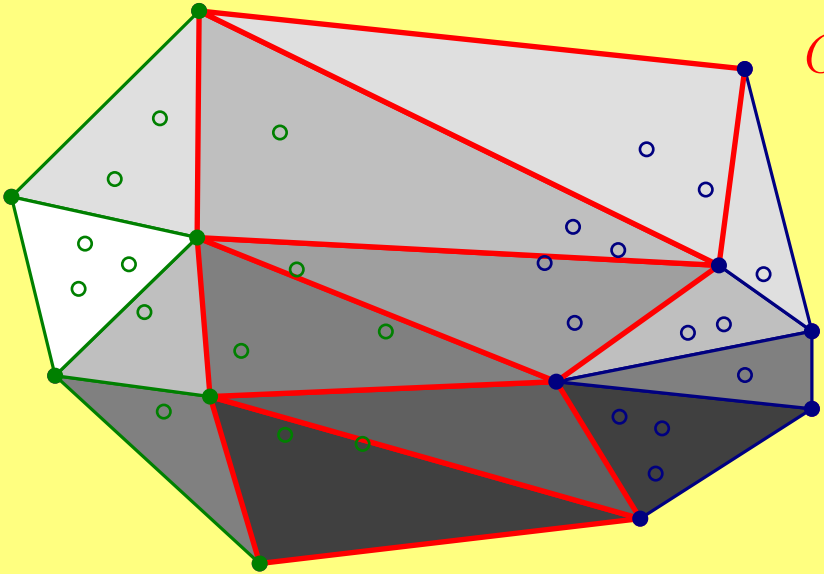


Images produites par Olivier Devillers

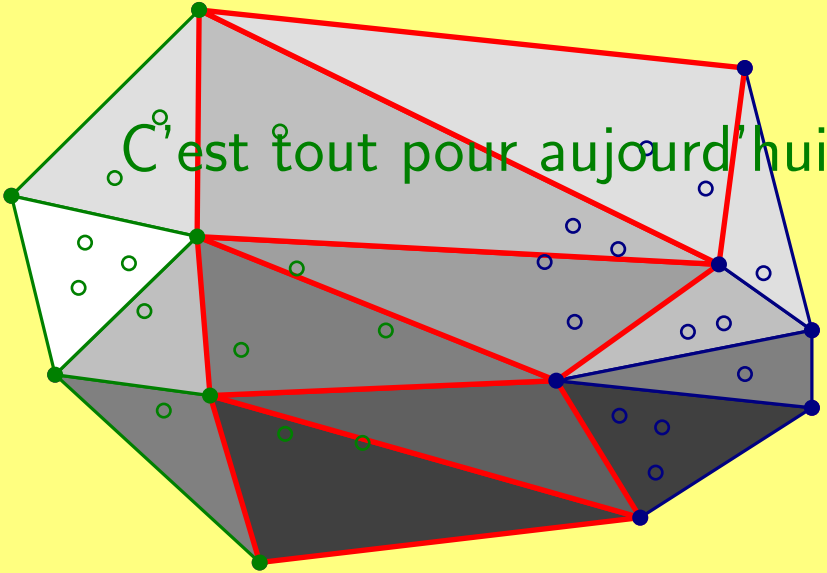
Algorithme division fusion

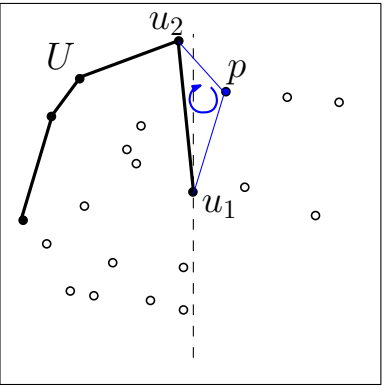
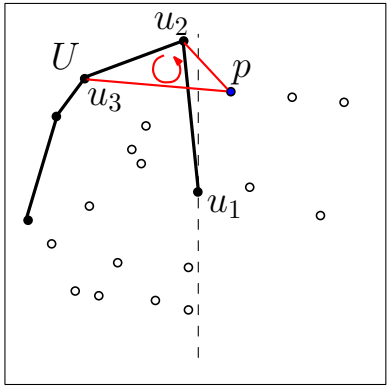
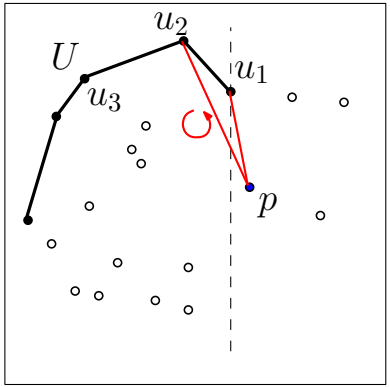
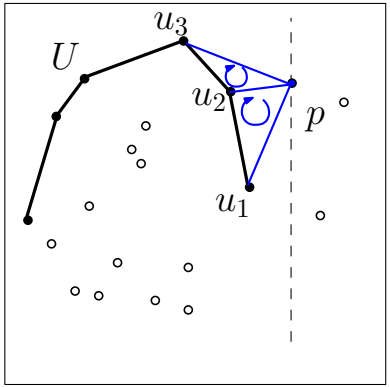
$O(n)$

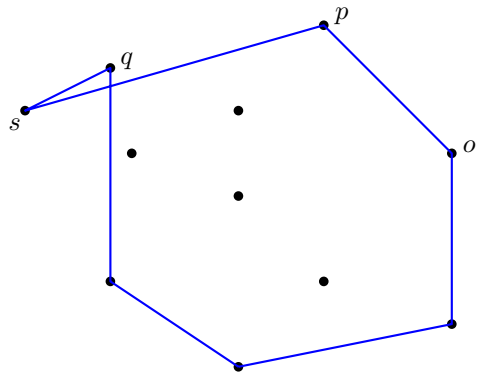
$O(n \log n)$

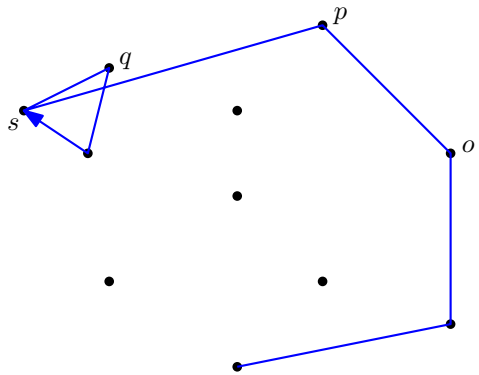


Images produites par Olivier Devillers

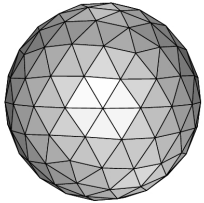

C'est tout pour aujourd'hui

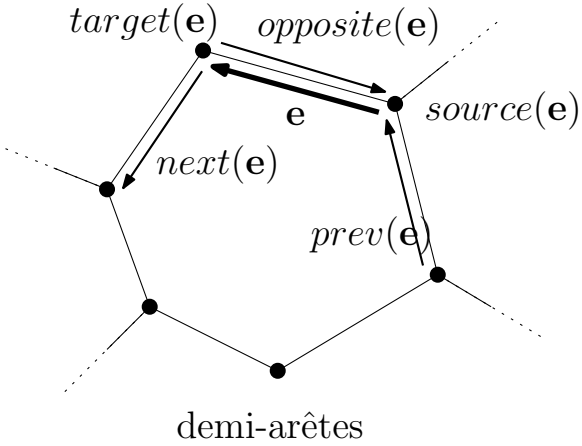

u_2
U
p
u_1

u_2
U
p
u_3
u_1

u_2
U
u_1
u_3
p

u_3
U
u_2
p
u_1


p
q
s
o


p
q
s
o

Structures de données géométriques





demi-arêtes

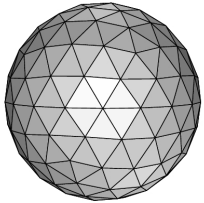
```
class Point{
  double x;
  double y;
}
```

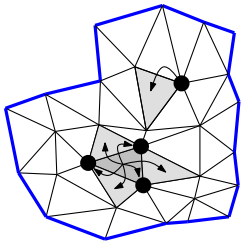
information géométrique

```
class Halfedge{
  Halfedge  prev, next, opposite;
  Vertex source, target;
  Face f;
}
class Vertex{
  Halfedge  e;
  Point p;
}
class Face{
  Halfedge  e;
}
```

information combinatoire

A base de triangles





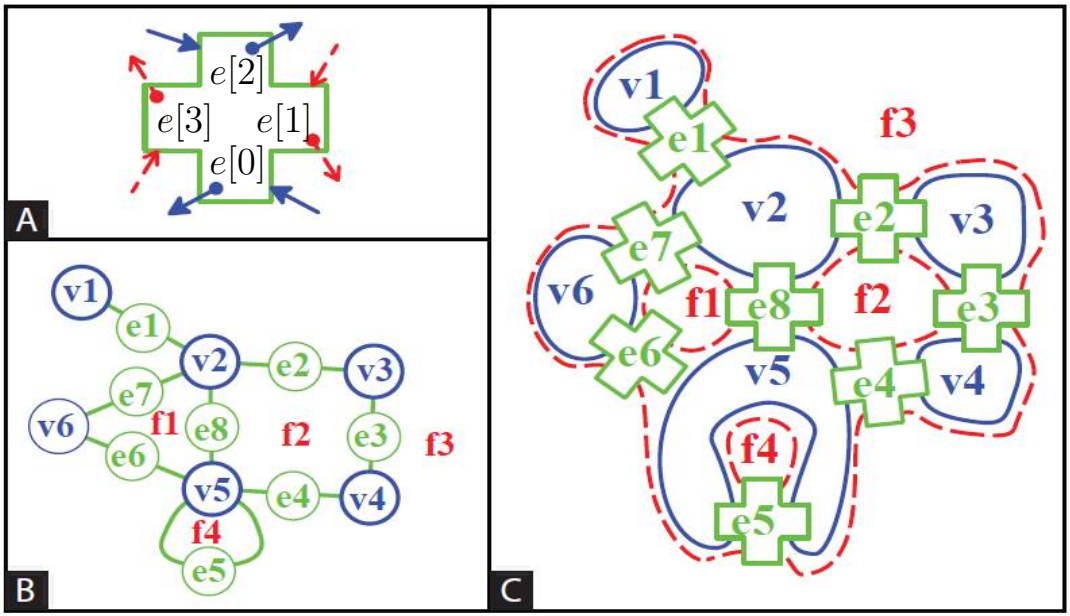
```
class Point{
  double x;
  double y;
}
```

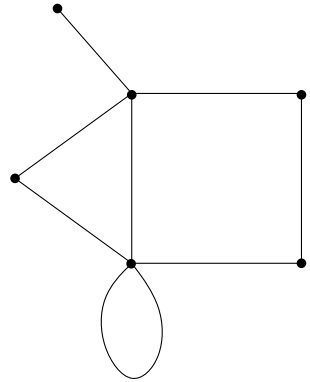
information géométrique

```
class Triangle{
  Triangle t1, t2, t3;
  Vertex v1, v2, v3;
}class Vertex{
  Triangle  root;
  Point p;
}
```

information combinatoire

Quad-edge

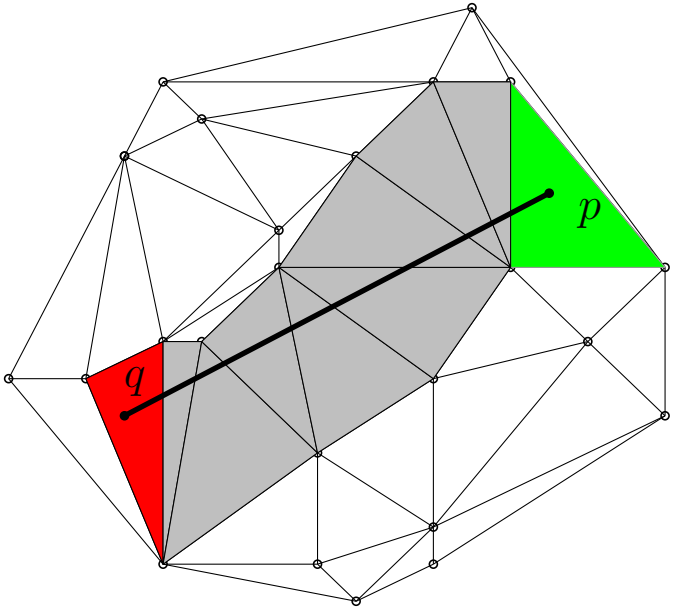

A
e[2]
e[3]
e[1]
e[0]
B
v1
e1
v2
e2
v3
e7
f1
e8
f2
e3
f3
v6
e6
v5
e4
v4
f4
e5
C
v1
e1
f3
v2
e2
v3
e7
v6
f1
e8
f2
e3
e6
v5
e4
v4
f4
e5

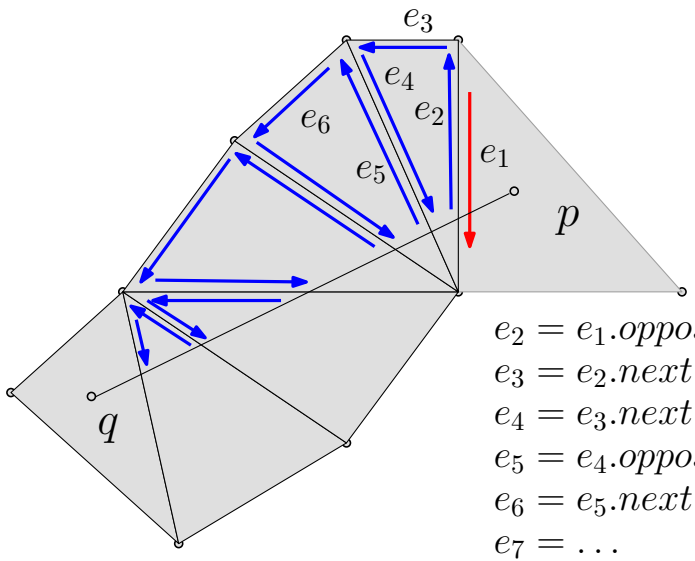


Planar Point location

par marche aléatoire

complexité $O(\sqrt{n})$

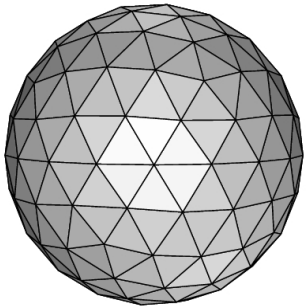




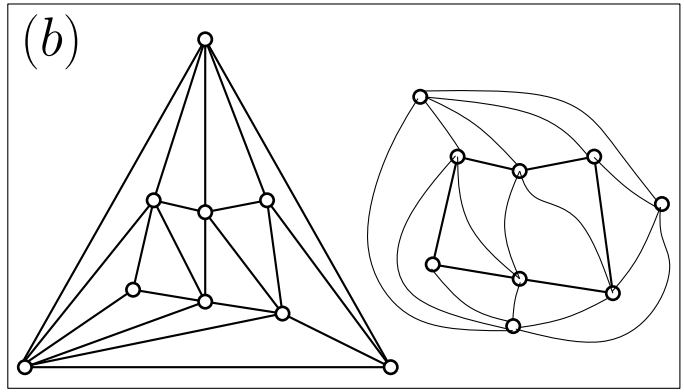
$e_2 = e_1.opposite$
$e_3 = e_2.next$
$e_4 = e_3.next$
$e_5 = e_4.opposite$
$e_6 = e_5.next$
$e_7 = \ldots$

Structures de données géométriques

(a)

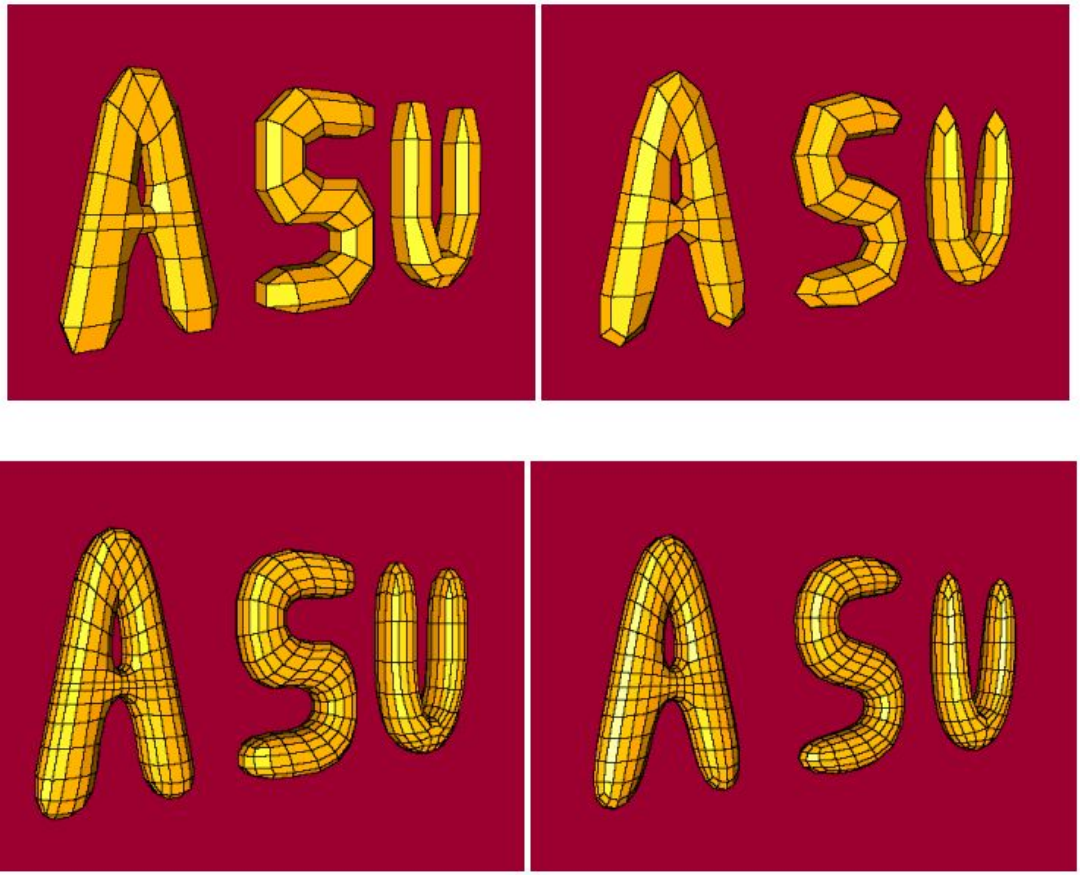


maillage triangulaire

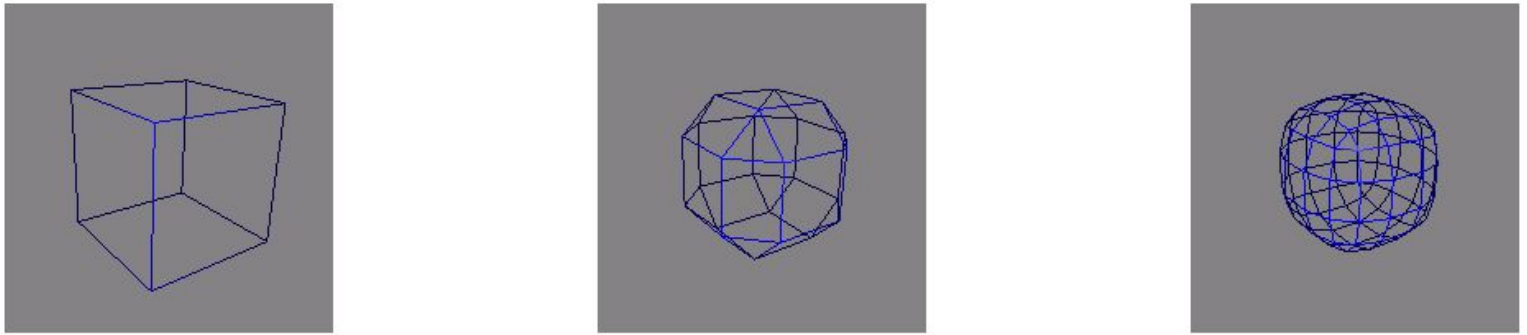


triangulation planaire

Surfaces de Subdivision (TD, exo)

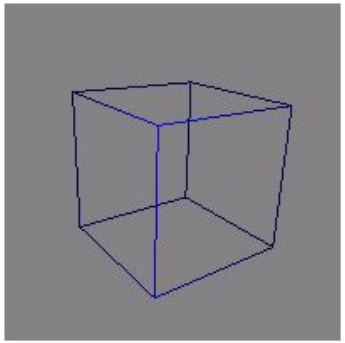
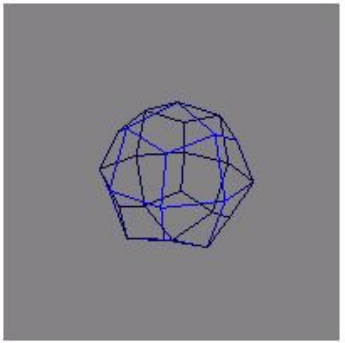
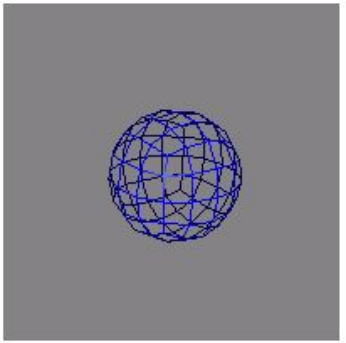


Doo-Sabin subdivision



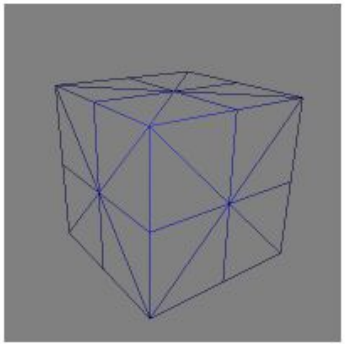
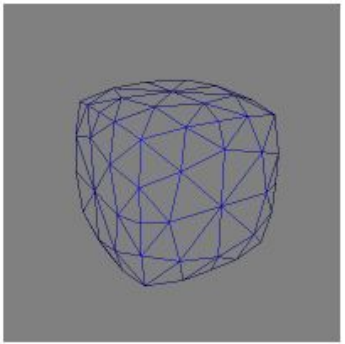
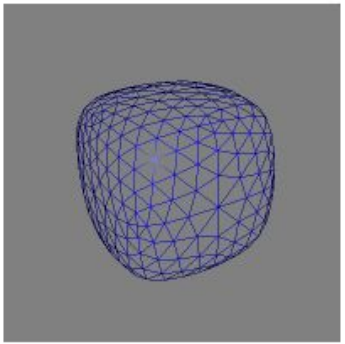
Cube, smoothed twice with repeated applications of the Doo-Sabin scheme

Catmull-Clark subdivision

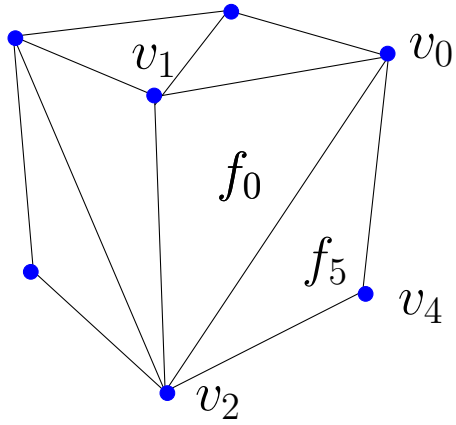
Cube, smoothed twice with repeated applications of the Catmull-Clark scheme

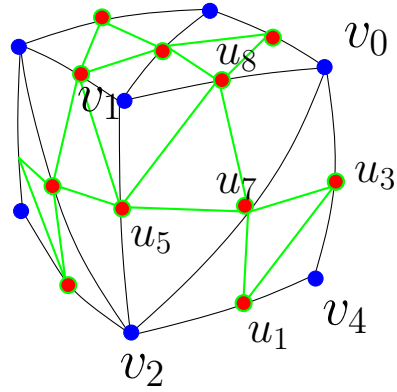
Loop subdivision (exo du TD)

Tessellated cube, smoothed twice with repeated applications of the Loop scheme (Rendered without hidden lines to keep the image clean)

Loop subdivision
updating the connectivity (incidence relations)





n sommets, e aretes, f faces

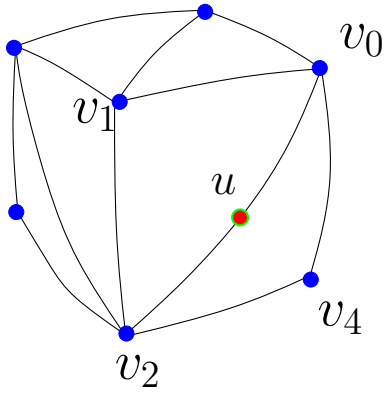
$n + e$ sommets, $4f$ faces

$$f_0 : \underset{\cdots}{v_0, v_1, v_2}$$

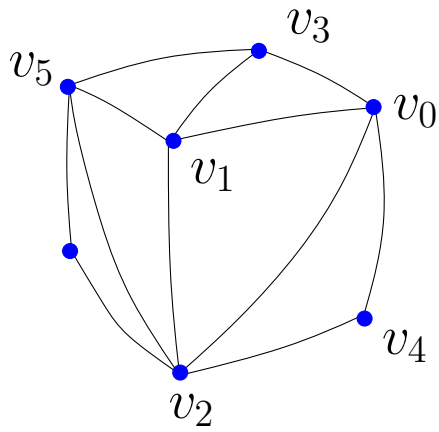
$$f_5 : \underset{\cdots}{v_2, v_4, v_0}$$

$$f_0 \to (f'_0, f'_1, f'_2, f'_3) \quad \left[\begin{array}{l} f'_0 : \underset{\cdots}{u_7, u_8, u_5} \\ f'_2 : \underset{\cdots}{u_7, u_5, v_2} \end{array} \right.$$

Loop subdivision
updating the geometry (point positions)



$$u = \tfrac{3}{8}v_0 + \tfrac{3}{8}v_2 + \tfrac{1}{8}v_1 + \tfrac{1}{8}v_3$$



$$v_i' = (1 - \alpha d)v_i + \alpha \, \Sigma_{j=1}^{d} \, v_{i_j}$$

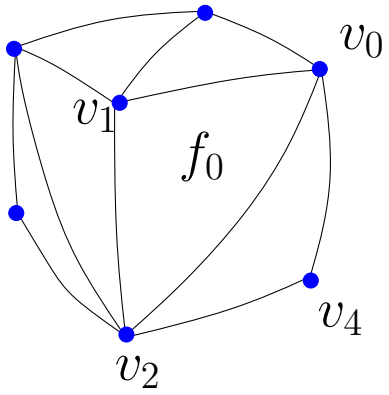
d est le degré de v_i
v_{i_j} est le j-th voisin de v_i

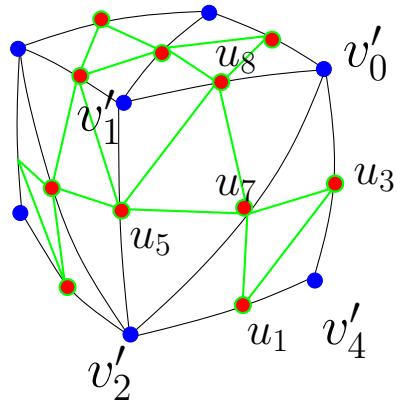
$$\begin{cases} \alpha = \frac{3}{16}, \text{ si } d = 3 \\ \alpha = \frac{3}{8d}, \text{ si } d > 3 \end{cases}$$

TD1 (exo): suggestion pour la solution

utiliser le constructeur suivant

$TriangulationDS\_2 < Point\_3 >$ `result=new` $TriangulationDS_2 < Point\_3 >$`(`$Point\_3$ `[], int[][]);`





Stoquer les sommets u_j dans une table de hachage

$f_0 : [v_0, v_1, v_2]$

$$\begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ \cdots \\ v_{n-1} \end{bmatrix}$$

$f'_0 : [u_5, u_7, u_8]$

$f'_1 : [u_5, u_8, v'_1]$

$f'_2 : [u_7, u_5, v'_2]$

$f'_3 : [u_8, u_7, v'_0]$

$f'_0 : [v'_{n+5}, v'_{n+5}, v'_{n+8}]$

$f'_1 : [v'_{n+5}, v'_{n+8}, v'_1]$

$f'_2 : [u_7, u_5, v'_2]$

$f'_3 : [u_8, u_7, v'_0]$

$$\begin{bmatrix} v'_0 \\ v'_1 \\ v'_2 \\ \cdots \\ v'_{n-1} \end{bmatrix} \quad \begin{bmatrix} v'_0 \\ v'_1 \\ v'_2 \\ \cdots \\ v'_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} u_0 \\ u_1 \\ \cdots \\ u_{e-1} \end{bmatrix} \quad \begin{bmatrix} v'_n \\ v'_{n+1} \\ \cdots \\ v'_{n+e-1} \end{bmatrix}$$