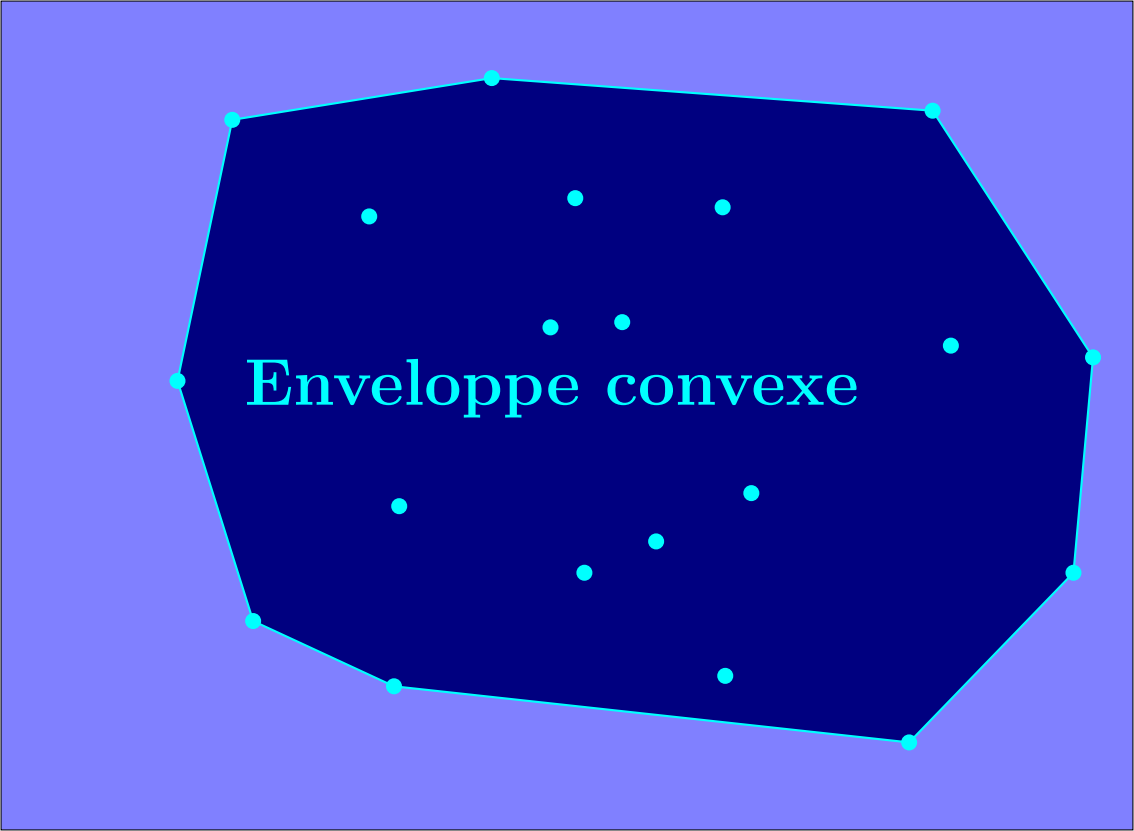
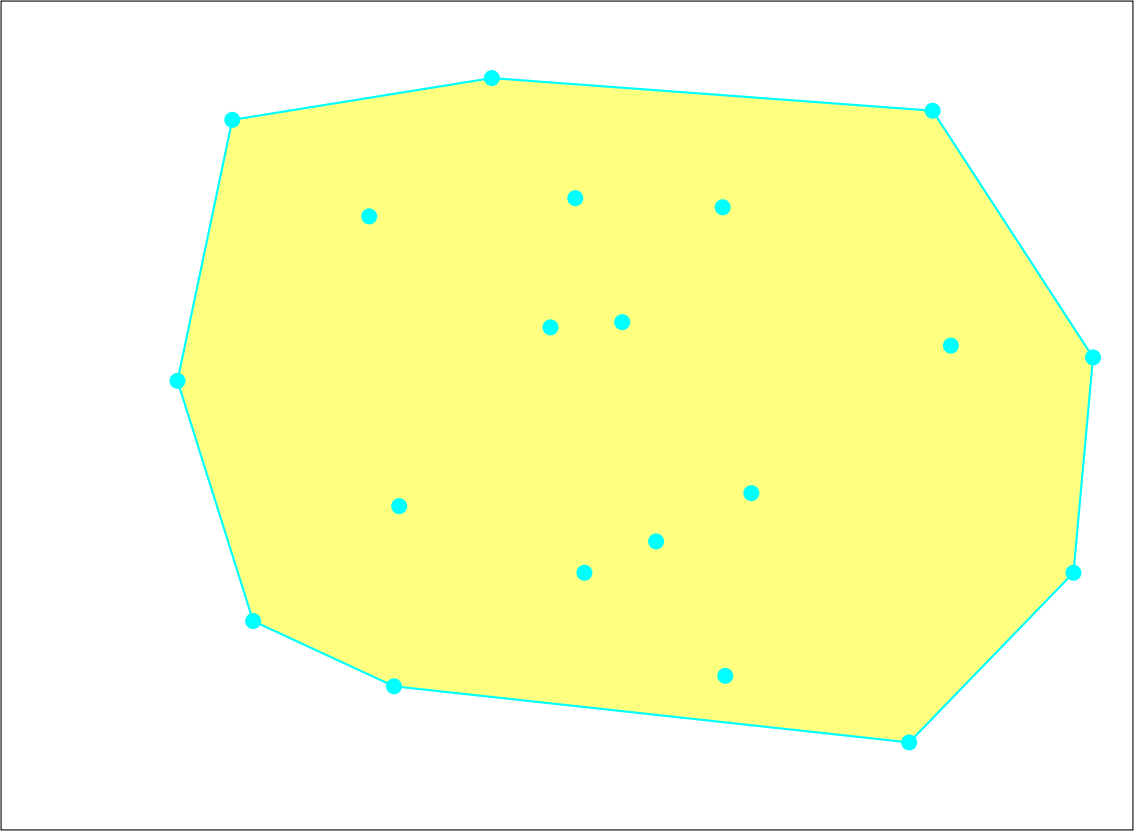
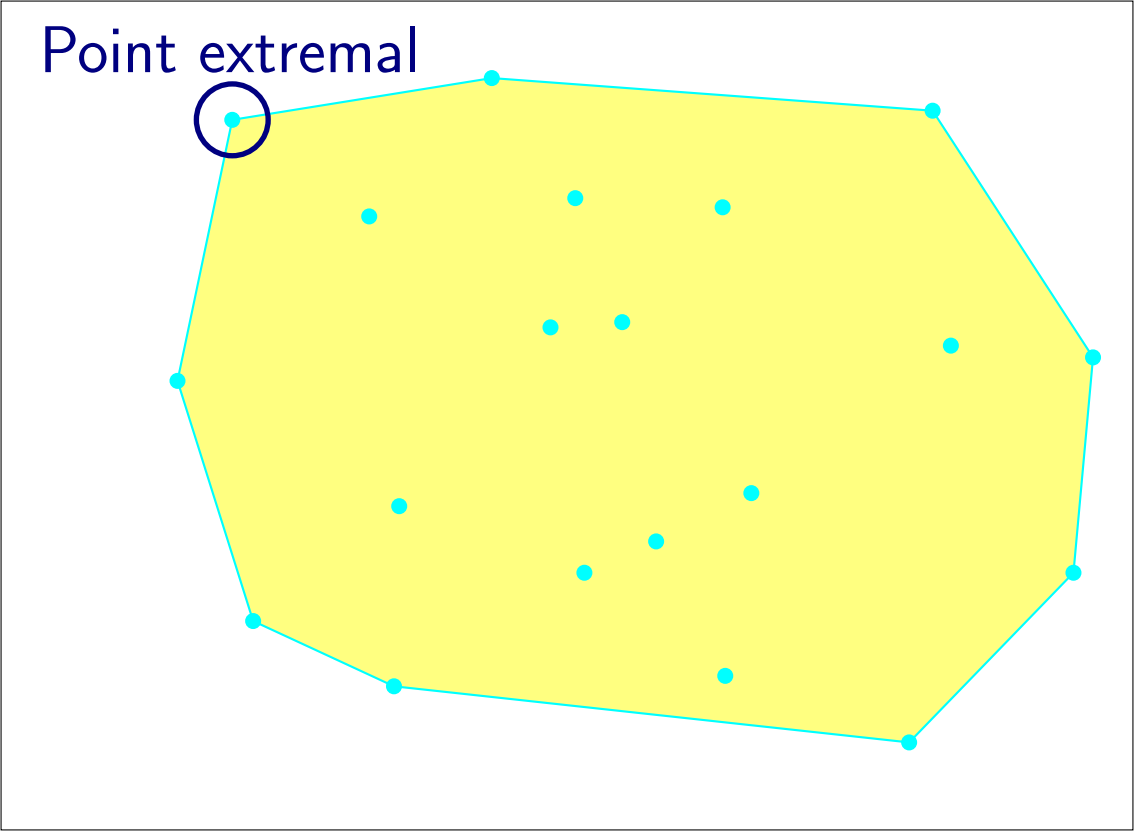
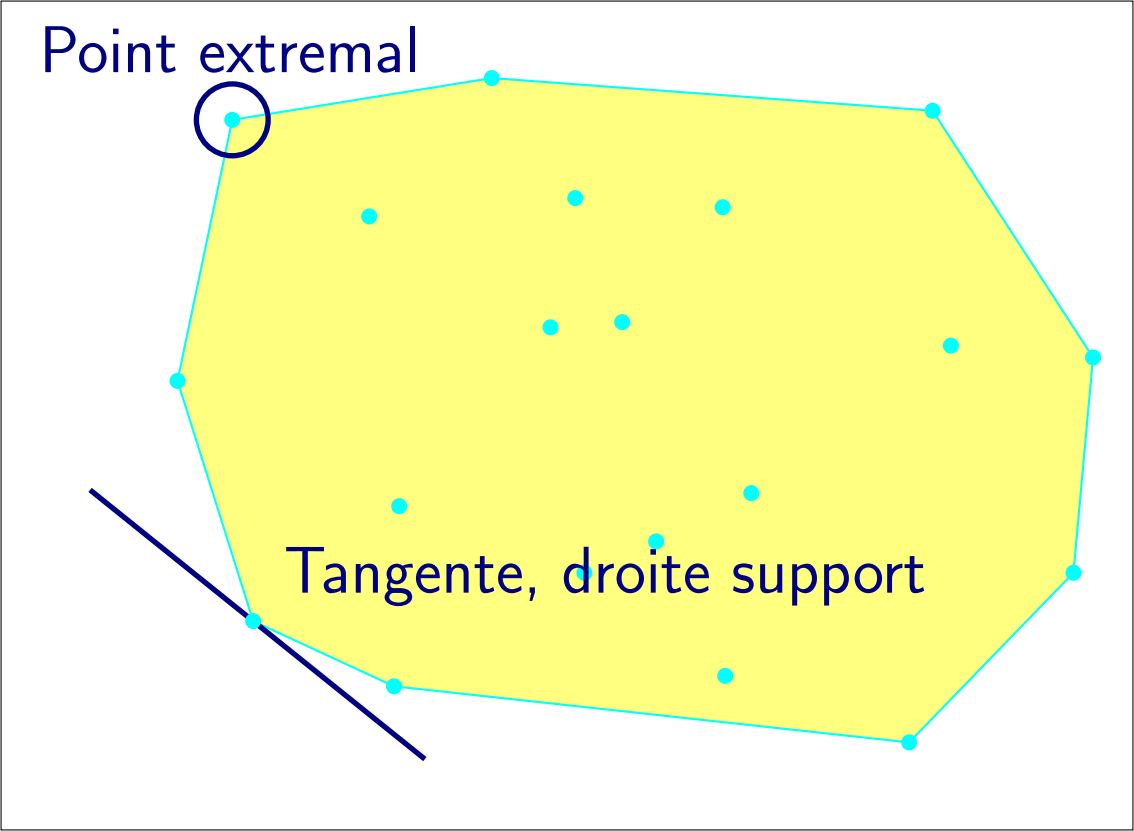

Enveloppe convexe

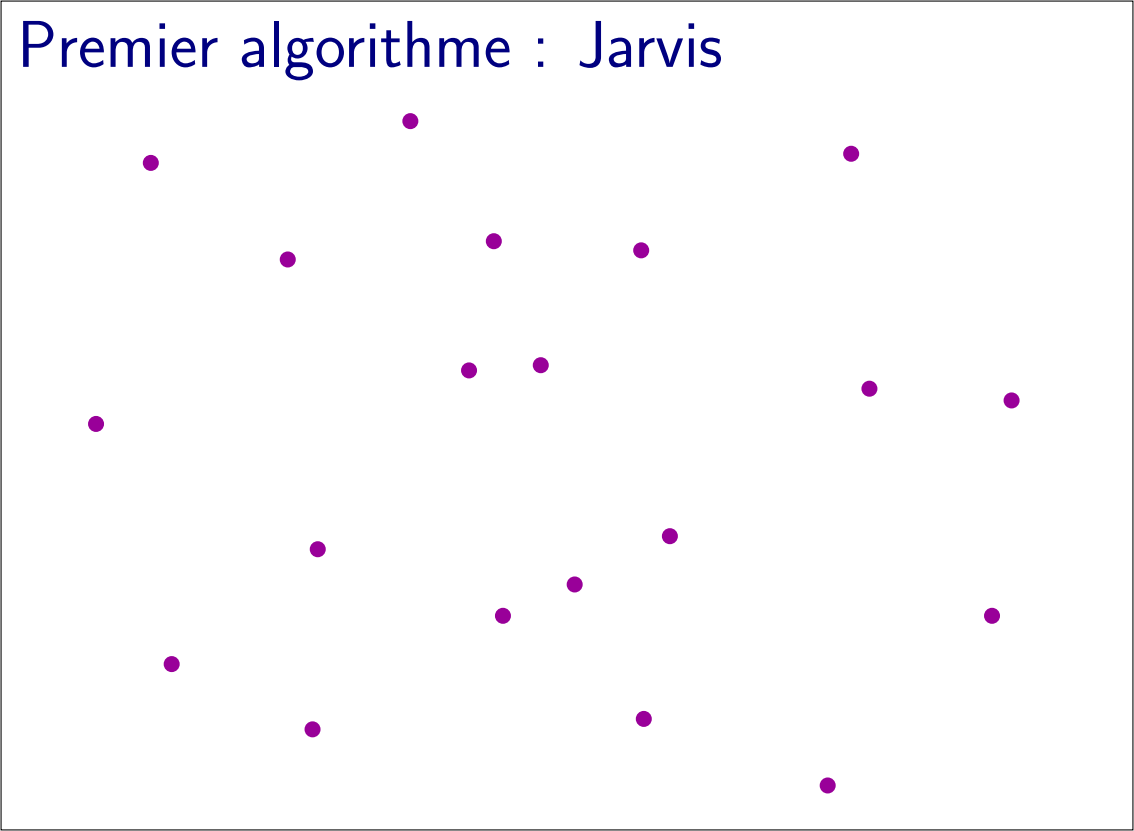

Enveloppe convexe



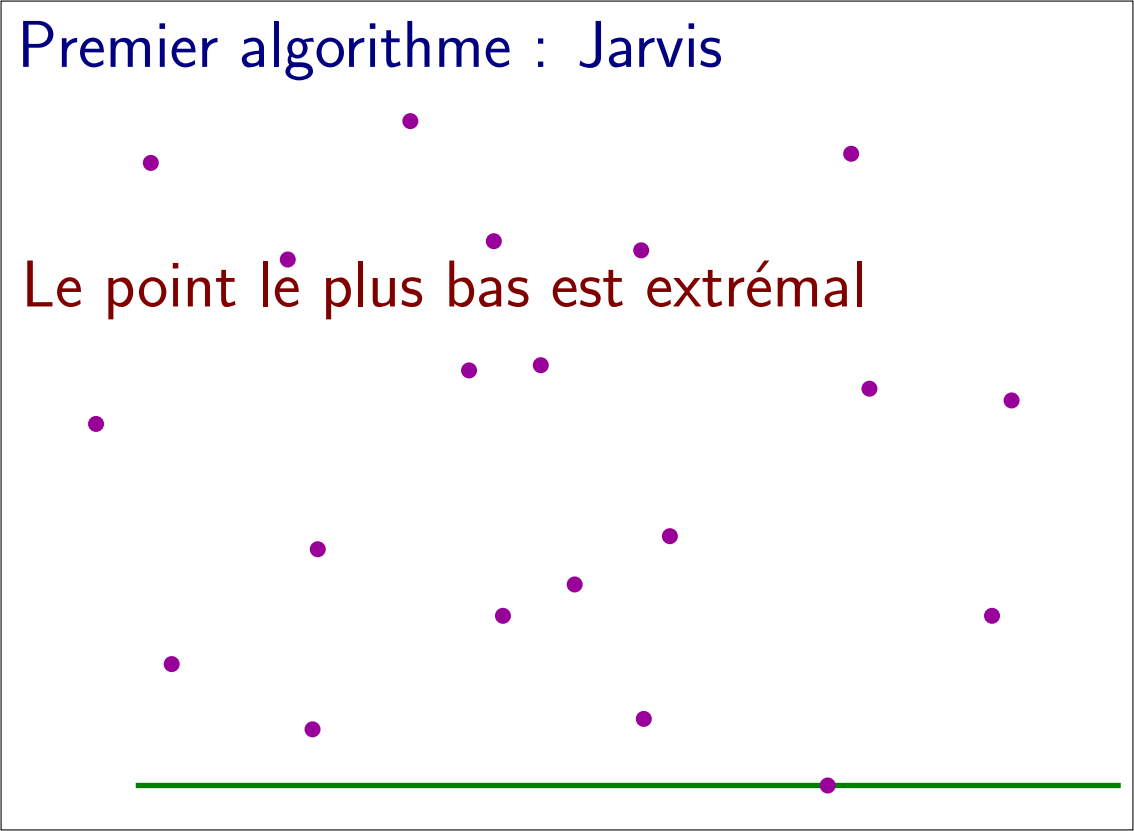

Point extremal


Point extremal
Tangente, droite support

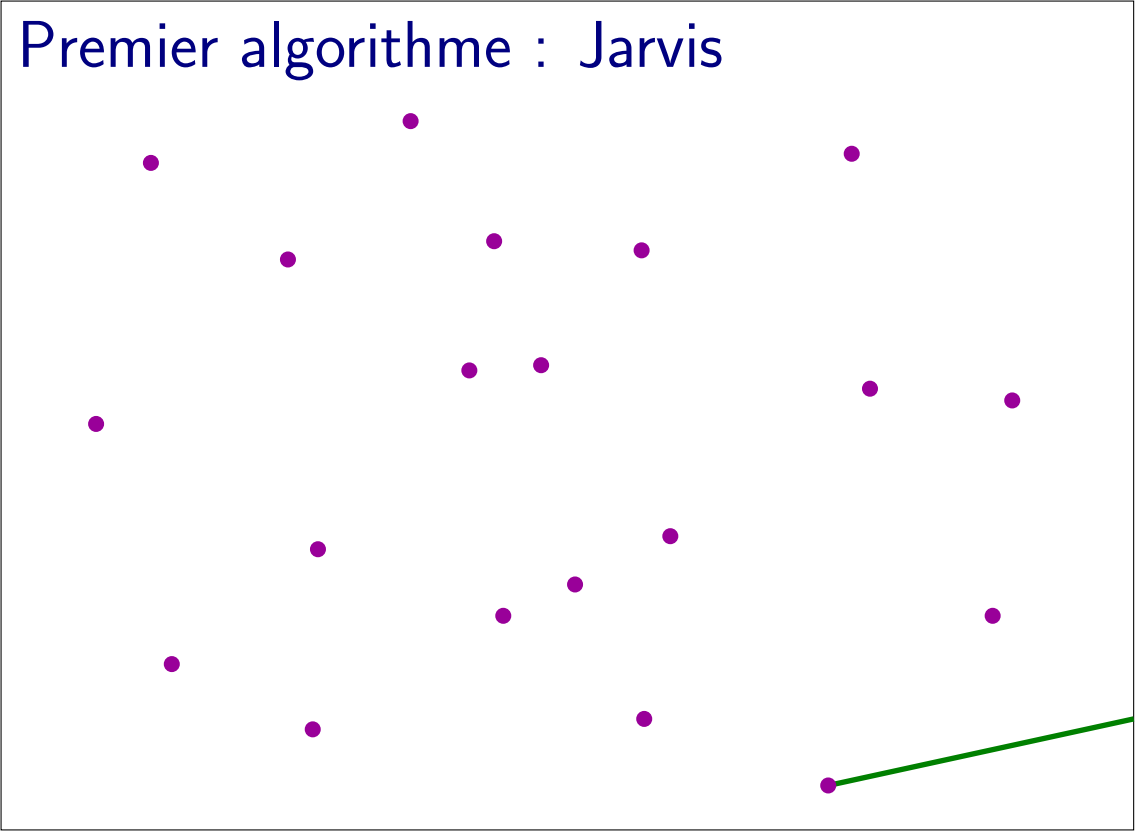
# Premier algorithme : Jarvis



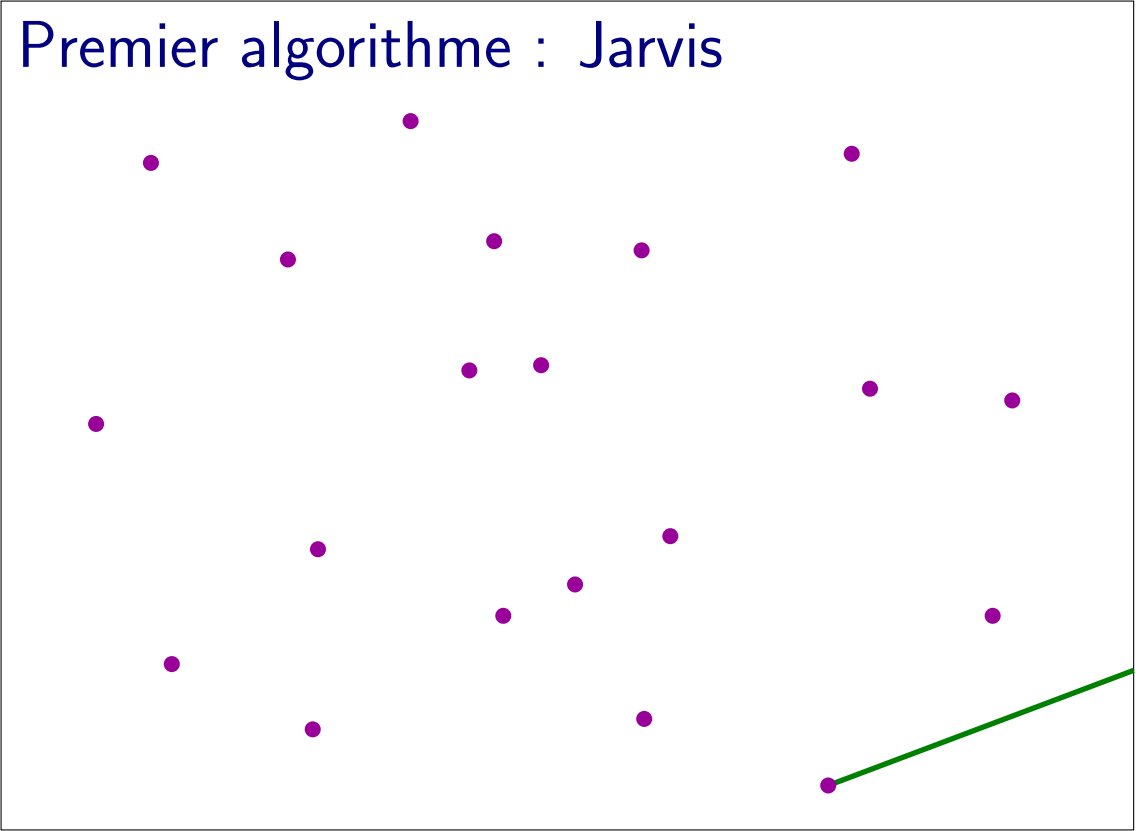
# Premier algorithme : Jarvis



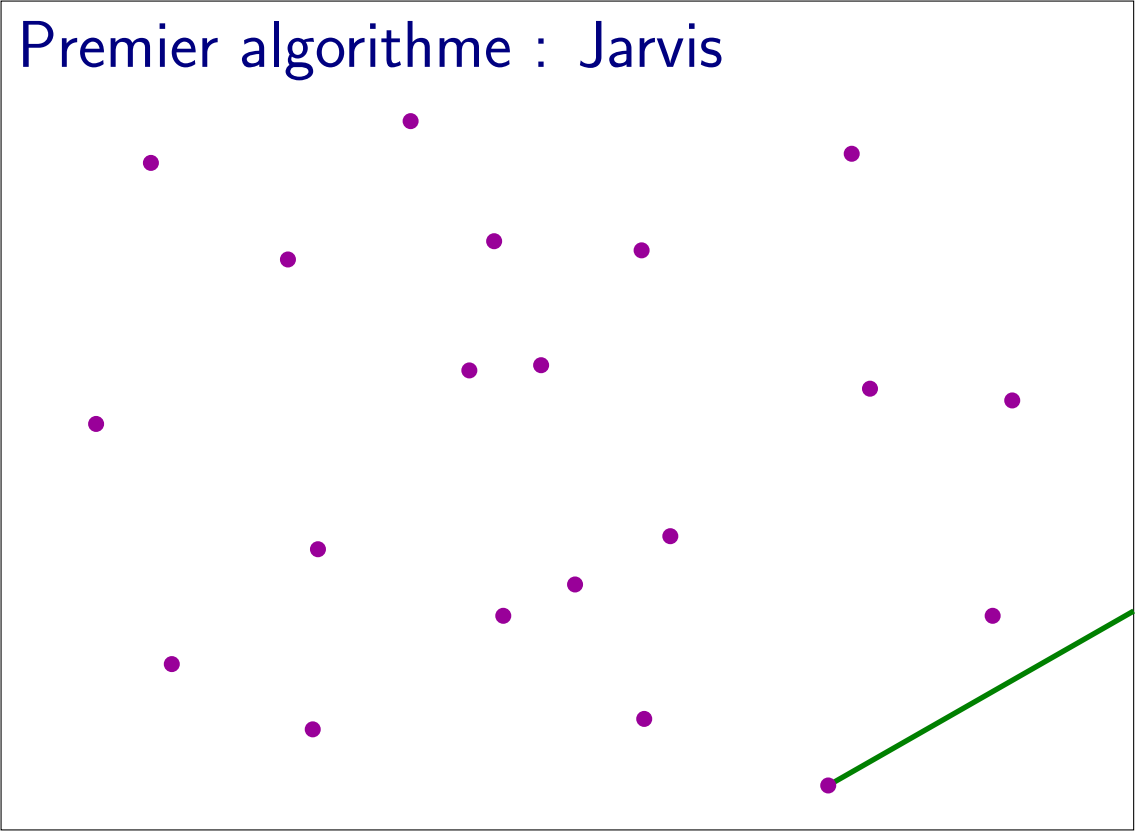
# Premier algorithme : Jarvis



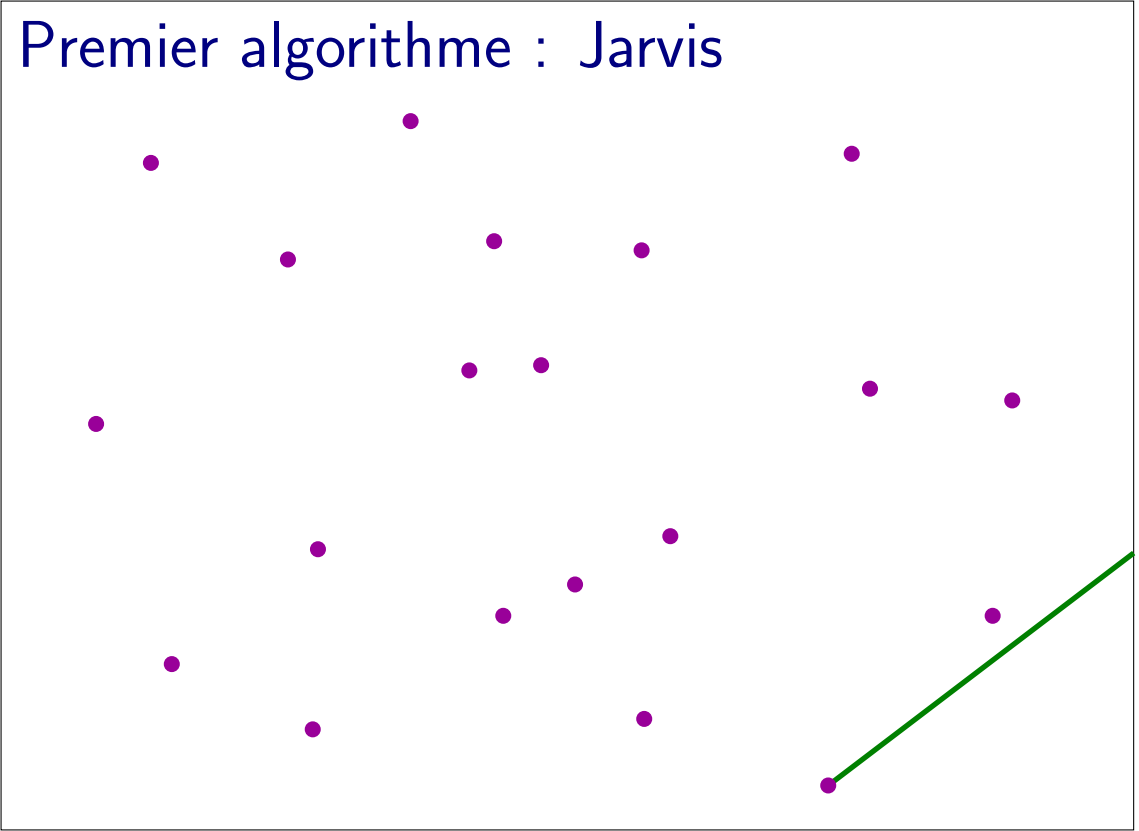
# Premier algorithme : Jarvis



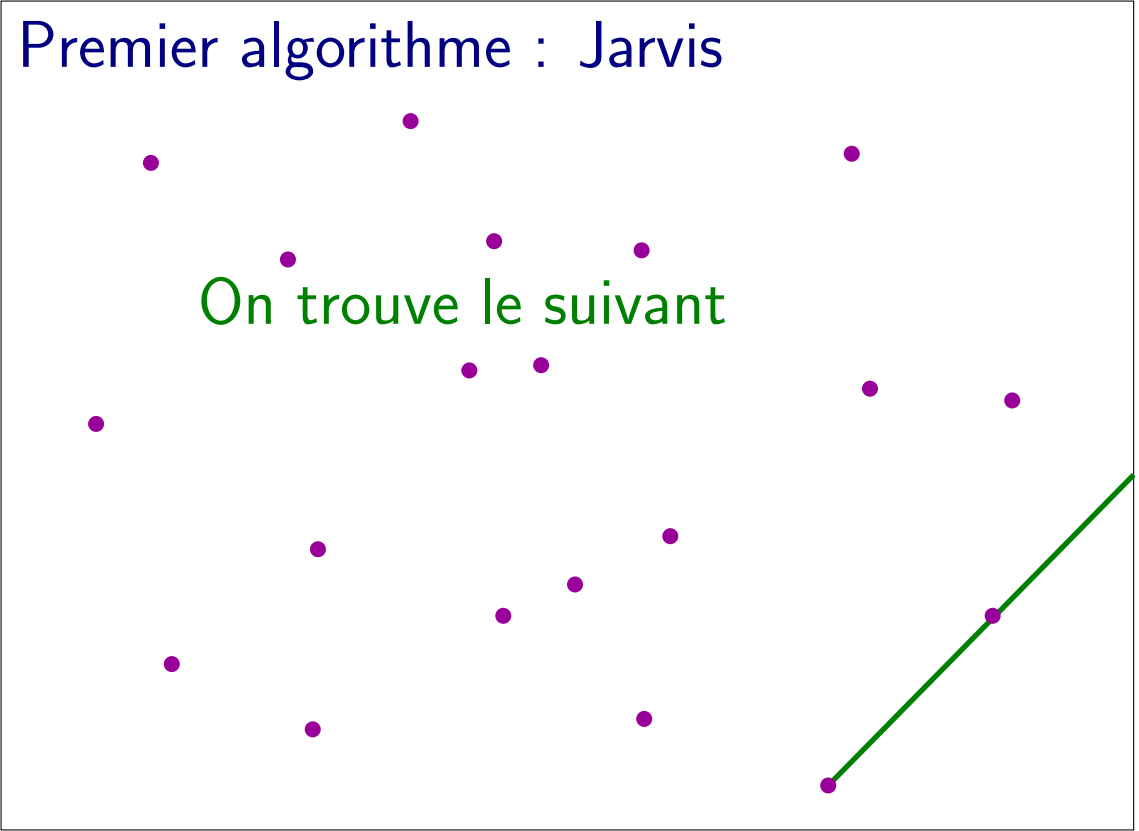
# Premier algorithme : Jarvis



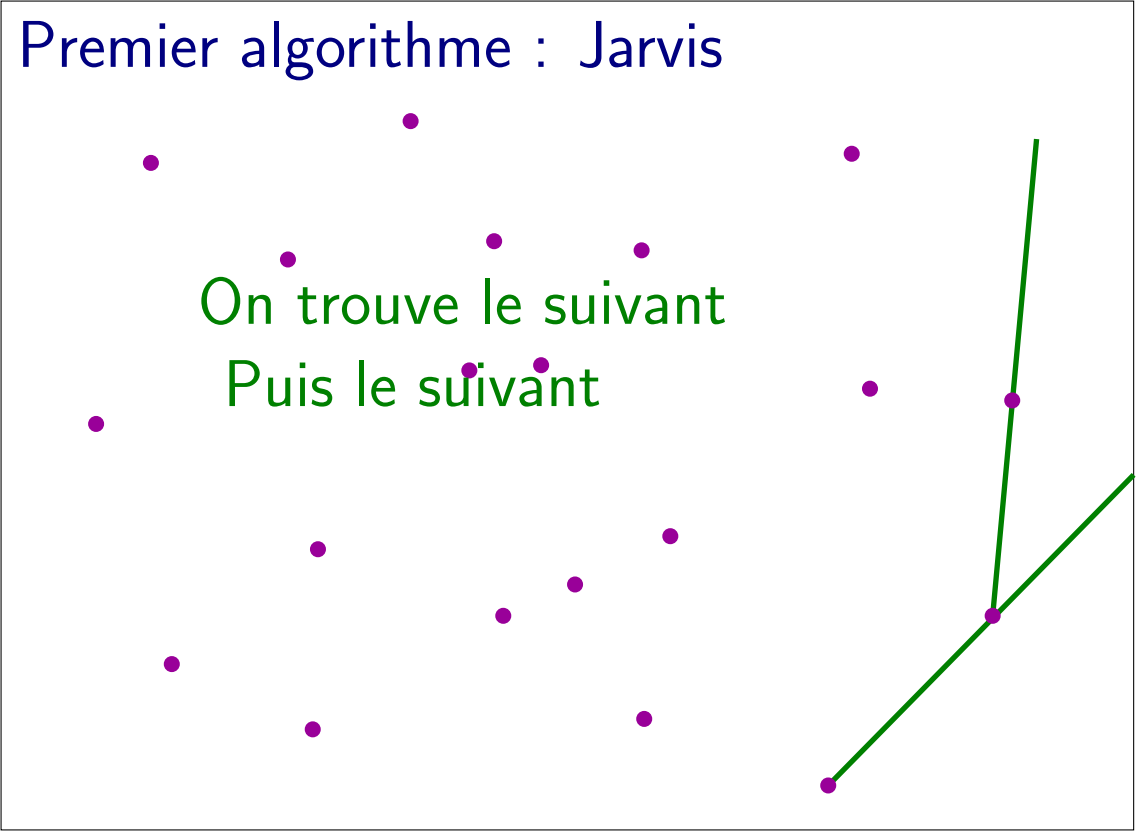
# Premier algorithme : Jarvis



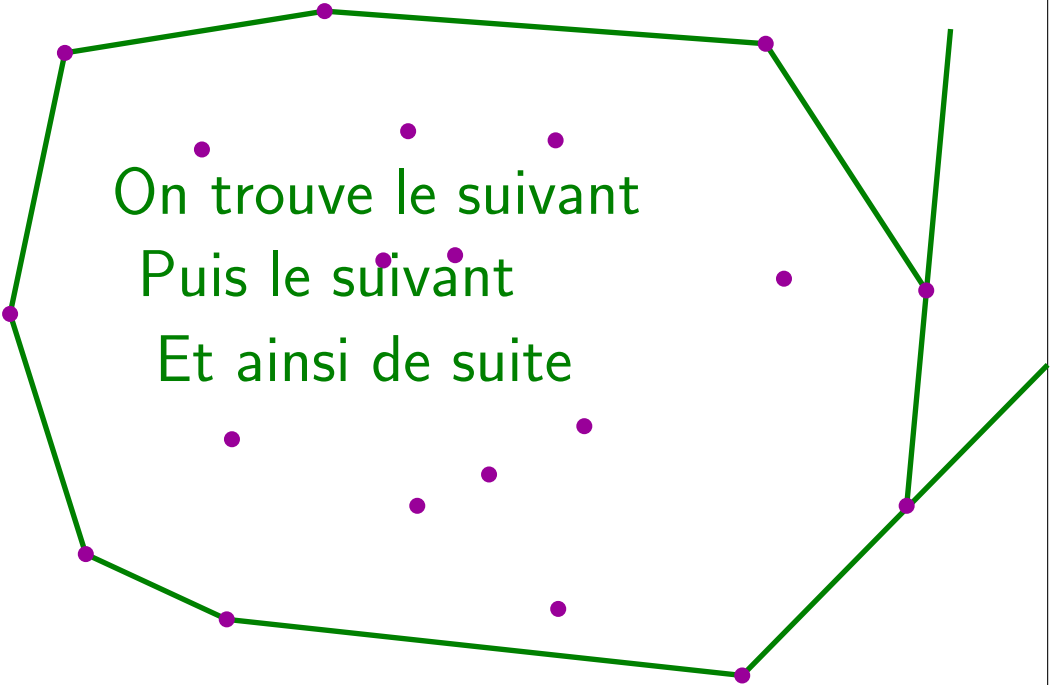
# Premier algorithme : Jarvis



# Premier algorithme : Jarvis



# Premier algorithme : Jarvis



# 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 $v \neq u$

# 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 $v \neq u$

# 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 $v \neq u$

$O(n)$

# 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 $v \neq u$

$O(n)$

# 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 $v \neq u$

# 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 $v \neq u$

$O(n)$

# 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 $v \neq u$

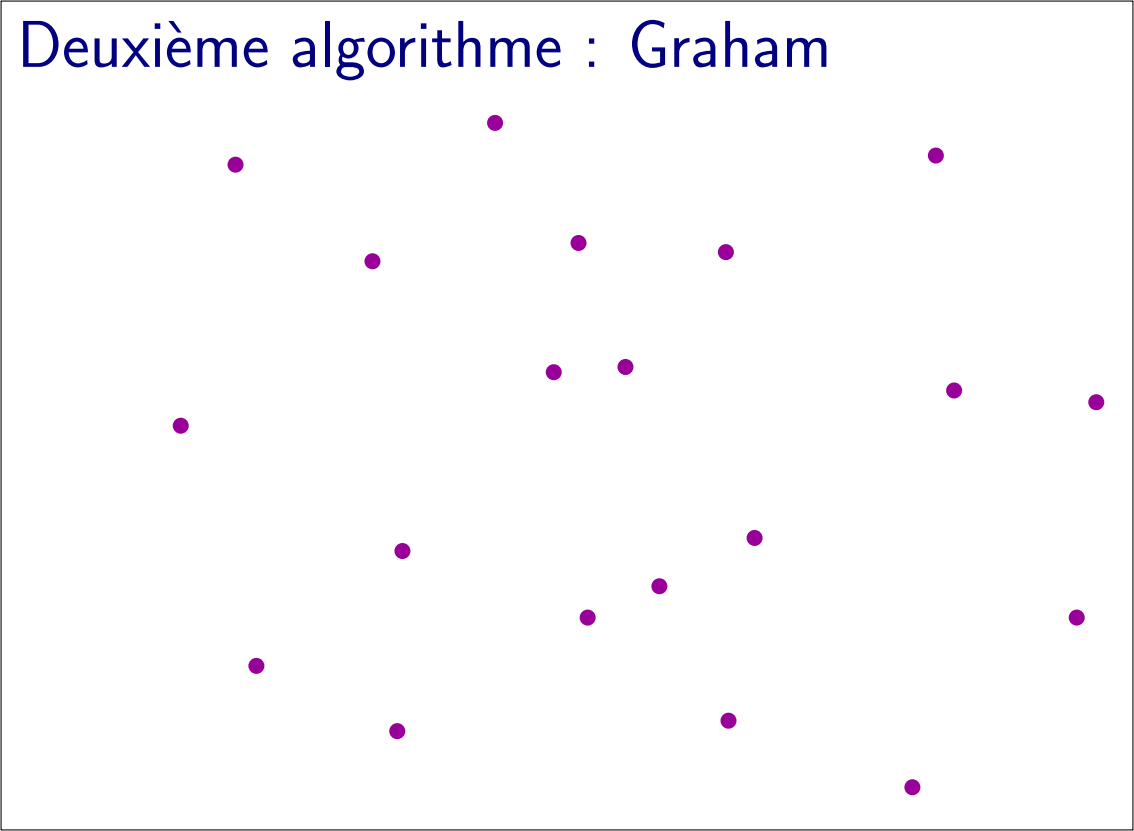
# Premier algorithme : Jarvis

## Complexité ?

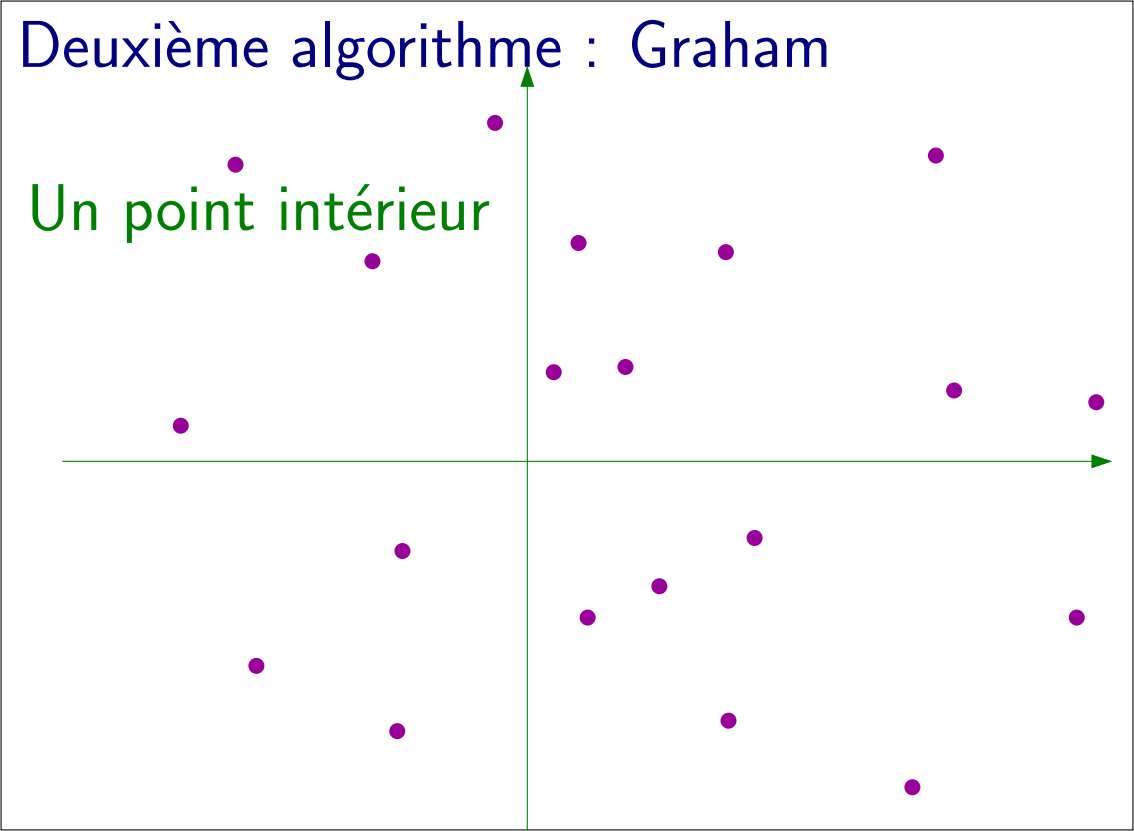
$O(nh)$

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 $v \neq u$

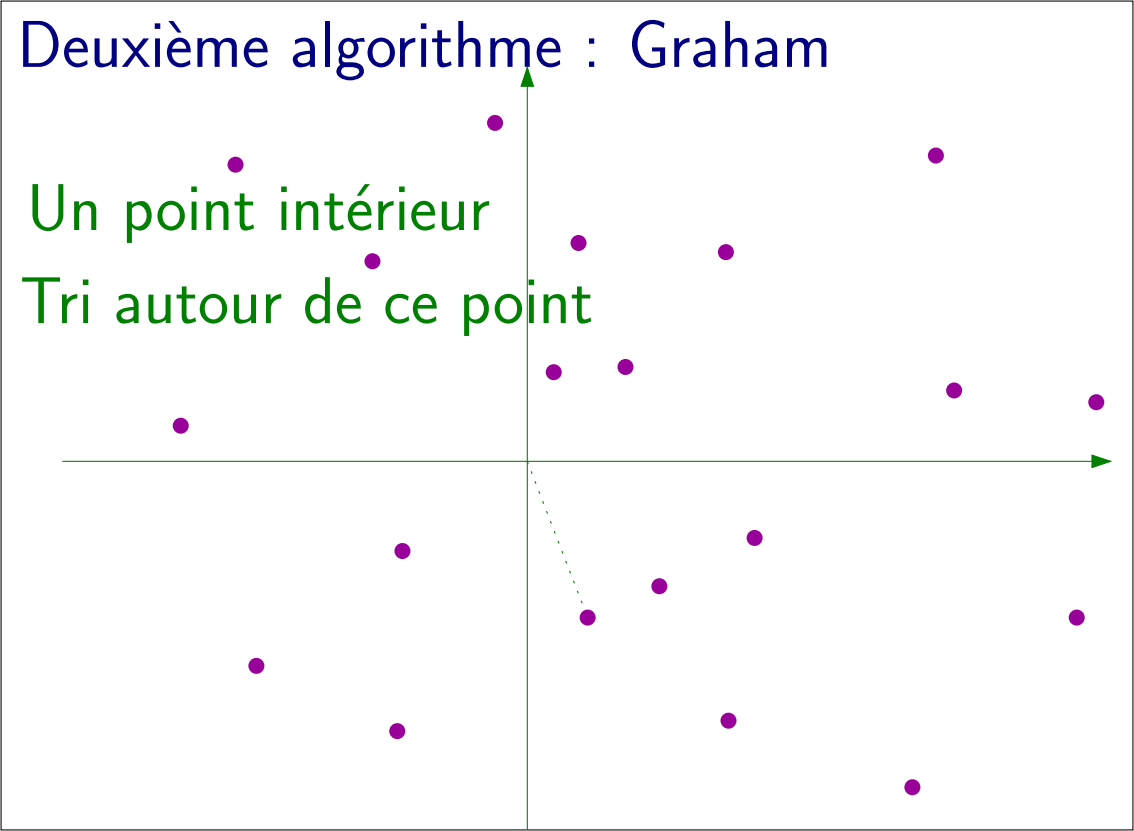
# Deuxième algorithme : Graham



# Deuxième algorithme : Graham



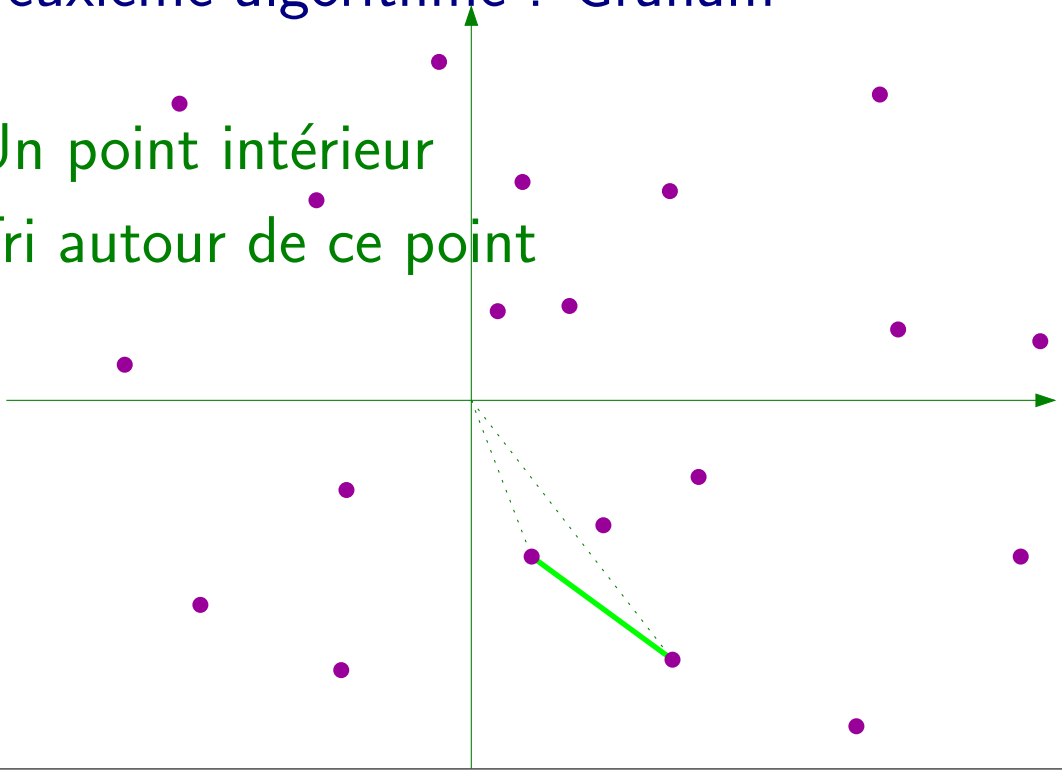
# Deuxième algorithme : Graham



# Deuxième algorithme : Graham

Un point intérieur

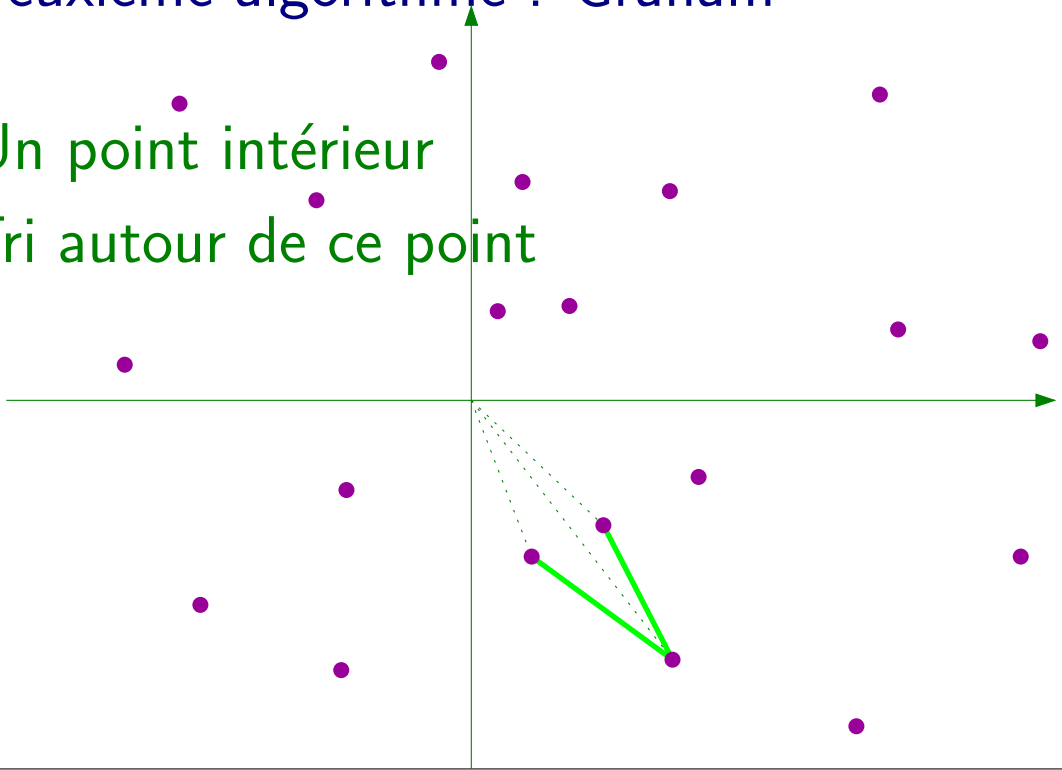
Tri autour de ce point



# Deuxième algorithme : Graham

Un point intérieur

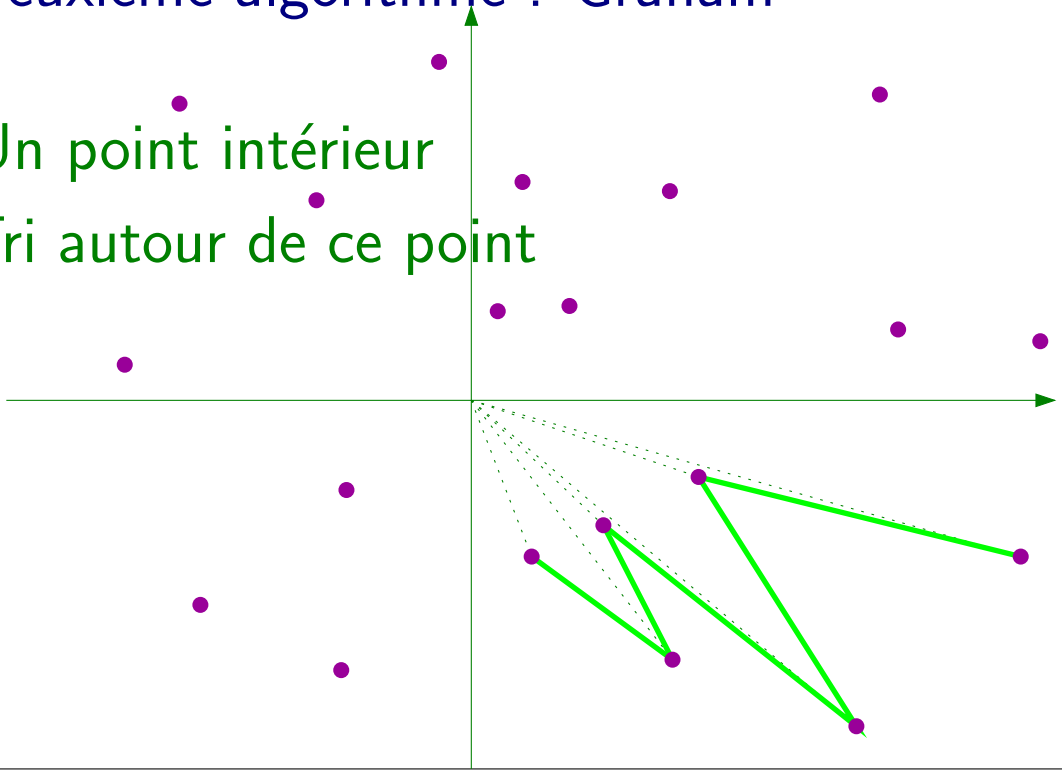
Tri autour de ce point



# Deuxième algorithme : Graham

Un point intérieur

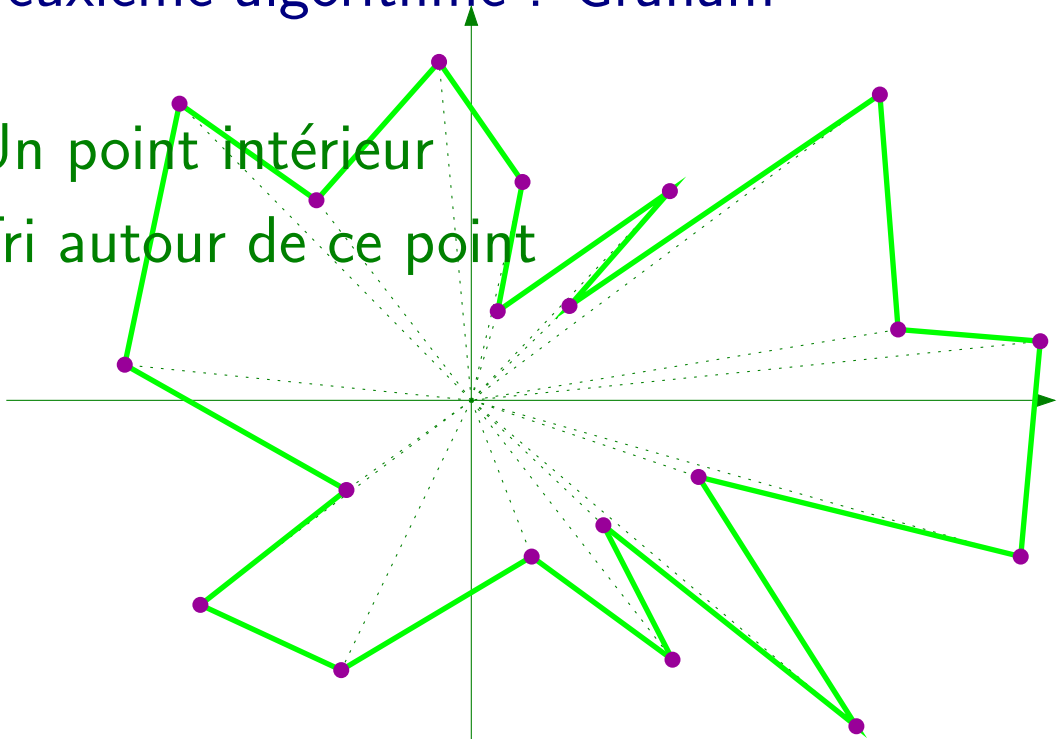
Tri autour de ce point



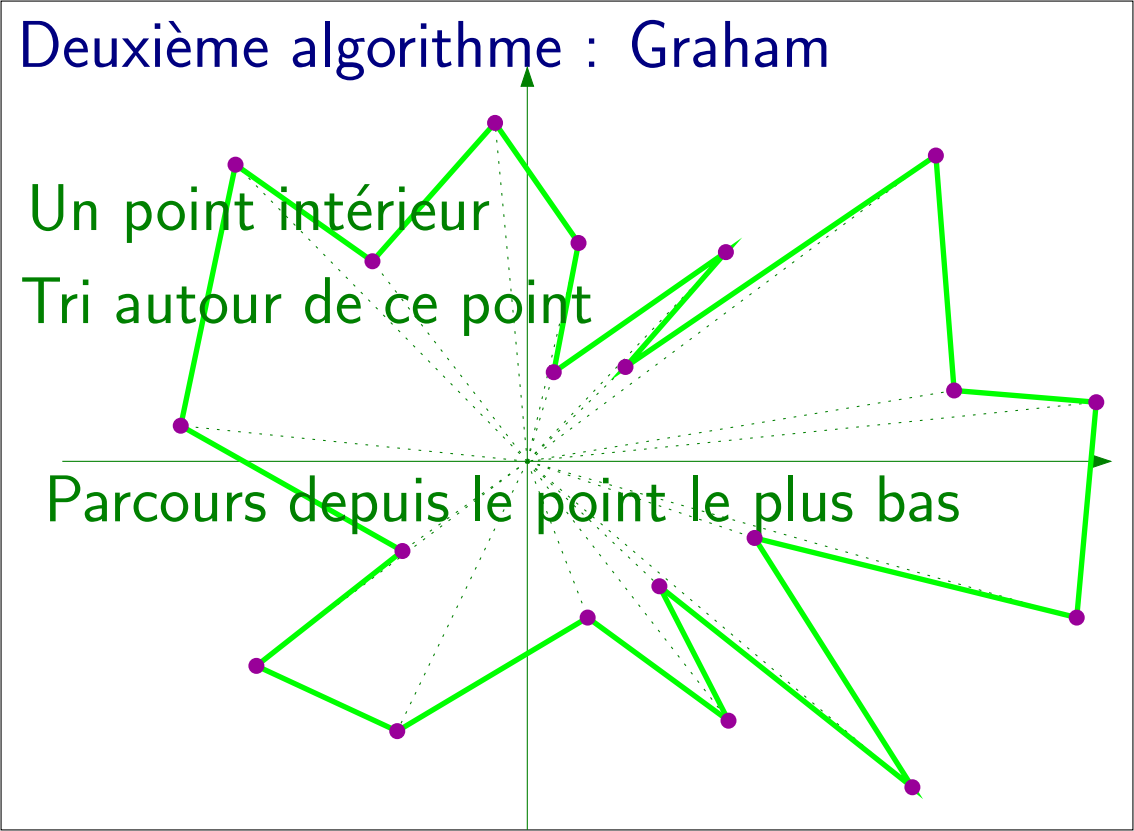
# Deuxième algorithme : Graham

Un point intérieur

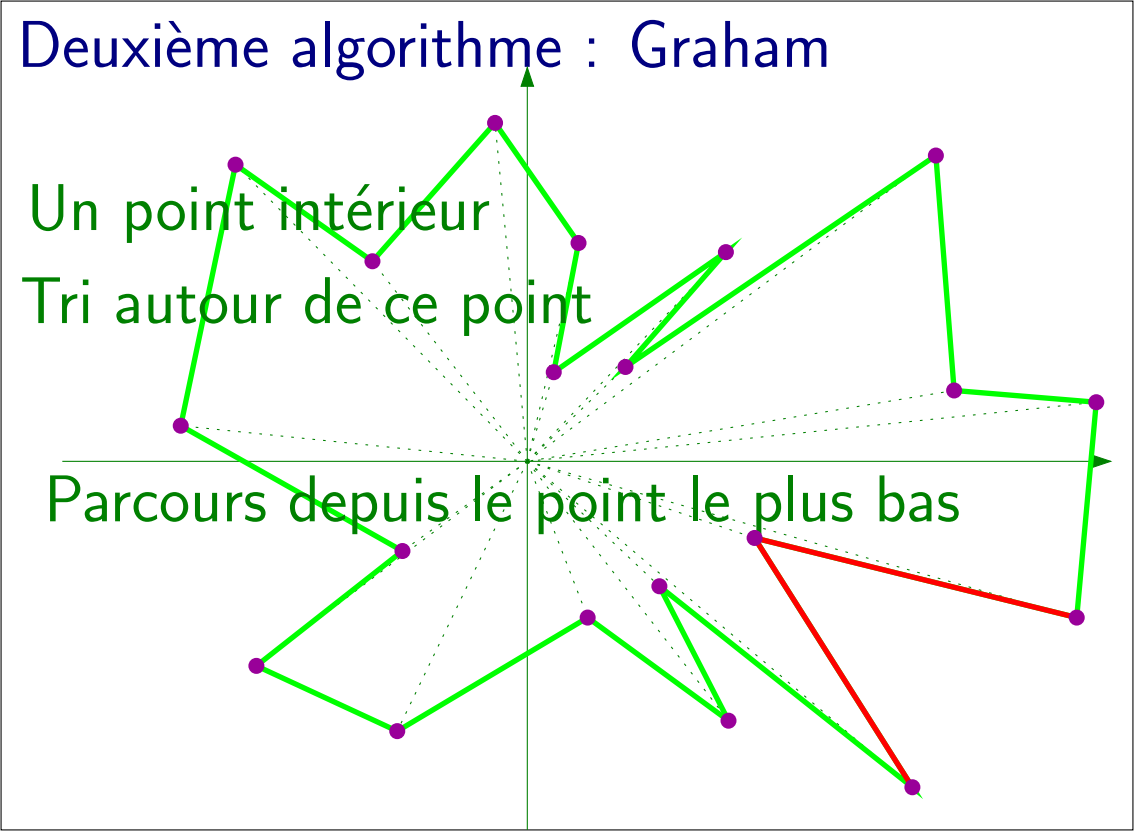
Tri autour de ce point



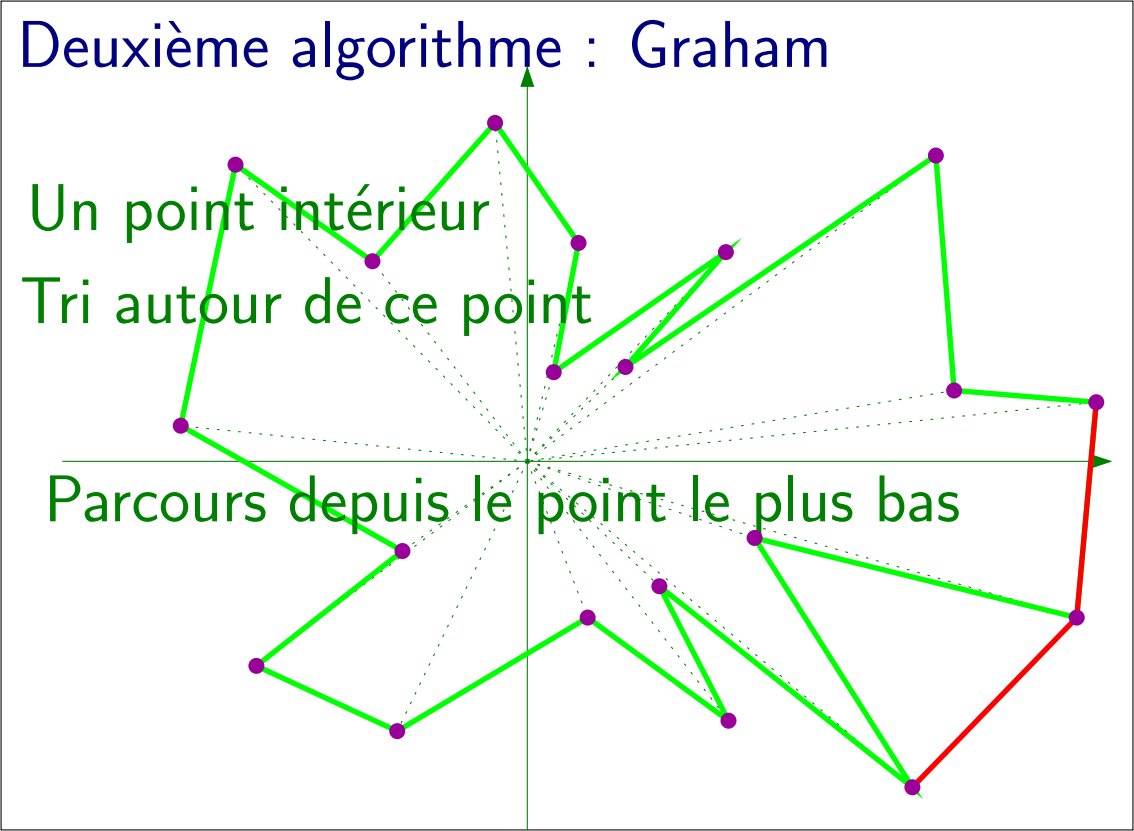
# Deuxième algorithme : Graham



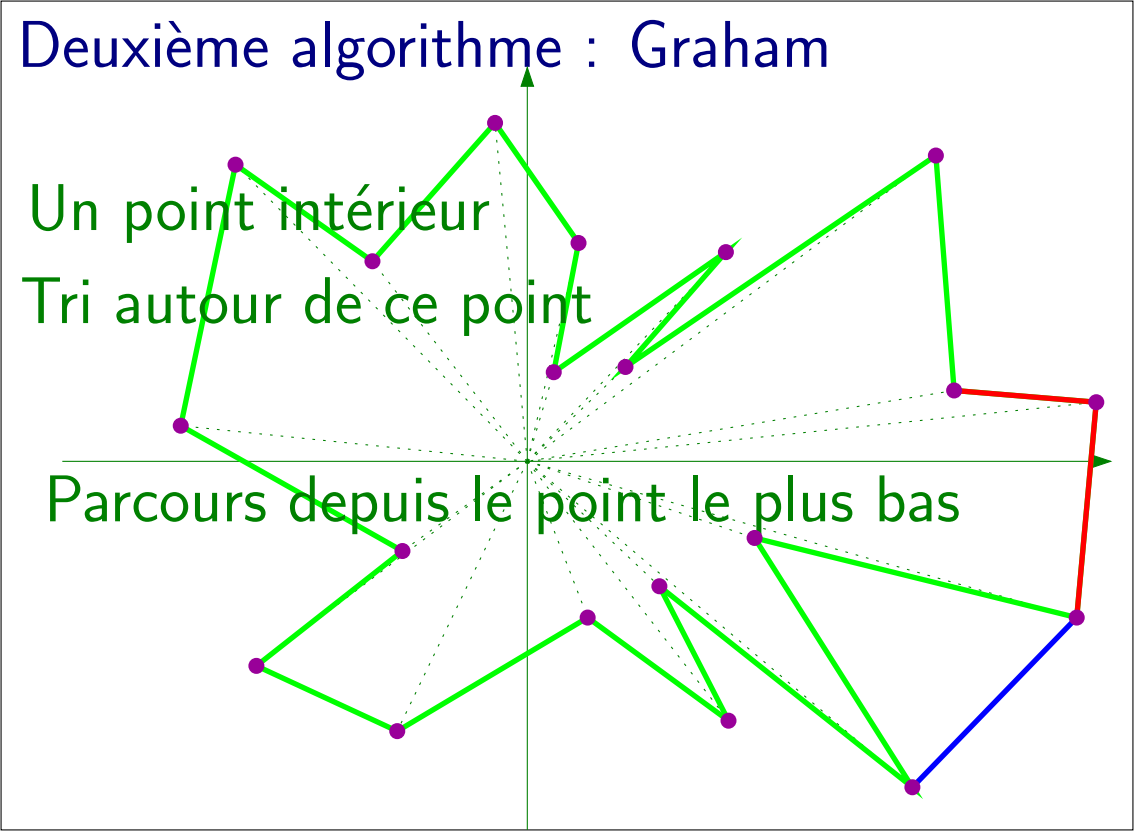
# Deuxième algorithme : Graham



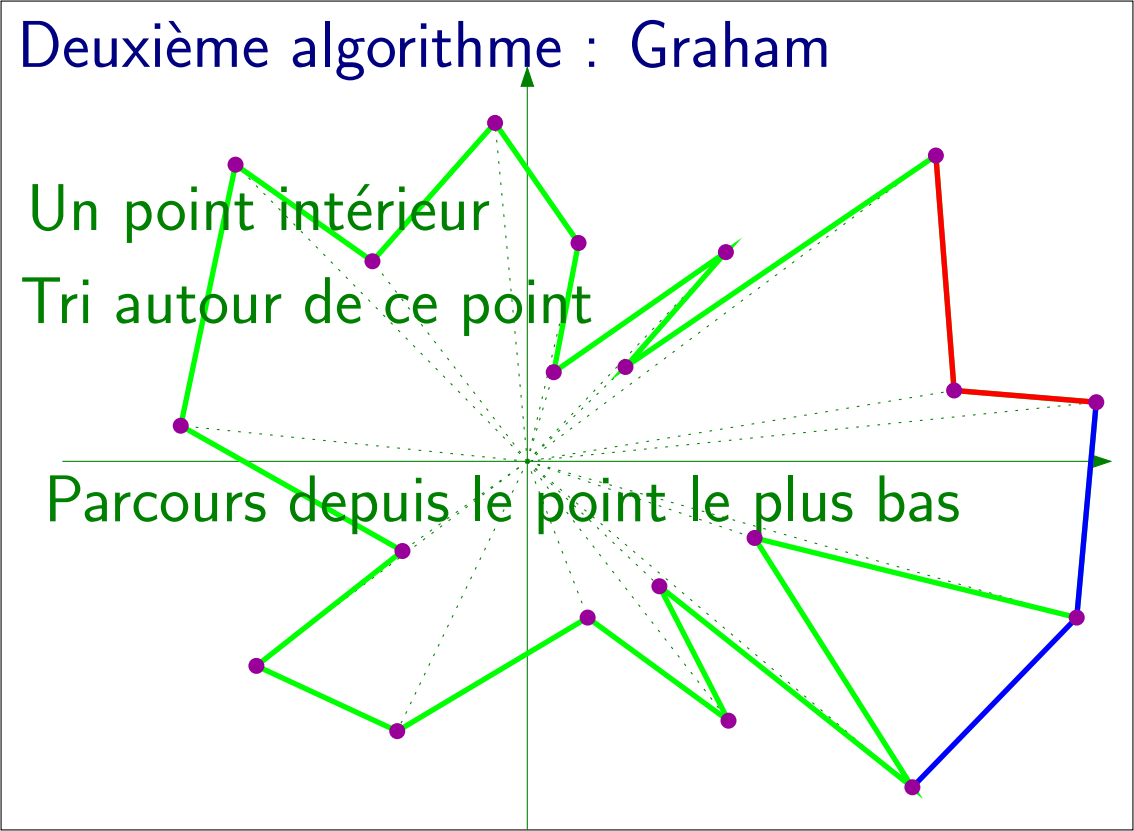
# Deuxième algorithme : Graham



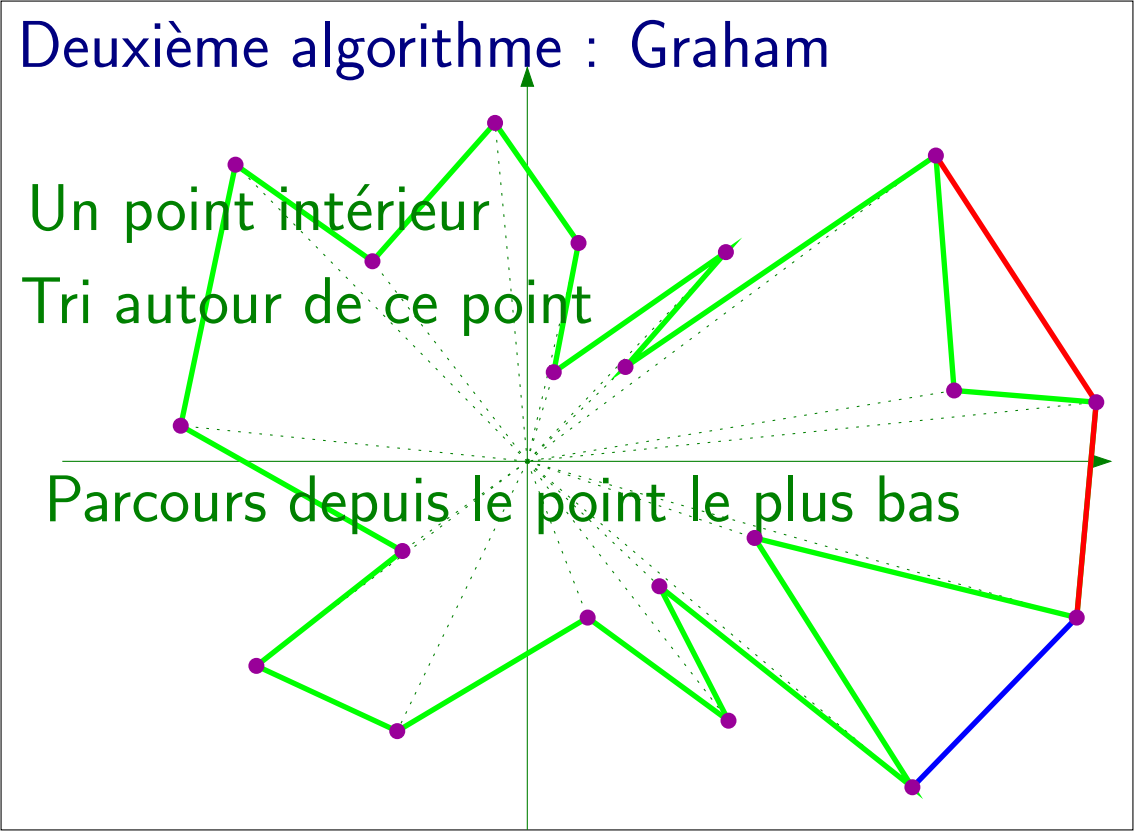
# Deuxième algorithme : Graham



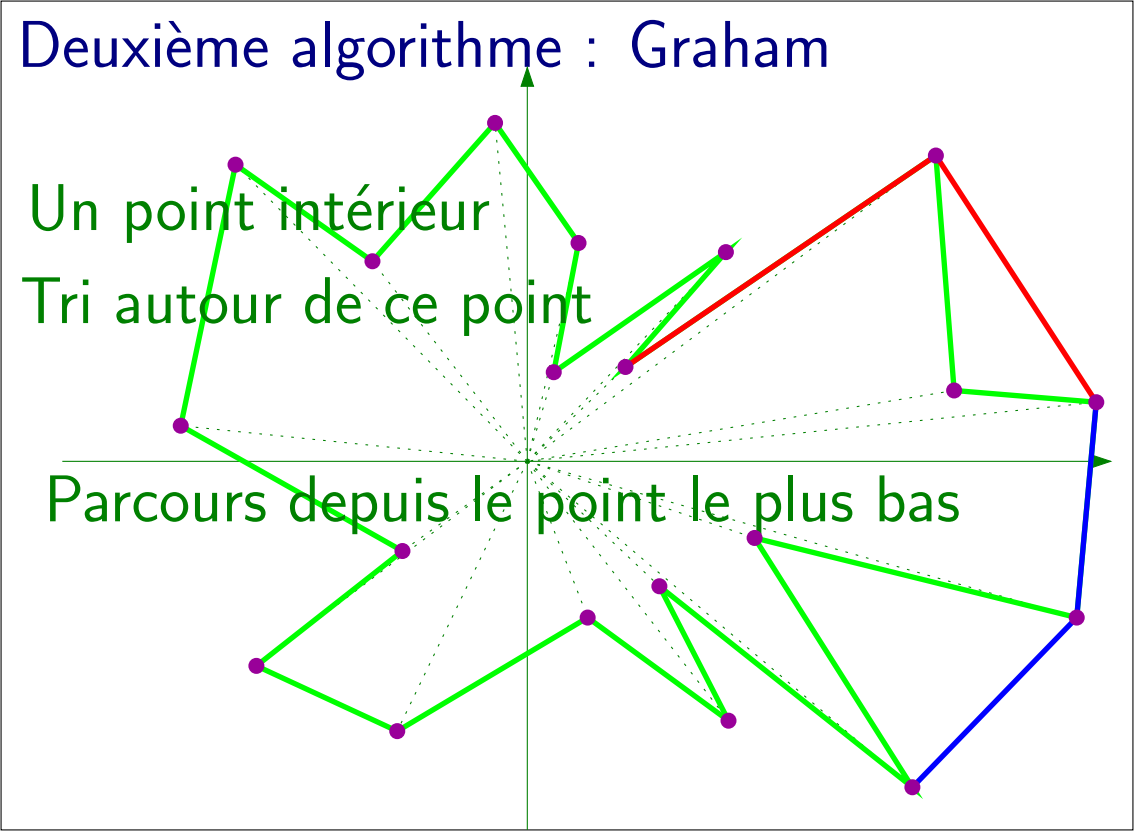
# Deuxième algorithme : Graham



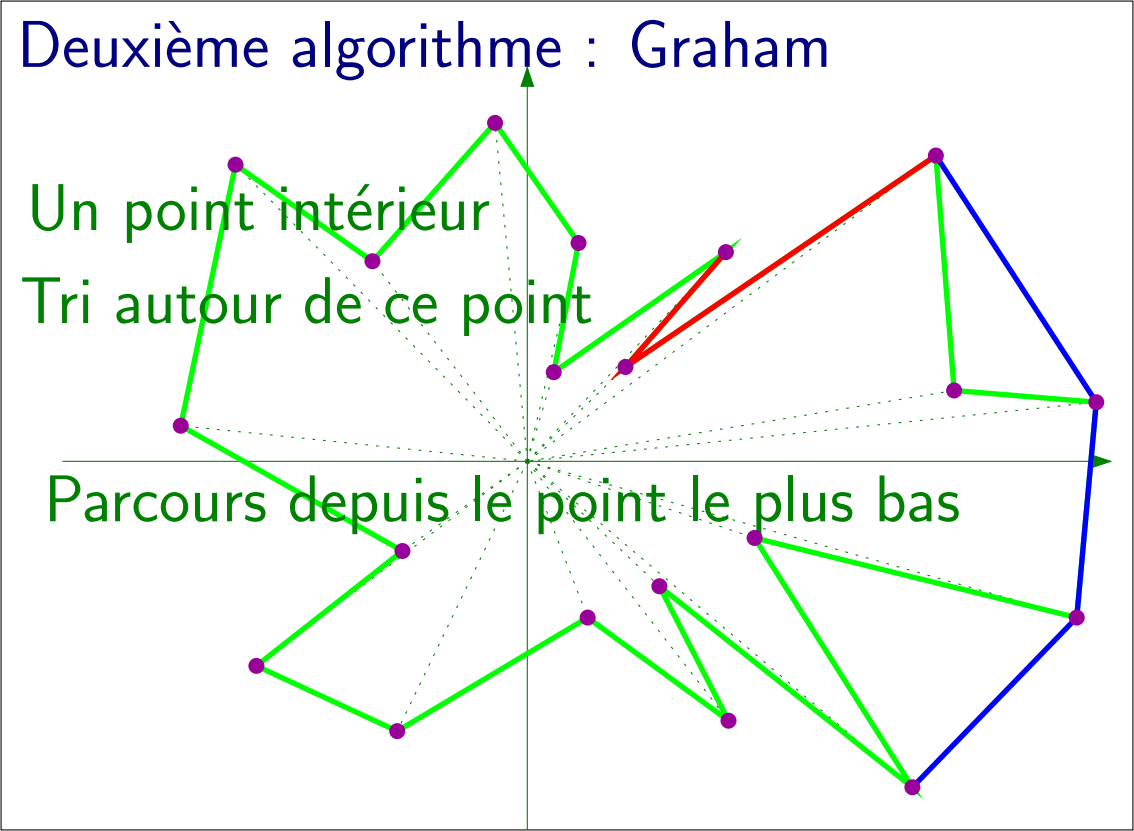
# Deuxième algorithme : Graham



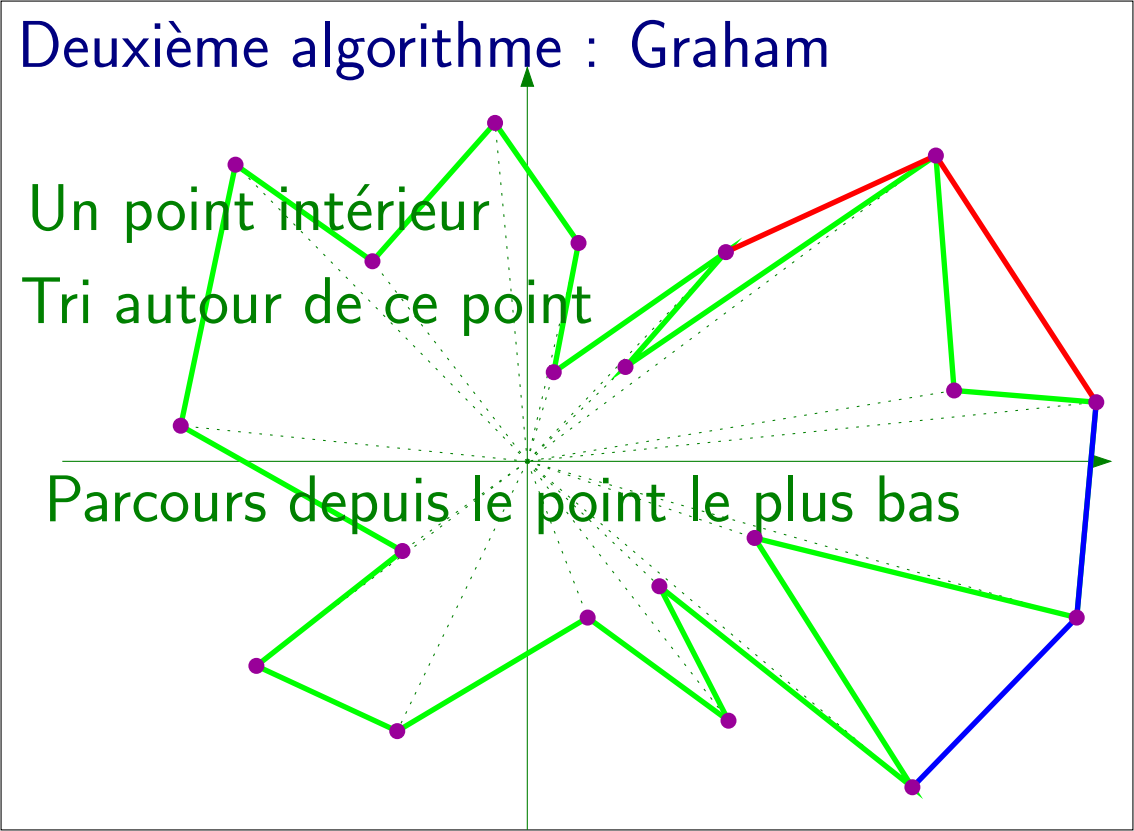
# Deuxième algorithme : Graham



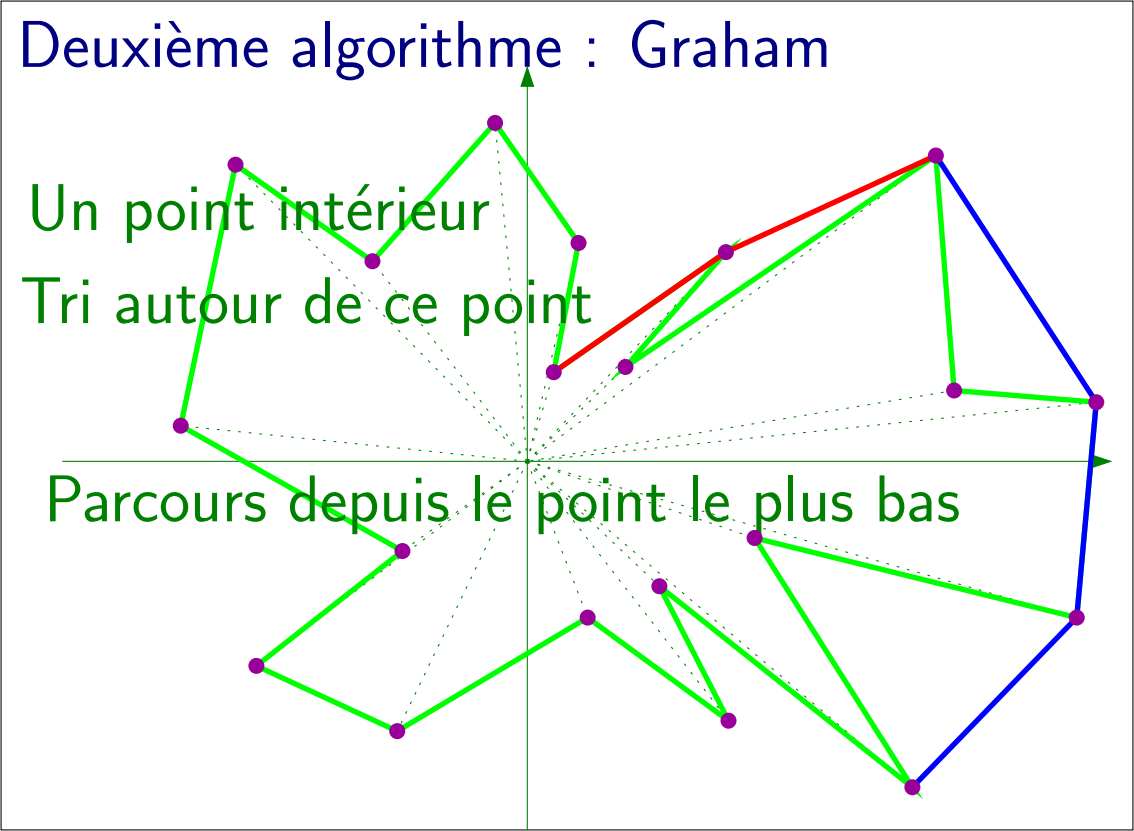
# Deuxième algorithme : Graham



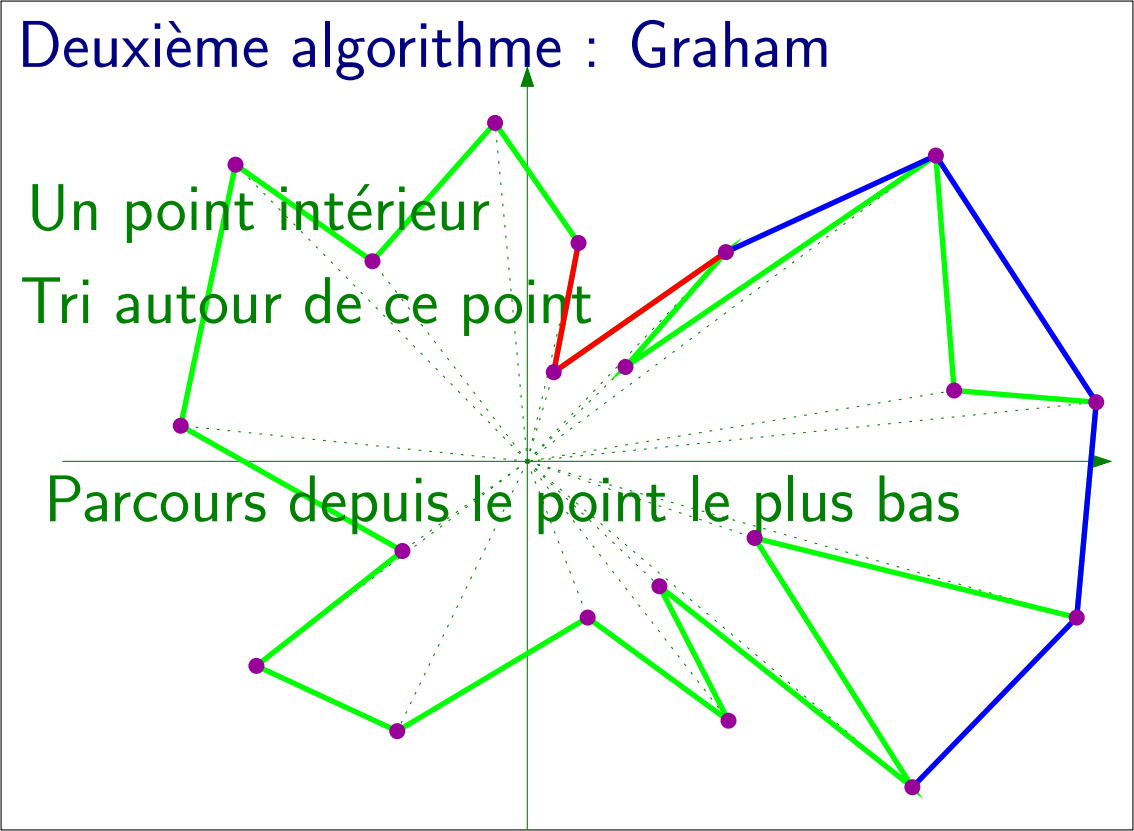
# Deuxième algorithme : Graham



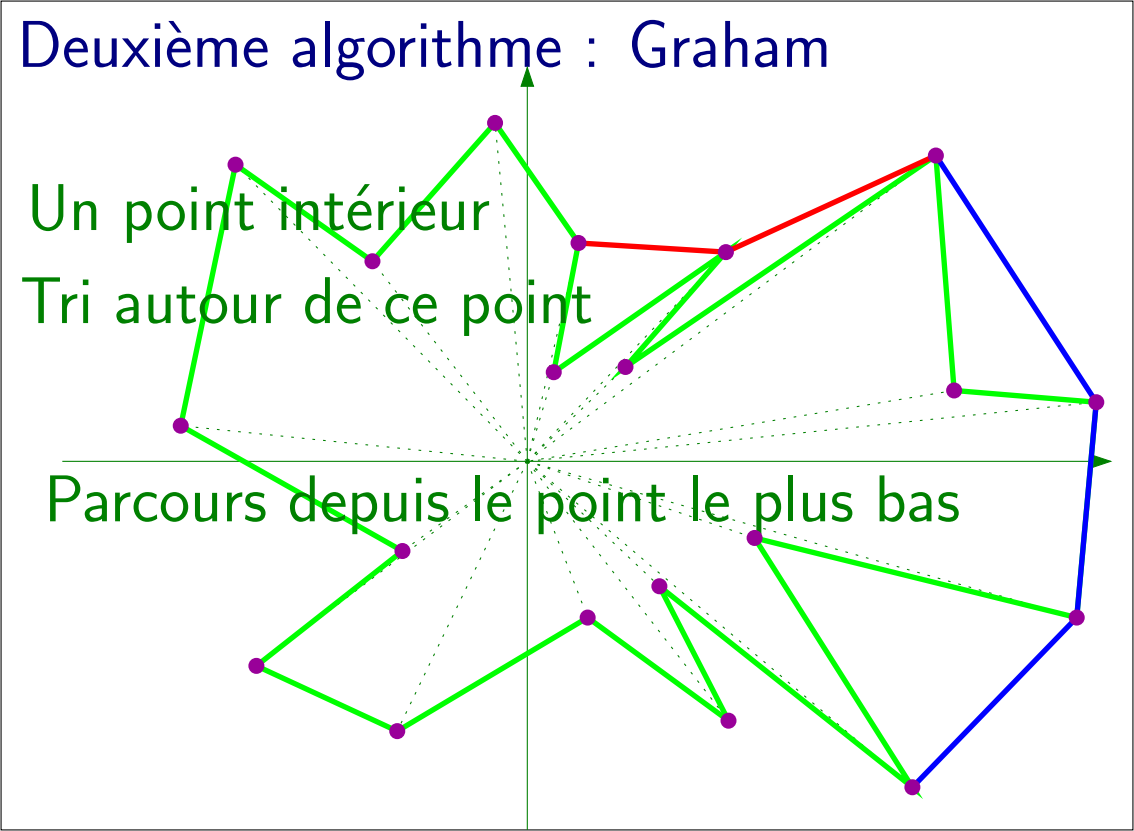
# Deuxième algorithme : Graham



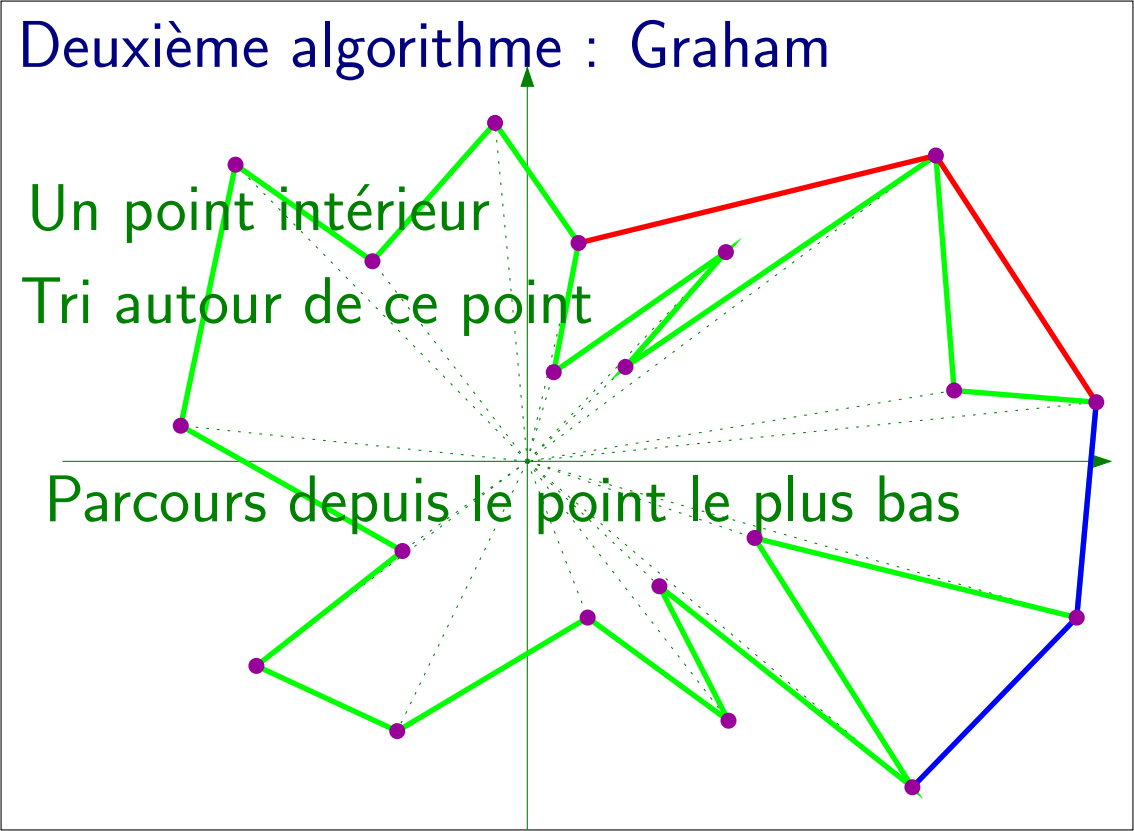
# Deuxième algorithme : Graham



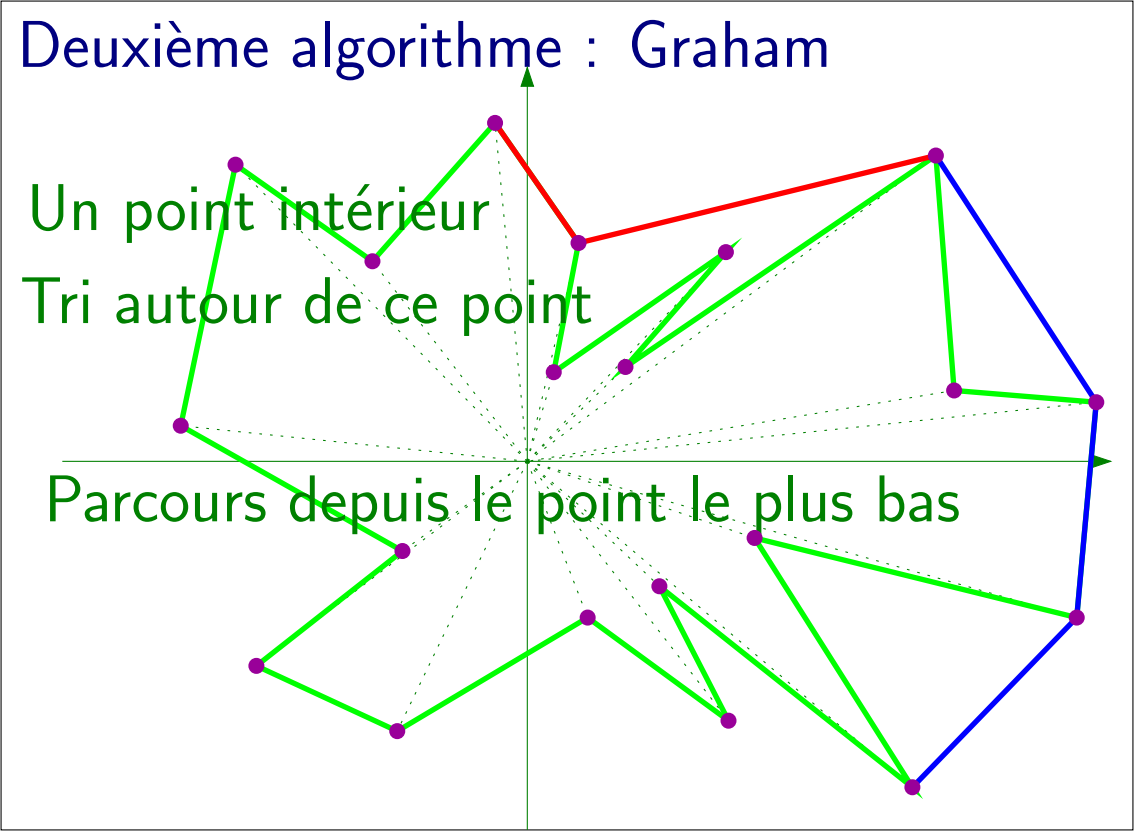
# Deuxième algorithme : Graham



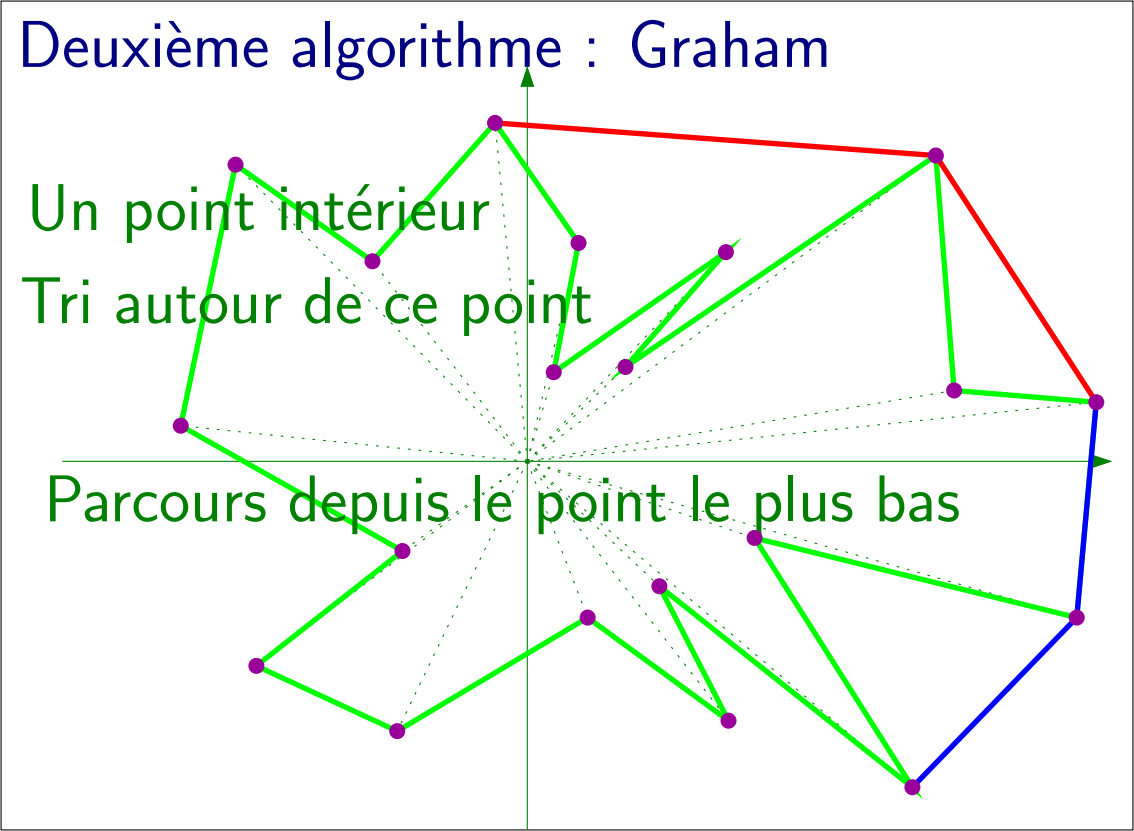
# Deuxième algorithme : Graham



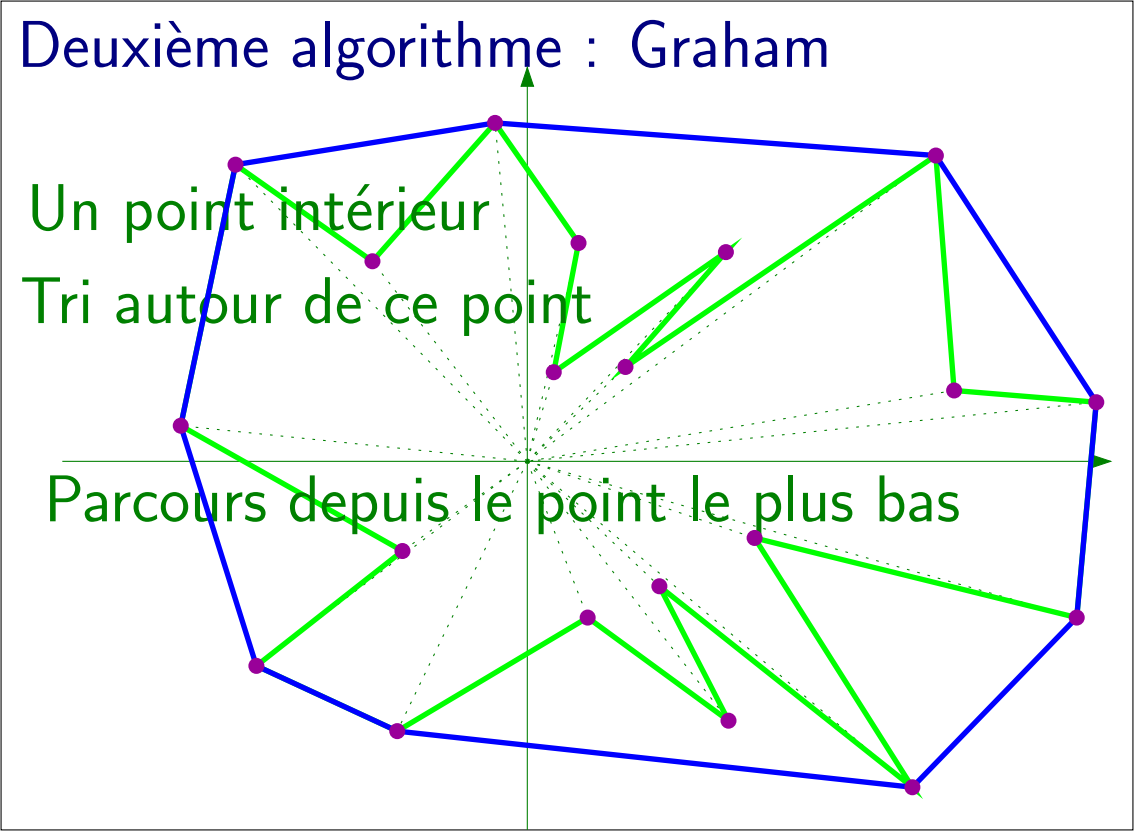
# Deuxième algorithme : Graham



# Deuxième algorithme : Graham



# Deuxième algorithme : Graham



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

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

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

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

$O(n \log n)$

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

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

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

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

# 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

# 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

# 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

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

$O(n)$

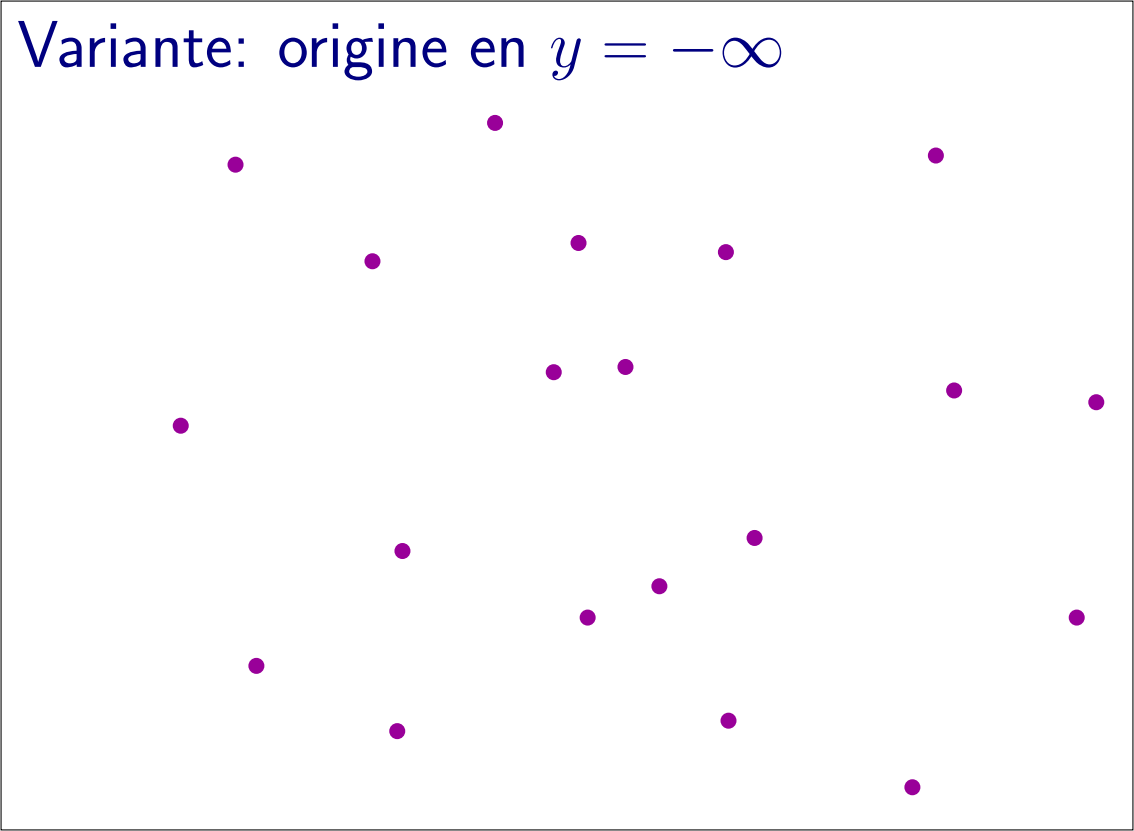
# Deuxième algorithme : Graham

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

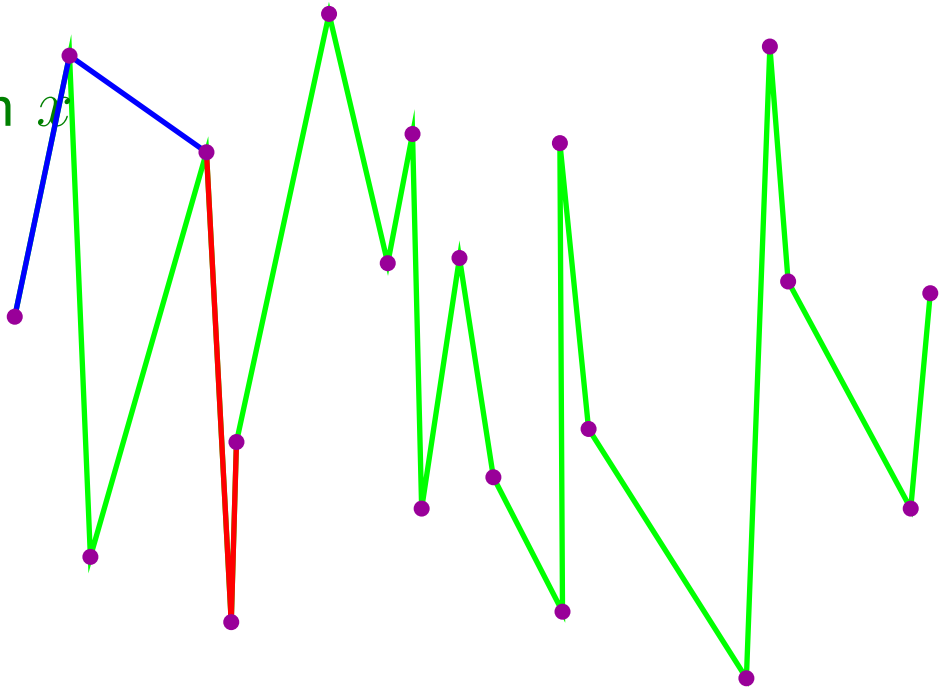
$O(n \log n)$

# Variante: origine en $y = -\infty$



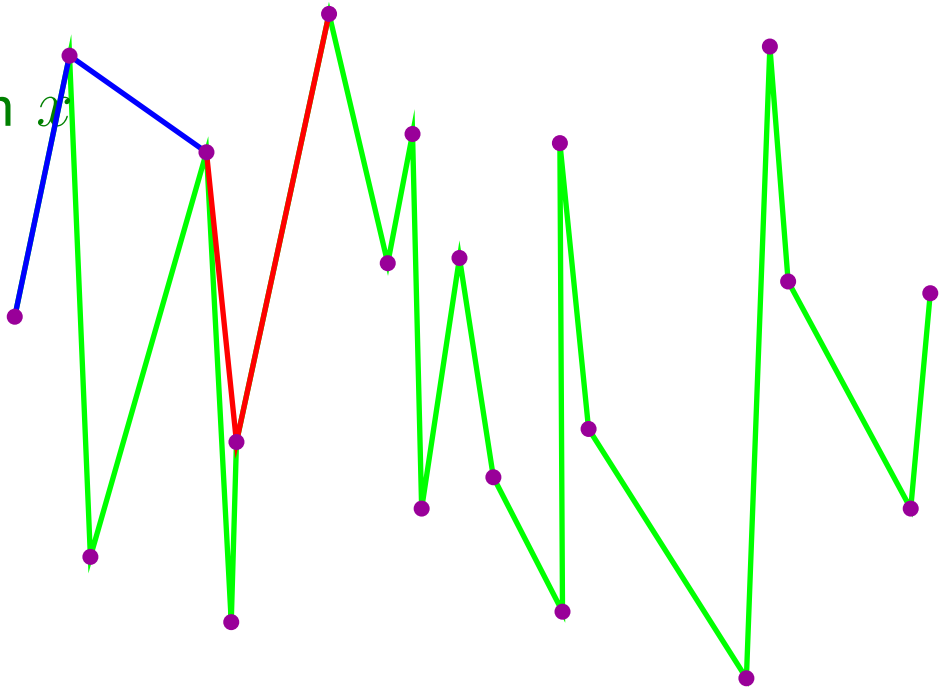
# Variante: origine en $y = -\infty$

Tri en $x$

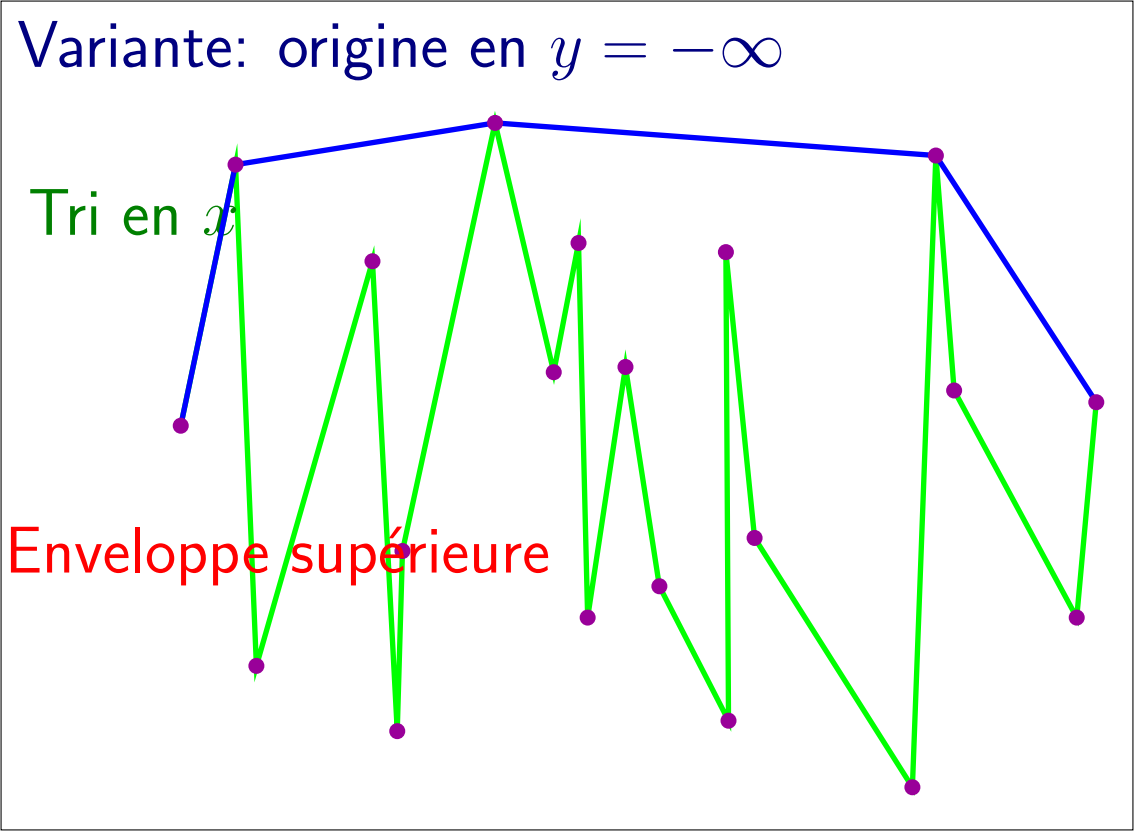


# Variante: origine en $y = -\infty$

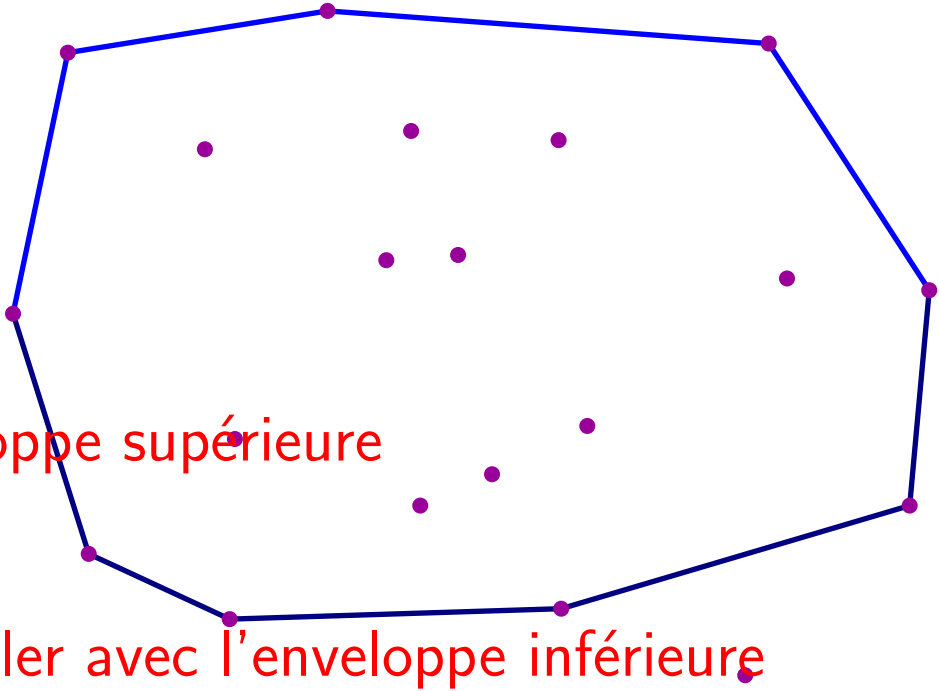
Tri en $x$



# Variante: origine en $y = -\infty$



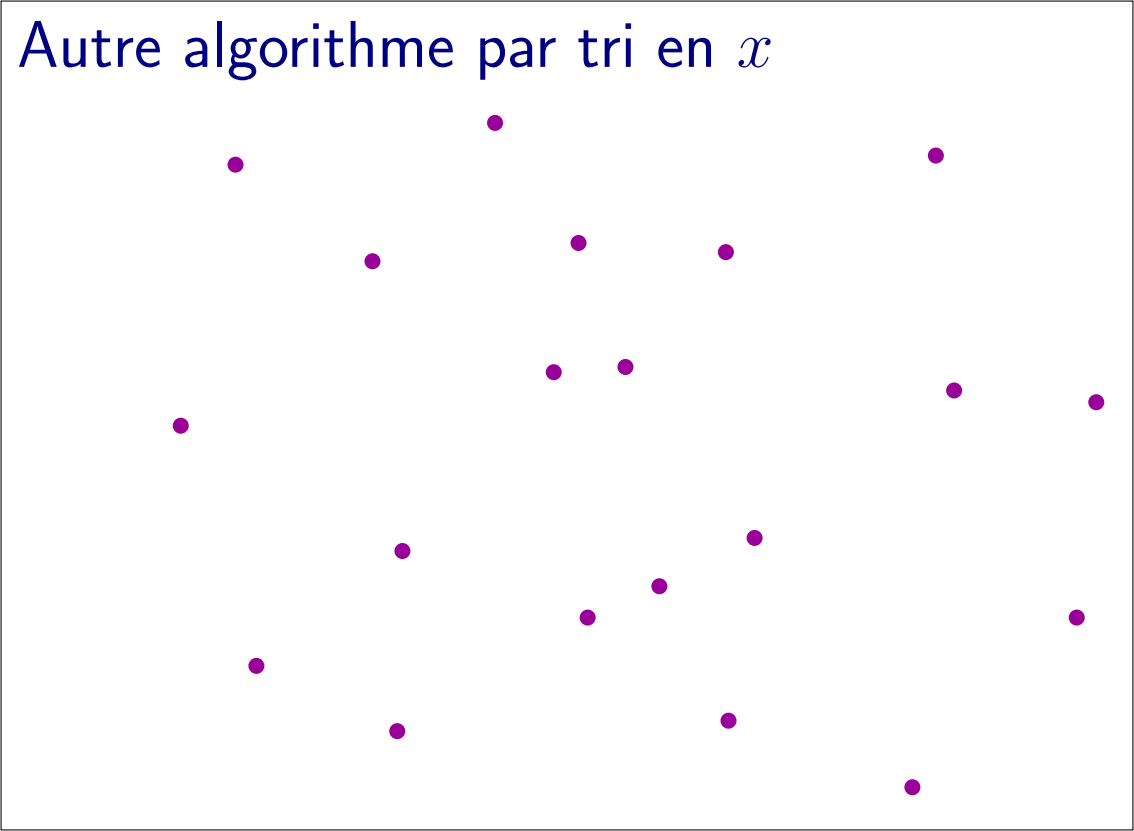
# Variante: origine en $y = -\infty$



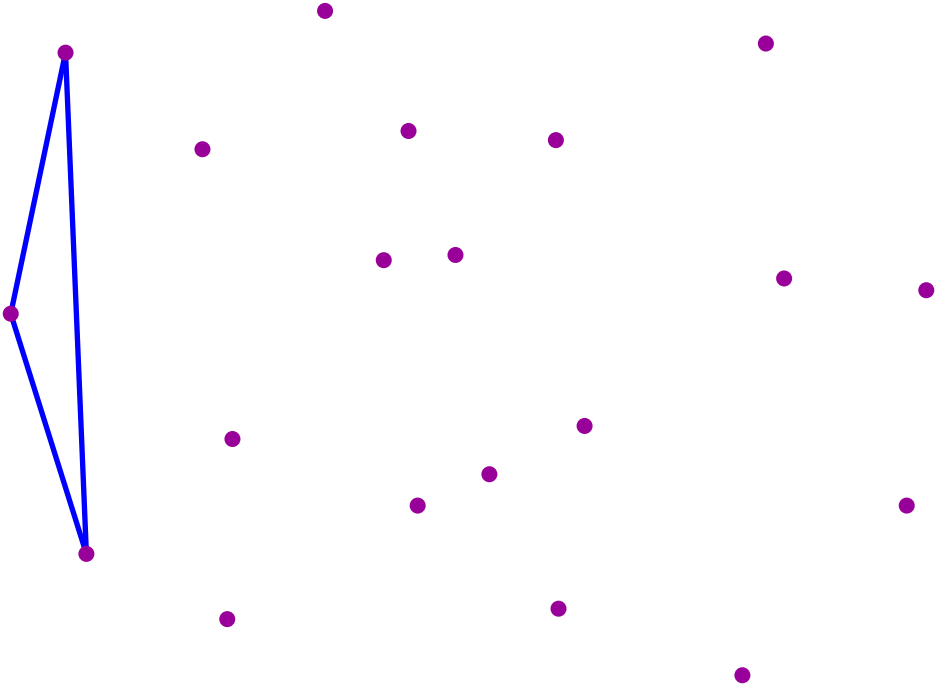
Enveloppe supérieure

À recoler avec l'enveloppe inférieure

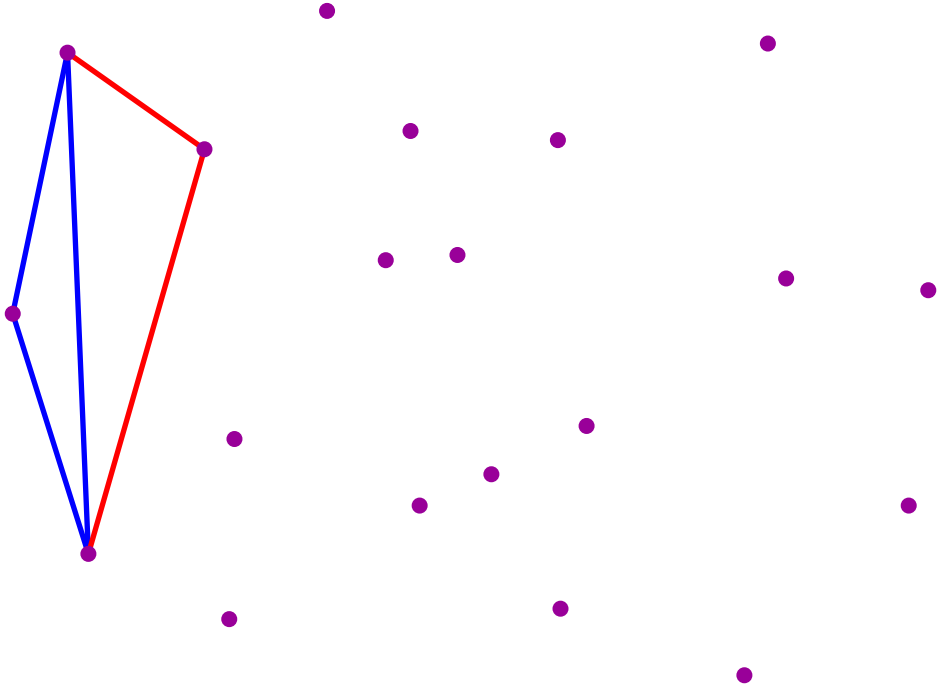
# Autre algorithme par tri en $x$



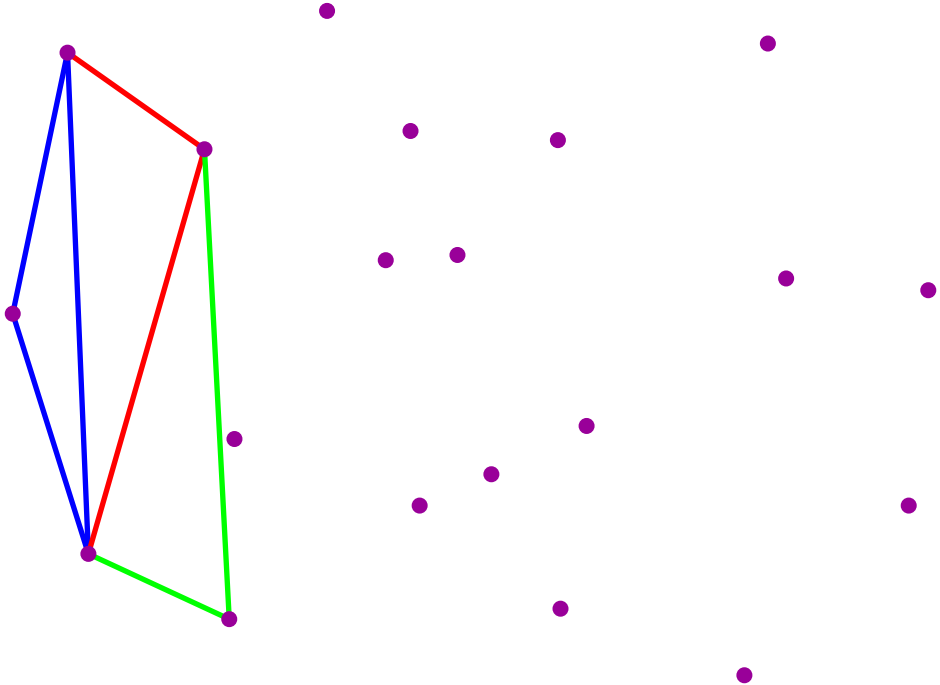
# Autre algorithme par tri en $x$



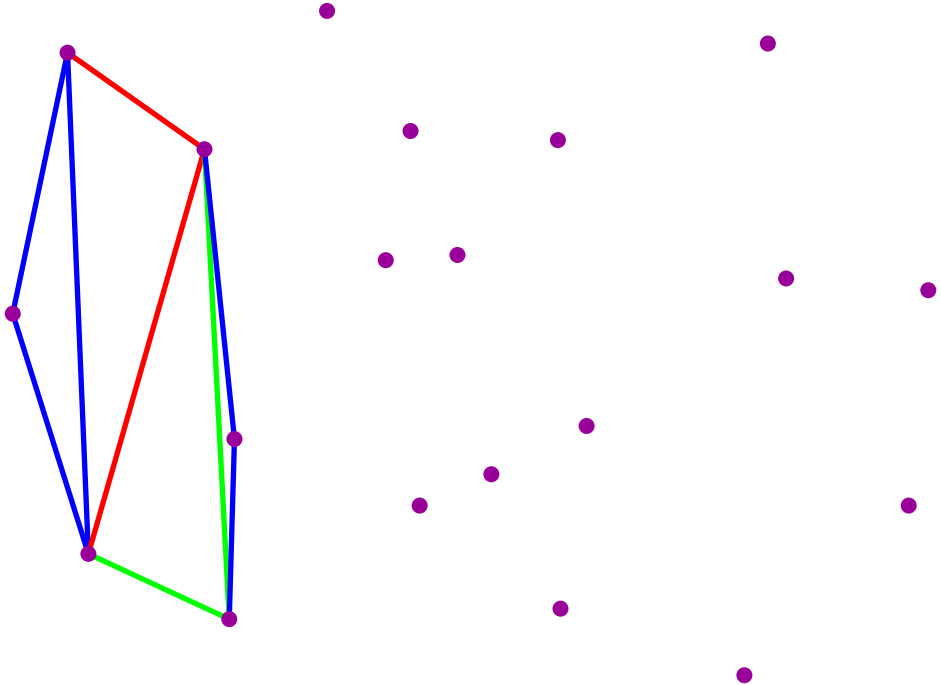
# Autre algorithme par tri en $x$



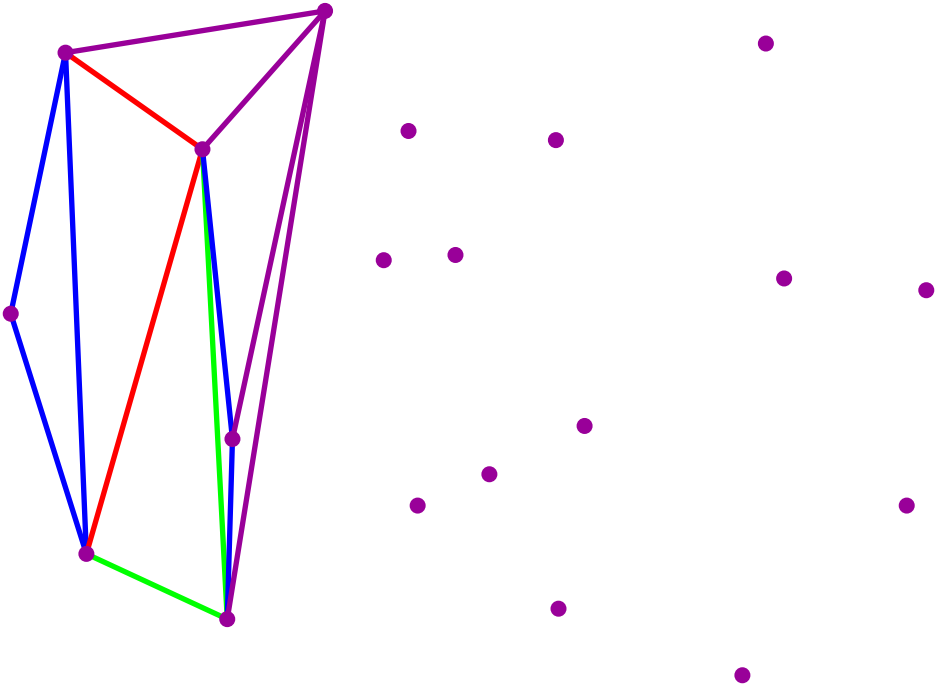
# Autre algorithme par tri en $x$



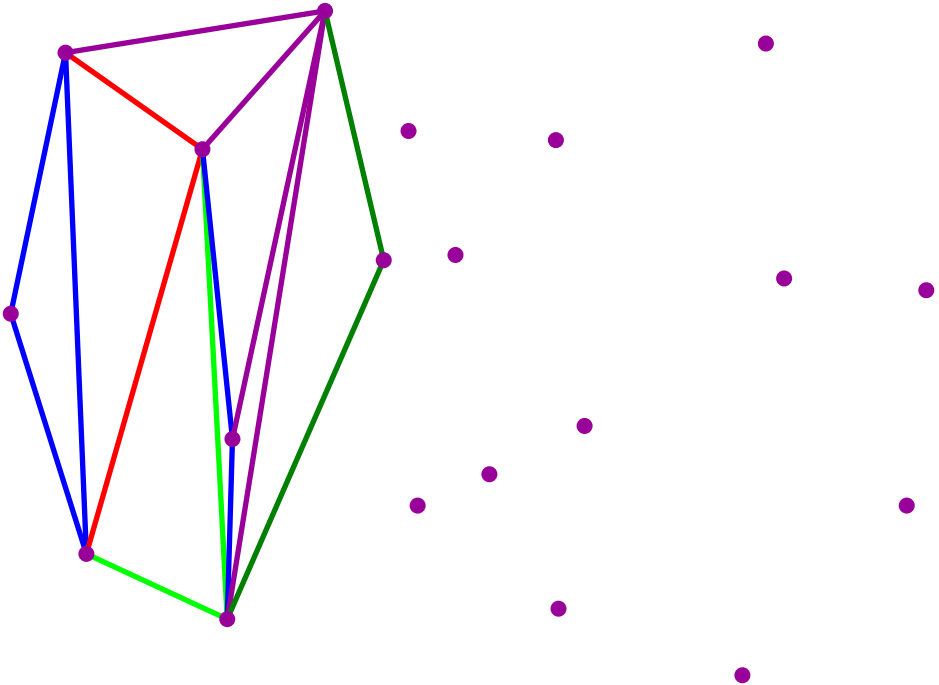
# Autre algorithme par tri en $x$



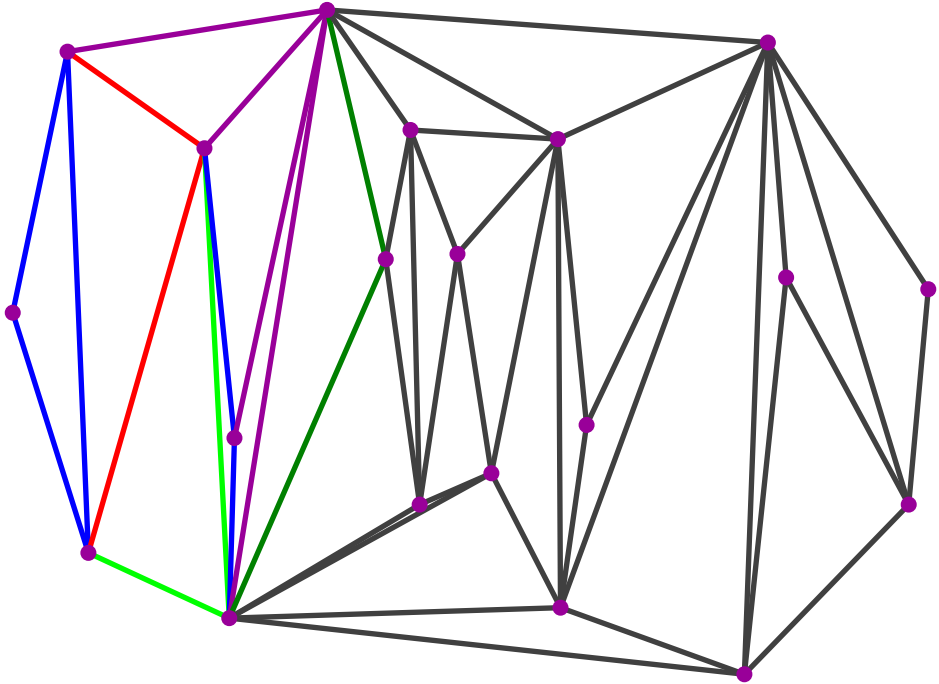
# Autre algorithme par tri en $x$



# Autre algorithme par tri en $x$



# Autre algorithme par tri en $x$



# 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

$u = u.suivant$;

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

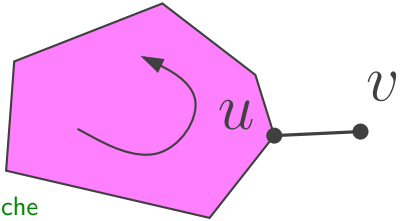
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

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

$u = v$;

# 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

$u = u.suivant$;

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

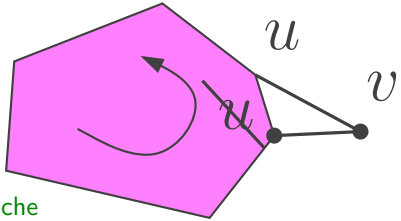
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

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

$u = v$;

# 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

$u = u.suivant$;

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

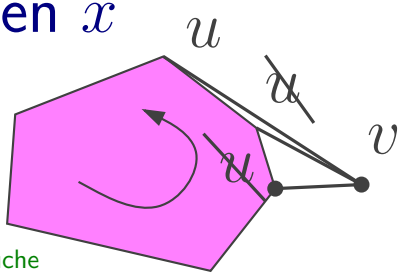
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

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

$u = v$;

# 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

$u = u.suivant$;

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

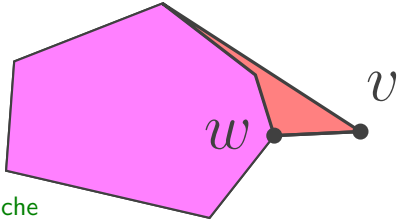
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

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

$u = v$;

# 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

$u = u.suivant$;

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

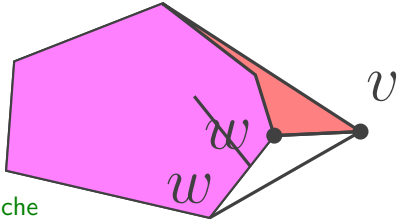
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

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

$u = v$;

# 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

$u = u.suivant$;

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

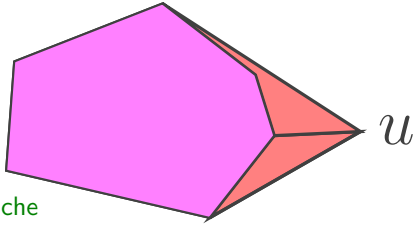
tant que $(v, w, w.pred)$ tourne à gauche

$w = w.pred$;

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

$u = v$;

# 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

$u = u.suivant$;

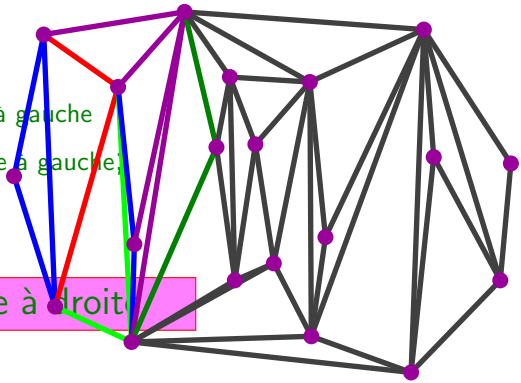
$v.suivant = u$; $u.pred = v$;

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

$w = w.pred$;

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

w=v;



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

$u = u.suivant$;

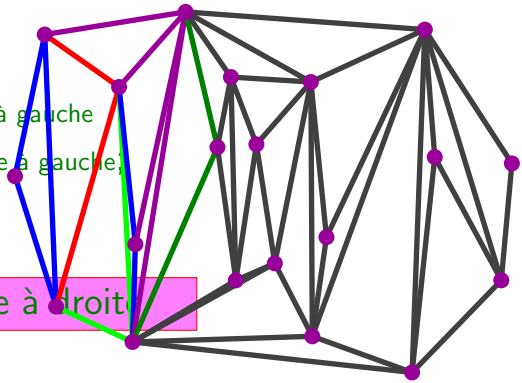
$v.suivant = u$; $u.pred = v$;

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 $x$

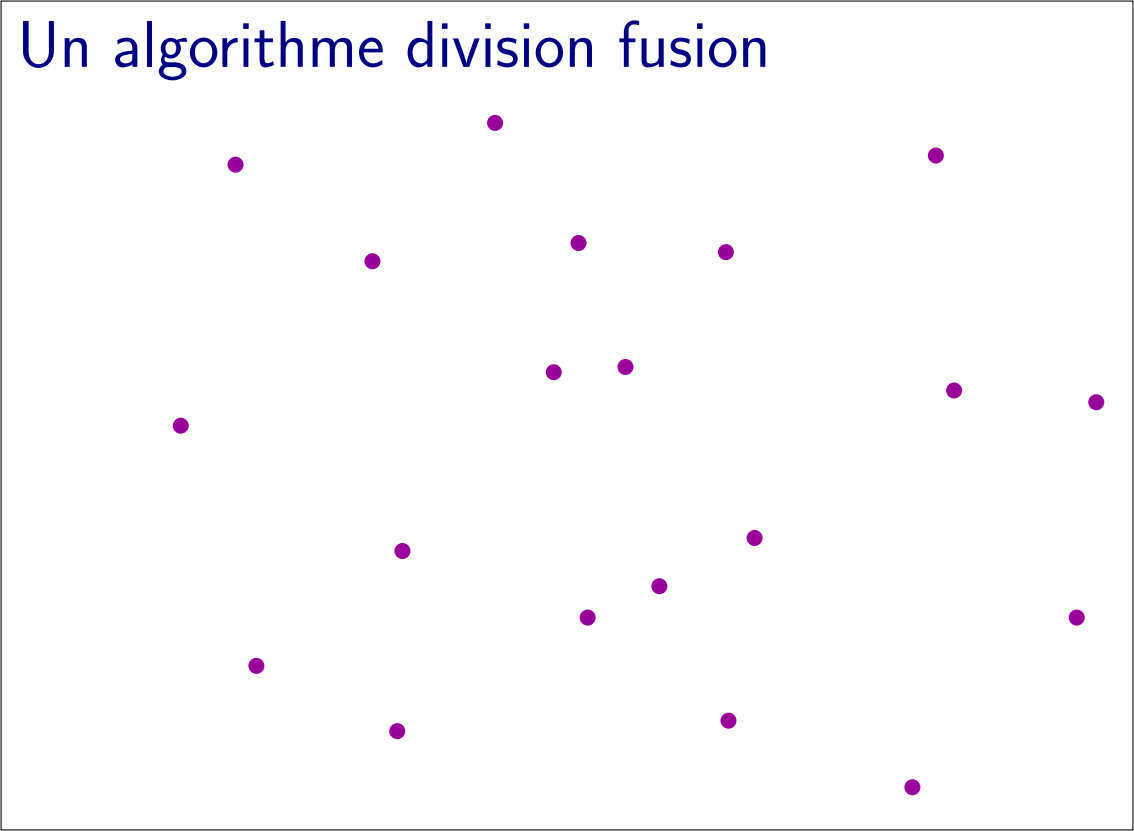
## Complexité

$O(n \log n)$

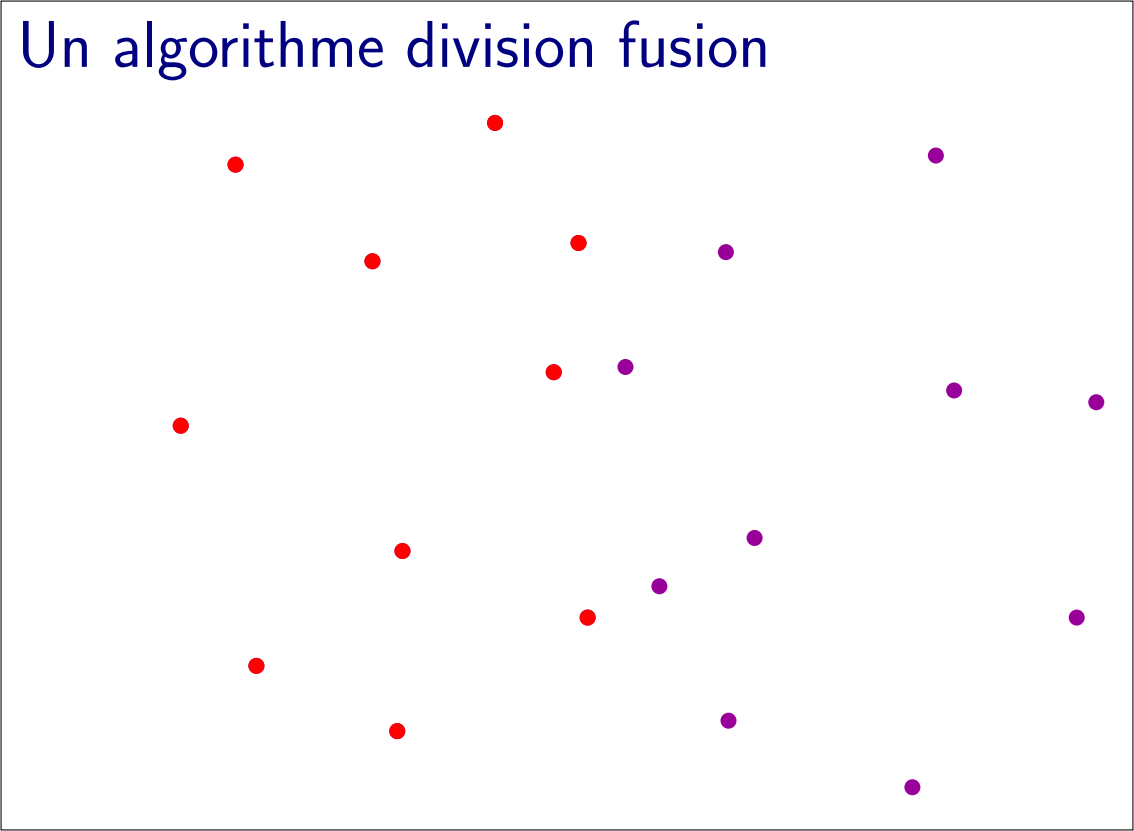
entrée : $S$ un ensemble de points.
trier $S$ en $x$;
initier une liste circulaire avec les 3 points les plus à gauche
tel que $u$ à droite et $u, u.suivant, u.suivant$ tourne à gauche;
Pour $v$ le prochain en $x$

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

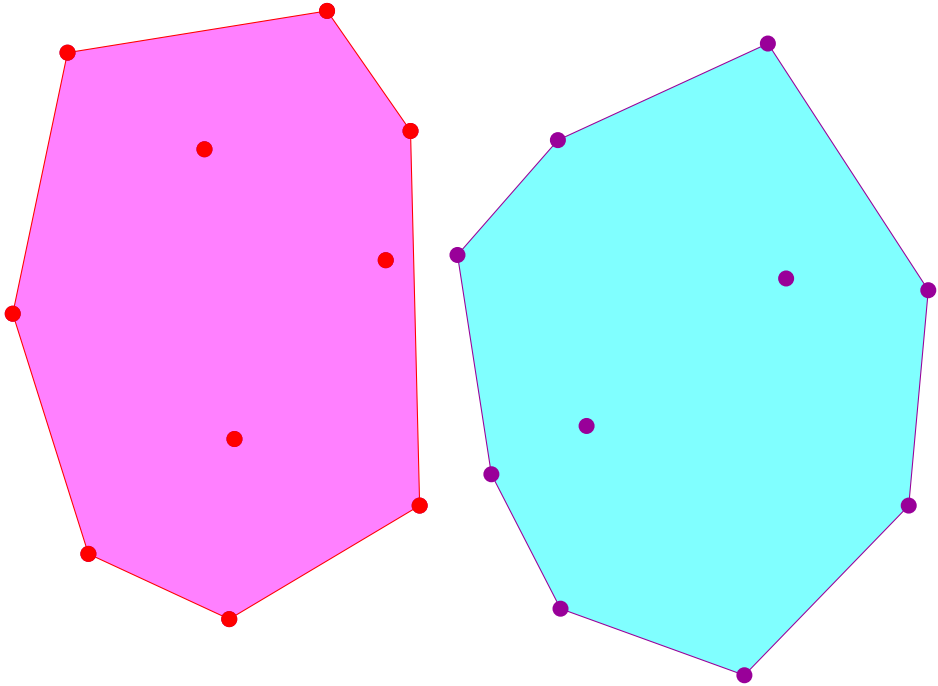
# Un algorithme division fusion



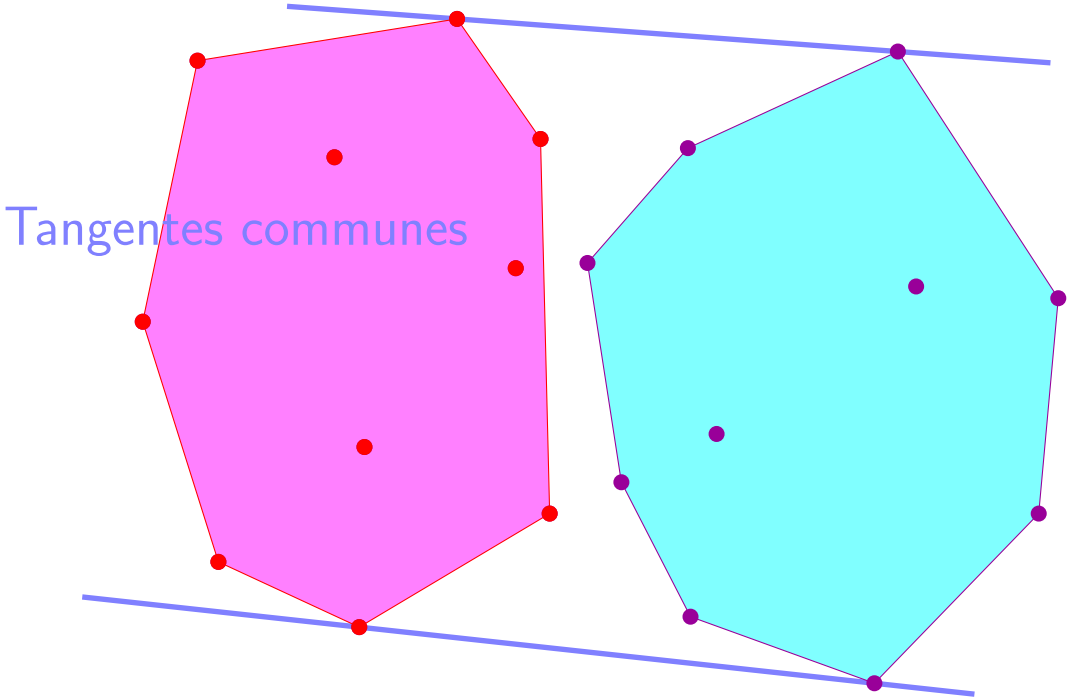
# Un algorithme division fusion



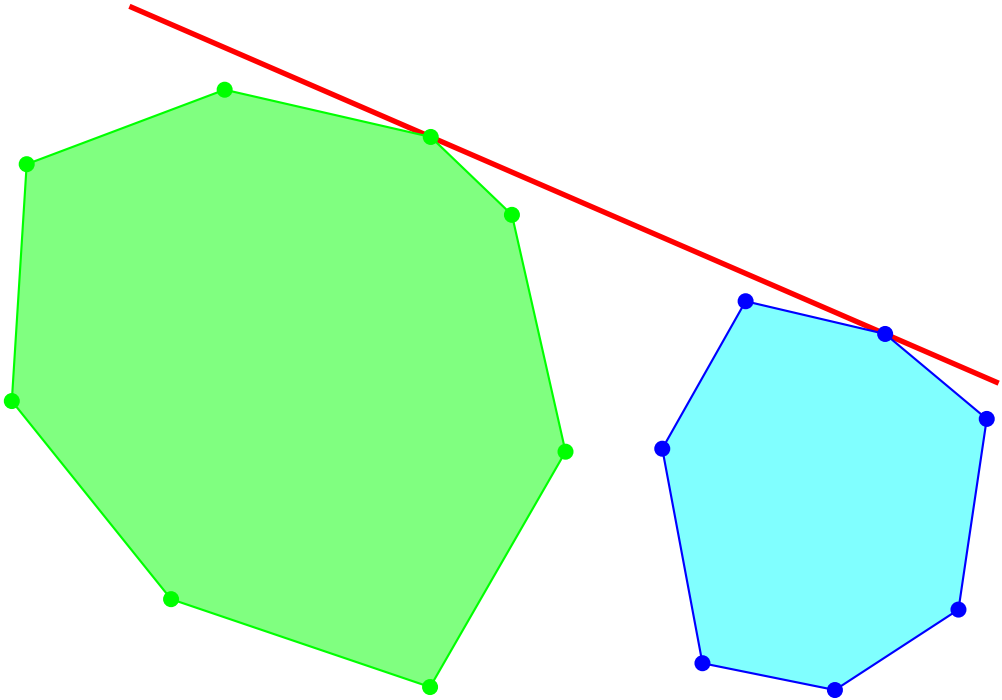
# Un algorithme division fusion



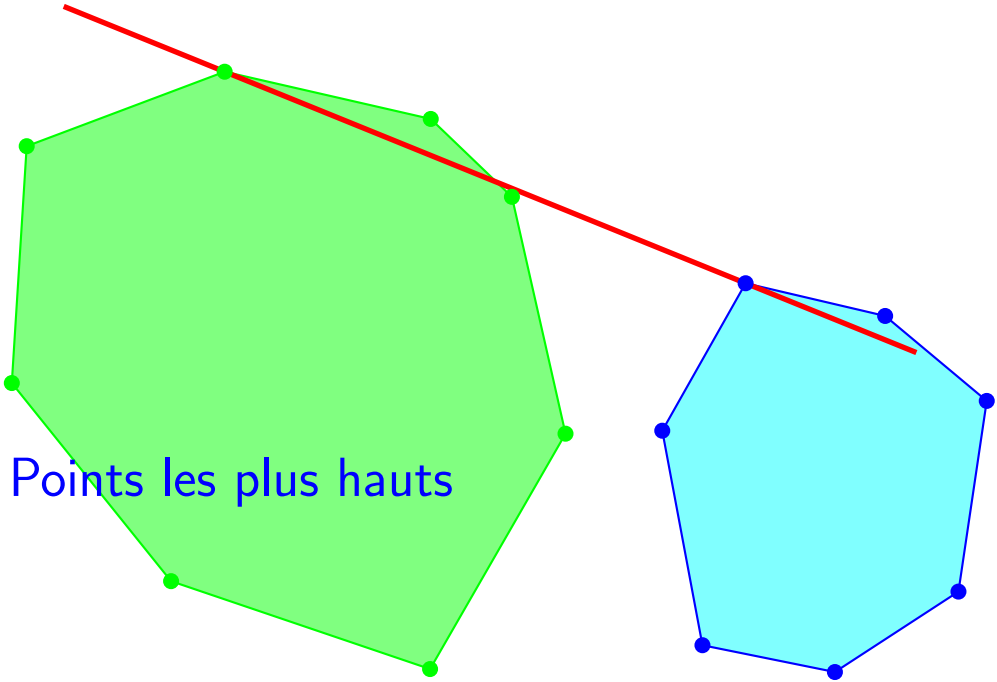
# Un algorithme division fusion



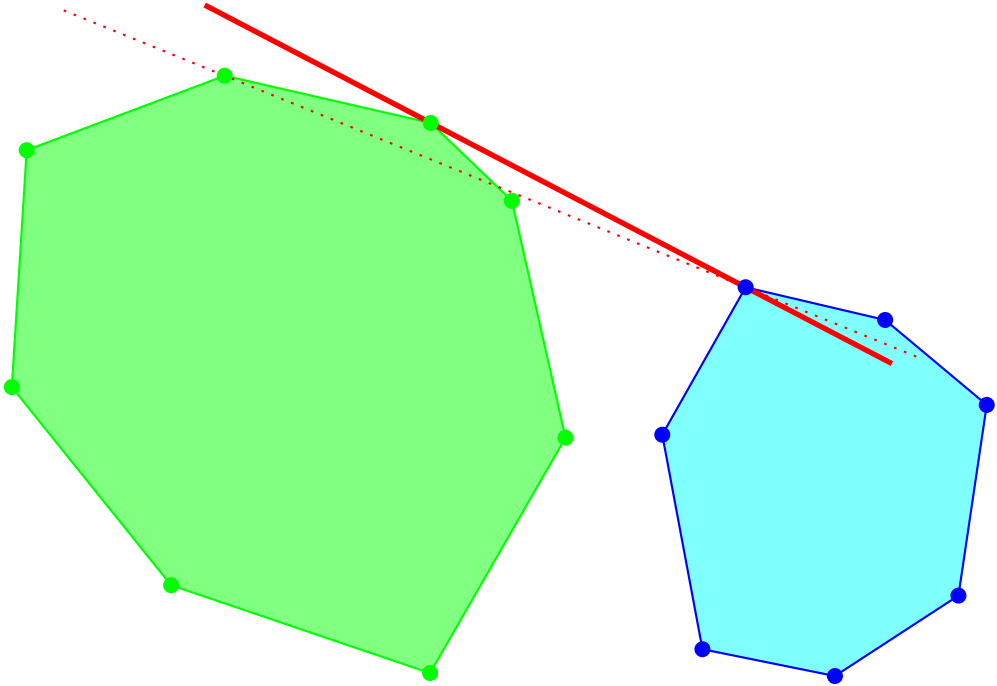
# Tangente supérieure



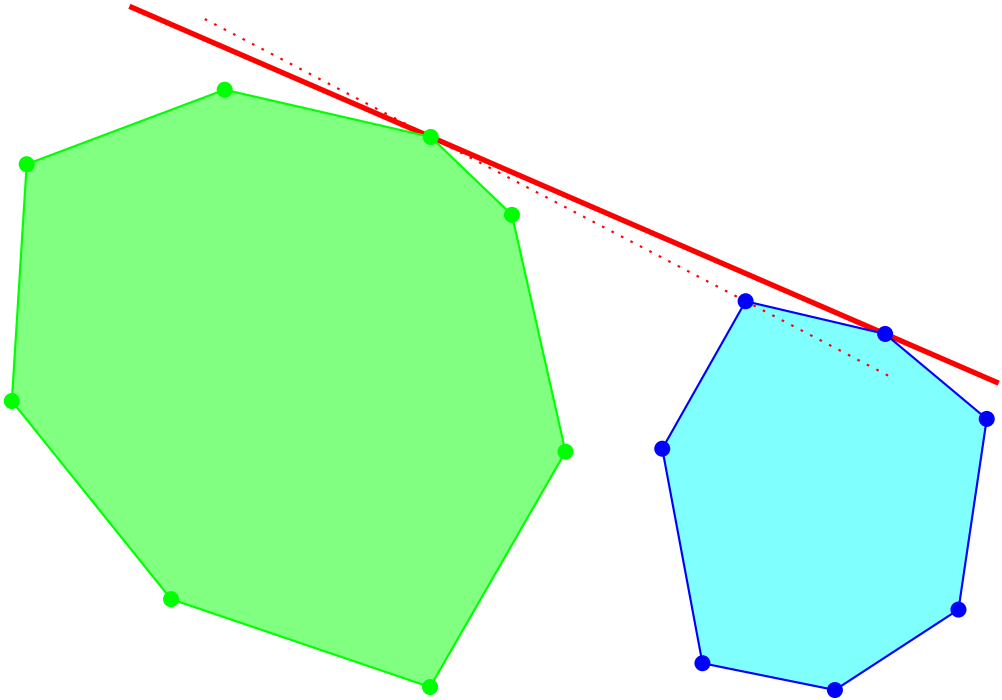
# Tangente supérieure



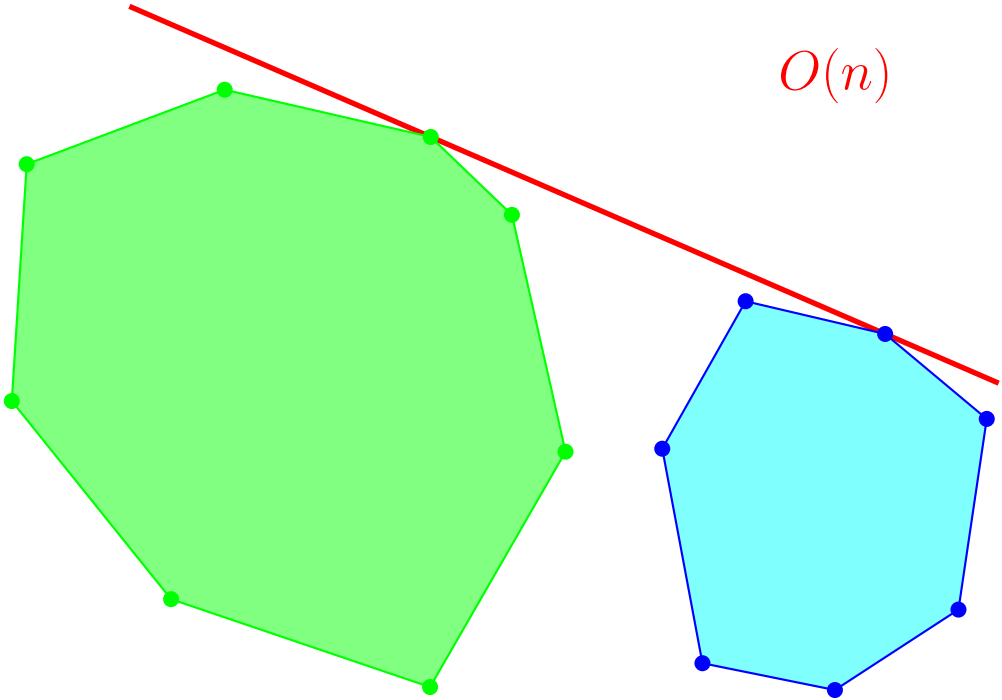
# Tangente supérieure



# 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

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

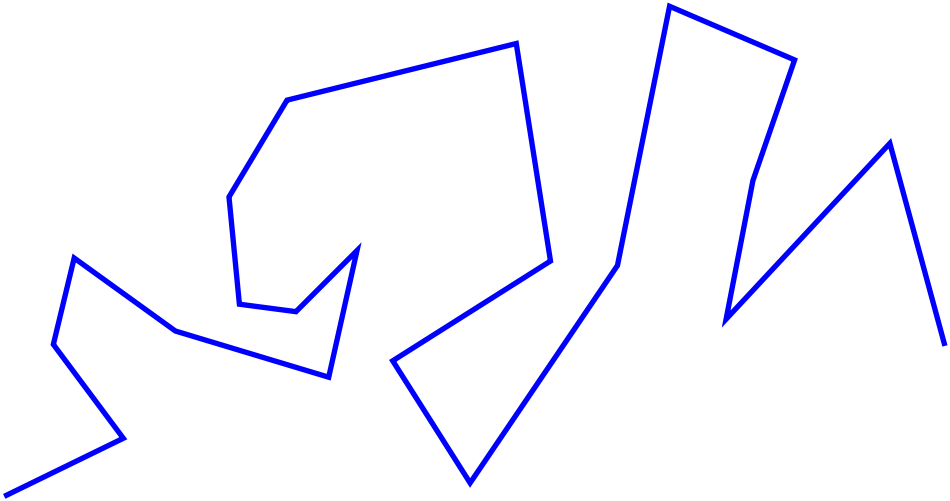
$$= O(n \log n)$$

Division et fusion en $O(n)$

Partition équilibré

(prétraitement $O(n \log n)$

# Cas particulier : polygone simple

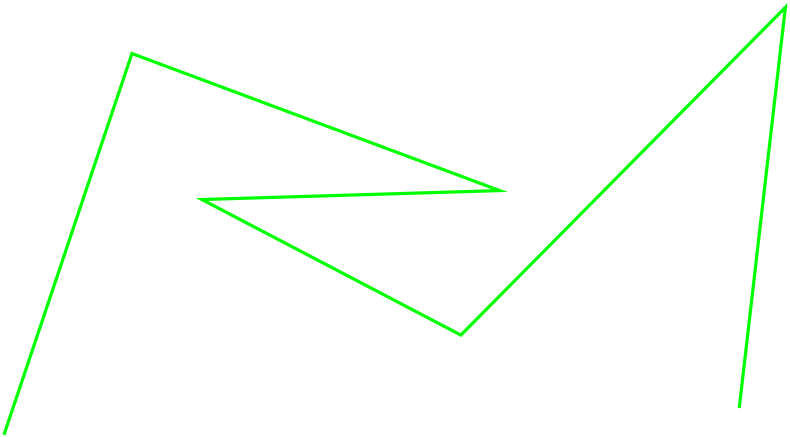


# Cas particulier : polygone simple

(déjà vu : chaine polygonale monotone)

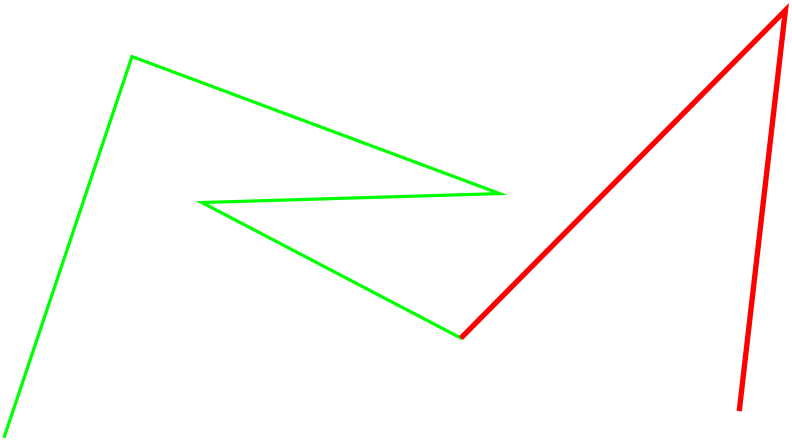
# Cas particulier : polygone simple

Graham marche pas



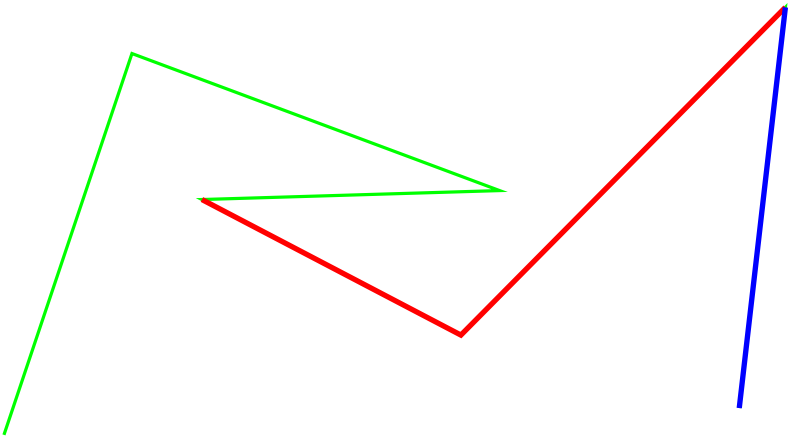
# Cas particulier : polygone simple

Graham marche pas



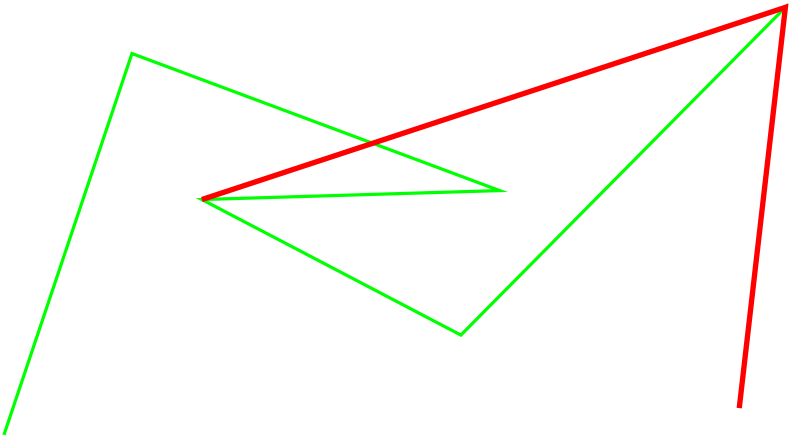
# Cas particulier : polygone simple

## Graham marche pas



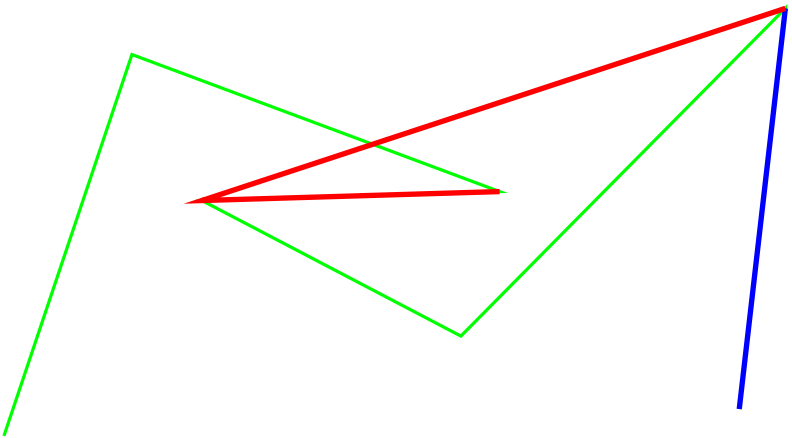
# Cas particulier : polygone simple

## Graham marche pas



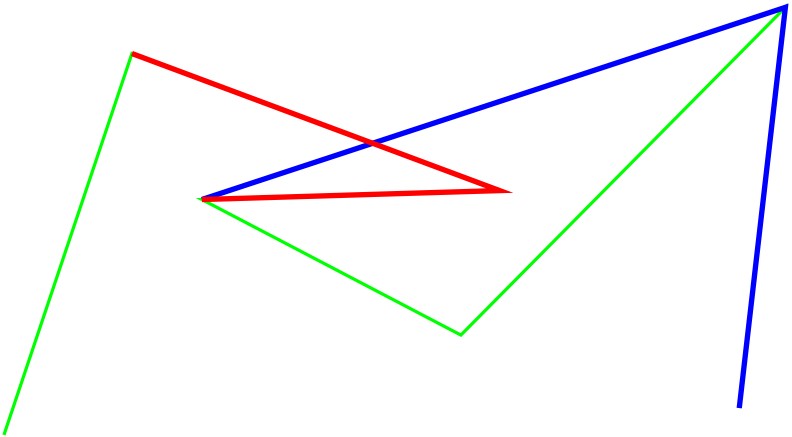
# Cas particulier : polygone simple

## Graham marche pas



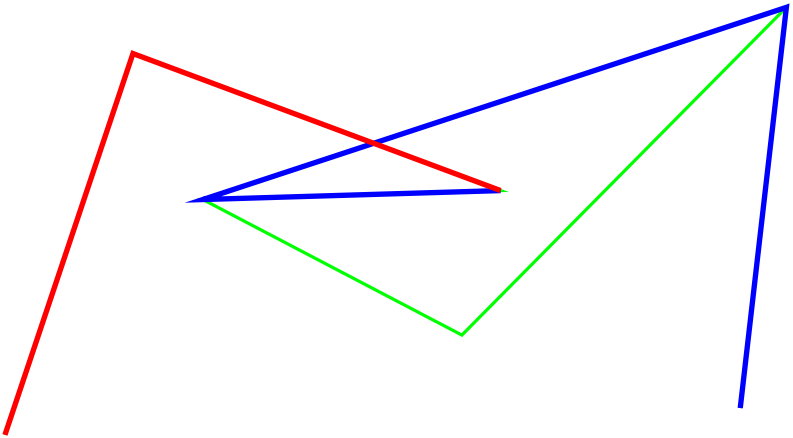
# Cas particulier : polygone simple

## Graham marche pas



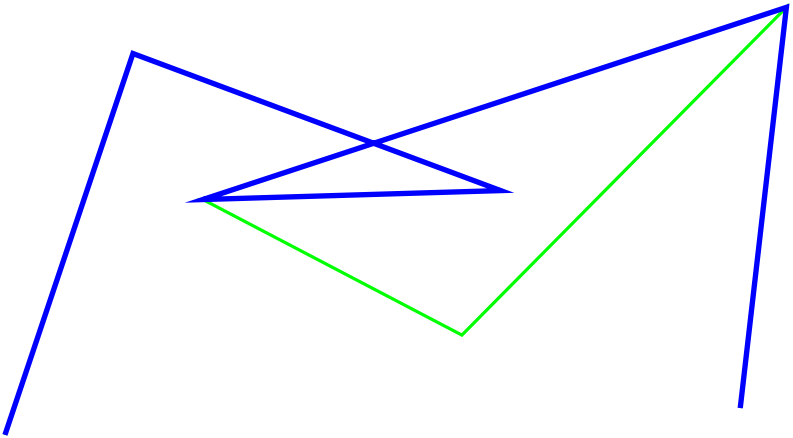
# Cas particulier : polygone simple

Graham marche pas

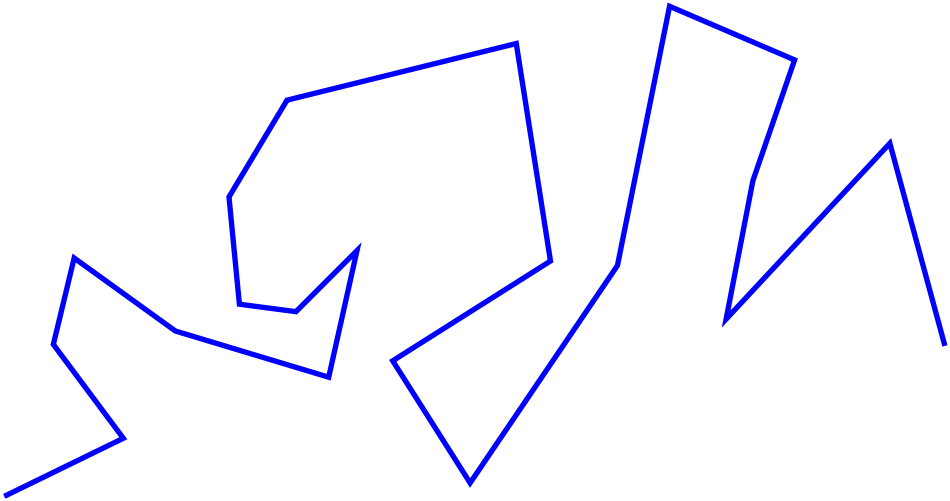


# Cas particulier : polygone simple

Graham marche pas

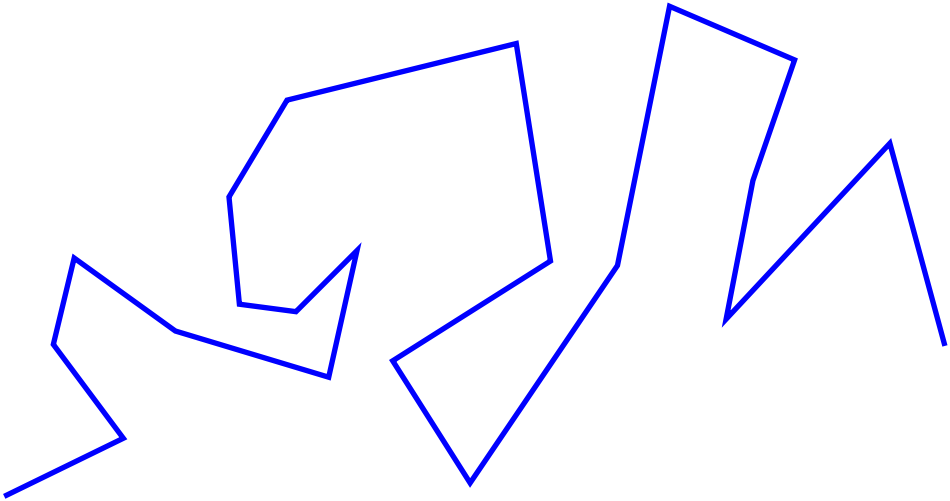


# Cas particulier : polygone simple

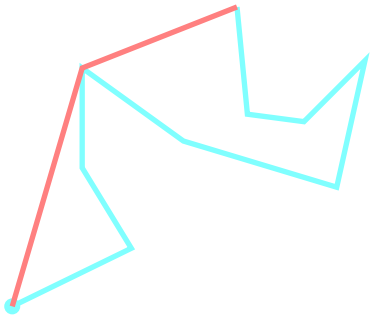


# Cas particulier : polygone simple

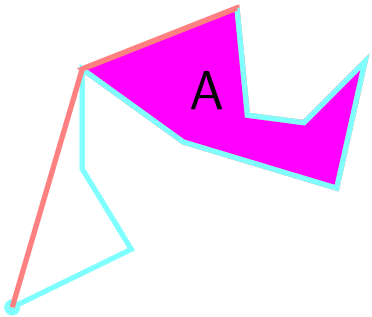
Principe : boucher les poches



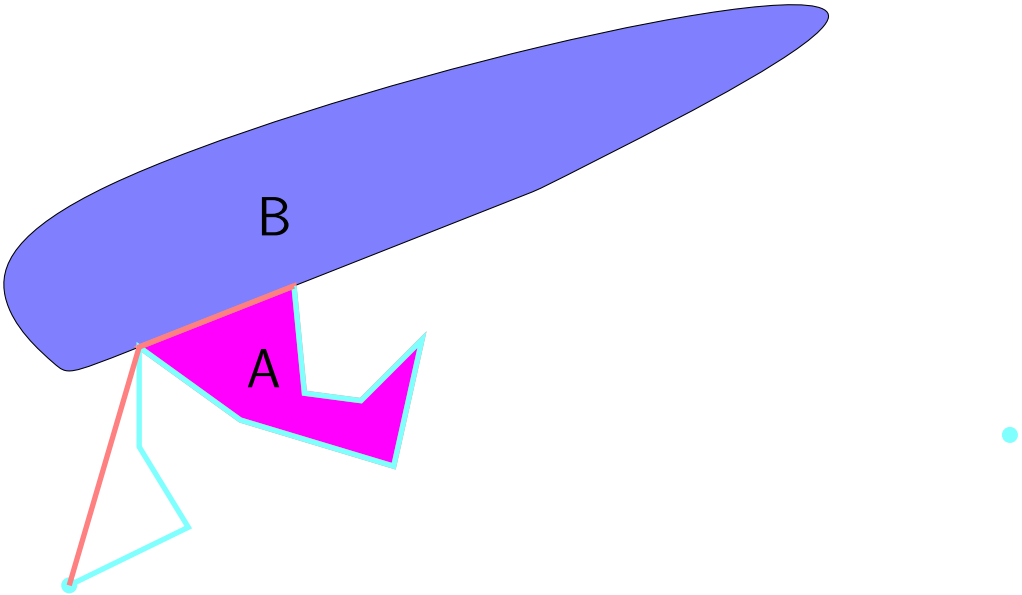
# Cas particulier : polygone simple



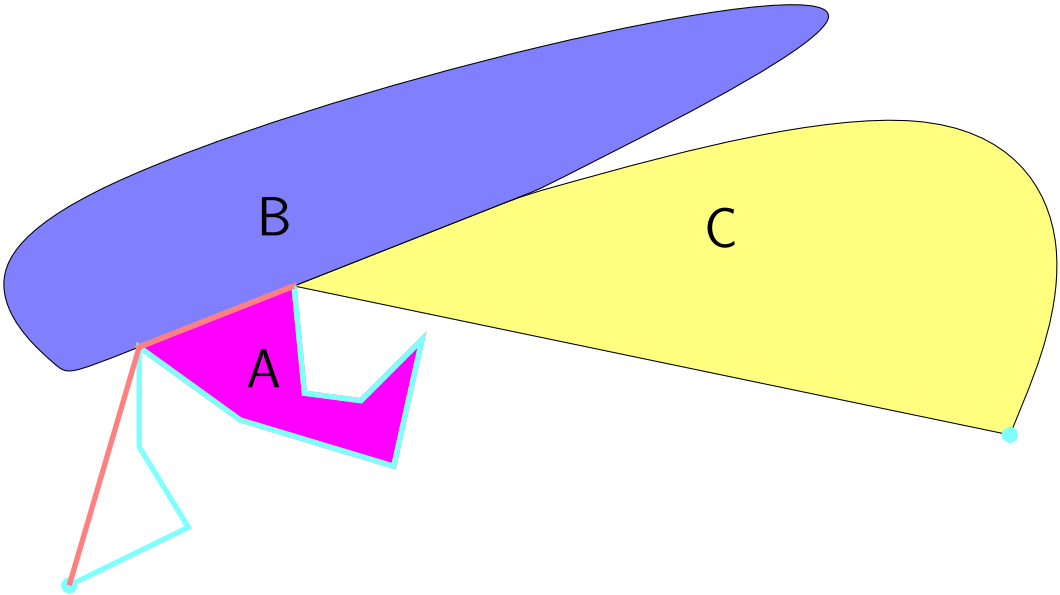
# Cas particulier : polygone simple



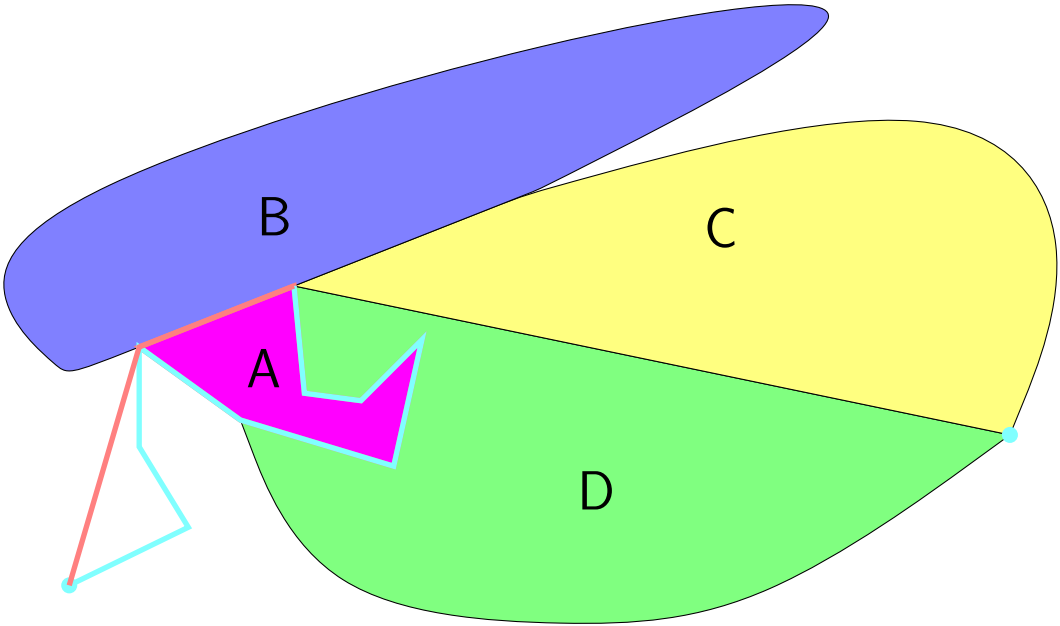
# Cas particulier : polygone simple



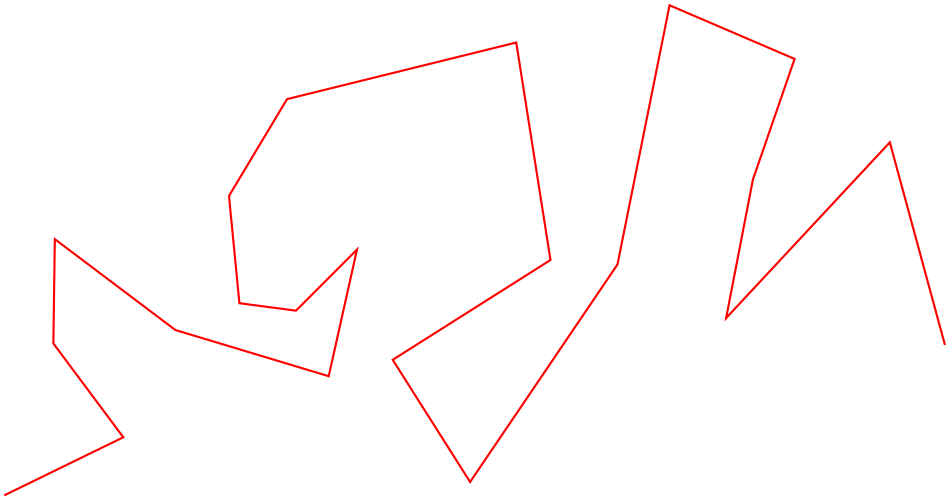
# Cas particulier : polygone simple



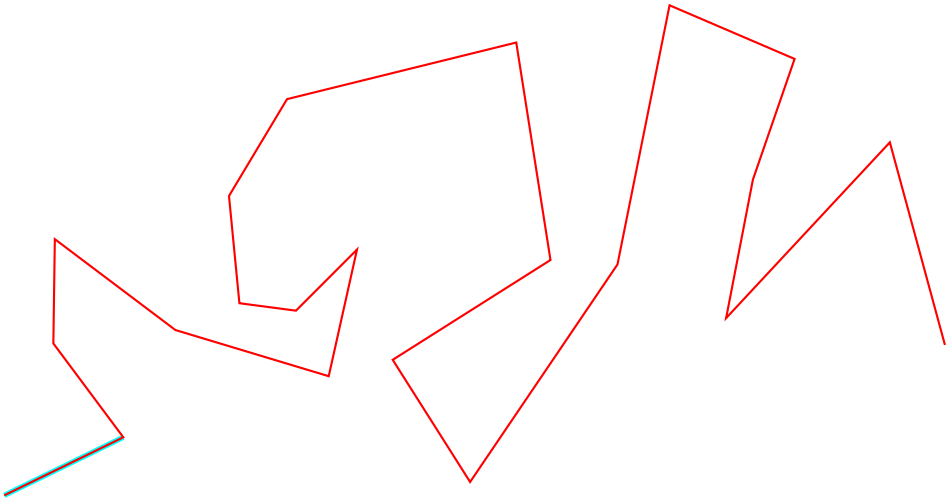
# Cas particulier : polygone simple



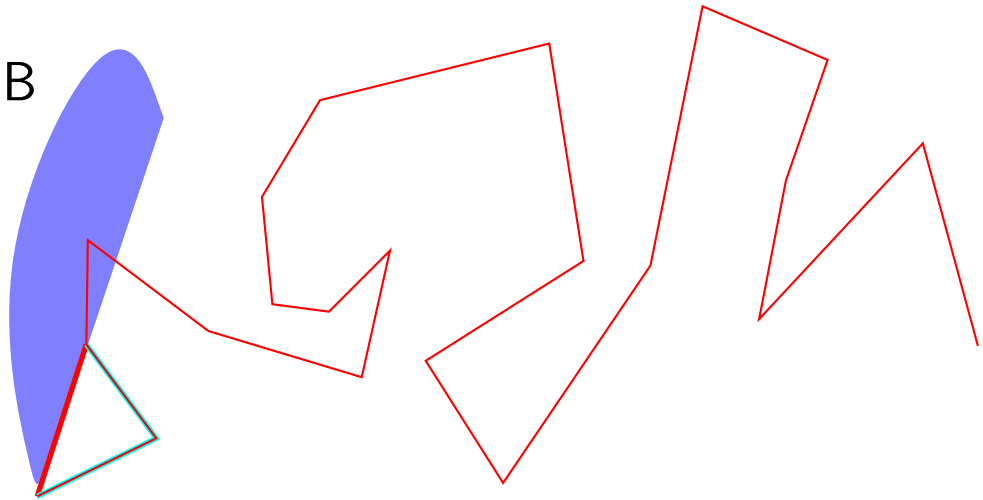
# Cas particulier : polygone simple



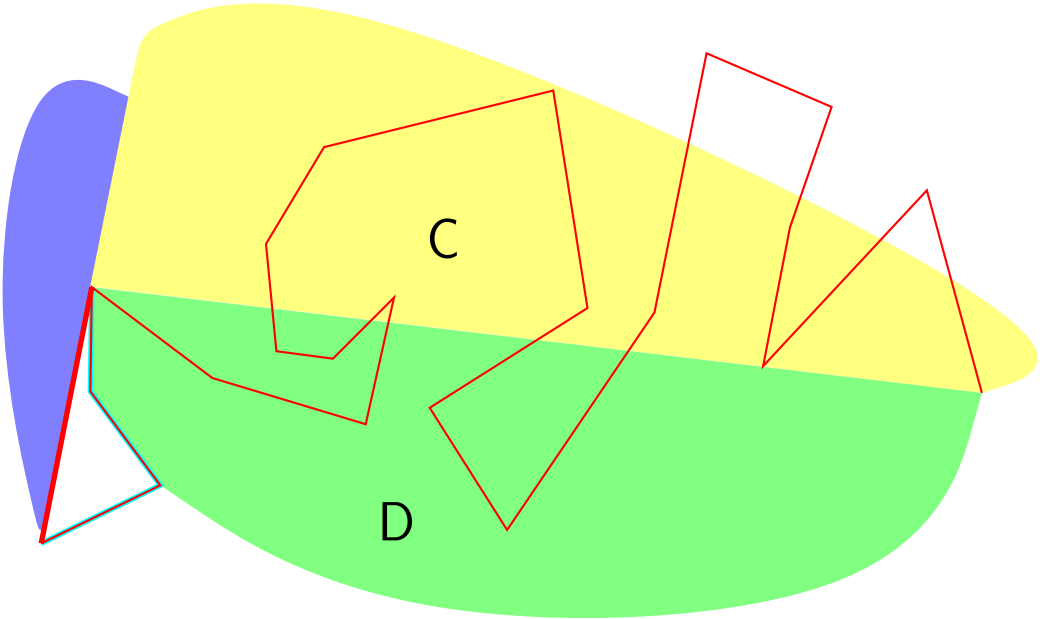
# Cas particulier : polygone simple



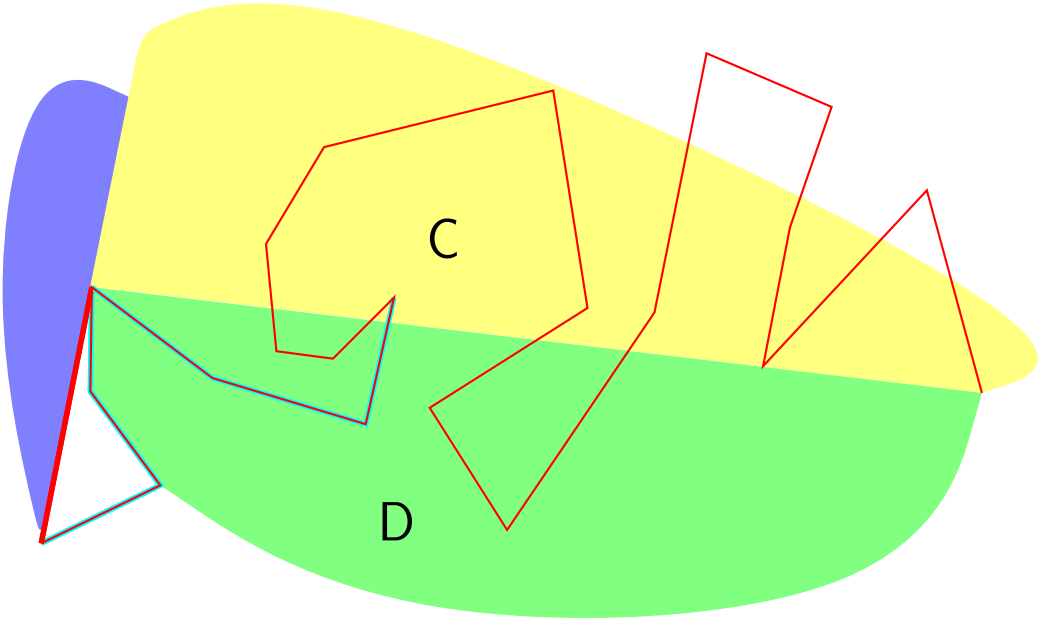
# Cas particulier : polygone simple



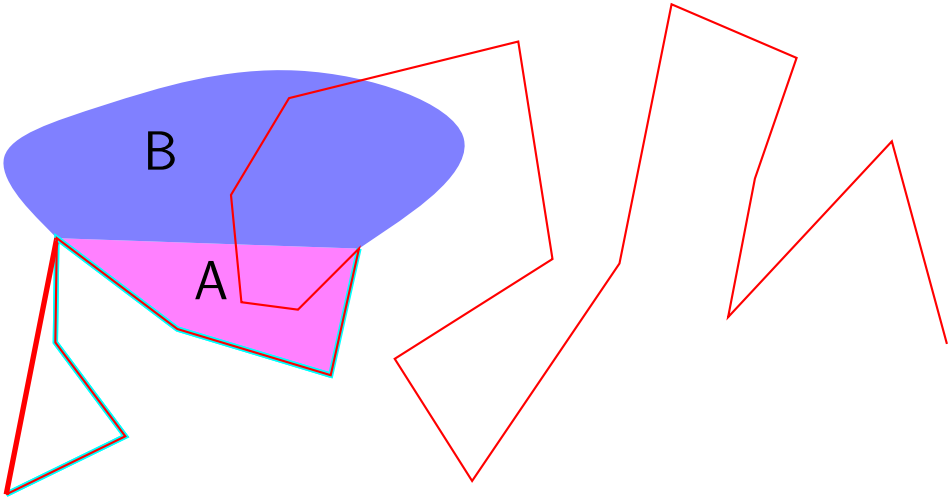
# Cas particulier : polygone simple



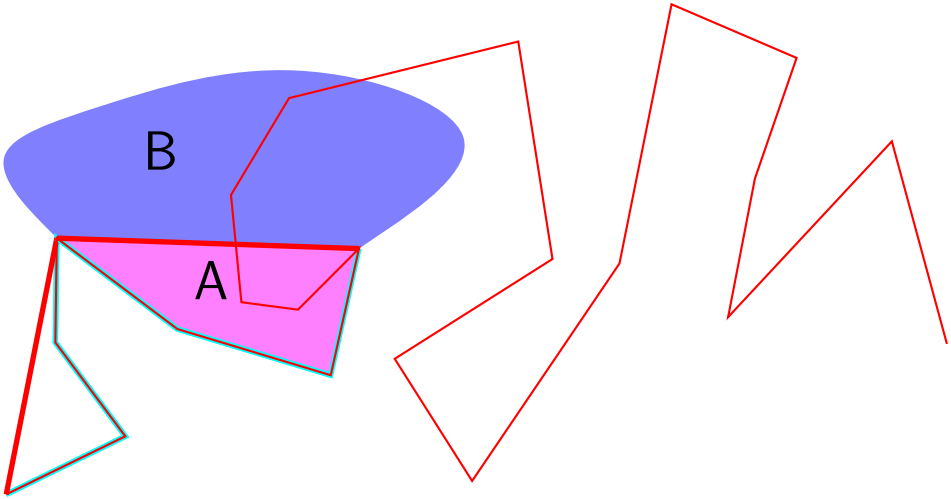
# Cas particulier : polygone simple



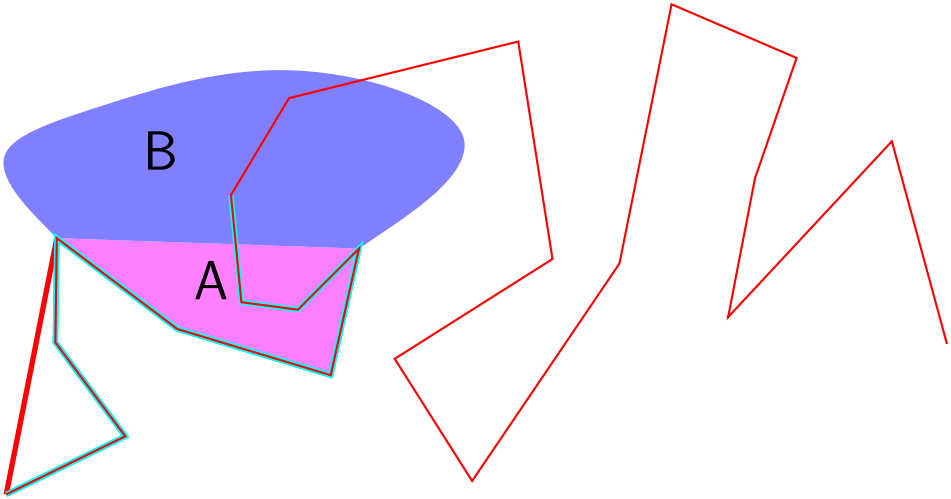
# Cas particulier : polygone simple



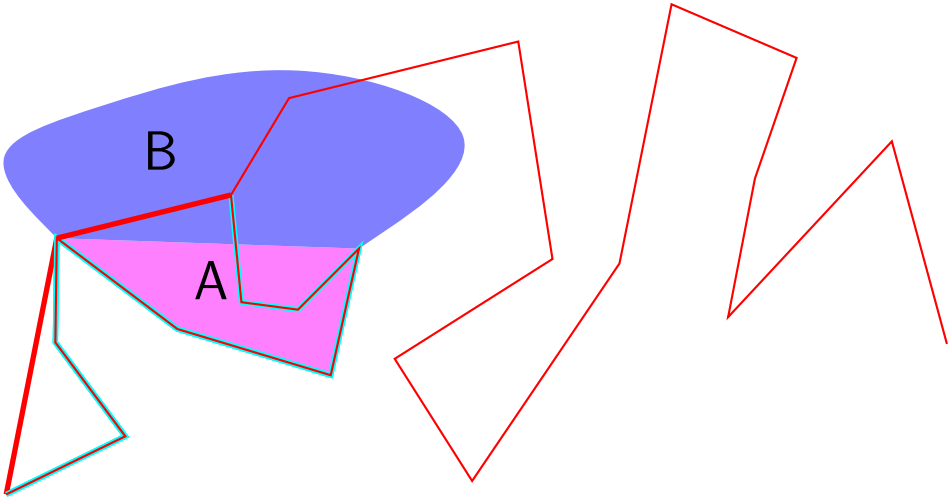
# Cas particulier : polygone simple



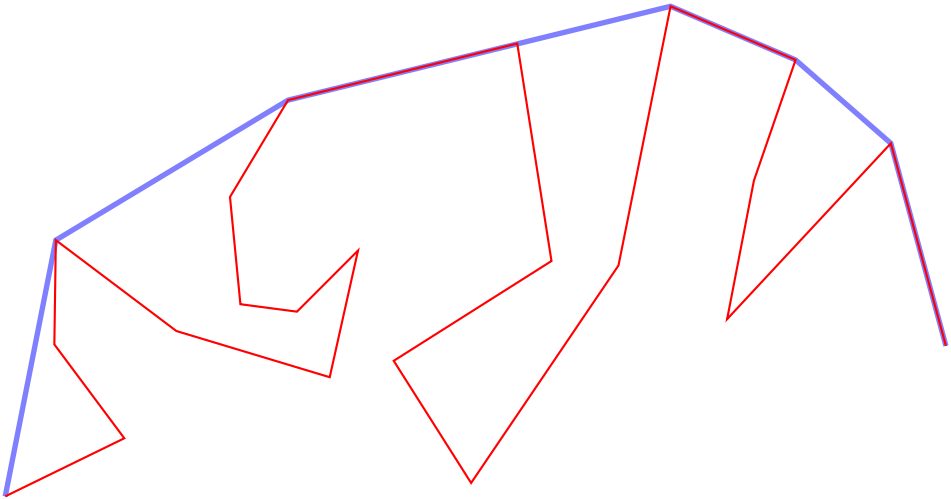
# Cas particulier : polygone simple



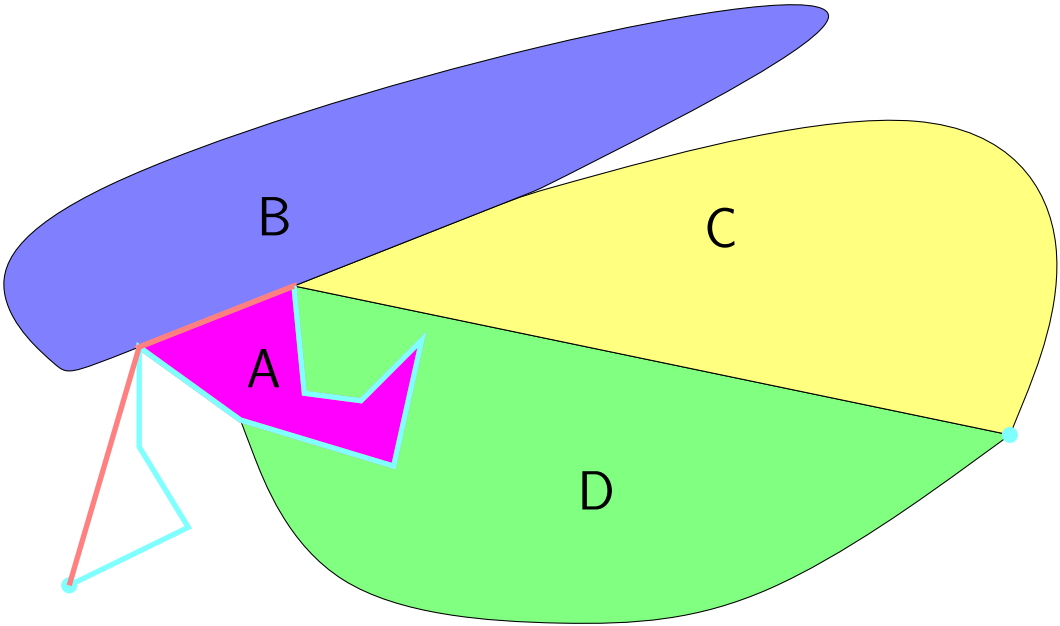
# Cas particulier : polygone simple



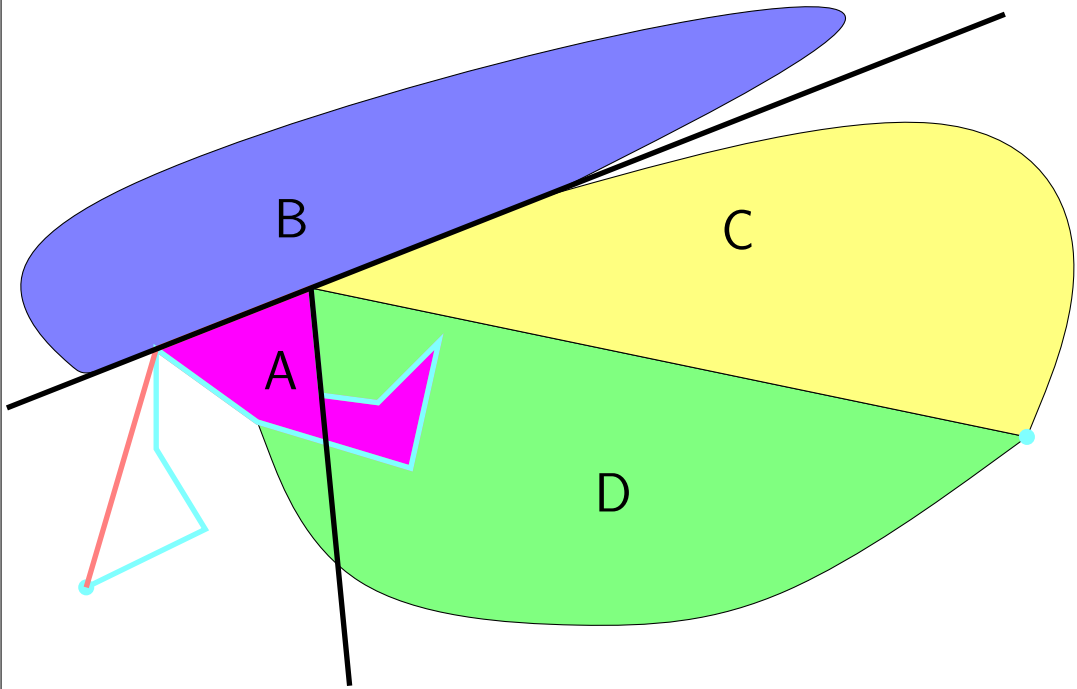
# Cas particulier : polygone simple



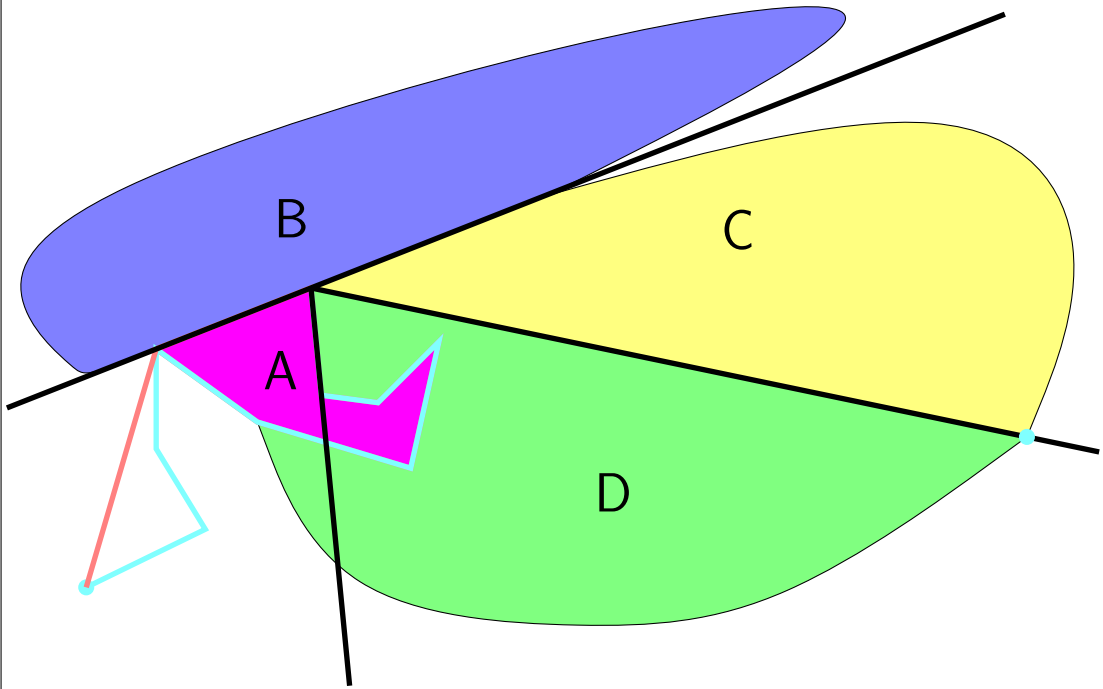
# Cas particulier : polygone simple



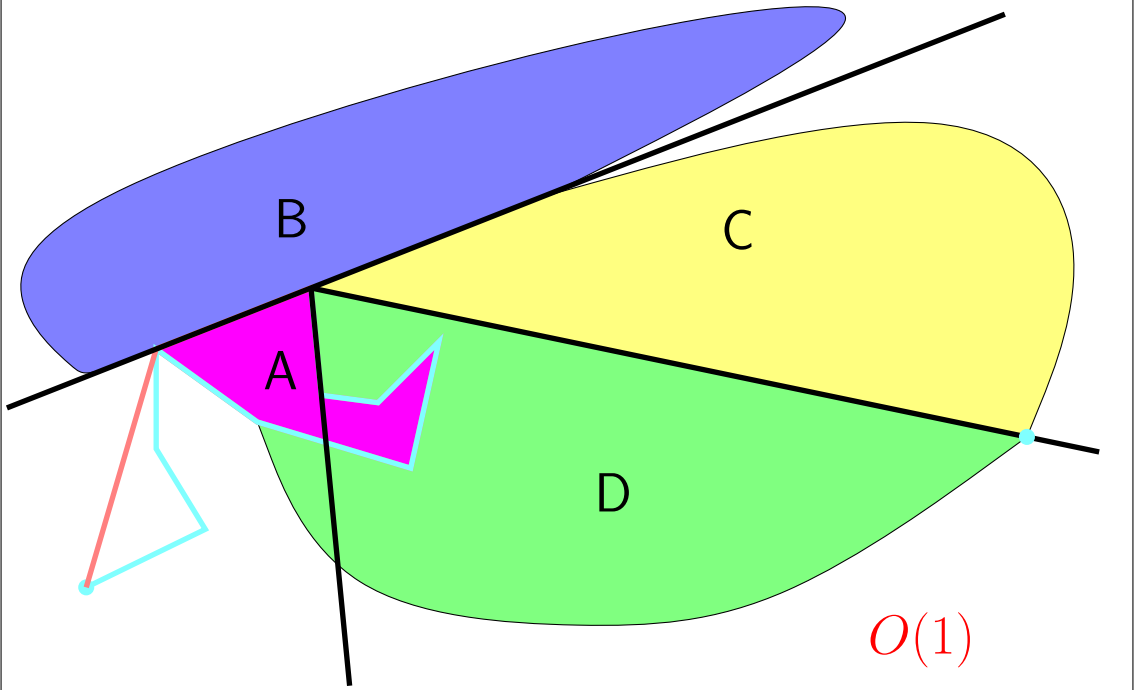
# Cas particulier : polygone simple



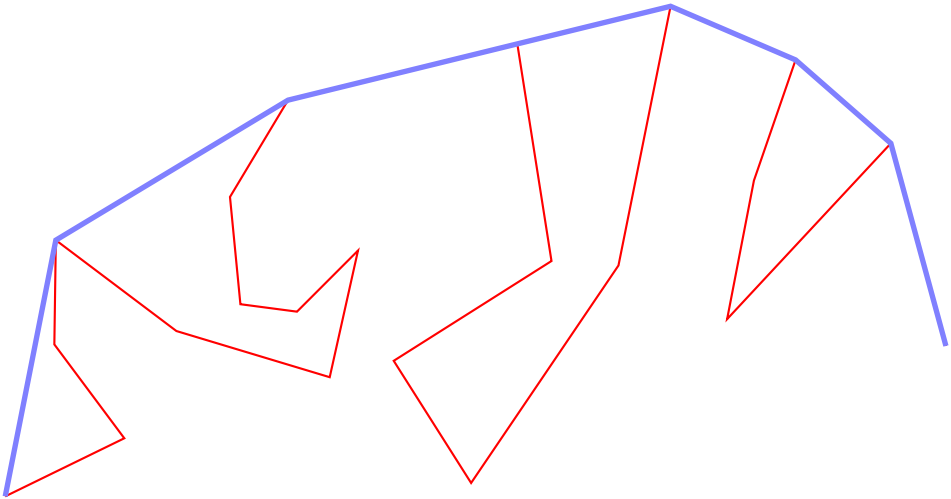
# Cas particulier : polygone simple



# Cas particulier : polygone simple

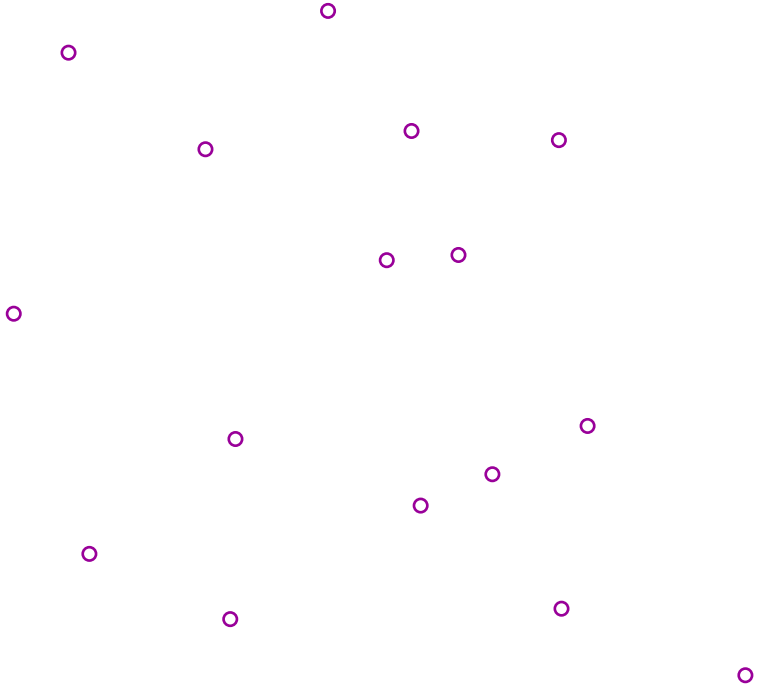


# Cas particulier : polygone simple

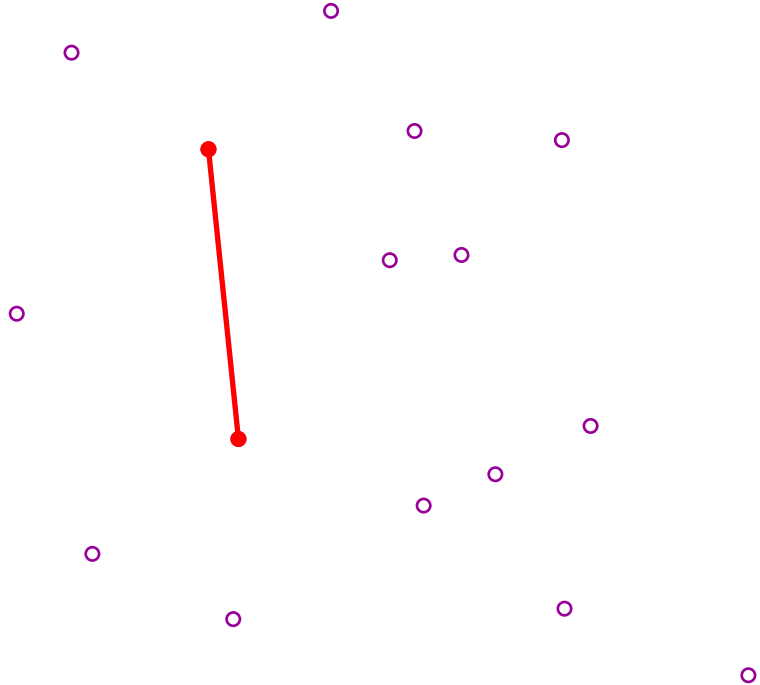


$O(n)$

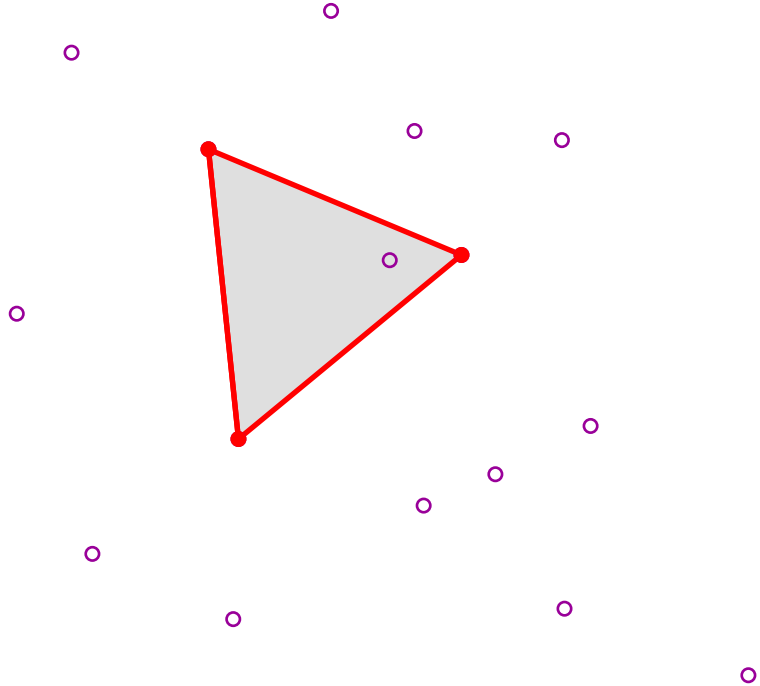
# Un algorithme incrémental



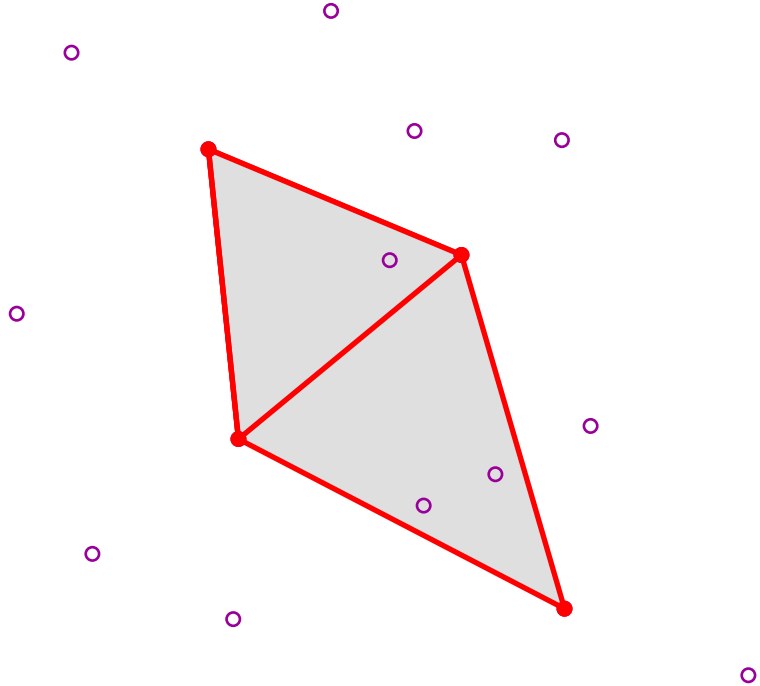
# Un algorithme incrémental



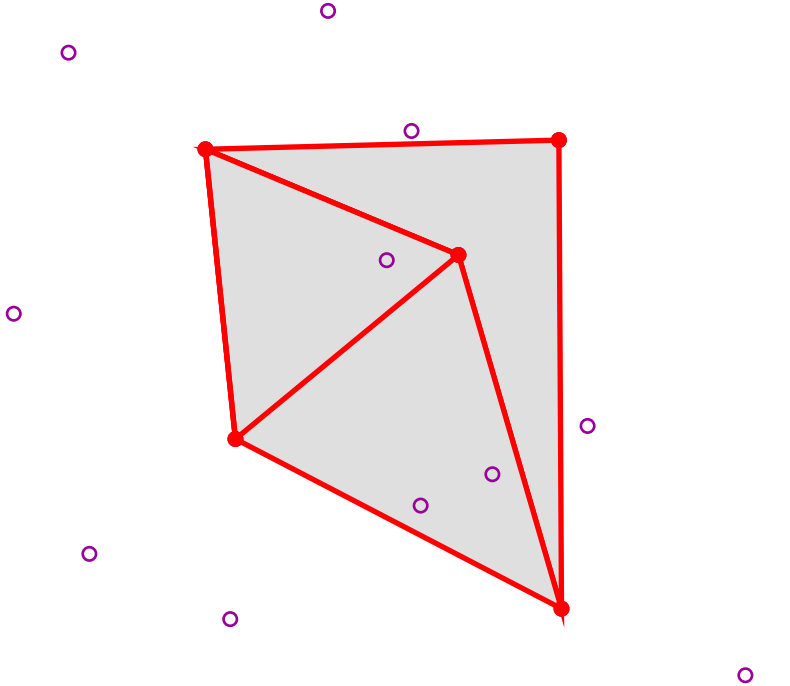
# Un algorithme incrémental



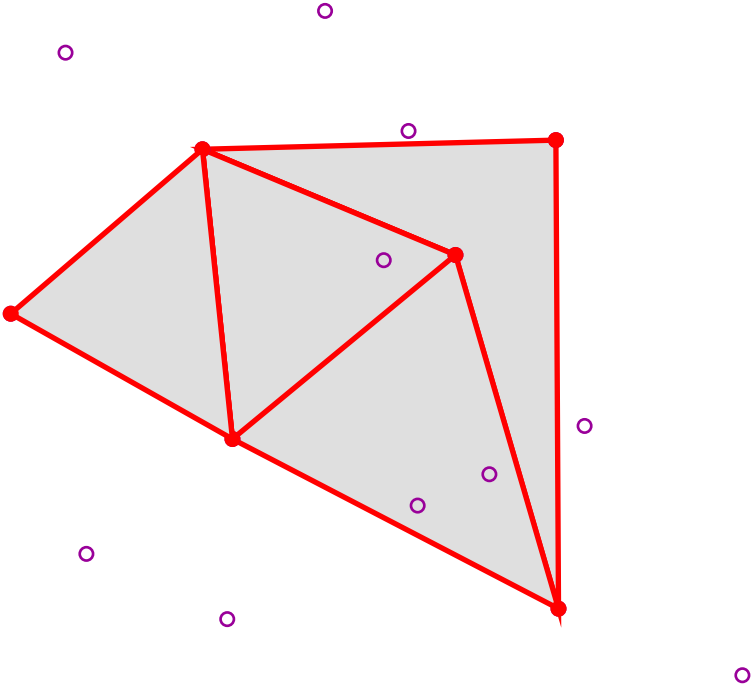
# Un algorithme incrémental



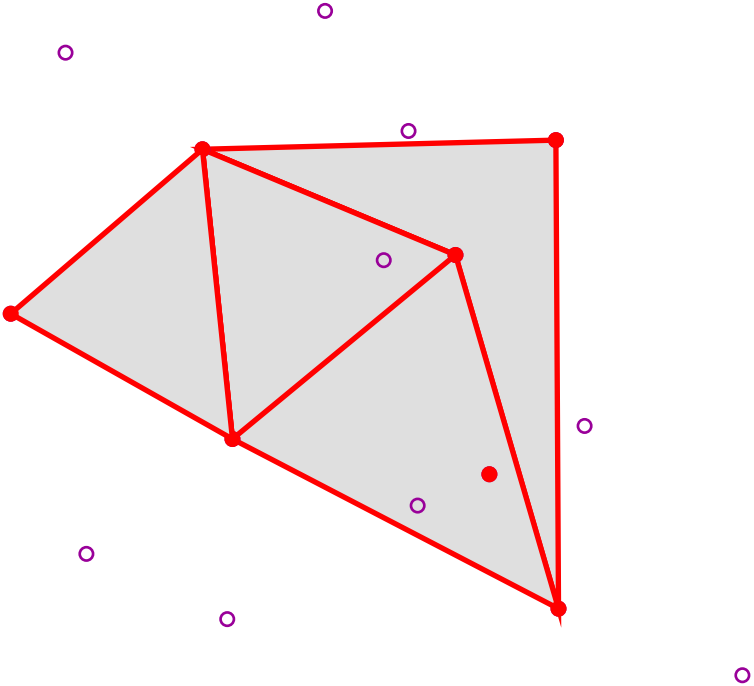
# Un algorithme incrémental



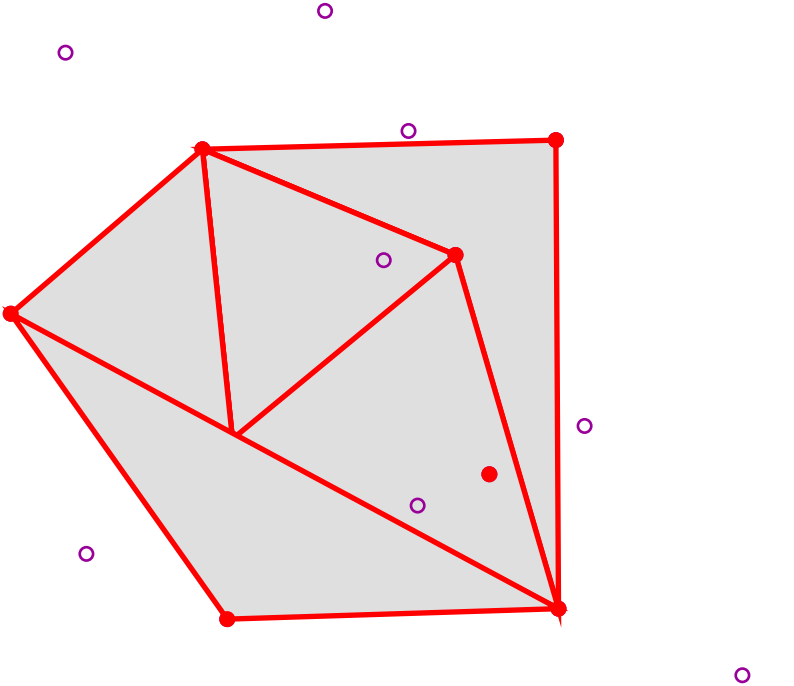
# Un algorithme incrémental



# Un algorithme incrémental

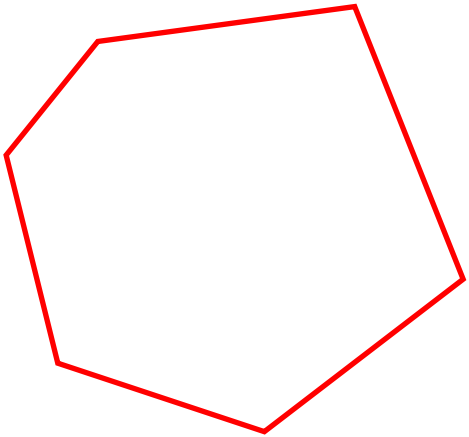


# Un algorithme incrémental

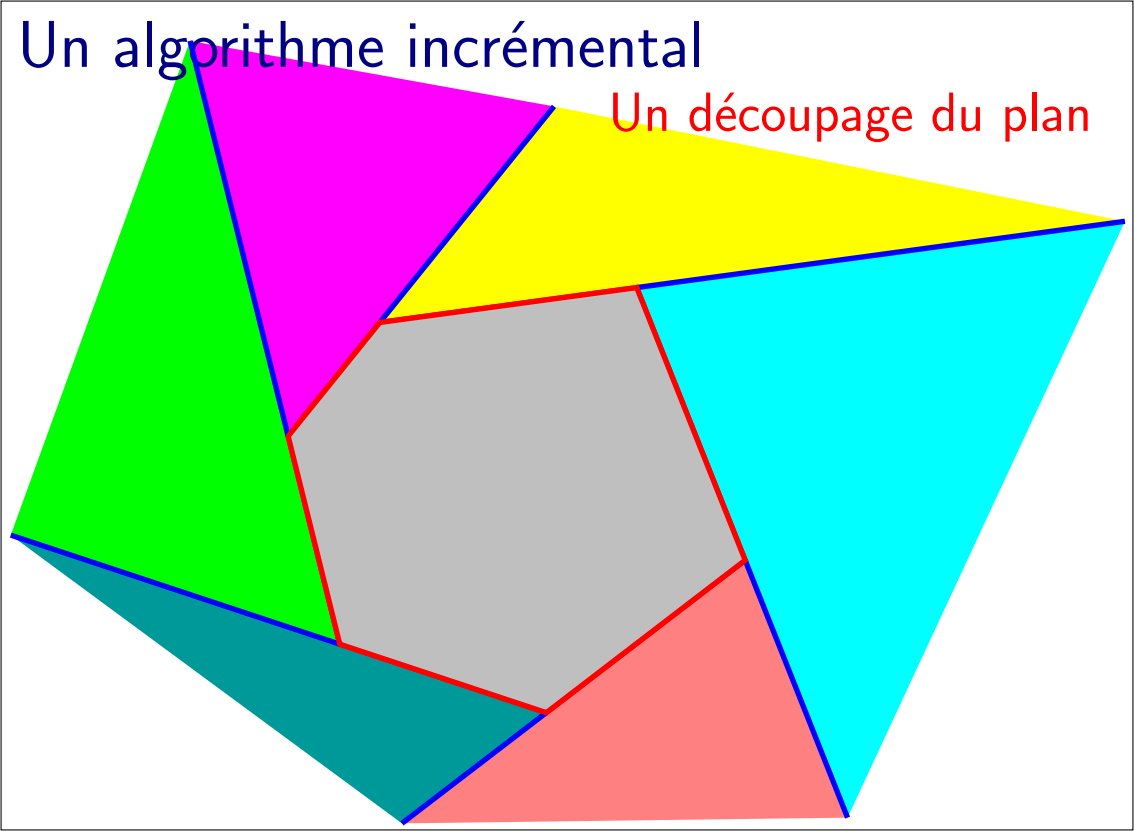


# Un algorithme incrémental

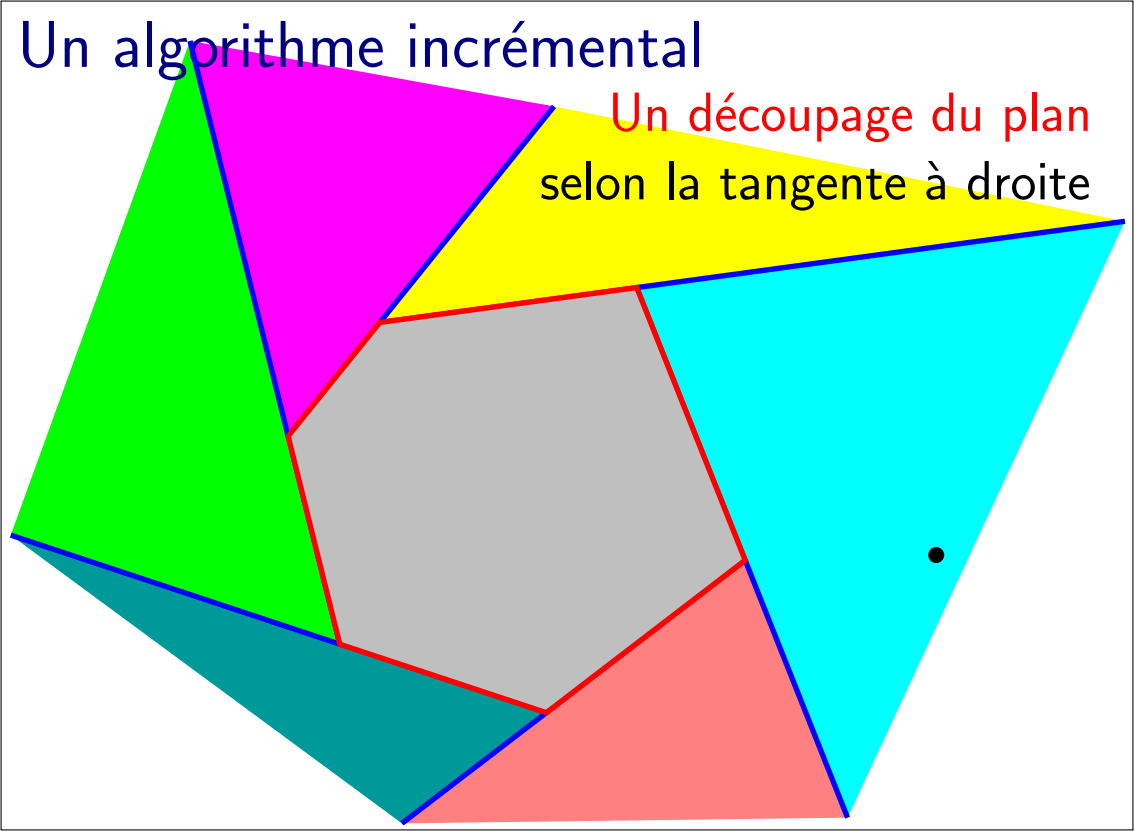
## Un découpage du plan



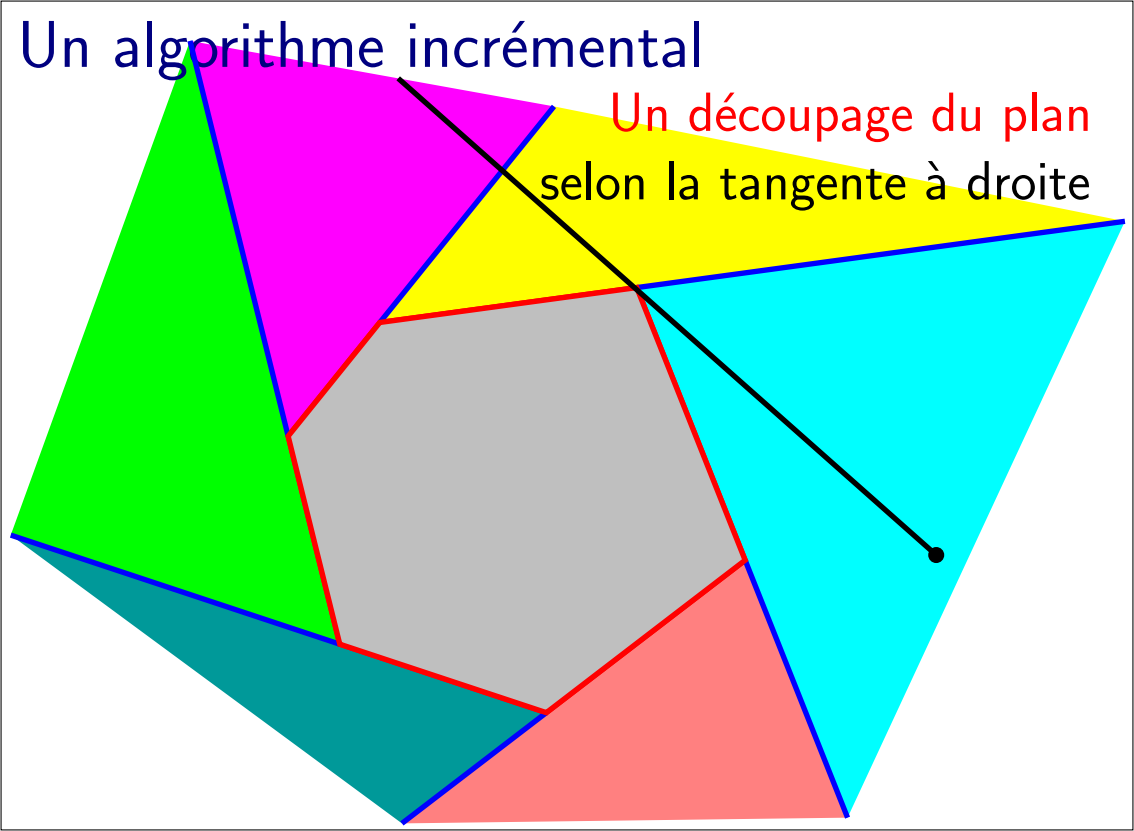
# Un algorithme incrémental



# Un algorithme incrémental

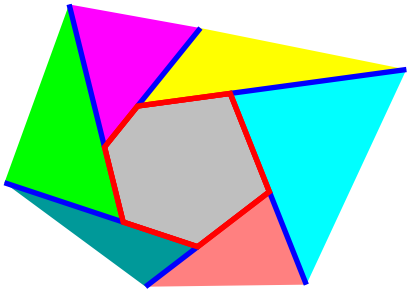


# Un algorithme incrémental



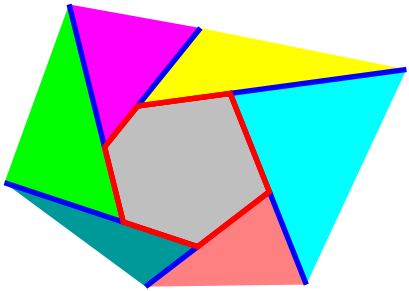
# Un algorithme incrémental

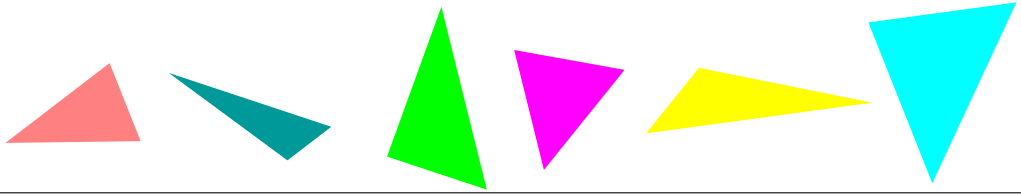
## Un découpage du plan



# Un algorithme incrémental

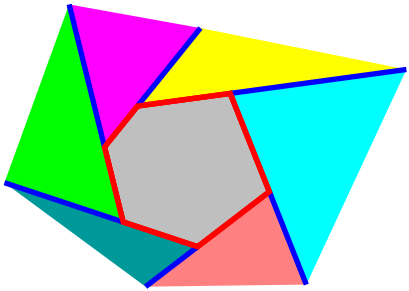
## Un découpage du plan

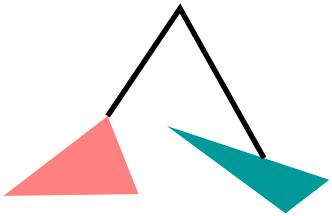


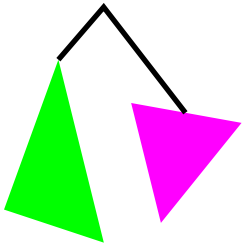


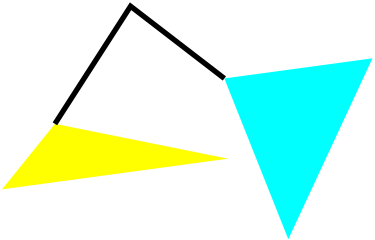
# Un algorithme incrémental

## Un découpage du plan



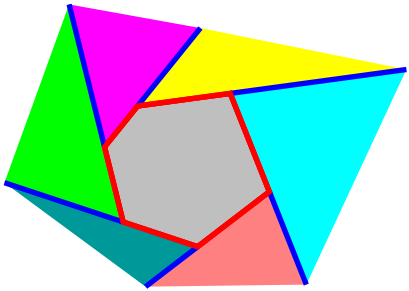


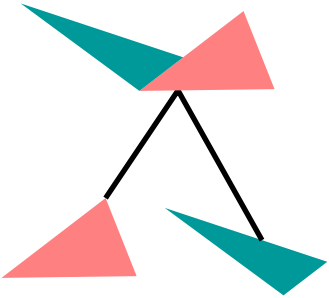


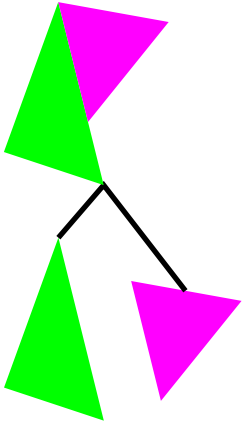


# Un algorithme incrémental

## Un découpage du plan

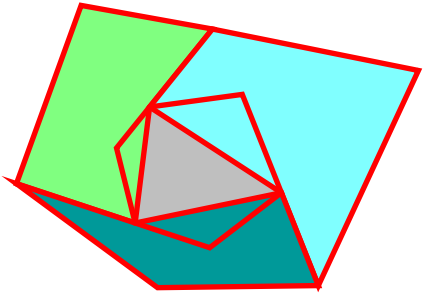


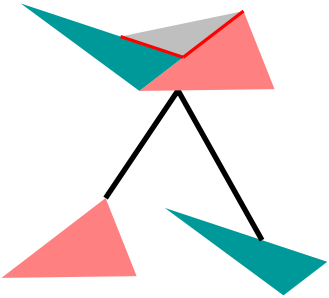


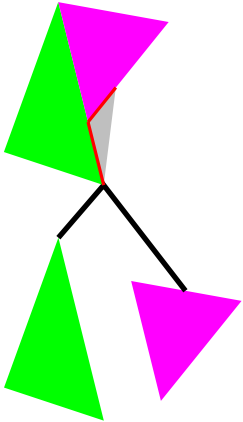


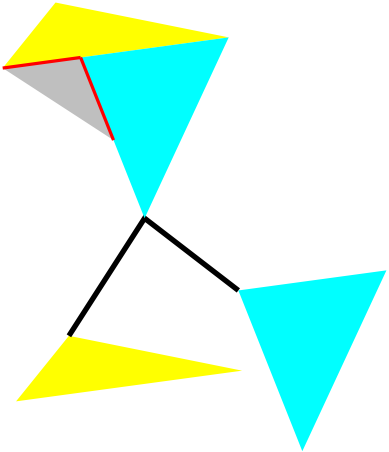
# Un algorithme incrémental

## Un découpage du plan



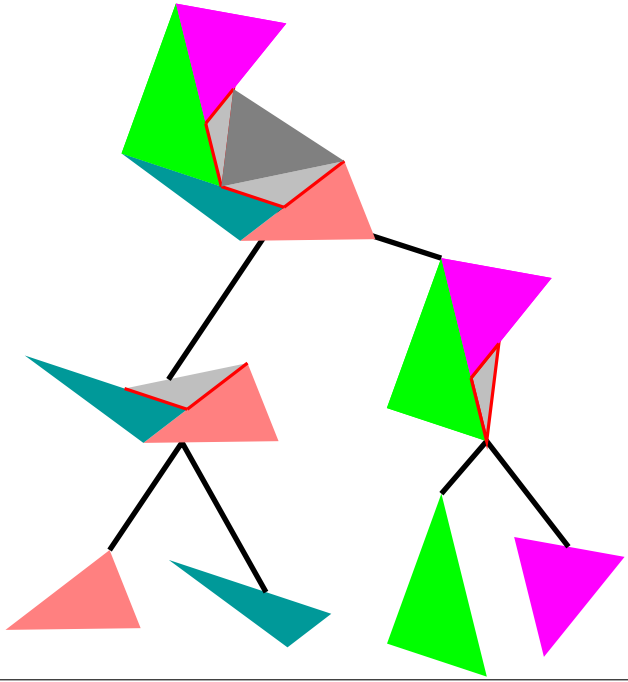


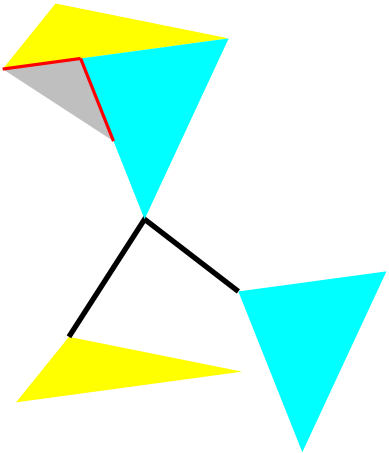


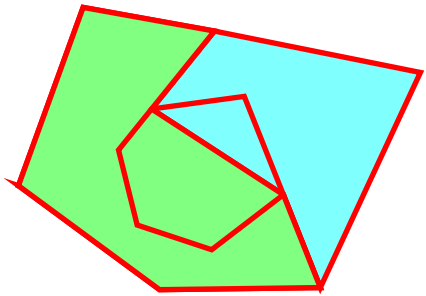


# Un algorithme incrémental

## Un découpage du plan

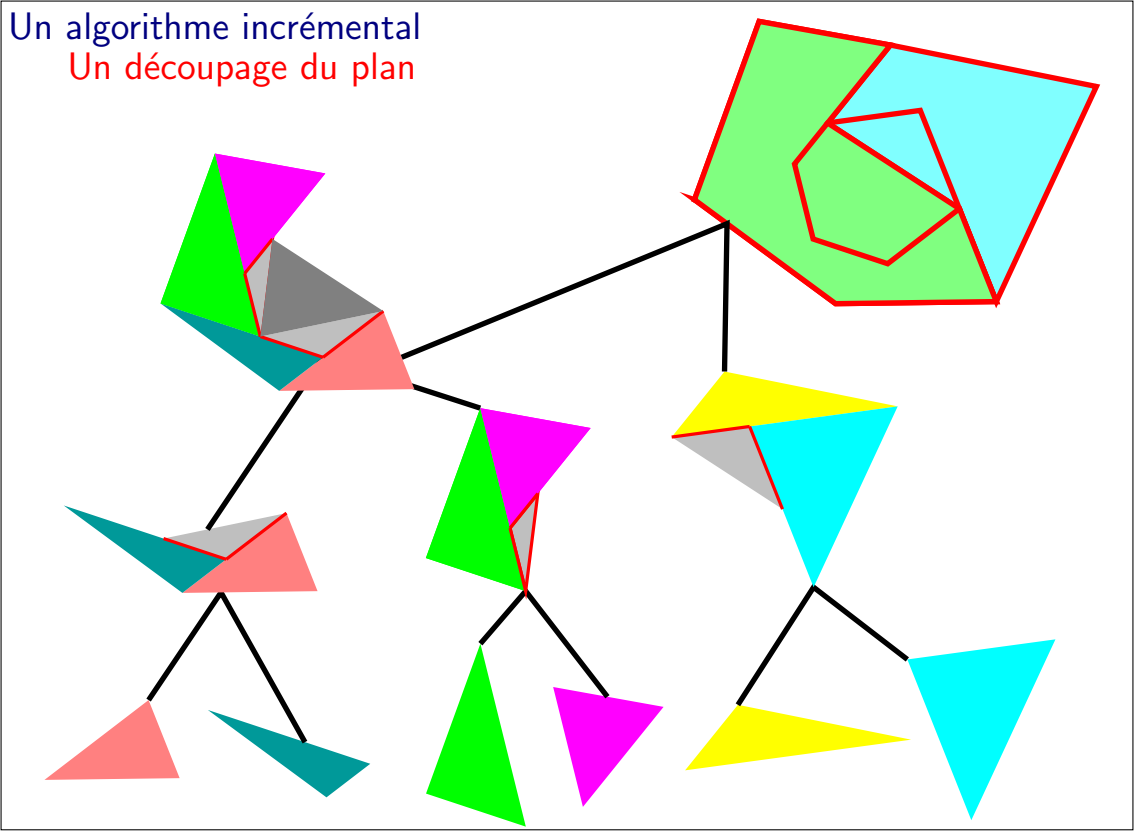




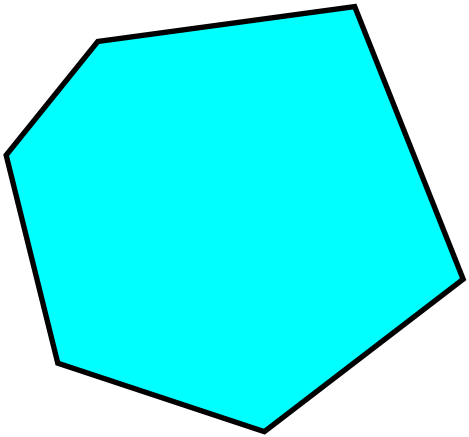


# Un algorithme incrémental

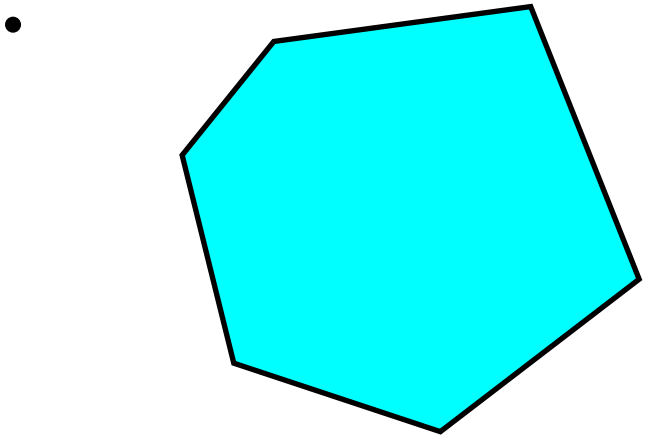
## Un découpage du plan



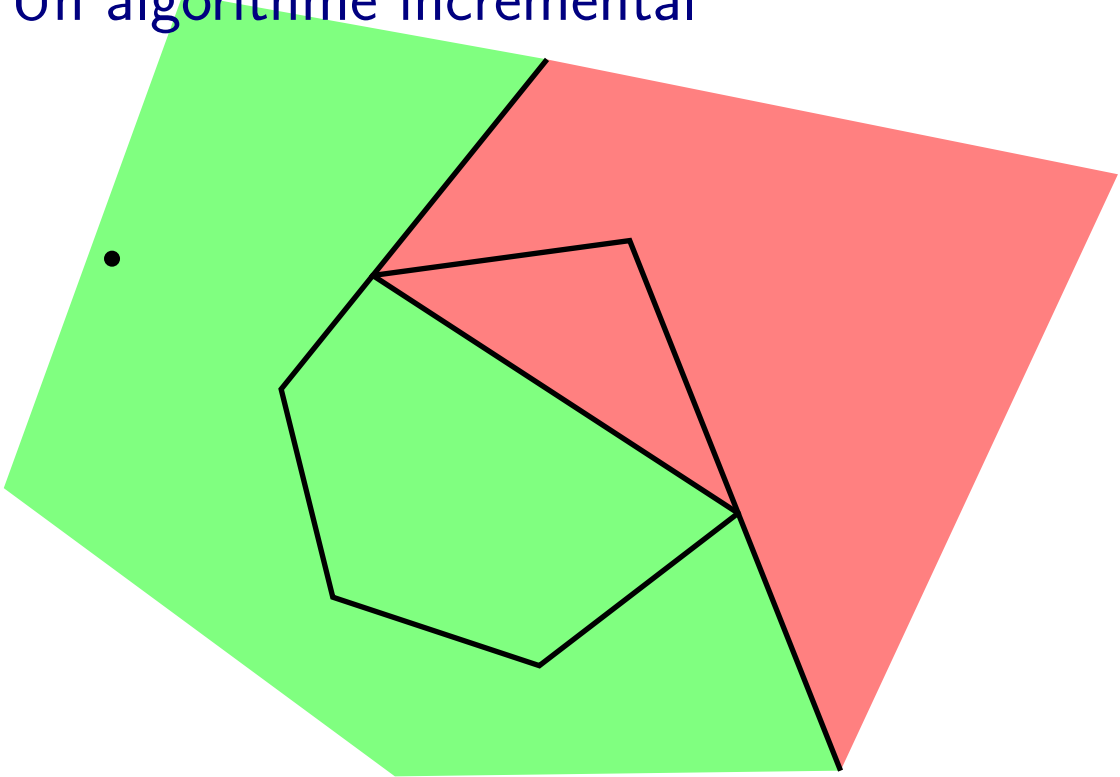
# Un algorithme incrémental



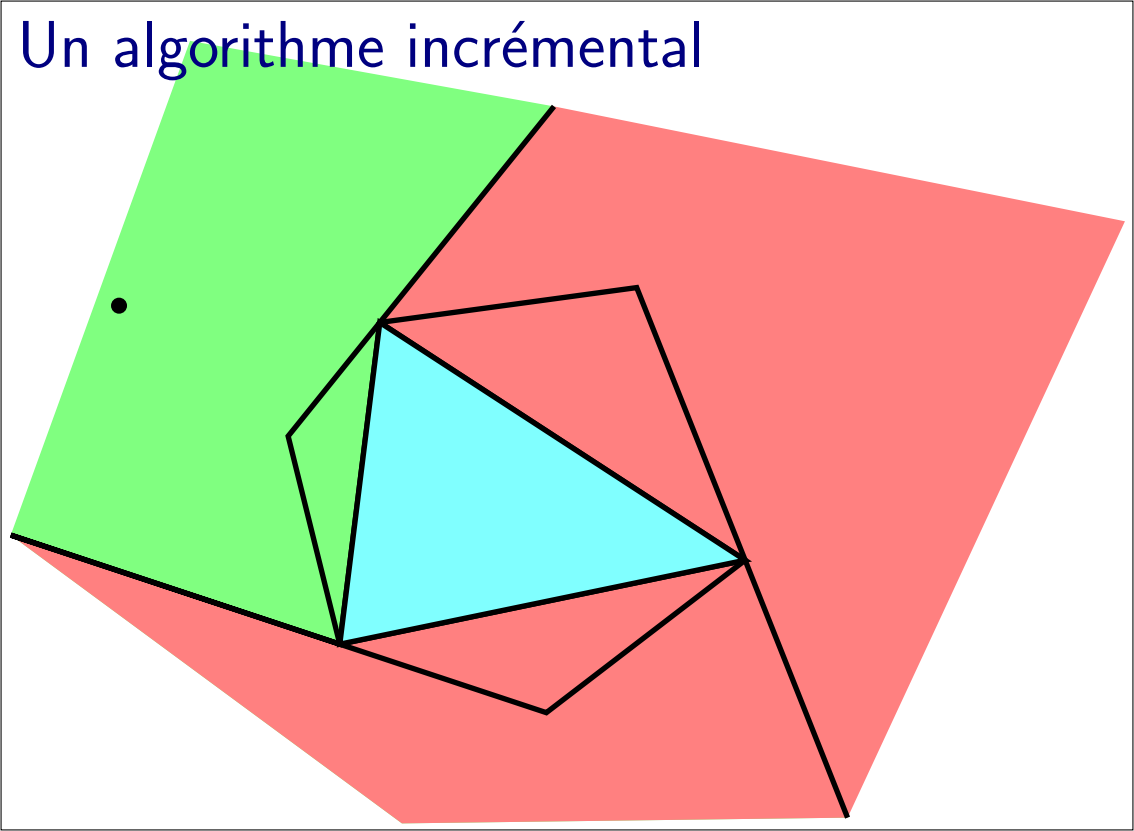
# Un algorithme incrémental



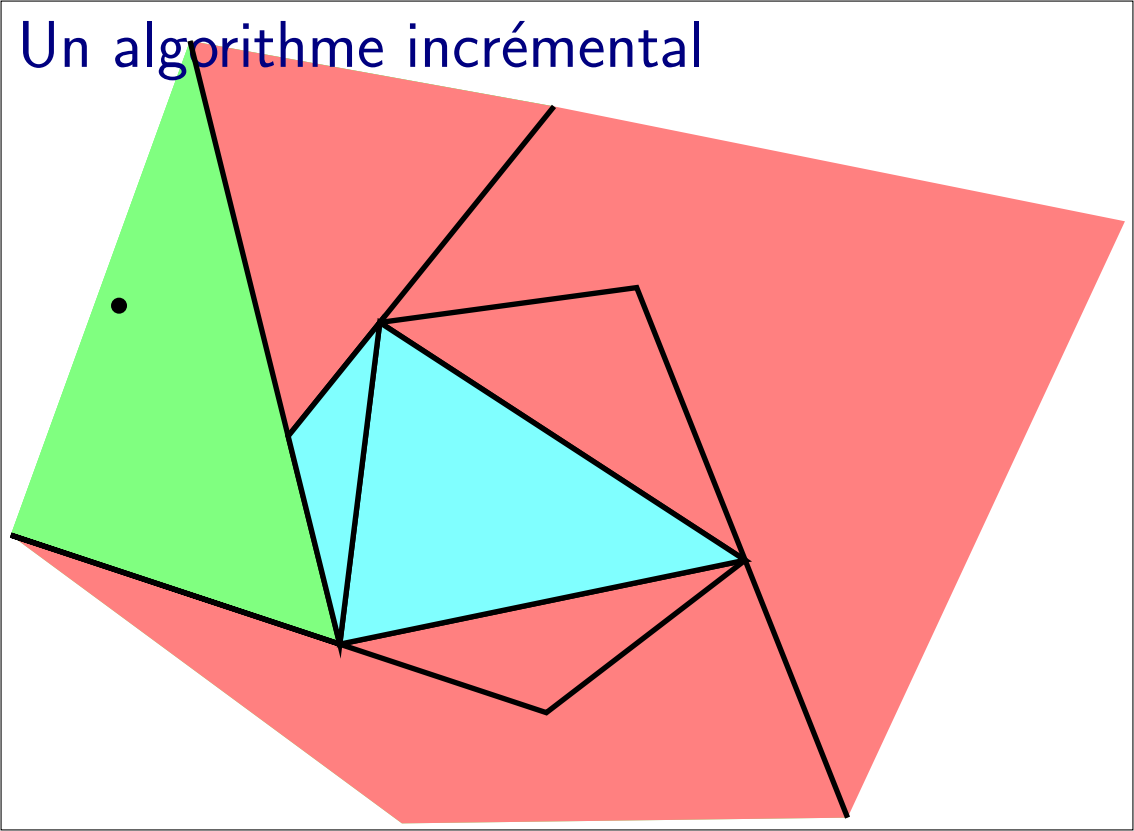
# Un algorithme incrémental



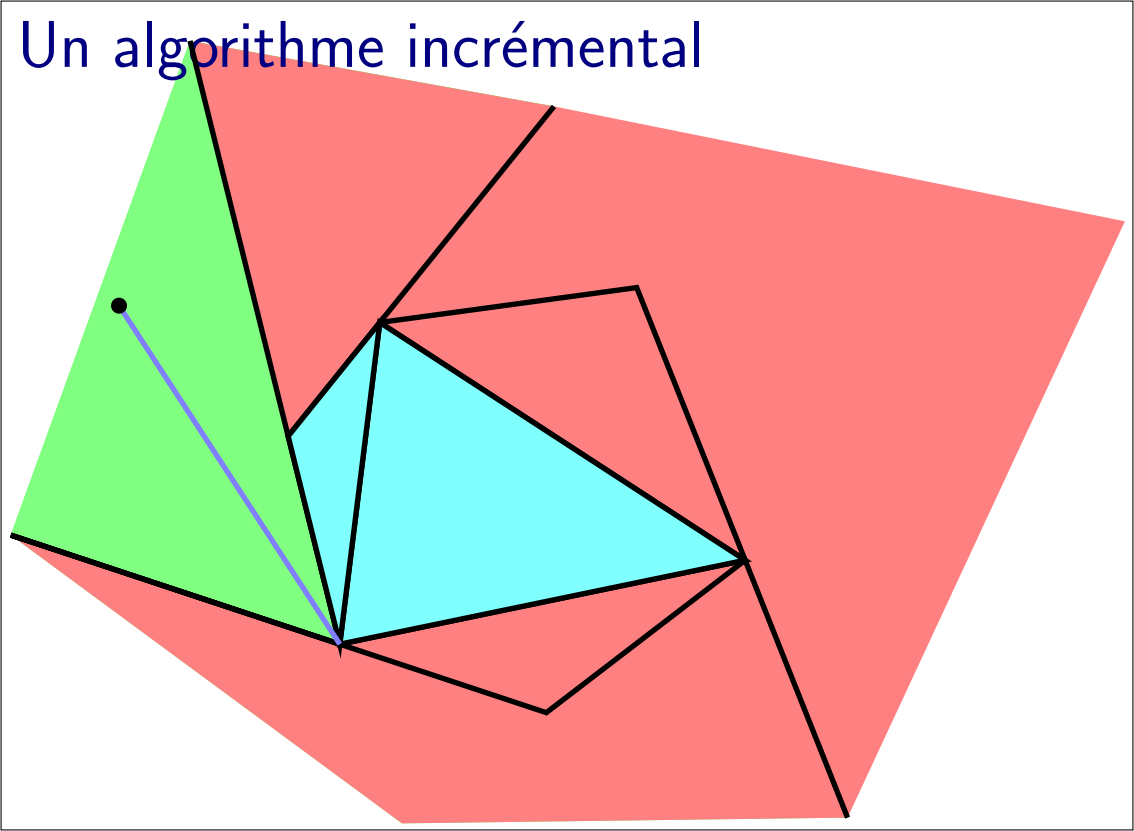
# Un algorithme incrémental



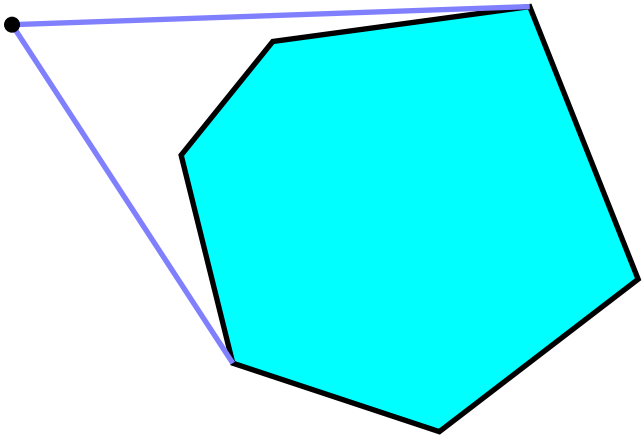
# Un algorithme incrémental



# Un algorithme incrémental

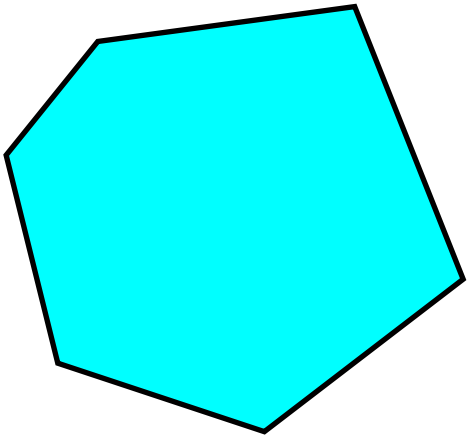


# Un algorithme incrémental

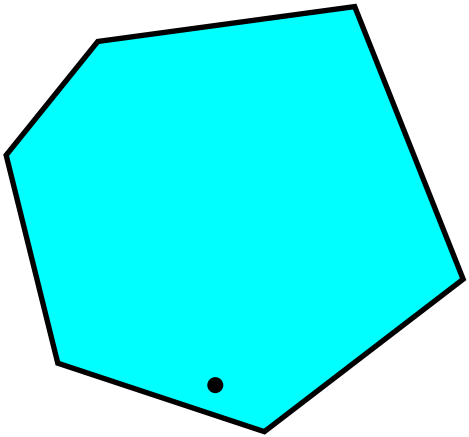


symétriquement

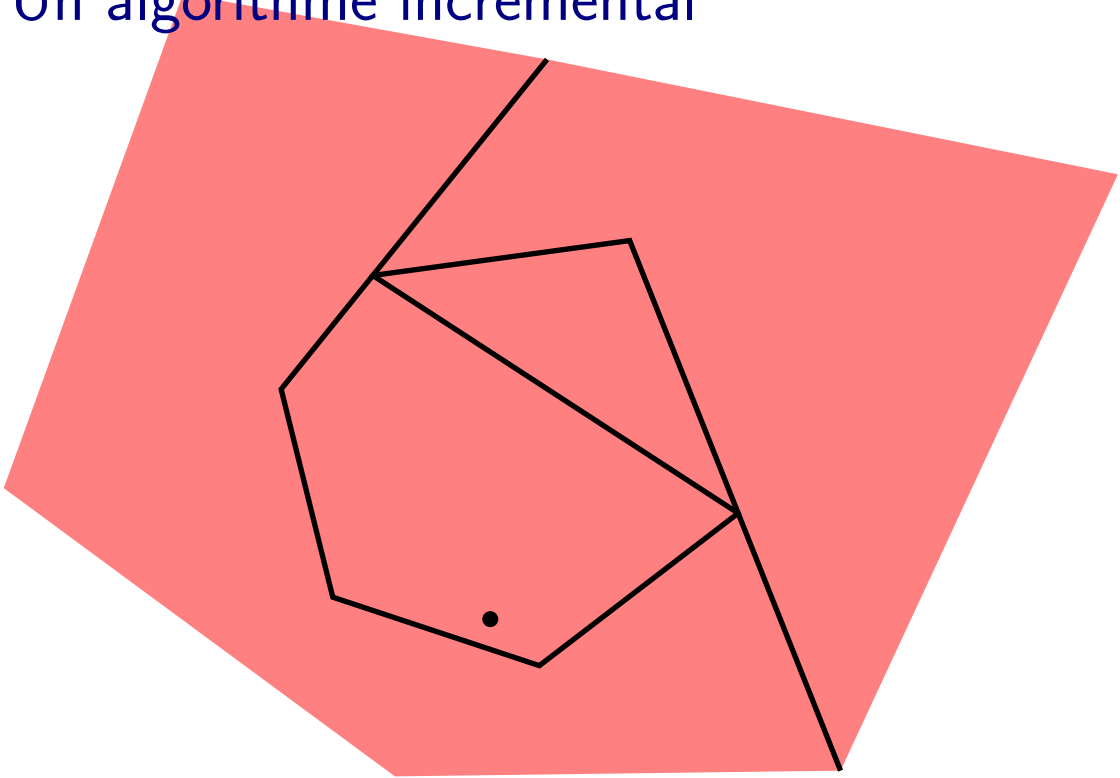
# Un algorithme incrémental



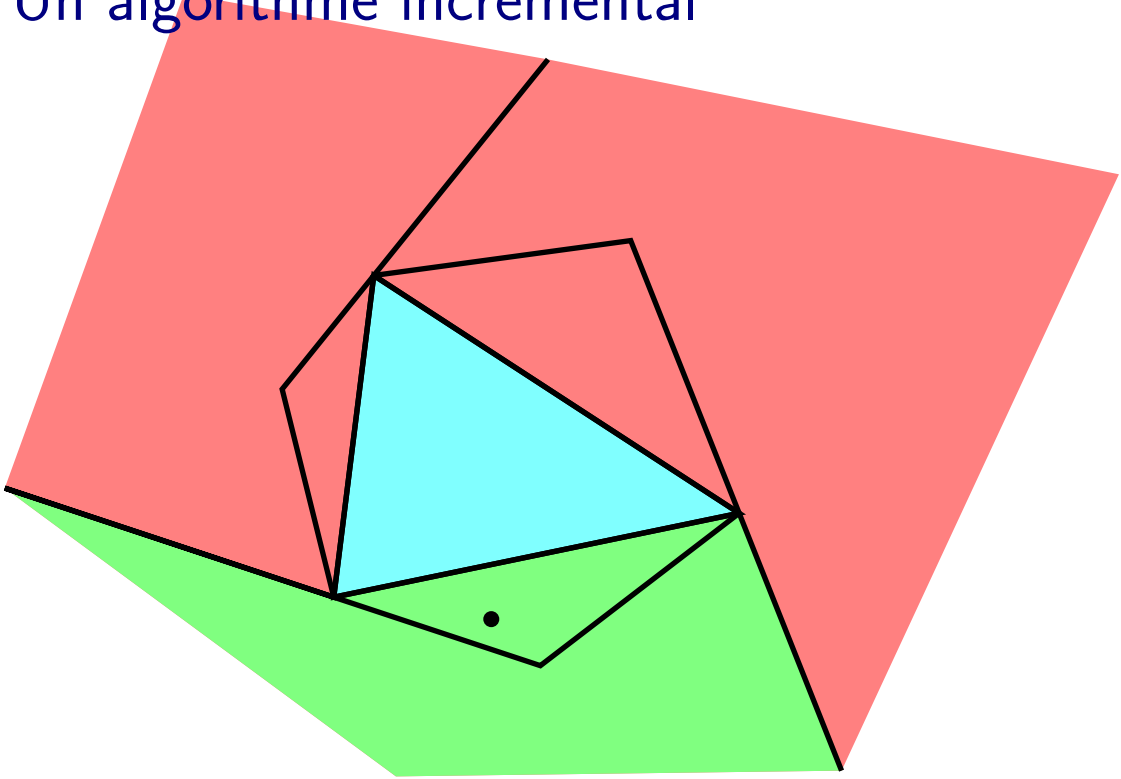
# Un algorithme incrémental



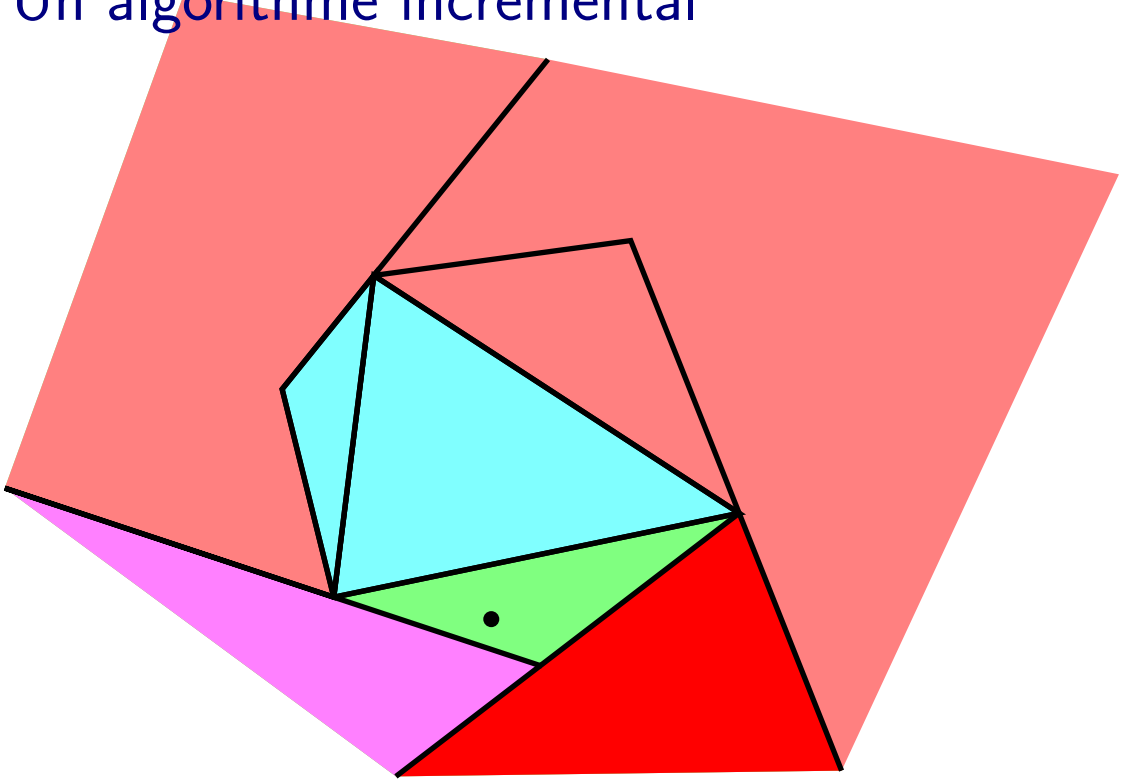
# Un algorithme incrémental



# Un algorithme incrémental

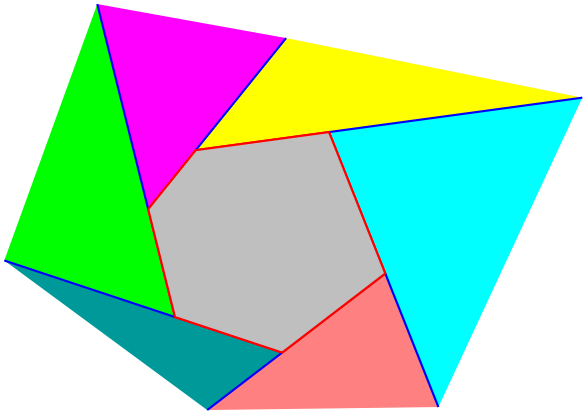


# Un algorithme incrémental



# Un algorithme incrémental

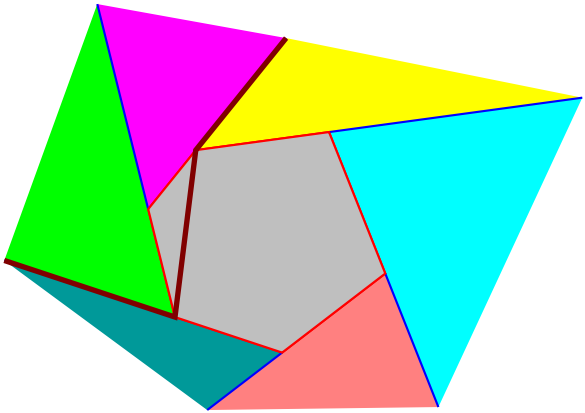
## Complexité



# Un algorithme incrémental

## Complexité

Complexité d'un nœud

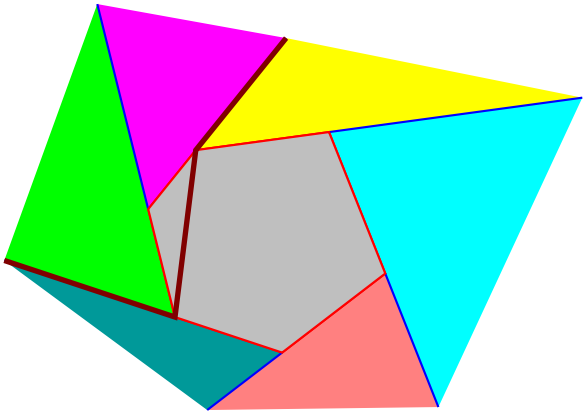


# Un algorithme incrémental

## Complexité

Complexité d'un nœud

$O(1)$



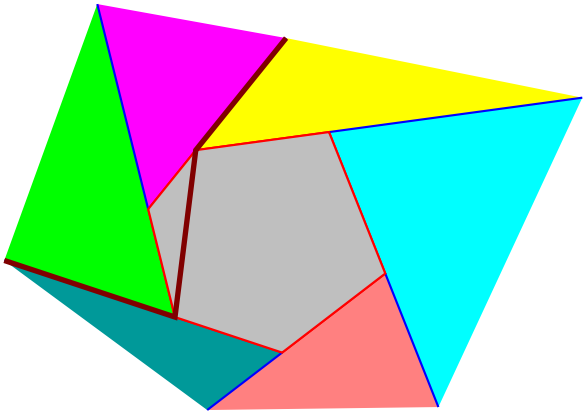
# Un algorithme incrémental

## Complexité

Complexité d'un nœud

$O(1)$

Arbre équilibré

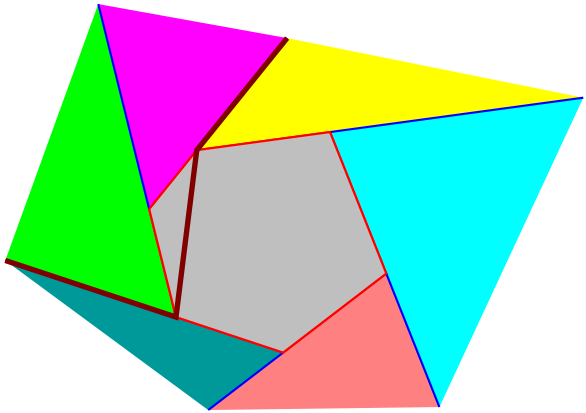


# Un algorithme incrémental

## Complexité

Complexité d'un nœud

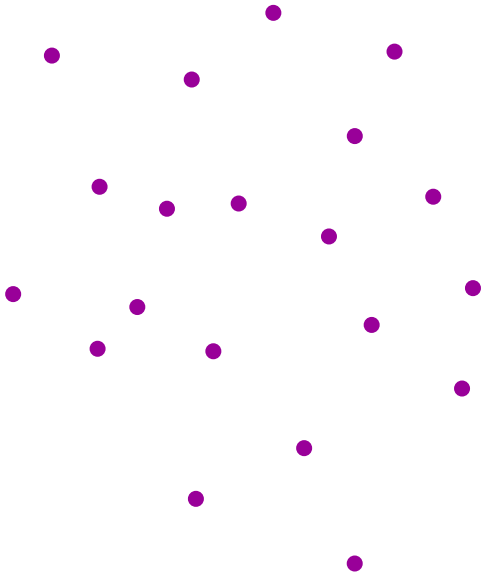
$O(1)$



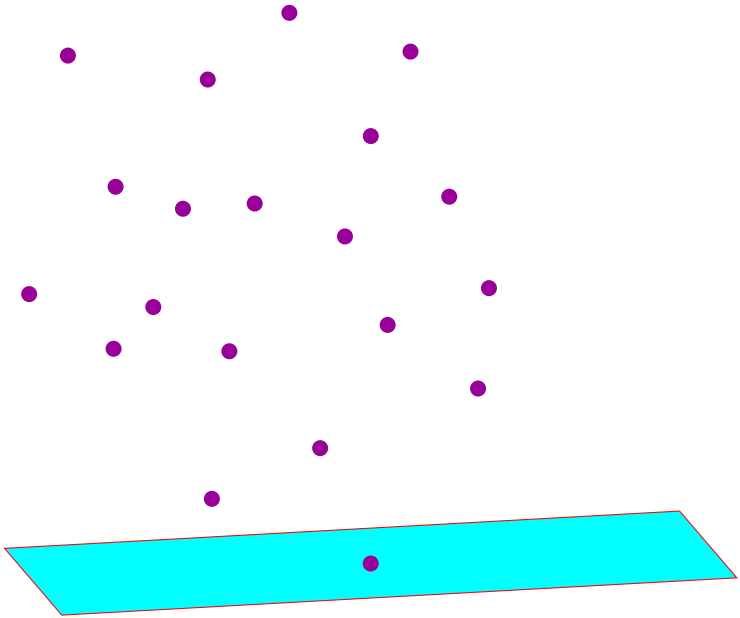
Arbre équilibré

$O(\log n)$ par insertion

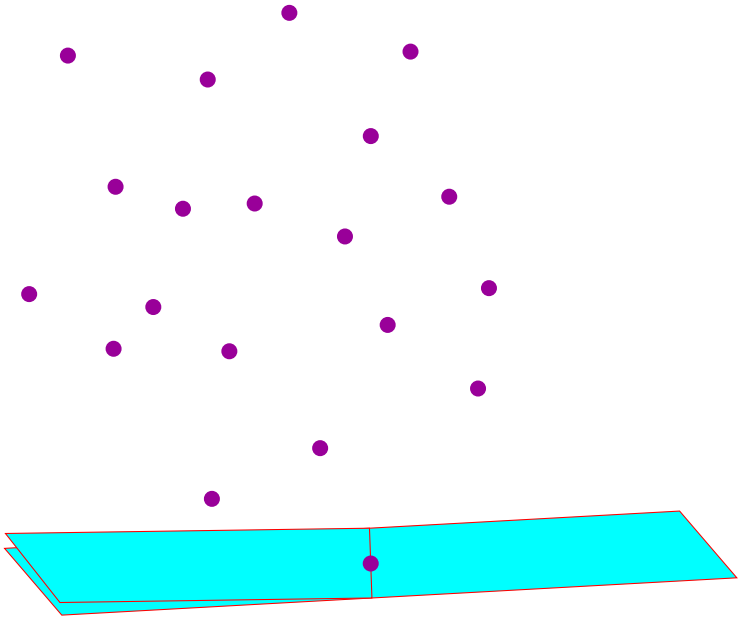
# L'algorithme du paquet cadeau



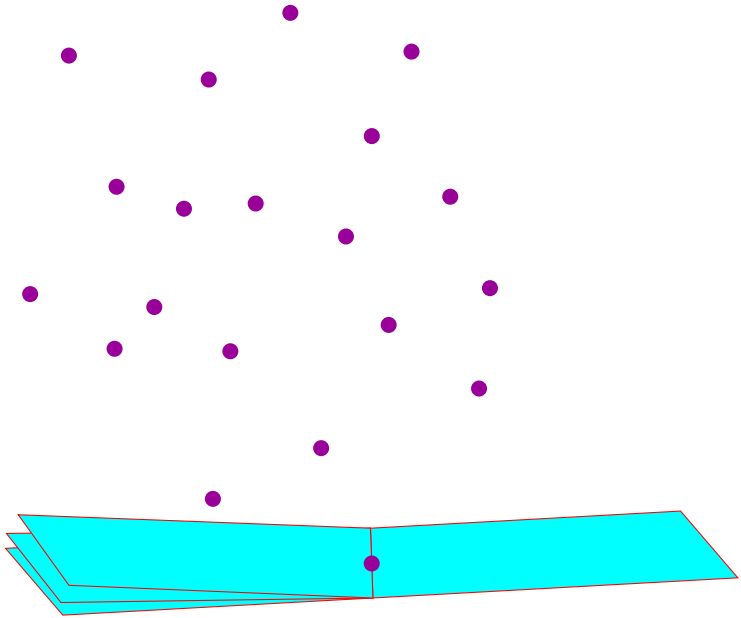
# L'algorithme du paquet cadeau



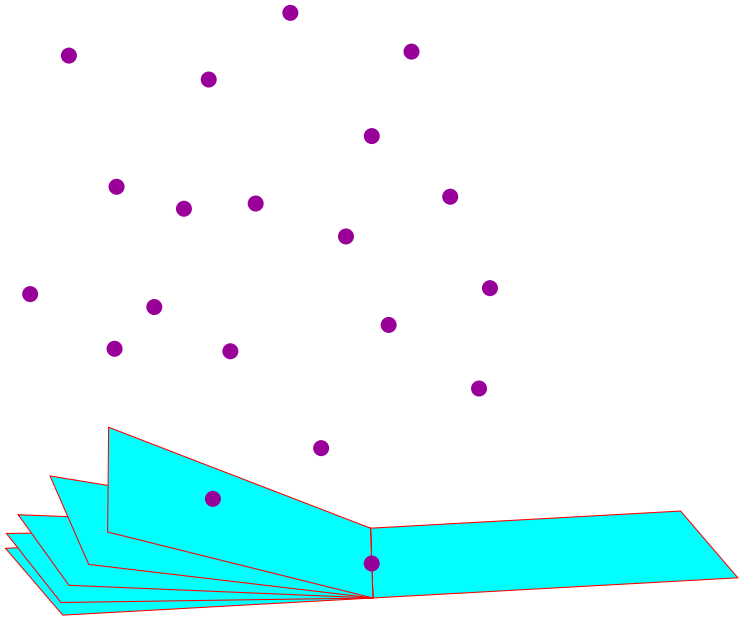
# L'algorithme du paquet cadeau



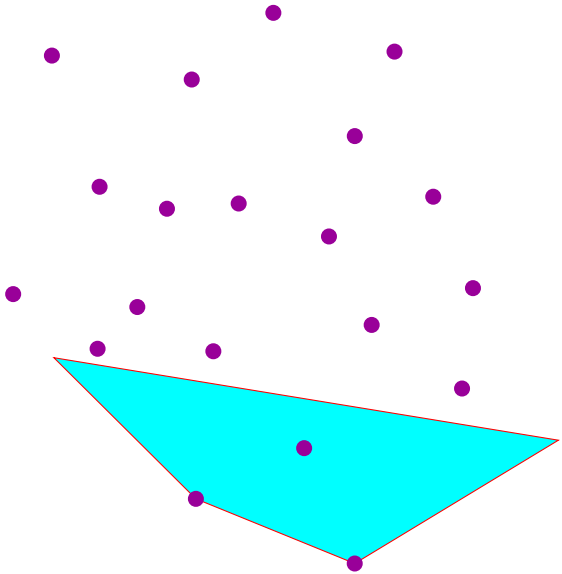
# L'algorithme du paquet cadeau



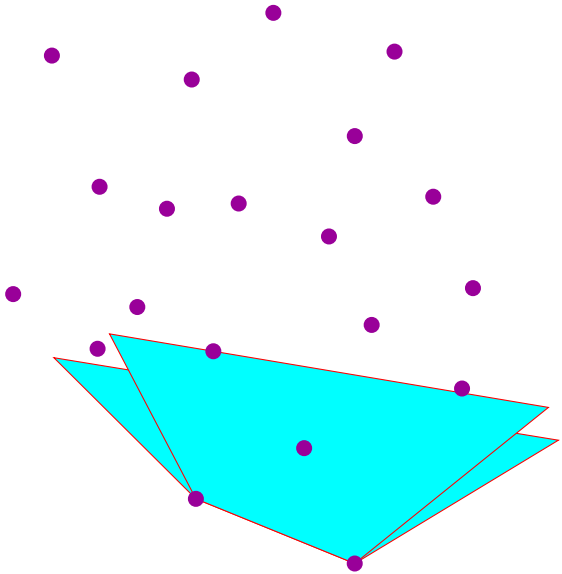
# L'algorithme du paquet cadeau



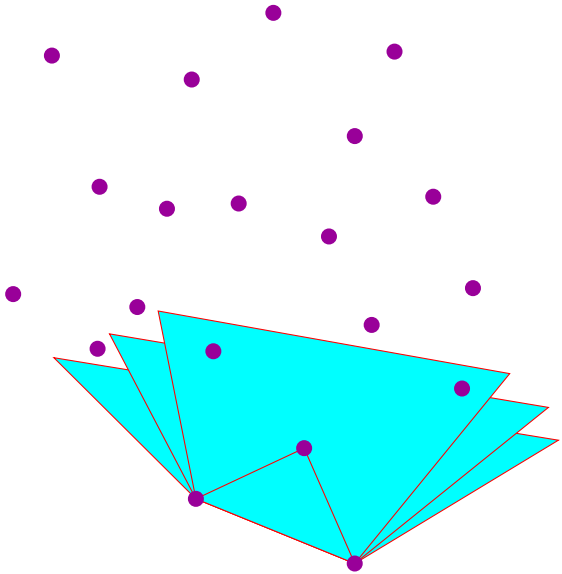
# L'algorithme du paquet cadeau



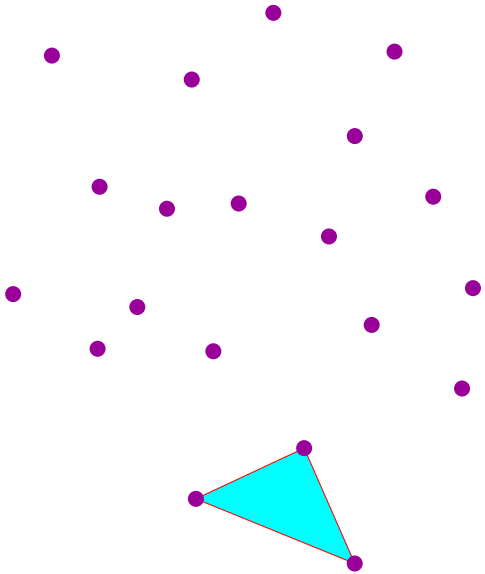
# L'algorithme du paquet cadeau



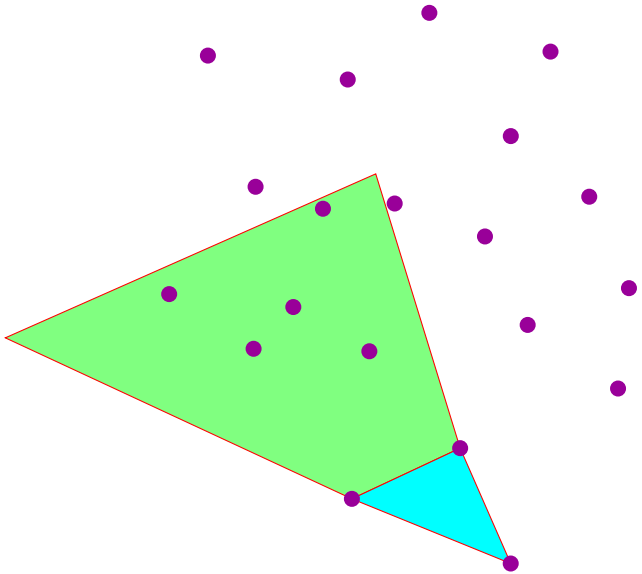
# L'algorithme du paquet cadeau



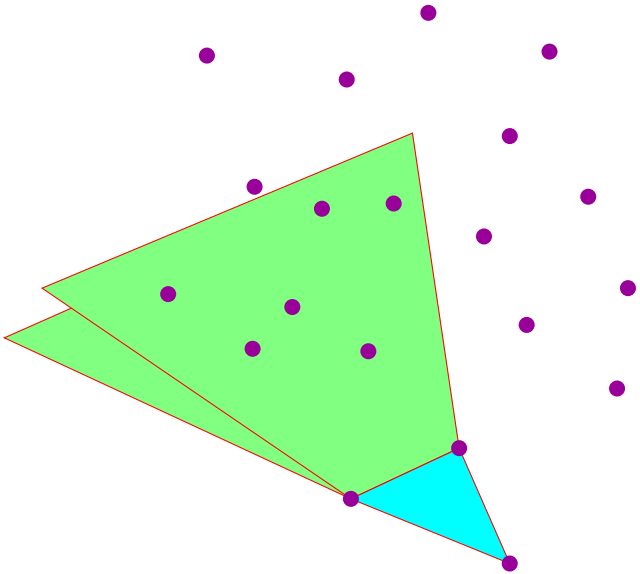
# L'algorithme du paquet cadeau



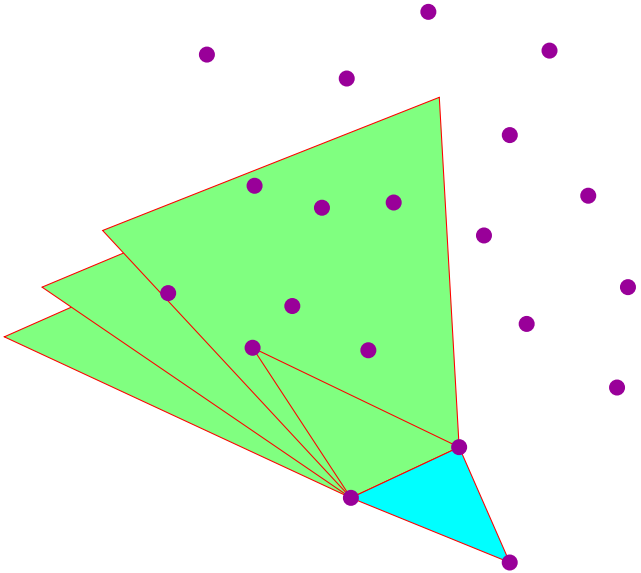
# L'algorithme du paquet cadeau



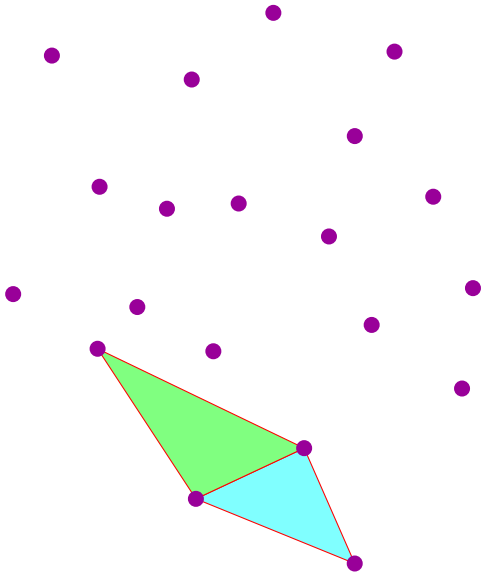
# L'algorithme du paquet cadeau



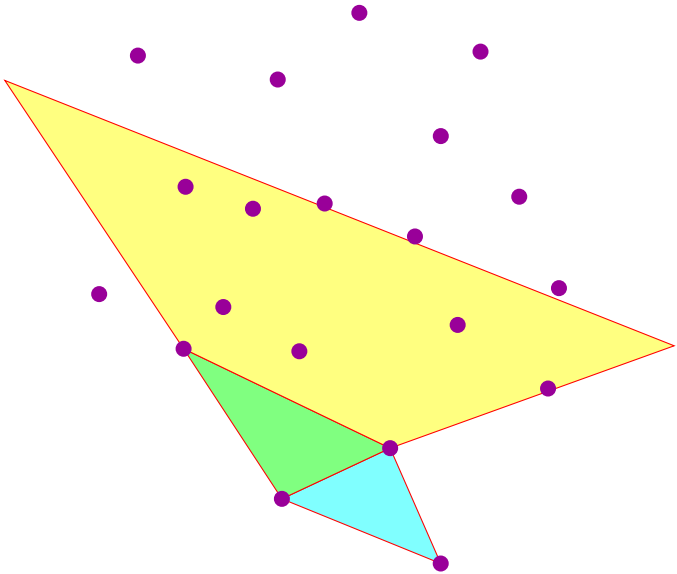
# L'algorithme du paquet cadeau



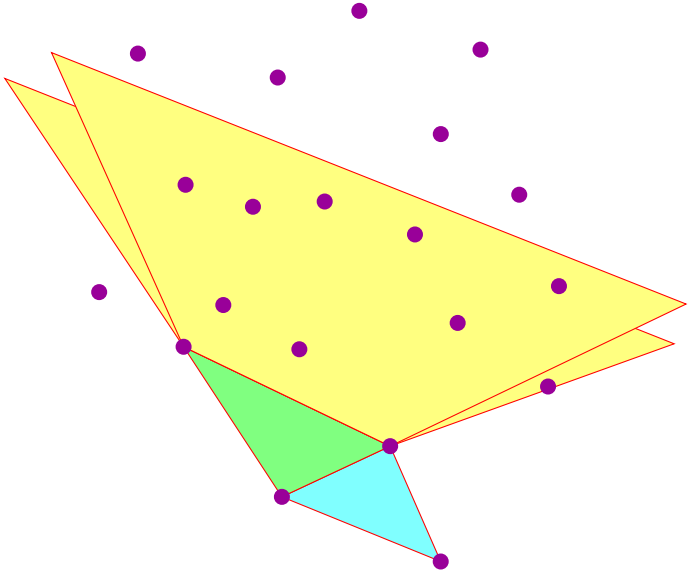
# L'algorithme du paquet cadeau



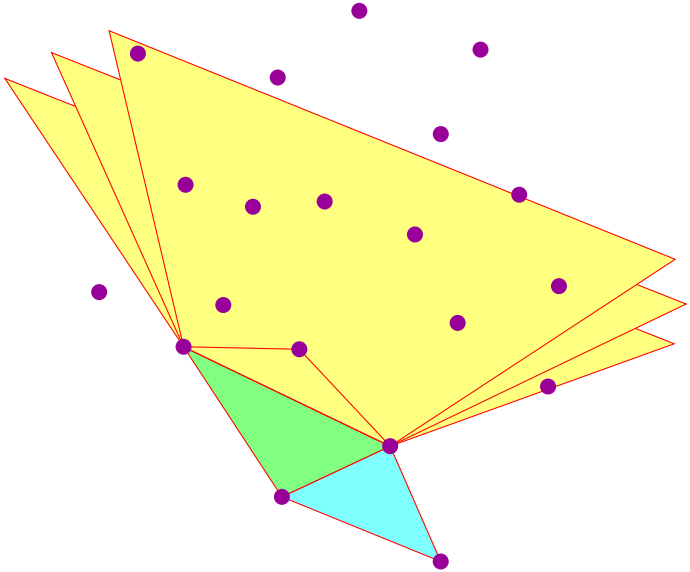
# L'algorithme du paquet cadeau



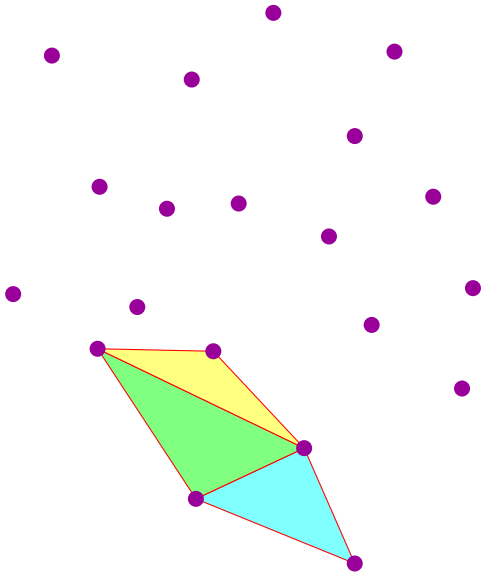
# L'algorithme du paquet cadeau



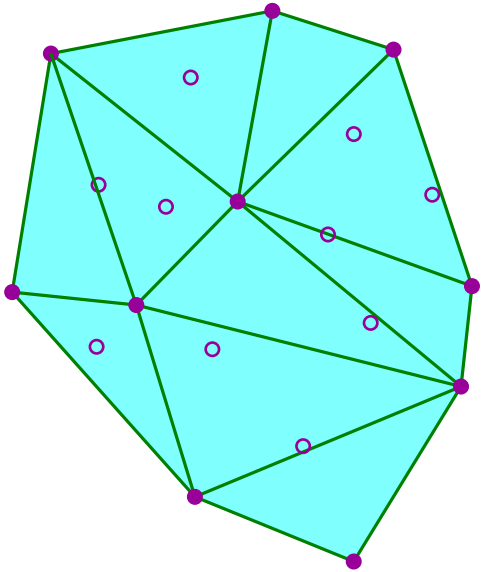
# L'algorithme du paquet cadeau



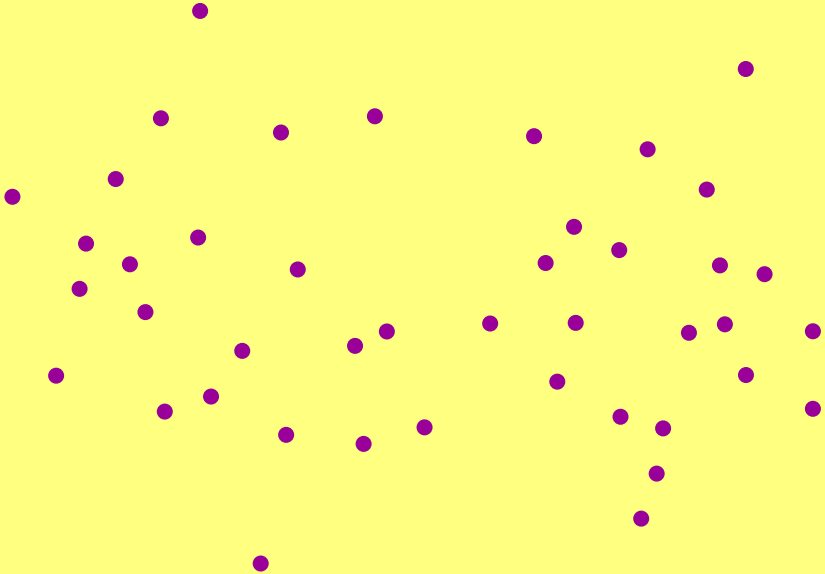
# L'algorithme du paquet cadeau



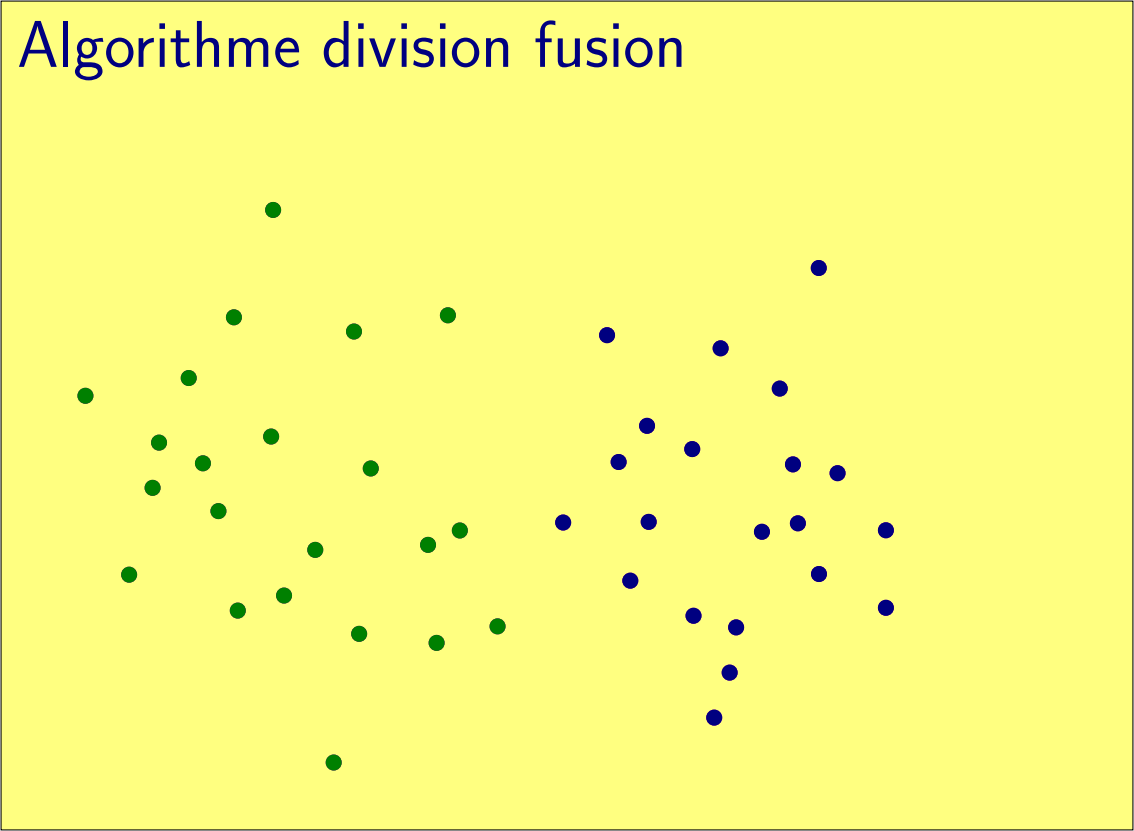
# L'algorithme du paquet cadeau



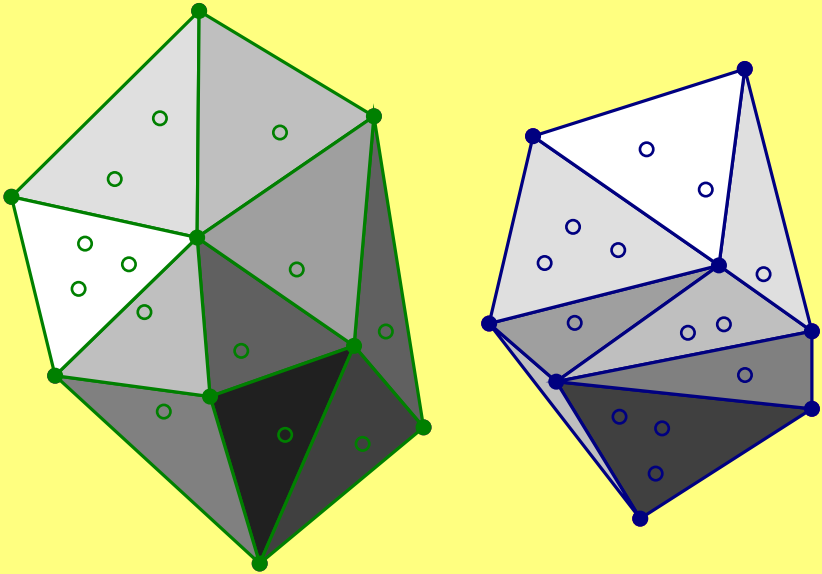
# Algorithme division fusion



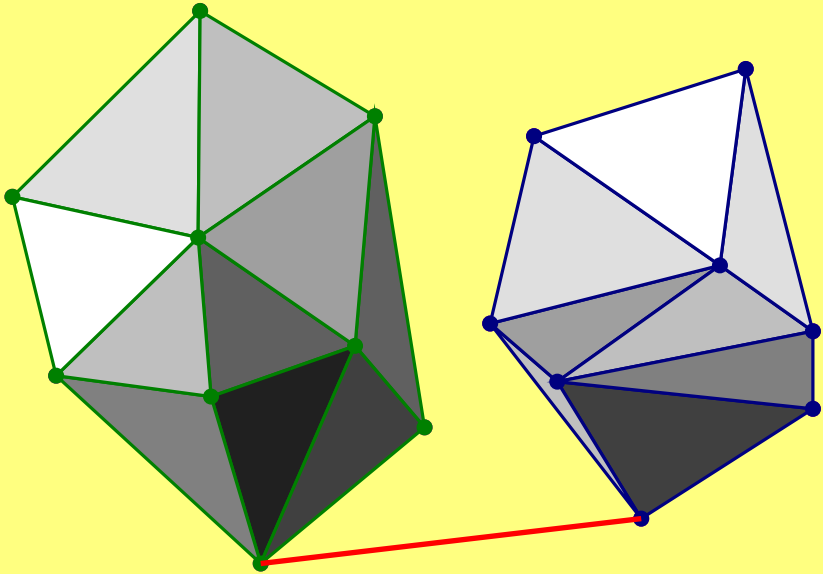
# Algorithme division fusion



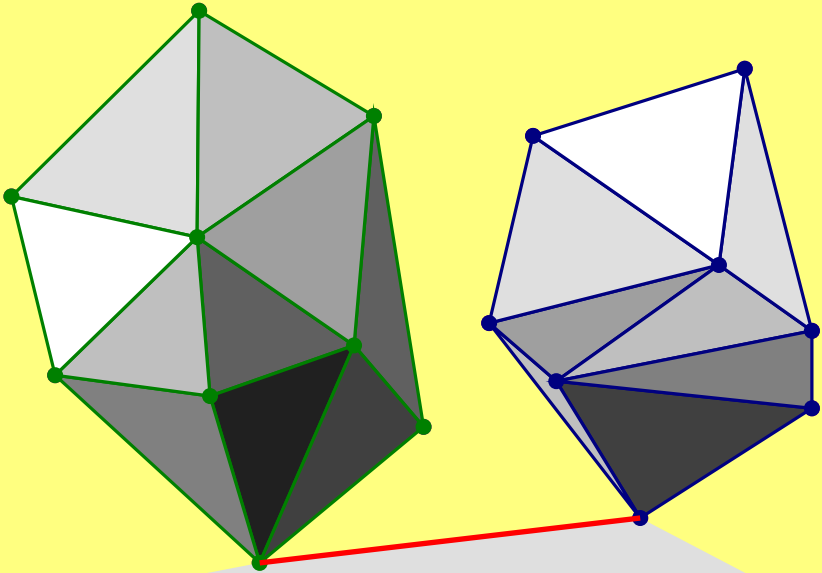
# Algorithme division fusion



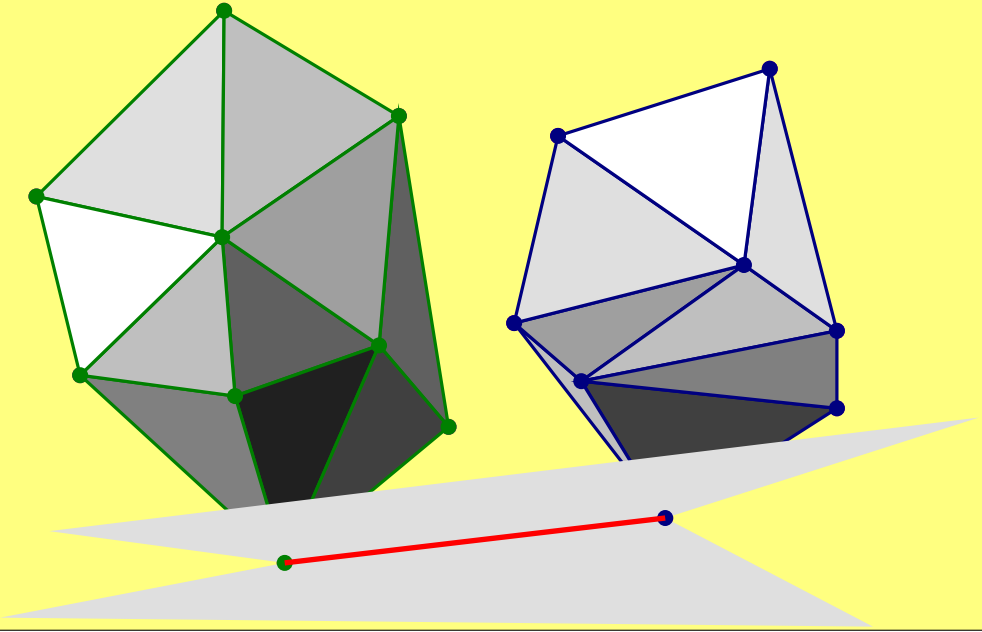
# Algorithme division fusion



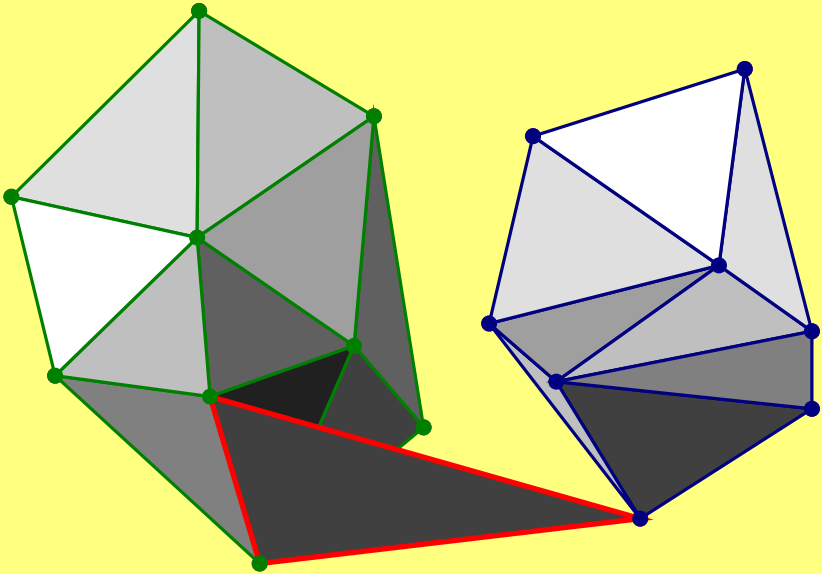
# Algorithme division fusion



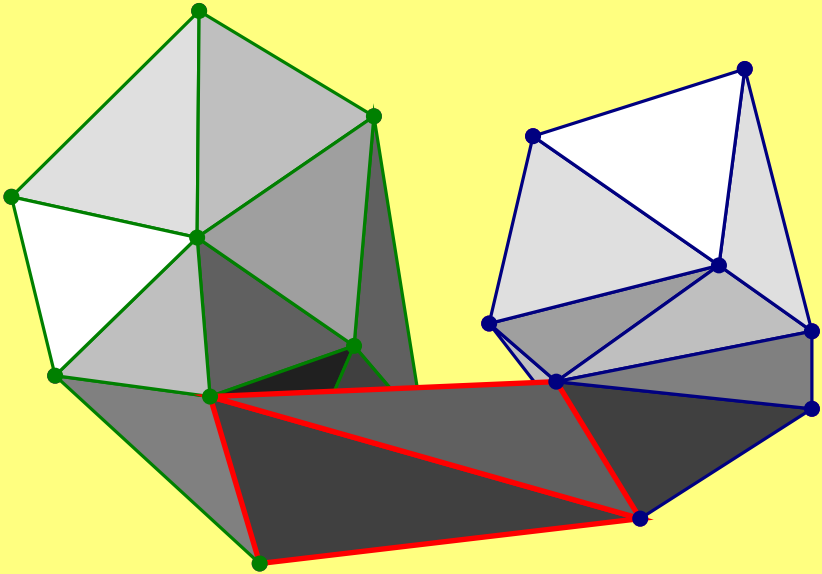
# Algorithme division fusion



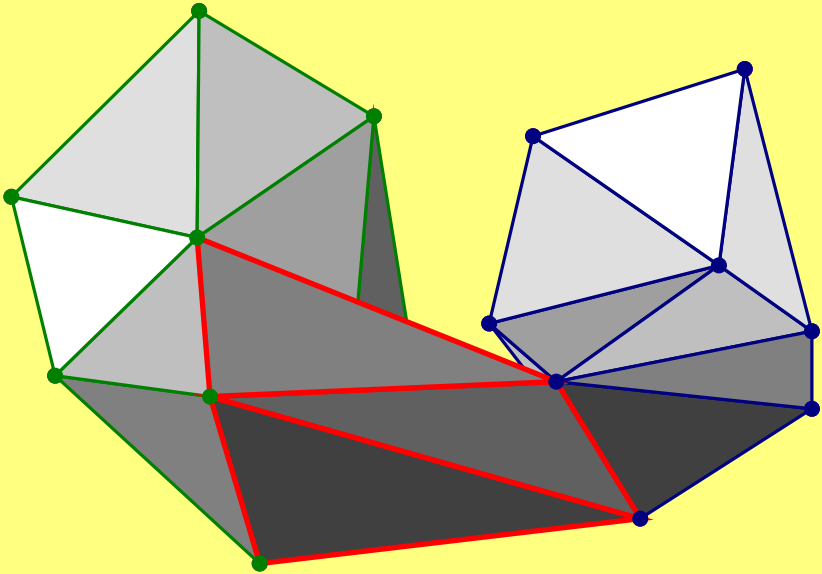
# Algorithme division fusion



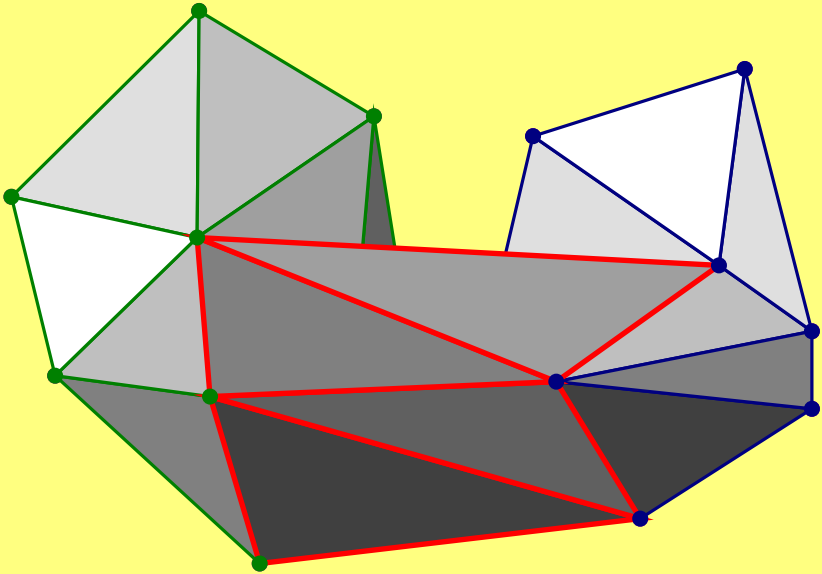
# Algorithme division fusion



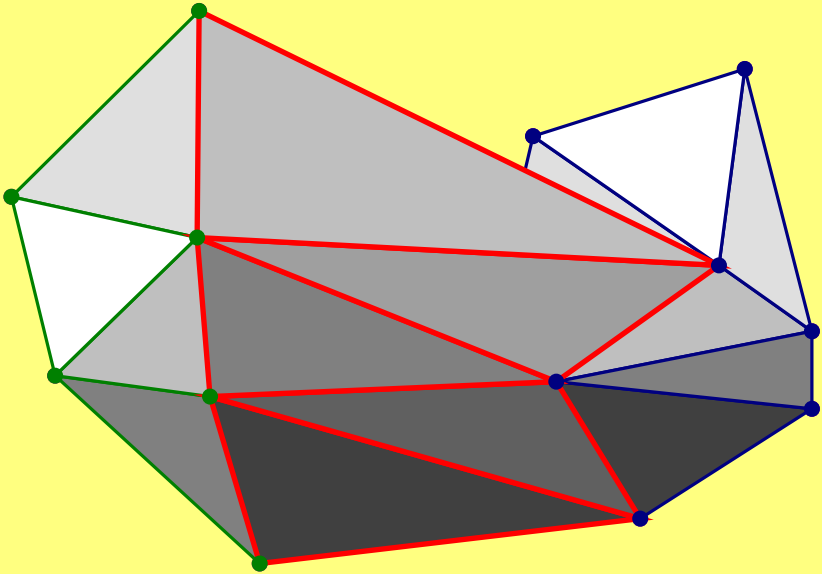
# Algorithme division fusion



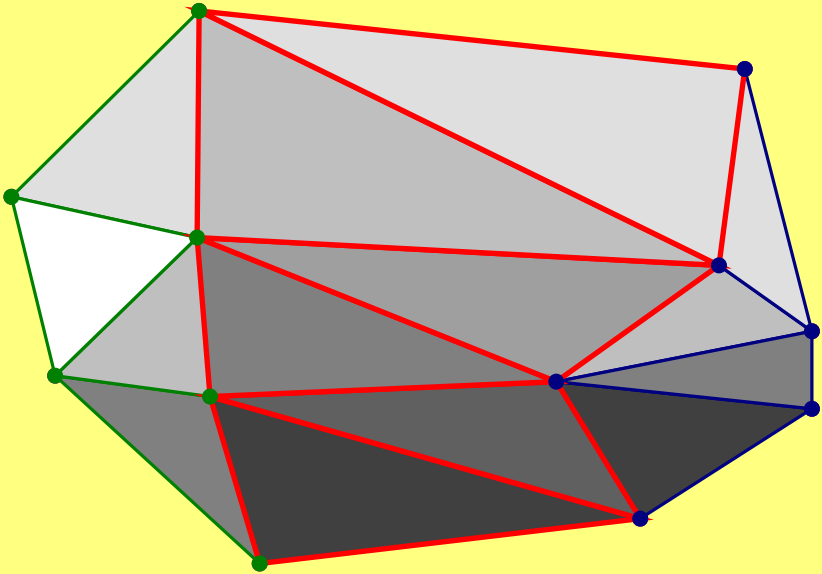
# Algorithme division fusion



# Algorithme division fusion

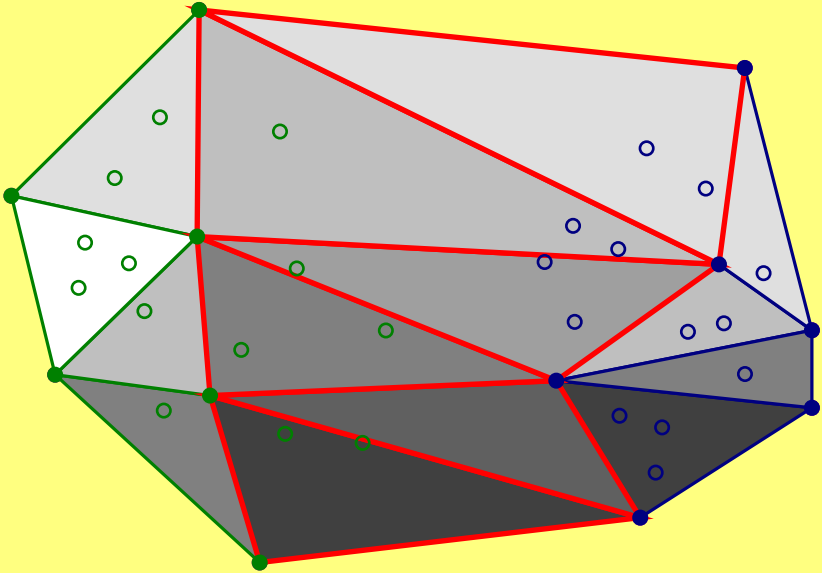


# Algorithme division fusion



# Algorithme division fusion

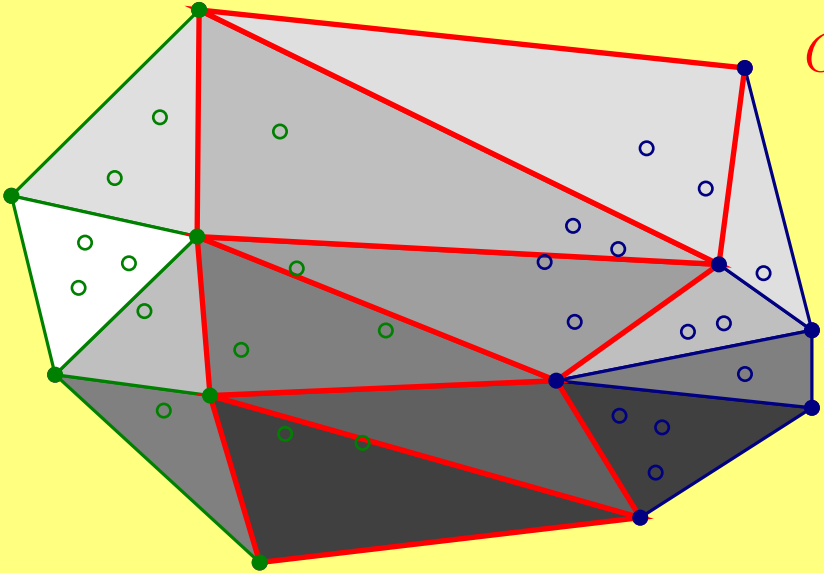
$O(n)$

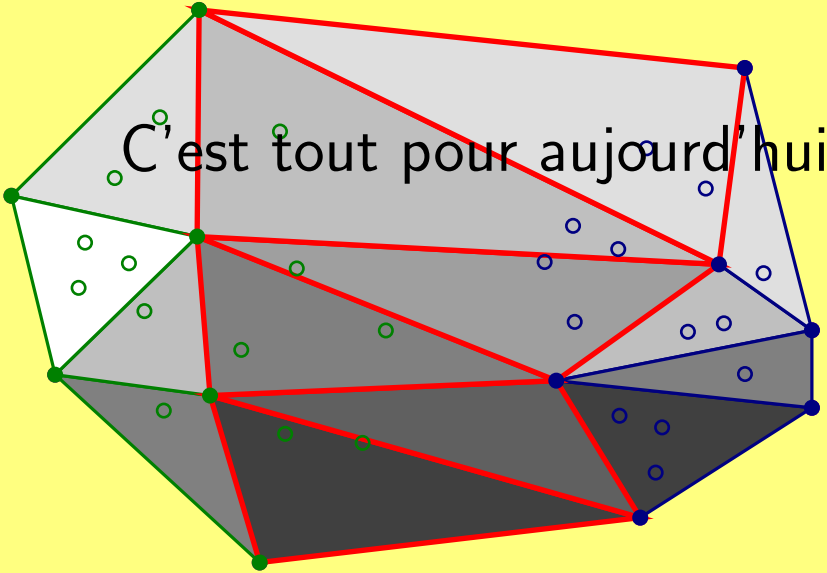


# Algorithme division fusion

$O(n)$

$O(n \log n)$




C'est tout pour aujourd'hui