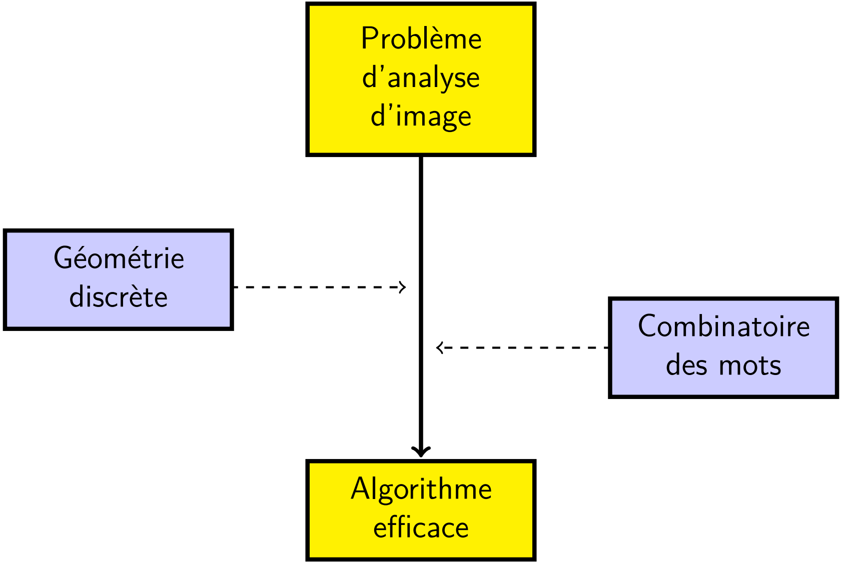
Le MLP : polygone de longueur minimale

Xavier Provençal

GTI320

27 juillet 2020, Montréal

Objectif


Problème d'analyse d'image
Géométrie discrète
Combinatoire des mots
Algorithme efficace

Combinatoire des mots

- Sous-branche des mathématiques discrètes.
- On considère un ensemble de symboles appelé **alphabet** où chaque symbole est appelé une **lettre** et on forme une suite de symboles, appellé un **mot**.
- Selon la manière dont un mot est construit, on étudie les propriétés combinatoires des motifs qui le composent : dénombrement, distribution, périodicité, etc.

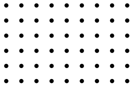
Combinatoire des mots

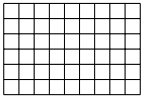
- Sous-branche des mathématiques discrètes.
- On considère un ensemble de symboles appelé **alphabet** où chaque symbole est appelé une **lettre** et on forme une suite de symboles, appellé un **mot**.
- Selon la manière dont un mot est construit, on étudie les propriétés combinatoires des motifs qui le composent : dénombrement, distribution, périodicité, etc.
- Exemple :
  - Alphabet $\{0, 1\}$, le mot de Thue-Morse est : $t_0 = 0,\ t_{n+1} = t_n\overline{t_n}$,

$$
\begin{array}{lcl}
t_0 & = & 0 \\
t_1 & = & 0\,1 \\
t_2 & = & 01\,10 \\
t_3 & = & 0110\,1001 \\
 & \vdots & \\
t_\infty & = & 0110100110010110100101100110100110010110011010010110100110010
\end{array}
$$

Géométrie discrète (digitale)

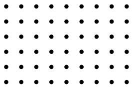
- Objectif : faire de la géométrie dans $\mathbb{Z}^d$.
- Visualisation :

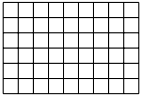




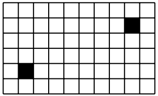
Géométrie discrète (digitale)

- Objectif : faire de la géométrie dans $\mathbb{Z}^d$.
- Visualisation :





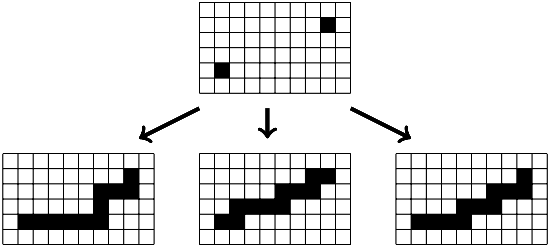
- Le cas des droites



Géométrie discrète (digitale)

- Objectif : faire de la géométrie dans $\mathbb{Z}^d$.
- Visualisation :

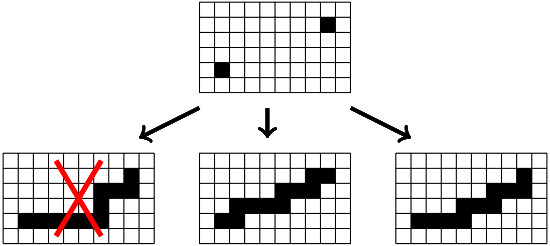
- Le cas des droites



Géométrie discrète (digitale)

- Objectif : faire de la géométrie dans $\mathbb{Z}^d$.
- Visualisation :

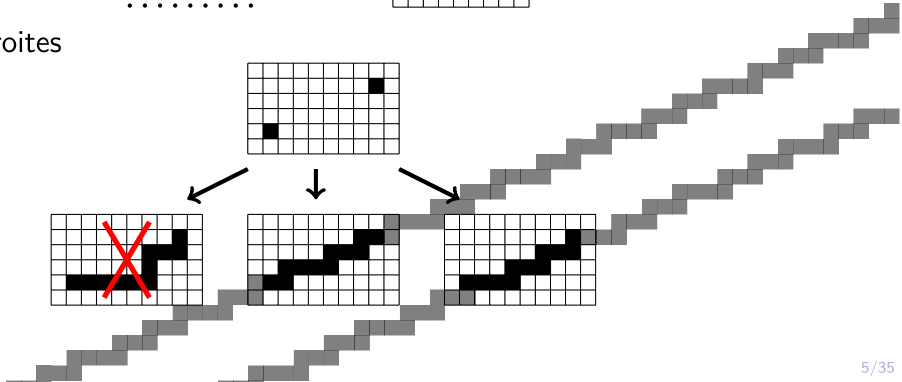
- Le cas des droites



Géométrie discrète (digitale)

- Objectif : faire de la géométrie dans $\mathbb{Z}^d$.
- Visualisation :

- Le cas des droites

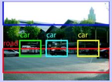


Analyse d'image

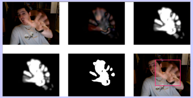
Definition (Wikipedia)

«**Image analysis** is the extraction of meaningful information from images; mainly from digital images by means of digital image processing techniques.»

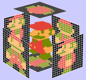
Object recognition



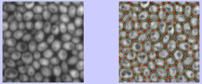
Motion detection



3D reconstruction



Segmentation



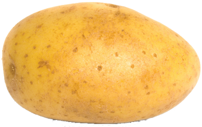
Medical imaging

Optical character recognition

(Sources des images : http://homes.cs.washington.edu/~bcr, http://biodynamics.ucsd.edu/ir/, http://nocreativity.com/blog/webcam-motion-detection-coolness/, http://wikipedia.org, http://www.blogsolute.com, http://www.free-ocr.com)

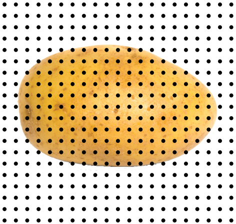
Discrétisation

Discrétisation : $\mathrm{Disc}(P) = P \cap \mathbb{Z}^d$.



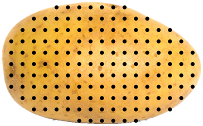
Discrétisation

Discrétisation : $\mathrm{Disc}(P) = P \cap \mathbb{Z}^d$.



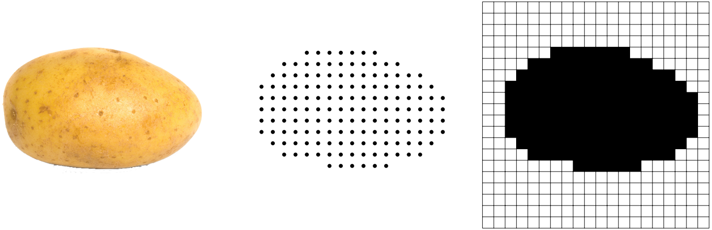
Discrétisation

Discrétisation : $\mathrm{Disc}(P) = P \cap \mathbb{Z}^d$.



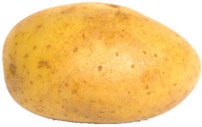
Discrétisation

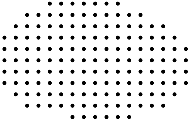
Discrétisation : $\mathrm{Disc}(P) = P \cap \mathbb{Z}^d$.

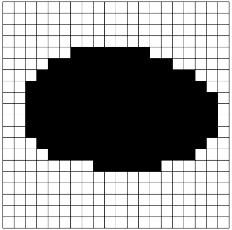


Discrétisation

Discrétisation : $\mathrm{Disc}(P) = P \cap \mathbb{Z}^d$.





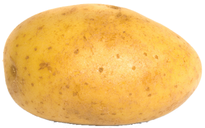


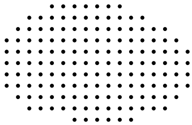
Étant donné $\mathrm{Disc}(P)$, que peut-on dire de P ?

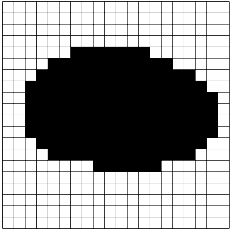
- Convexité ?
- Aire ?
- Périmètre ?
- Courbure ?
- ...

Discrétisation

Discrétisation : $\mathrm{Disc}(P) = P \cap \mathbb{Z}^d$.

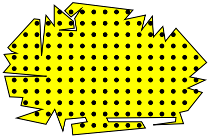






Étant donné $\mathrm{Disc}(P)$, que peut-on dire de P ?

- Convexité ?
- Aire ?
- Périmètre ?
- Courbure ?
- ...



Estimateur

- Étant donné une fonction f définie sur les régions* de $\mathbb{R}^n$, un **estimateur** est une fonction $\hat{f}$ définie sur les discrétisations telle que :

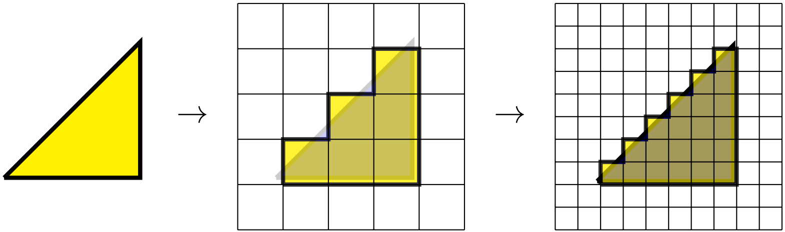
$$\lim_{h \to 0} \hat{f}\left(\mathrm{Disc}_h(P)\right) = f(P)$$

où h est le pas de discrétisation. (*certaines conditions s'appliquent)

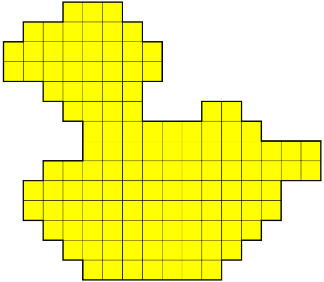
- Exemple :

La surface des pixels est un estimateur de surface

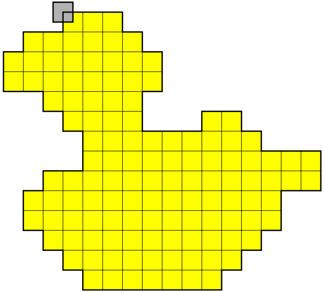
Le périmètre des pixels n'est pas un estimateur de périmètre



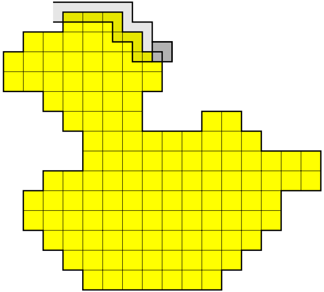
Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



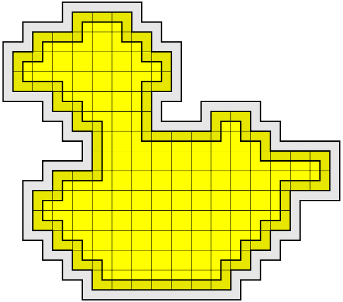
Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



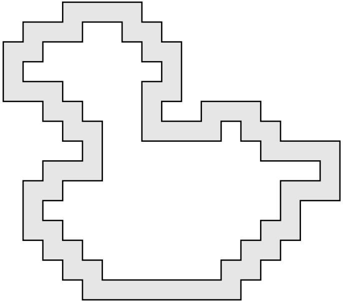
Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



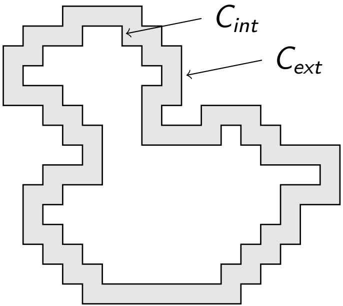
Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



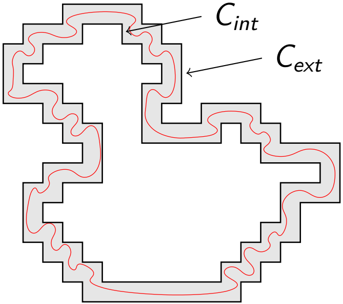
Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



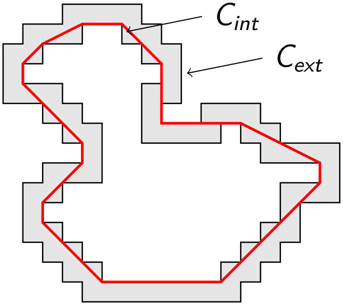
Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)

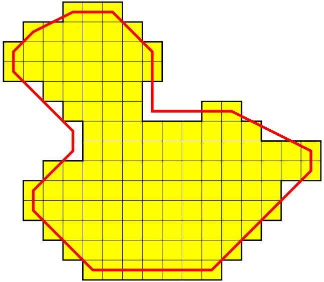


Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



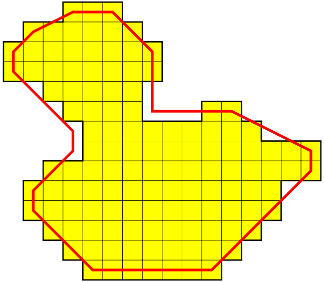
- Le **polygone de longueur minimum** (MLP) est la courbe la plus courte qui fait le tour de $\mathrm{Disc}(P)$ en restant dans la bande définie par C_{int} et C_{ext}.

Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



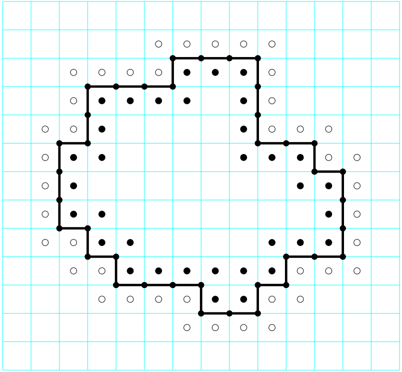
- Le **polygone de longueur minimum** (MLP) est la courbe la plus courte qui fait le tour de $\mathrm{Disc}(P)$ en restant dans la bande définie par C_{int} et C_{ext}.

Le MLP (Montanari 1970) (Sklansky, Chazin, Hansen 1972)



- Le **polygone de longueur minimum** (MLP) est la courbe la plus courte qui fait le tour de $\mathrm{Disc}(P)$ en restant dans la bande définie par C_{int} et C_{ext}.
  - bon estimateur de longueur,
  - bon estimateur de tangente,
  - décrit la convexité de l'objet,
  - réversible*.

Pixels intérieurs et extérieurs

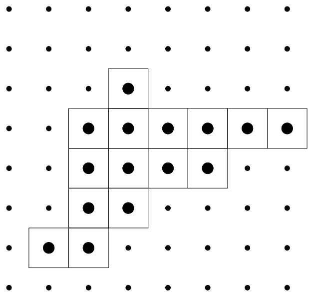


● pixels **intérieurs**.

○ pixels **extérieurs**.

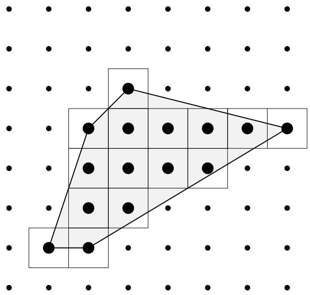
Enveloppe convexe

- L'**enveloppe convexe** d'un ensemble de pixels est le plus petit ensemble convexe qui contient les centres de chacun des pixels.



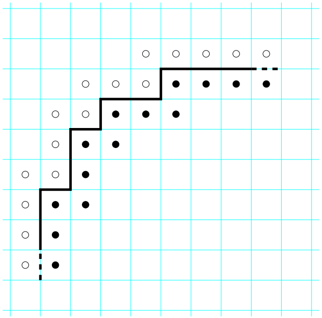
Enveloppe convexe

- L'**enveloppe convexe** d'un ensemble de pixels est le plus petit ensemble convexe qui contient les centres de chacun des pixels.



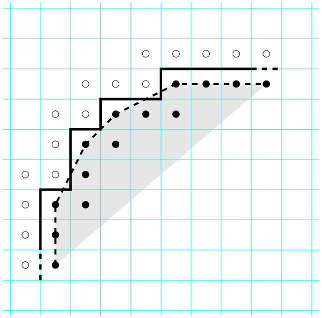
Construction du MLP (Sloboda, Stoer 1994) (Lachaud, P. 2009)

- Le MLP est construit par morceau, selon la convexité locale :



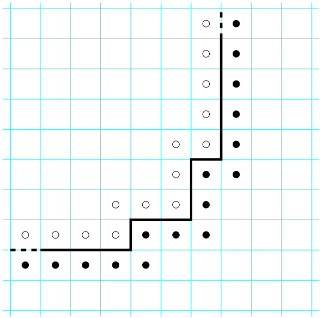
Construction du MLP (Sloboda, Stoer 1994) (Lachaud, P. 2009)

- Le MLP est construit par morceau, selon la convexité locale :
  - Sur une parti convexe, le MLP suit l'enveloppe convexe des pixels intérieurs.



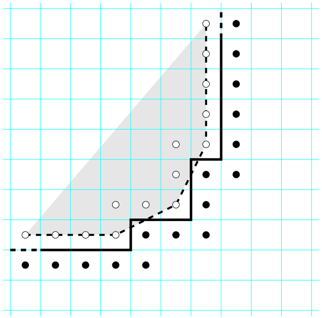
Construction du MLP (Sloboda, Stoer 1994) (Lachaud, P. 2009)

- Le MLP est construit par morceau, selon la convexité locale :
  - Sur une parti convexe, le MLP suit l'enveloppe convexe des pixels intérieurs.



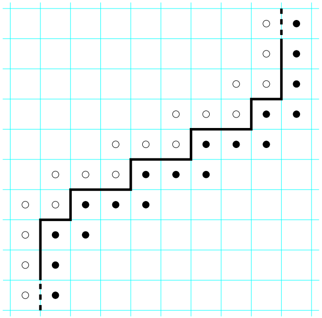
Construction du MLP (Sloboda, Stoer 1994) (Lachaud, P. 2009)

- Le MLP est construit par morceau, selon la convexité locale :
  - Sur une parti convexe, le MLP suit l'enveloppe convexe des pixels intérieurs.
  - Sur une parti concave, le MLP suit l'enveloppe convexe des pixels extérieurs.



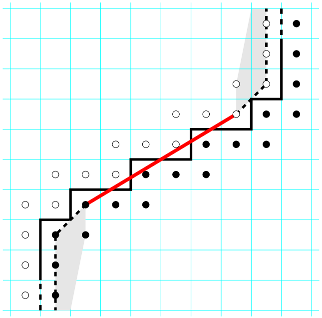
Construction du MLP (Sloboda, Stoer 1994) (Lachaud, P. 2009)

- Le MLP est construit par morceau, selon la convexité locale :
  - Sur une parti convexe, le MLP suit l'enveloppe convexe des pixels intérieurs.
  - Sur une parti concave, le MLP suit l'enveloppe convexe des pixels extérieurs.



Construction du MLP (Sloboda, Stoer 1994) (Lachaud, P. 2009)

- Le MLP est construit par morceau, selon la convexité locale :
  - Sur une parti convexe, le MLP suit l'enveloppe convexe des pixels intérieurs.
  - Sur une parti concave, le MLP suit l'enveloppe convexe des pixels extérieurs.
  - Sur une zone d'inflexion, le MLP forme une ligne droite qui relie les deux enveloppes convexes.



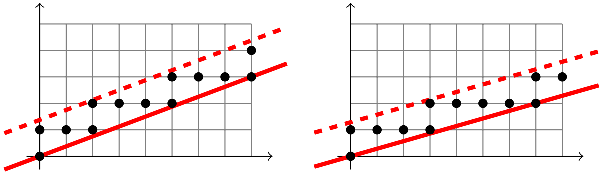
Droite discrète (Reveillès 1991), (Kovalev 1990)

- Une **droite discrète** est l'ensemble des points entiers (x, y) tels que :

$$0 \leq ax + by + \mu < |a| + |b|$$

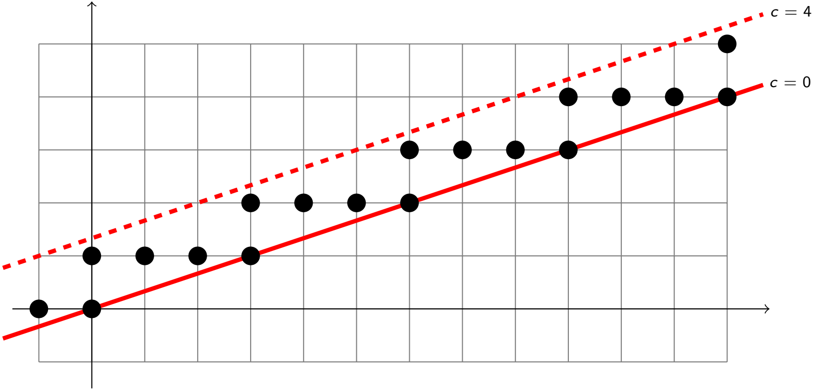
  où

  - $\begin{bmatrix} a \\ b \end{bmatrix}$ est le **vecteur normal**,
  - $-b/a$ est la **pente**,
  - μ est le **décalage**.



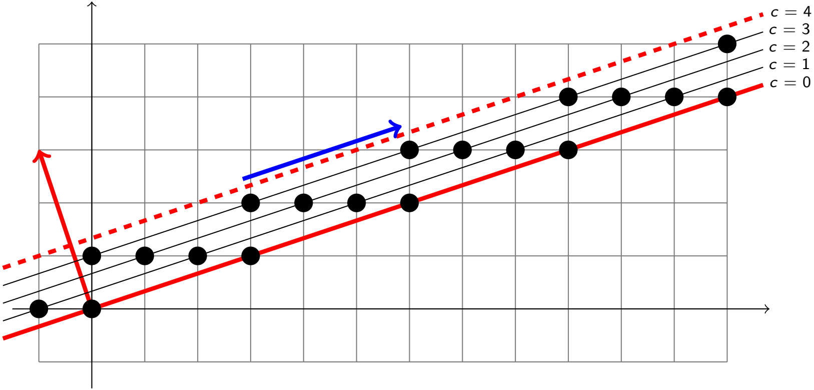
Structure périodique d'une DD

- Les points d'une DD sont réparties périodiquement sur des droites parallèles en fonction de la valeur de $c = \begin{bmatrix} a \\ b \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix}$
- Par exemple pour $(a, b) = (-1, 3)$:



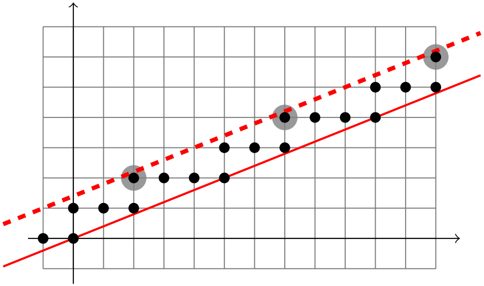
Structure périodique d'une DD

- Les points d'une DD sont réparties périodiquement sur des droites parallèles en fonction de la valeur de $c = \begin{bmatrix} a \\ b \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix}$
- Par exemple pour $(a, b) = (-1, 3)$:



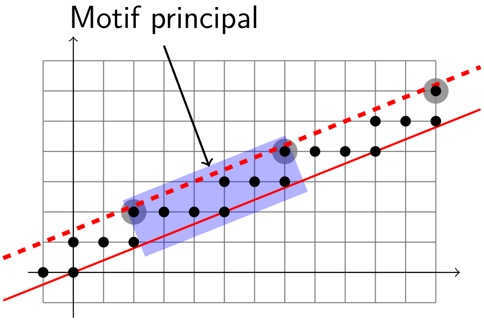
Motif principal d'une droite discrète

- A point (x, y) est un **point d'appui supérieur** si $\begin{bmatrix} x \\ y \end{bmatrix} \bullet \begin{bmatrix} a \\ b \end{bmatrix}$ est maximal.



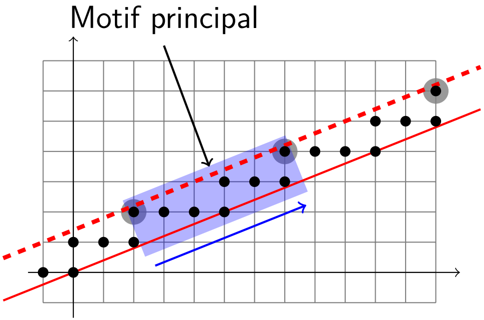
Motif principal d'une droite discrète

- A point (x, y) est un **point d'appui supérieur** si $\begin{bmatrix} x \\ y \end{bmatrix} \bullet \begin{bmatrix} a \\ b \end{bmatrix}$ est maximal.
- Le **motif principal** d'une droite discrète est l'ensemble des points bornés par deux points d'appui supérieurs consécutifs.



Motif principal d'une droite discrète

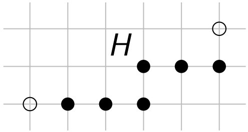
- A point (x, y) est un **point d'appui supérieur** si $\begin{bmatrix} x \\ y \end{bmatrix} \bullet \begin{bmatrix} a \\ b \end{bmatrix}$ est maximal.
- Le **motif principal** d'une droite discrète est l'ensemble des points bornés par deux points d'appui supérieurs consécutifs.



Construction récursive du motif principal

- $\circ$: points d'appui supérieurs
- Soit H le points le plus *haut* parmi $\{\bullet\}$.

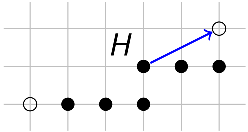
Motif principal de pente $2/5$.



Construction récursive du motif principal

- $\bigcirc$: points d'appui supérieurs
- Soit H le points le plus *haut* parmi $\{\bullet\}$.
- Un vecteur de H à un point d'appui supérieur est appelé **vecteur de Bezout**.

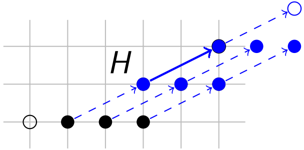
Motif principal de pente $2/5$.



Construction récursive du motif principal

- ○ : points d'appui supérieurs
- Soit H le points le plus *haut* parmi $\{\bullet\}$.
- Un vecteur de H à un point d'appui supérieur est appelé **vecteur de Bezout**.

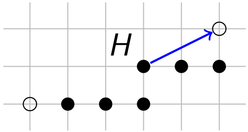
Motif principal de pente $2/5$.

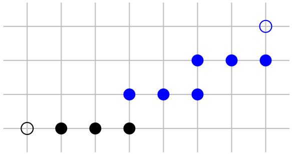


Construction récursive du motif principal

- ○ : points d'appui supérieurs
- Soit H le points le plus *haut* parmi $\{\bullet\}$.
- Un vecteur de H à un point d'appui supérieur est appelé **vecteur de Bezout**.

Motif principal de pente $2/5$.



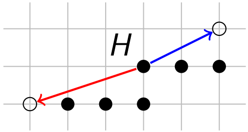


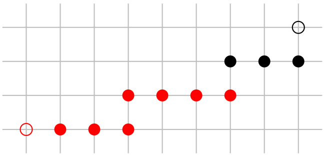
Motif principal de pente $3/8$.

Construction récursive du motif principal

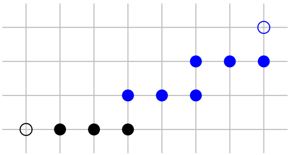
- ○ : points d'appui supérieurs
- Soit H le points le plus *haut* parmi $\{\bullet\}$.
- Un vecteur de H à un point d'appui supérieur est appelé **vecteur de Bezout**.

Motif principal de pente $2/5$.



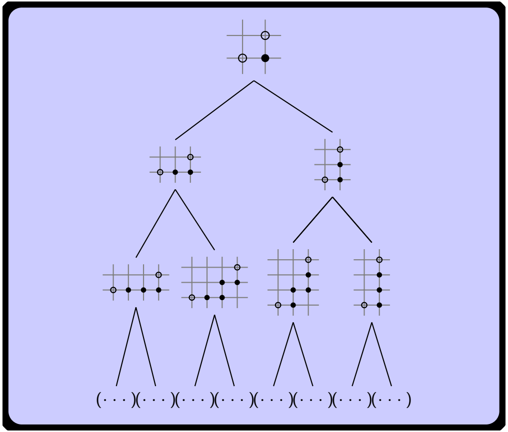


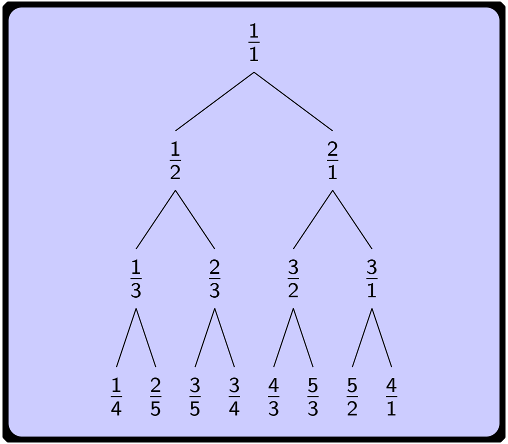
Motif principal de pente $3/7$.



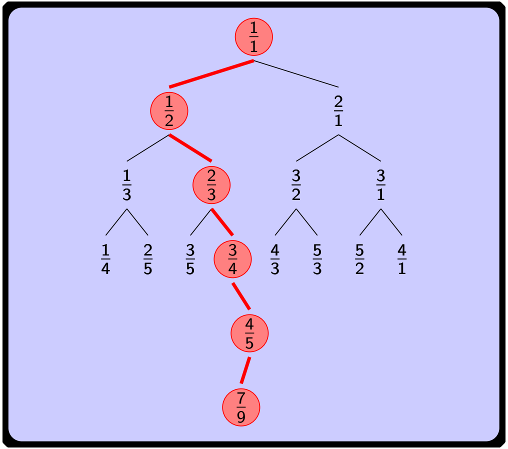
Motif principal de pente $3/8$.

Construction récursive du motif principal





Construction guidée par Euclide



Euclid algorithm	Pente du motif
$(7, 9)$	$(1, 1)$
$\downarrow$	$\downarrow$
$(7, 2)$	$(1, 2)$
$\downarrow$	$\downarrow$
$(5, 2)$	$(2, 3)$
$\downarrow$	$\downarrow$
$(3, 2)$	$(3, 4)$
$\downarrow$	$\downarrow$
$(1, 2)$	$(4, 5)$
$\downarrow$	$\downarrow$
$(1, 1)$	$(7, 9)$

Forme matricielle

	Algo Euclide	Pente motif
n	v_n	p_n
0	$(\underline{7},9)$	$(1,1)$
	$\downarrow$	$\downarrow$
1	$(7,\underline{2})$	$(1,2)$
	$\downarrow$	$\downarrow$
2	$(5,\underline{2})$	$(2,3)$
	$\downarrow$	$\downarrow$
3	$(3,\underline{2})$	$(3,4)$
	$\downarrow$	$\downarrow$
4	$(\underline{1},2)$	$(4,5)$
	$\downarrow$	$\downarrow$
5	$(1,1)$	$(7,9)$

Algorithme d'Euclide
Given a vector (x,y), return

- $\begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$ si $x < y$,
- $\begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$ si $x > y$,
- **stop** si $x = y$.

Étant donné un vecteur v, on pose :

- $v_0 = v$,
- For all $n \geq 1$: $\begin{cases} M_n = \textbf{Euclid}(v_{n-1}) \\ v_n = M_n v_{n-1}. \end{cases}$

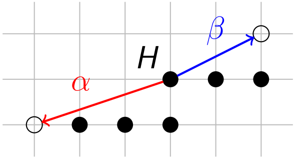
- $v_n = M_n M_{n-1} \cdots M_1 v$
- $p_n = M_1^{-1} M_2^{-1} \cdots M_n^{-1} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$

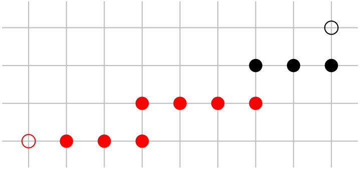
Forme matricielle

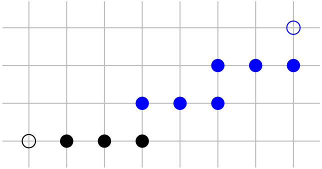
Soient α le vecteur de Bézout gauche et β le vecteur de Bézout droit,

$$M_1^\top M_2^\top \cdots M_n^\top = [\ \alpha\ \beta\ ]$$

$$M_1^\top \cdots M_n^\top e_1 = \alpha, \qquad M_1^\top \cdots M_n^\top e_2 = \beta.$$



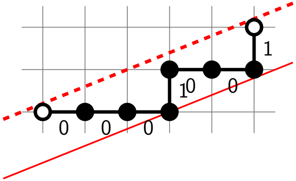




Mot de Christoffel

Definition ([Christoffel 1875])

Un **mot de Christoffel** encodes le motif principal d'une droite discrète sur un alphabet à deux lettres.

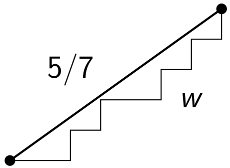


Mot de Christoffel de **pente** $2/5$: 0001001.

Ordre lexicographique

Propriété

L'ordre lexicographique sur les mots de Christoffel correspond l'ordre sur les pentes

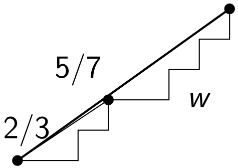


- Pente $5/7$: 001010010101

Ordre lexicographique

Propriété

L'ordre lexicographique sur les mots de Christoffel correspond l'ordre sur les pentes

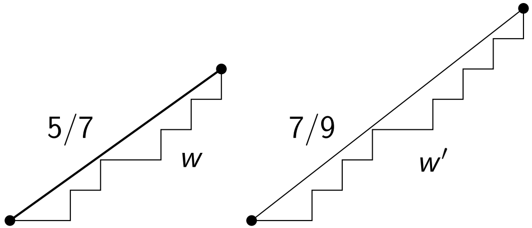


- Pente 5/7 : 001010010101
- Pente 2/3 : 00101

Ordre lexicographique

Propriété

L'ordre lexicographique sur les mots de Christoffel correspond l'ordre sur les pentes

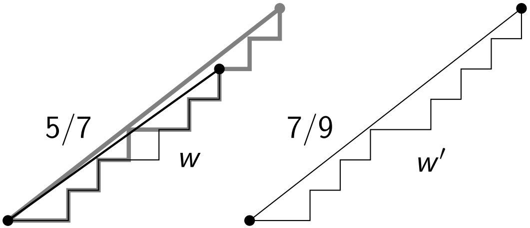


- Pente 5/7 : 001010010101
- Pente 2/3 : 00101
- Pente 7/9 : 0010101001010101

Ordre lexicographique

Propriété

L'ordre lexicographique sur les mots de Christoffel correspond l'ordre sur les pentes



- Pente 5/7 : 001010010101
- Pente 2/3 : 00101
- Pente 7/9 : 0010101001010101

Lyndon words

Definition ([Lyndon 54])

Un mot w est un **mot de Lyndon** ssi pour tous ses suffixes propres s de w

$$w <_{\textbf{Lex}} s$$

Exemples :

1. 00101 des Lyndon car $00101 <_{\textbf{Lex}} \{0101, 101, 01, 1\}$,
2. 01001 n'est pas Lyndon car $001 <_{\textbf{Lex}} 01001$.
3. 001001 n'est pas Lyndon car $001 <_{\textbf{Lex}} 001001$.

Lyndon words

Definition ([Lyndon 54])

Un mot w est un **mot de Lyndon** ssi pour tous ses suffixes propres s de w

$$w <_{\textbf{Lex}} s$$

Exemples :

1. 00101 des Lyndon car $00101 <_{\textbf{Lex}} \{0101, 101, 01, 1\}$,
2. 01001 n'est pas Lyndon car $001 <_{\textbf{Lex}} 01001$.
3. 001001 n'est pas Lyndon car $001 <_{\textbf{Lex}} 001001$.

Théorème ([Chen, Fox, Lyndon 58])

Tout mot admet une ***unique*** *factorisation en mots de Lyndon décroissants.*

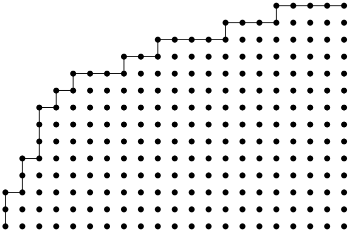
Exemple :

$$\begin{aligned} & 1\,1\,0\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0 \\ = \;& 1 \cdot 1 \cdot 0\,1\,1 \cdot 0\,1\,1 \cdot 0\,0\,1\,0\,0\,1\,1 \cdot 0 \cdot 0 \cdot 0 \\ = \;& (1)^2 \cdot (0\,1\,1)^2 \cdot (0\,0\,1\,0\,0\,1\,1)^1 \cdot (0)^3. \end{aligned}$$

Vision combinatoire de la convexité

Théorème ([Brlek, Lachaud, P., Reutenauer 09])

La partie nord-ouest d'une forme digitale est convexe ssi la factorisation de Lyndon du mot qui encode son bord est formée uniquement de mots de Christoffel.
De plus, les points de factorisation correspondent aux points de l'envloppe convexe.

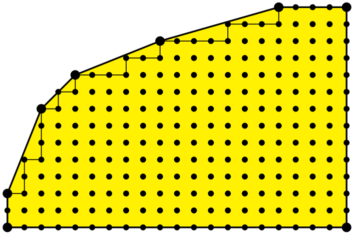


$$1101101110101000100100001000100000$$
$$=(1)^2 \cdot 0110111 \cdot (01)^2 \cdot 001001 \cdot 000010001 \cdot (0)^4$$

Vision combinatoire de la convexité

Théorème ([Brlek, Lachaud, P., Reutenauer 09])

La partie nord-ouest d'une forme digitale est convexe ssi la factorisation de Lyndon du mot qui encode son bord est formée uniquement de mots de Christoffel.
De plus, les points de factorisation correspondent aux points de l'envloppe convexe.



$$1101101110101000100100001000\,10000$$
$$= (1)^2 \cdot 0110111 \cdot (01)^2 \cdot 001001 \cdot 000010001 \cdot (0)^4$$

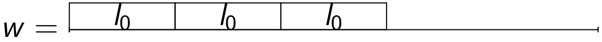
Algorithme de Duval (Duval 1983)

Identification du premier mot de la factorisation de Lyndon.

$w =$

Algorithme de Duval (Duval 1983)

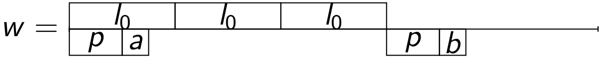
Identification du premier mot de la factorisation de Lyndon.



1. Soit l_0 un préfixe de w qui est un mot de Lyndon et k son nombre de répétitions.

Algorithme de Duval (Duval 1983)

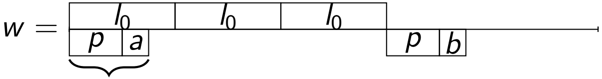
Identification du premier mot de la factorisation de Lyndon.



1. Soit l_0 un préfixe de w qui est un mot de Lyndon et k son nombre de répétitions.
2. On identifie b, la première lettre qui n'est pas une répétition de l_0.

Algorithme de Duval (Duval 1983)

Identification du premier mot de la factorisation de Lyndon.



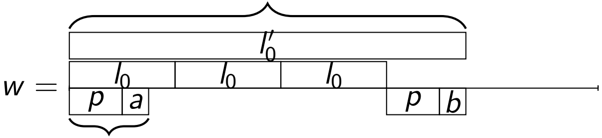
$a > b \implies$ Lyndon fact. starts by l_0^k

1. Soit l_0 un préfixe de w qui est un mot de Lyndon et k son nombre de répétitions.
2. On identifie b, la première lettre qui n'est pas une répétition de l_0.
3. Si b est plus petite, alors l_0 est le premier mot de la factorisation,

Algorithme de Duval (Duval 1983)

Identification du premier mot de la factorisation de Lyndon.

$b > a \implies l_0^k pb$ est Lyndon



$a > b \implies$ Lyndon fact. starts by l_0^k

1. Soit l_0 un préfixe de w qui est un mot de Lyndon et k son nombre de répétitions.
2. On identifie b, la première lettre qui n'est pas une répétition de l_0.
3. Si b est plus petite, alors l_0 est le premier mot de la factorisation, sinon, $l_0^k pb$ est un mot de Lyndon.

Algorithme de Duval

$$w = \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1:j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{rcccccccccccccccccc}
i: & \downarrow & & & & & & & & & & & & & & & & & \\
w = & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & \uparrow & & & & & & & & & & & & & & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1:j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{rcccccccccccccccccc}
i: & & \downarrow & & & & & & & & & & & & & & & & \\
w = & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & \uparrow & & & & & & & & & & & & & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1:j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{lcccccccccccccccc}
i: & \downarrow & & & & & & & & & & & & & & & \\
w = & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & \uparrow & & & & & & & & & & & & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{lcccccccccccccccc}
i: & \downarrow & & & & & & & & & & & & & & & \\
w = & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & \uparrow & & & & & & & & & & & & &
\end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1:j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rcccccccccccccccc}
i : & \downarrow & & & & & & & & & & & & & & & \\
w = & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j : & & & & \uparrow & & & & & & & & & & & &
\end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{lcccccccccccccccc}
i: & & \downarrow & & & & & & & & & & & & & & \\
w = & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & \uparrow & & & & & & & & & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{lcccccccccccccccc}
i: & & & \downarrow & & & & & & & & & & & & & \\
w = & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & & \uparrow & & & & & & & & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rcccccccccccccccc}
i: & & & & \downarrow & & & & & & & & & & & & \\
w = & & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & & & & \uparrow & & & & & & & & &
\end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1:j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rcccccccccccccccc}
i: & & & & & \downarrow & & & & & & & & & & & \\
w = & & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & & & & \uparrow & & & & & & & &
\end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{lcccccccccc}
i: & \downarrow & & & & & & & & & \\
w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & \uparrow & & & & & & & &
\end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rcccccccccc} i : & & \downarrow & & & & & & & & \\ w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ j : & & & \uparrow & & & & & & & \end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rcccccccccc} i: & \downarrow & & & & & & & & & \\ w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ j: & & & & \uparrow & & & & & & \end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1:j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{rcccccccccc}
i: & & \downarrow & & & & & & & & \\
w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & \uparrow & & & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{rcccccccccc}
i: & & & \downarrow & & & & & & & \\
w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & & \uparrow & & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w

$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{rcccccccccc}
i: & & & & \downarrow & & & & & & \\
w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & & & \uparrow & & &
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rcccccccccc}
i: & \downarrow & & & & & & & & & \\
w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & & & & \uparrow & &
\end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
    si $w_i = w_j$ **alors**
        $++i$;
    sinon
        $i \leftarrow 1$;
    $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rcccccccccc} i : & & \downarrow & & & & & & & & \\ w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ j : & & & & & & & & & \uparrow & \end{array}$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{rcccccccccc}
i: & & & \downarrow & & & & & & & \\
w = & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
j: & & & & & & & & & & \uparrow
\end{array}
$$

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$\begin{array}{rccc} i: & \downarrow & & \\ w = & 0 & 0 & 0 \\ j: & & \uparrow & \end{array}$$

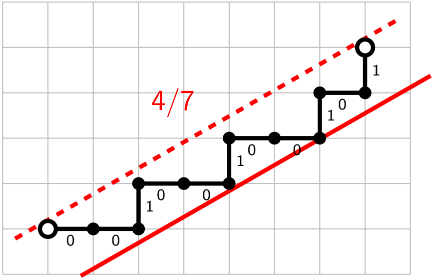
Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
    si $w_i = w_j$ **alors**
        $++i$;
    sinon
        $i \leftarrow 1$;
    $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Algorithme de Duval

$$
\begin{array}{ll}
i : & \quad \downarrow \quad \\
w = & 0 \quad 0 \quad 0 \\
j : & \quad \quad \uparrow
\end{array}
$$

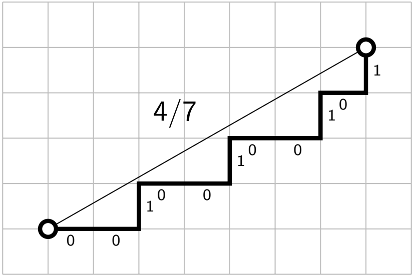
Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w
$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Factorisation standard d'un mot de Christoffel



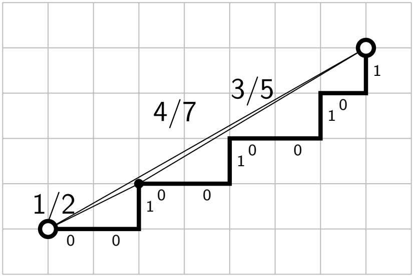
00100100101 est le mot de Christoffel de **pente** $4/7$.

Factorisation standard d'un mot de Christoffel



00100100101 est le mot de Christoffel de **pente** 4/7.

Factorisation standard d'un mot de Christoffel



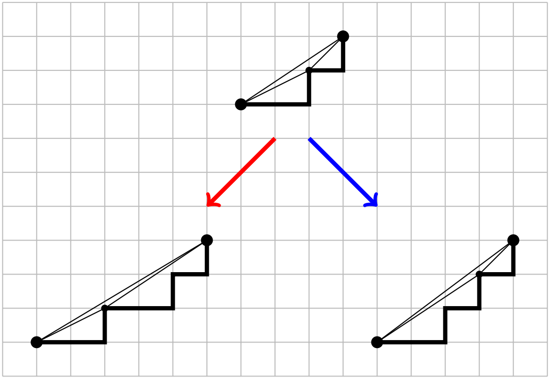
$001 \cdot 00100101$ est le mot de Christoffel de **pente** $4/7$.

> **Théorème ([Borel, Laubie 93])**
>
> *Tout mot de Christoffel autre que 0 et 1 s'écrit de manière* ***unique*** *comme un la concaténation de deux mots de Christoffel.*

Ceci est appelé la **factorisation standard**, notée $w = (u, v)$.

Christoffel Tree

Si (u, v) est une factorisation standard, alors (u, uv) et (uv, v) sont aussi des factorisation standard de mots de Christoffel.



Préfixes imbriqués

Propriété

Un mot de Christoffel C qui admet $w = (u, v)$ comme préfixe propre, possède également un préfixe $w^k v = (w, w^{k-1}v)$.

Ainsi, pour identifier le plus long préfixe qui est un mot de Christoffel :

Préfixes imbriqués

Propriété

Un mot de Christoffel C qui admet $w = (u, v)$ comme préfixe propre, possède également un préfixe $w^k v = (w, w^{k-1}v)$.

Ainsi, pour identifier le plus long préfixe qui est un mot de Christoffel :

Préfixes imbriqués

Propriété

Un mot de Christoffel C qui admet $w = (u, v)$ comme préfixe propre, possède également un préfixe $w^k v = (w, w^{k-1} v)$.

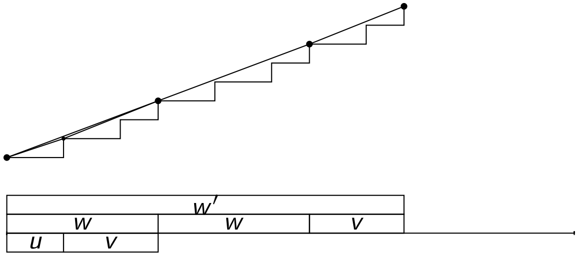
Ainsi, pour identifier le plus long préfixe qui est un mot de Christoffel :

Préfixes imbriqués

Propriété

Un mot de Christoffel C qui admet $w = (u, v)$ comme préfixe propre, possède également un préfixe $w^k v = (w, w^{k-1}v)$.

Ainsi, pour identifier le plus long préfixe qui est un mot de Christoffel :

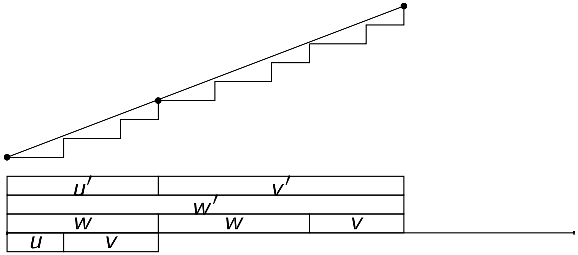


Préfixes imbriqués

Propriété

Un mot de Christoffel C qui admet $w = (u, v)$ comme préfixe propre, possède également un préfixe $w^k v = (w, w^{k-1}v)$.

Ainsi, pour identifier le plus long préfixe qui est un mot de Christoffel :

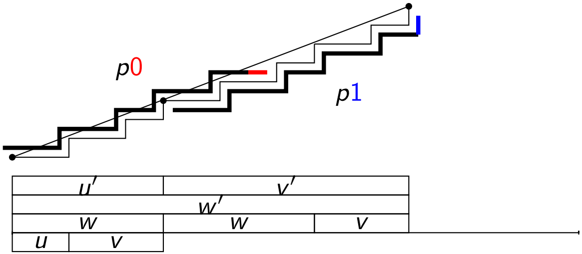


Préfixes imbriqués

Propriété

Un mot de Christoffel C qui admet $w = (u, v)$ comme préfixe propre, possède également un préfixe $w^k v = (w, w^{k-1}v)$.

Ainsi, pour identifier le plus long préfixe qui est un mot de Christoffel :



Corollary

A Christoffel word $w = (u, v)$ has a prefix $p0$ such that $v = p1$.

Duval++ (Lachaud, P., 2009)

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w si celui-ci est un mot de Christoffel, **faux** sinon.

$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
    si $w_i = w_j$ **alors**
        $++i$;
    sinon
        $i \leftarrow 1$;
    $++j$;
retourner $\left(w[1 : j-i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Duval++ (Lachaud, P., 2009)

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w si celui-ci est un mot de Christoffel, **faux** sinon.

$i \leftarrow 1$; $j \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
  sinon si $j \neq q$ **alors**
    // w n'encode pas une région convexe
    retourner ***faux***;
  sinon
    $i \leftarrow 1$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

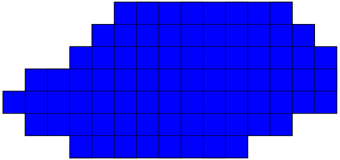
Duval++ (Lachaud, P., 2009)

Entrées : w un mot sur un alphabet ordonné
Output : le premier terme de la factorisation de Lyndon de w si celui-ci est un mot de Christoffel, **faux** sinon.

$i \leftarrow 1$; $j \leftarrow 2$; $p \leftarrow 1$; $q \leftarrow 2$;
tant que $j \leq n$ ***et*** $w_i \leq w_j$ **faire**
  si $w_i = w_j$ **alors**
    $++i$;
    si $j = q$ **alors**
      $q \leftarrow q + p$;
  sinon si $j \neq q$ **alors**
    // w n'encode pas une région convexe
    retourner ***faux***;
  sinon
    $i \leftarrow 1$; $q \leftarrow 2q - p$; $p \leftarrow j$;
  $++j$;
retourner $\left(w[1 : j - i], \lfloor \frac{j-1}{j-i} \rfloor\right)$;

Mot de contour

- On utilise un alphabet à quatre lettre $\{0, 1, 2, 3\}$ pour encoder les pas dans chacune des directions
$$0 :\rightarrow,\ 1 :\uparrow,\ 2 :\leftarrow, 3 :\downarrow$$
- Le bord d'un forme digitale est encodé par un mot sur $\{0, 1, 2, 3\}$.

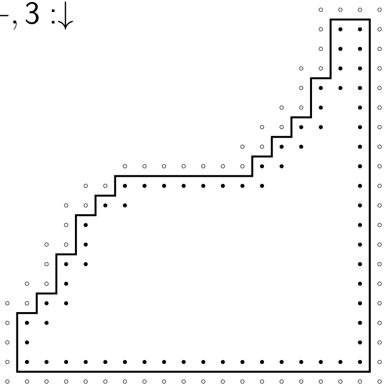


- 1010010101000000003030333223223222222212212

Calcul du MLP

$0 : \rightarrow,\ 1 : \uparrow,\ 2 : \leftarrow, 3 : \downarrow$

$3 < 0 < 1 < 2$

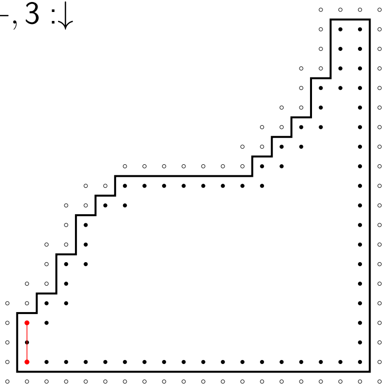


111010110110101000000010101011011100333333333333333333222222222222222222

Calcul du MLP

$0 : \rightarrow,\ 1 : \uparrow,\ 2 : \leftarrow, 3 : \downarrow$

$3 < 0 < 1 < 2$



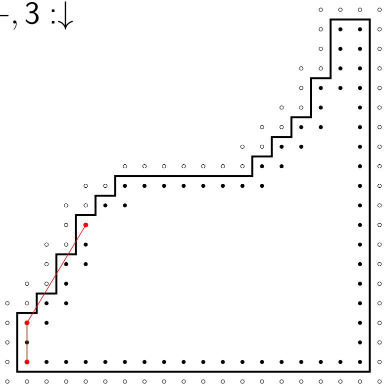
0101101101010000000101010110111100333333333333333333222222222222222222

$(1)^3$

Calcul du MLP

$0 :\rightarrow,\ 1 :\uparrow,\ 2 :\leftarrow, 3 :\downarrow$

$3 < 0 < 1 < 2$



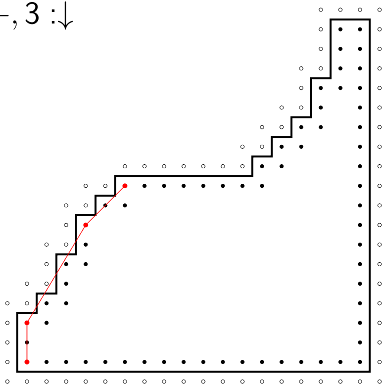
0101000000010101011011100333333333333333333222222222222222222

$(1)^3 \cdot (01011011)$

Calcul du MLP

$0 :\rightarrow,\ 1 :\uparrow,\ 2 :\leftarrow, 3 :\downarrow$

$3 < 0 < 1 < 2$



000000010101011011100333333333333333333222222222222222222

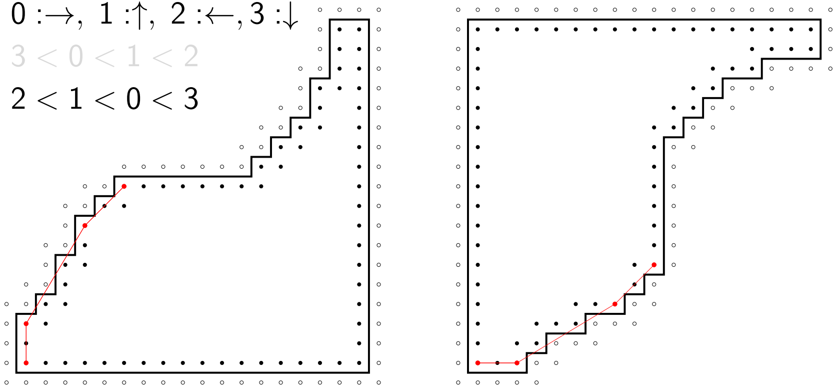
$(1)^3 \cdot (01011011) \cdot (0101)$

Calcul du MLP

$0 : \rightarrow,\ 1 : \uparrow,\ 2 : \leftarrow, 3 : \downarrow$

$3 < 0 < 1 < 2$

$2 < 1 < 0 < 3$



000000010101011011100333333333333333333222222222222222222

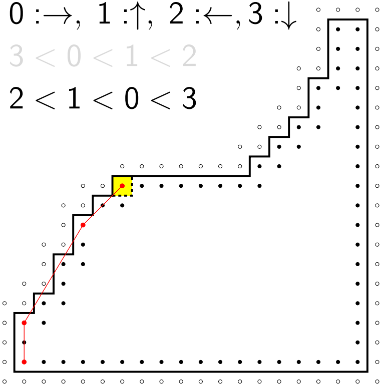
$(1)^3 \cdot (01011011) \cdot (0101)$

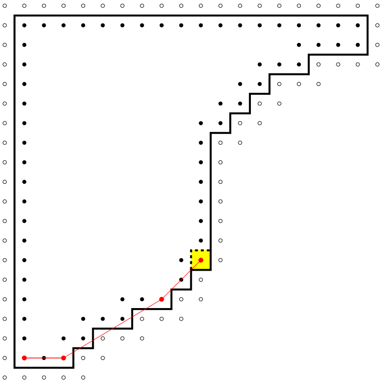
Calcul du MLP

0 :→, 1 :↑, 2 :←, 3 :↓

$3 < 0 < 1 < 2$

$2 < 1 < 0 < 3$





1000000101010110111003333333333333333332222222222222222222

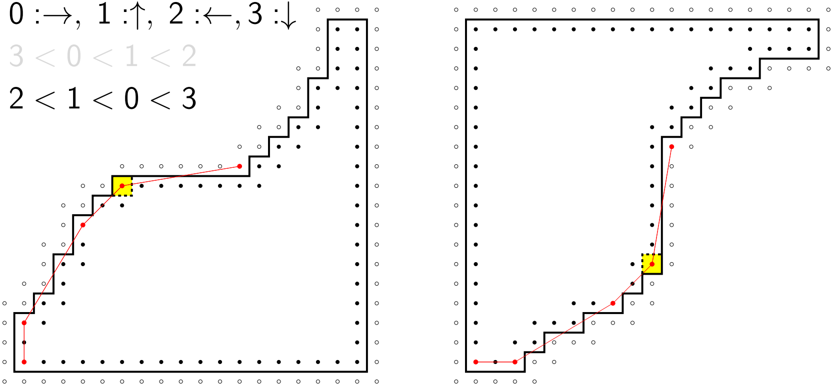
$(1)^3 \cdot (01011011) \cdot (0101)$

Calcul du MLP

0 :$\rightarrow$, 1 :$\uparrow$, 2 :$\leftarrow$, 3 :$\downarrow$

$3 < 0 < 1 < 2$

$2 < 1 < 0 < 3$



101010110111003333333333333333332222222222222222222

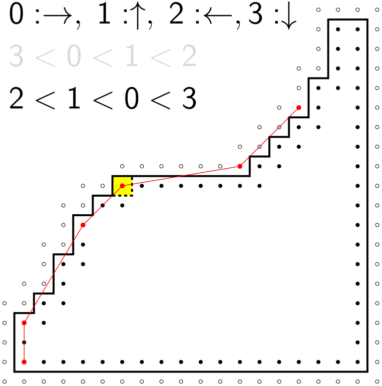
$(1)^3 \cdot (01011011) \cdot (0101) \cdot (1000000)$

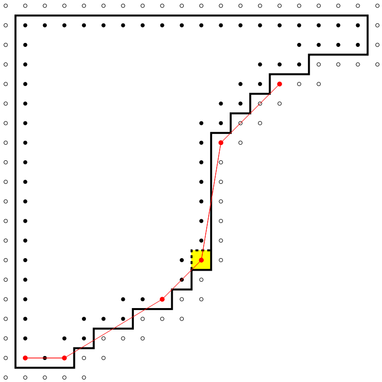
Calcul du MLP

$0 : \rightarrow,\ 1 : \uparrow,\ 2 : \leftarrow, 3 : \downarrow$

$3 < 0 < 1 < 2$

$2 < 1 < 0 < 3$





110111003333333333333333332222222222222222222

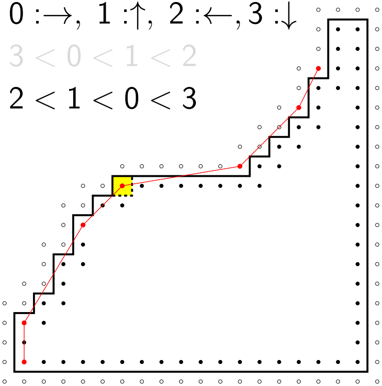
$(1)^3 \cdot (01011011) \cdot (0101) \cdot (1000000) \cdot (101010)$

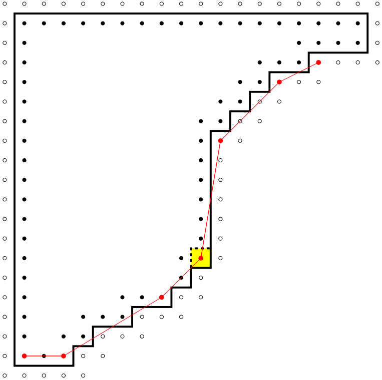
Calcul du MLP

0 :→, 1 :↑, 2 :←, 3 :↓

$3 < 0 < 1 < 2$

$2 < 1 < 0 < 3$





111003333333333333333332222222222222222222

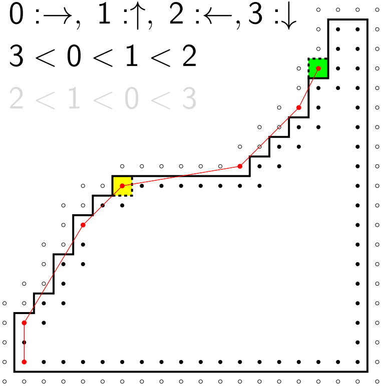
$(1)^3 \cdot (01011011) \cdot (0101) \cdot (1000000) \cdot (101010) \cdot (110)$

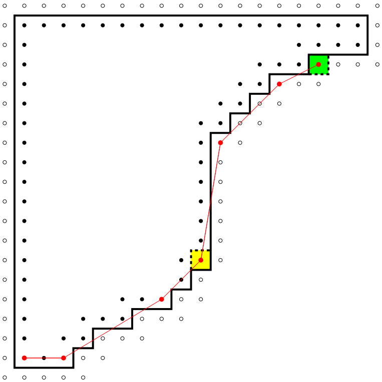
Calcul du MLP

$0 : \rightarrow,\ 1 : \uparrow,\ 2 : \leftarrow, 3 : \downarrow$

$3 < 0 < 1 < 2$

$2 < 1 < 0 < 3$





0110033333333333333333332222222222222222222

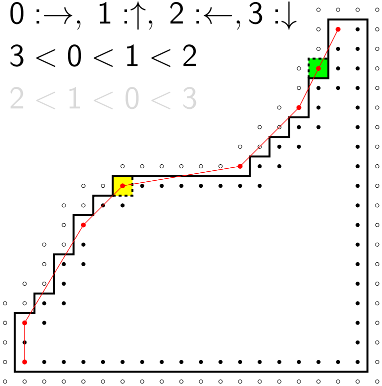
$(1)^3 \cdot (01011011) \cdot (0101) \cdot (1000000) \cdot (101010) \cdot (110)$

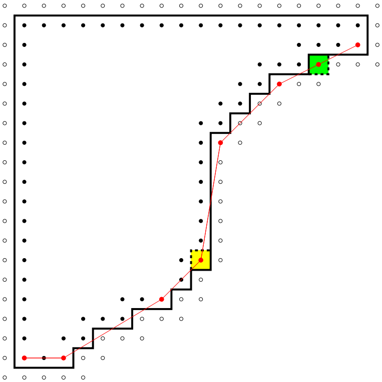
Calcul du MLP

$0 :\rightarrow,\ 1 :\uparrow,\ 2 :\leftarrow, 3 :\downarrow$

$3 < 0 < 1 < 2$

$2 < 1 < 0 < 3$





00333333333333333333322222222222222222

$(1)^3 \cdot (01011011) \cdot (0101) \cdot (1000000) \cdot (101010) \cdot (110) \cdot (011)$