BROUILLON - DES COEFFICIENTS DE BACHET-BÉZOUT POUR LES HUMAINS

CHRISTOPHE BAL

Document, avec son source LaTeX, disponible sur la page
https://github.com/bc-writing/drafts.

Mentions « légales »

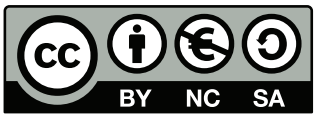


Table des matières

Date: 10 Sept. 2019 - 21 Sept. 2019.

1. Où allons-nous ?

Un résultat classique d'arithmétique dit qu'étant donné $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$, il existe $(u\,;v) \in \mathbb{Z}\times\mathbb{Z}$ tel que $au+bv = \text{pgcd}(a\,;b)$. Les entiers u et v seront appelés *« coefficients de Bachet-Bézout »* et $au + bv = \text{pgcd}(a\,;b)$ sera nommée *« relation de Bachet-Bézout »*. Notons qu'il n'y a pas unicité car nous avons par exemple :

$$(-3) \times 12 + 1 \times 42 = 4 \times 12 + (-1) \times 42 = 6 = \text{pgcd}(12\,;42)$$

Nous allons voir comment trouver de tels entiers u et v tout d'abord de façon humainement rapide puis ensuite via un algorithme programmable efficace.

2. L'algorithme « human friendly » appliqué de façon magique

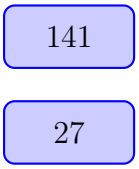
2.1. **Un exemple complet façon *« diaporama »*.** Sur le lieu de téléchargement de ce document se trouve un fichier `PDF` de chemin relatif `bezout-coef-for-human/slide-version.pdf` présentant la méthode sous la forme d'un diaporama. Nous vous conseillons de le regarder avant de lire les explications ci-après.

2.2. **Phase 1 – Au début était l'algorithme d'Euclide...** Pour chercher des coefficients de Bachet-Bézout pour $(a\,;b) = (27\,;141)$, on commence par appliquer l'algorithme d'Euclide *« verticalement »* comme suit.

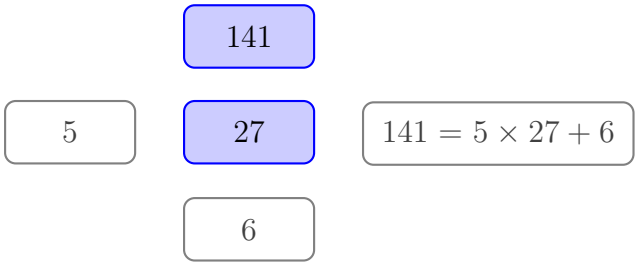
141

27

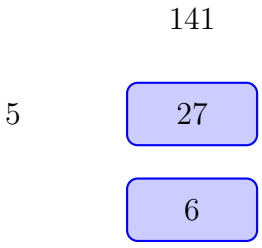
Étape 1 : le plus grand naturel est mis au-dessus.



Étape 2 : deux naturels à diviser.



Étape 3 : première division euclidienne.



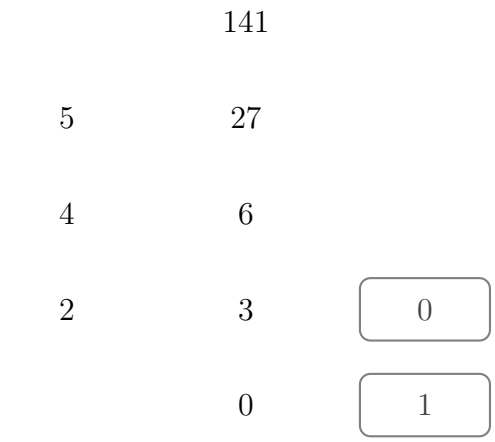
Étape 4 : on passe aux deux naturels suivants.

En répétant ce processus, nous arrivons à la représentation suivante où nous n'avons pas besoin de garder la trace des divisions euclidiennes.

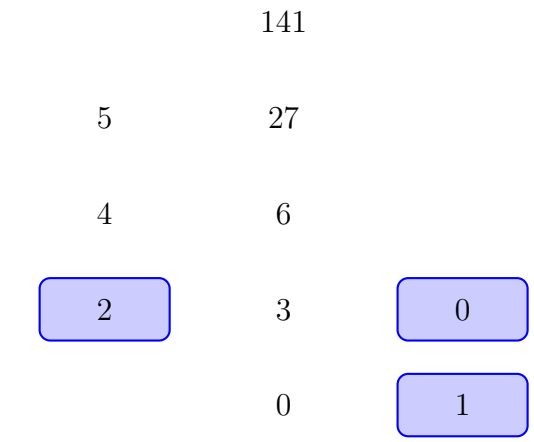
	141
5	27
4	6
2	3
	0

Étape finale (1re phase) : l'algorithme d'Euclide « vertical ».

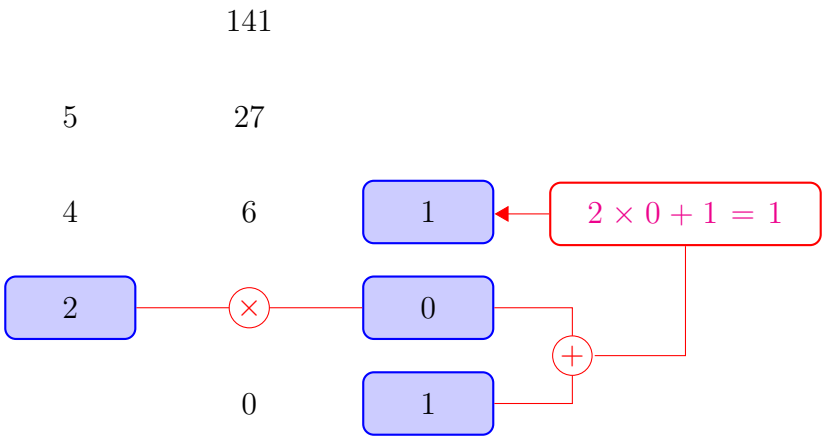
2.3. **Phase 2 – Remontée facile des étapes.** La méthode classique consiste à remonter les calculs. Mais comment faire cette remontée tout en évitant un claquage neuronal ? L'astuce est la suivante.



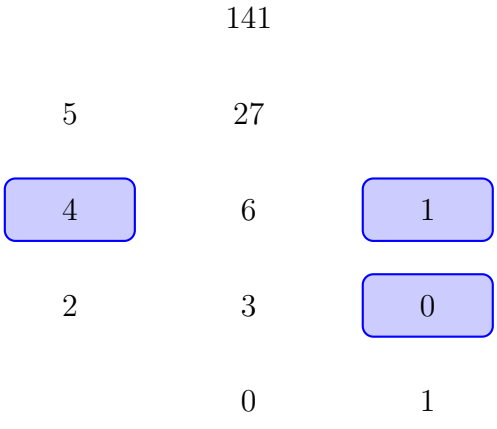
Étape 1 : ajout d'une nouvelle colonne.



Étape 2 : on n'utilise pas la colonne centrale.



Étape 3 : on fait une sorte de division « inversée ».



Étape 4 : on passe aux trois naturels suivants.

En répétant ce processus, nous arrivons à la représentation suivante.

	141	21
5	27	4
4	6	1
2	3	0
	0	1

Étape finale (2e phase) : remontée en voie libre des calculs.

Remarque 1. *En remontant les calculs sur la colonne centrale, on dispose d'un moyen simple de construire deux entiers a et b de* pgcd *fixé et avec des valeurs des quotients intermédiaires q_k choisis.*

2.4. Et voilà comment conclure !

	141	21
5	27	4
4	6	1
2	3	0
	0	1

$141 \times 4 - 21 \times 27 = -3$

Étape finale (la vraie) : on finit avec un produit en croix.

Des coefficients de Bachet-Bézout s'obtiennent sans souci via l'équivalence suivante où nous avons $3 = \text{pgcd}(27, 141)$.

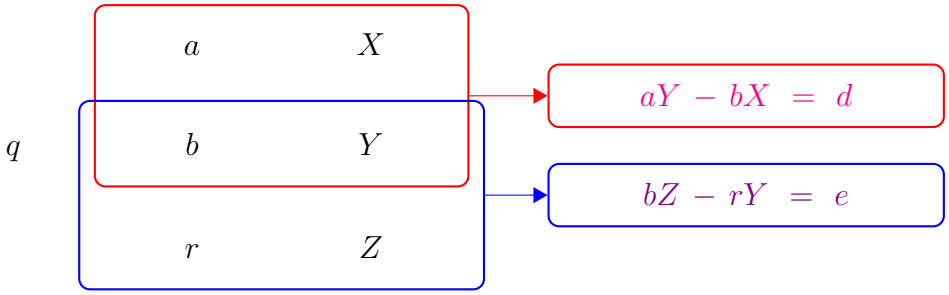
$$141 \times 4 - 27 \times 21 = -3 \iff 27 \times 21 - 141 \times 4 = 3$$

Nous allons voir, dans la section qui suit, que l'on obtient forcément à la fin $\pm\text{pgcd}(27, 141)$.

En pratique, nous n'avons pas besoin de détailler les calculs comme nous l'avons fait à certains moments afin d'expliquer comment procéder. Avec ceci en tête, on comprend toute l'efficacité de la méthode présentée, mais pas encore justifiée, car il suffit de garder une trace minimale, mais complète, des étapes tout en ayant à chaque étape des opérations assez simples à effectuer. Il reste à démontrer que notre méthode marche à tous les coups. Ceci est le propos de la section suivante.

3. Pourquoi cela marche-t-il ?

3.1. **Avec des arguments élémentaires.** Commençons par une preuve vérificative qui malheureusement ne nous permet pas de voir d'où vient l'astuce *(nous explorerons ceci dans la sous-section suivante)*.



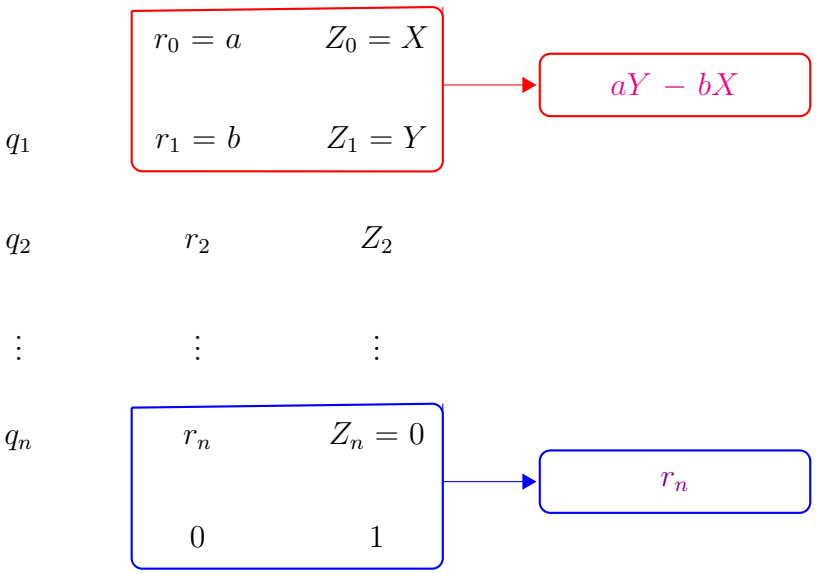
Calculs faits dans les deux phases.

Par construction, nous avons $a = qb + r$ et $X = qY + Z$. Ceci nous donne :

$$\begin{aligned} d &= aY - bX \\ &= (qb + r)Y - b(qY + Z) \\ &= rY - bZ \\ &= -e \end{aligned}$$

Donc si l'on fait *« glisser »* des carrés sur les deux colonnes de droite, les produits en croix dans ces carrés ne différeront que par leur signe.

La représentation symbolique *« complète »* ci-dessous donne $aY - bX = \pm \operatorname{pgcd}(a\,;b)$ car le dernier reste non nul de l'algorithme d'Euclide est $\operatorname{pgcd}(a\,;b)$. Ceci prouve la validité de la méthode dans le cas général. On comprend au passage l'ajout initial du 0 et du 1 dans la 3e colonne. Bien entendu, (-1) aurait pu convenir à la place de 1, et l'on constate que 0 peut être remplacé par n'importe quelle valeur entière.



Représentation symbolique au complet.

3.2. **Avec des matrices pour y voir plus clair.** Oublions tout ce que nous avons vu précédemment. Soit $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$ avec $a \geq b$. Nous cherchons $(u\,;v) \in \mathbb{Z} \times \mathbb{Z}$ tel que $au + bv = \operatorname{pgcd}(a\,;b)$. La petite astuce est de noter que $au + bv = \det M$ où $M = \begin{pmatrix} a & -v \\ b & u \end{pmatrix}$. Nous allons poser $X = -v$ et $Y = u$ de sorte que $M = \begin{pmatrix} a & X \\ b & Y \end{pmatrix}$ et raisonner en supposant l'existence de u et v [1].

Soit ensuite $a = qb + r$ la division euclidienne de a par b. L'algorithme d'Euclide nous fait alors travailler avec $(b\,;r)$ au lieu de $(a\,;b)$. Comme $r = a - qb$, on peut considérer la matrice $N = \begin{pmatrix} a - qb & X - qY \\ b & Y \end{pmatrix}$ qui vérifie $\det N = \det M$ puis, afin d'avoir b en haut, la matrice $P = \begin{pmatrix} b & Y \\ a - qb & X - qY \end{pmatrix}$ qui vérifie $\det P = -\det M$.

Notant $Z = X - qY$, de sorte que $P = \begin{pmatrix} b & Y \\ r & Z \end{pmatrix}$, nous avons $X = Z + qY$. Ceci justifie la construction utilisée lors de la phase de remontée.

Pour passer à une nouvelle preuve, notons que $\begin{pmatrix} b & Y \\ r & Z \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & -q \end{pmatrix} \cdot \begin{pmatrix} a & X \\ b & Y \end{pmatrix}$ puis introduisons les notations suivantes.

- $r_0 = a$, $r_1 = b$, $Z_0 = X$ et $Z_1 = Y$ où Z_0 et Z_1 ne sont pas connus pour le moment.
- Pour $k \in \mathbb{N}^*$, on note $r_{k-1} = r_k q_k + r_{k+1}$ la division euclidienne de r_{k-1} par r_k, puis ensuite on pose $Z_{k+1} = Z_{k-1} - Z_k q_k$ de sorte que $Z_{k-1} = Z_k q_k + Z_{k+1}$.
- On note enfin $M_k = \begin{pmatrix} r_k & Z_k \\ r_{k+1} & Z_{k+1} \end{pmatrix}$ pour $k \in \mathbb{N}$ et $Q_k = \begin{pmatrix} 0 & 1 \\ 1 & -q_k \end{pmatrix}$ pour $k \in \mathbb{N}^*$ de sorte que nous avons $M_{k+1} = Q_{k+1} \cdot M_k$ pour $k \in \mathbb{N}$. Il est immédiat que Q_k est inversible d'inverse $R_k = \begin{pmatrix} q_k & 1 \\ 1 & 0 \end{pmatrix}$ avec $\det R_k = -1$.

L'algorithme d'Euclide nous donne l'existence de $n \in \mathbb{N}$ un indice minimal tel que $r_{n+1} = 0$, et que pour cet indice nous avons $r_n = \operatorname{pgcd}(a\,;b)$.

Comme $M_n = \prod_{k=n}^{1} Q_k \cdot M_0$, nous avons $M_0 = \prod_{k=1}^{n} R_k \cdot M_n$ avec $M_0 = \begin{pmatrix} r_0 & Z_0 \\ r_1 & Z_1 \end{pmatrix} = \begin{pmatrix} a & X \\ b & Y \end{pmatrix}$ et $M_n = \begin{pmatrix} r_n & Z_n \\ 0 & Z_{n+1} \end{pmatrix} = \begin{pmatrix} \operatorname{pgcd}(a\,;b) & Z_n \\ 0 & Z_{n+1} \end{pmatrix}$.

Comme $\det M_0 = \pm 1 \det M_n$ car $\det R_k = -1$, il suffit de choisir $Z_{n+1} = 1$, avec Z_n quelconque, pour avoir $aY - bX = \pm \operatorname{pgcd}(a\,;b)$. Voilà comment découvrir la méthode visuelle vue précédemment où le choix particulier $Z_n = 0$ sert juste à simplifier les premiers calculs.

Remarque 2. *Comme Z_n peut être quelconque, nous pouvons produire une infinité de coefficients de Bachet-Bézout. En effet, les étapes de* « remontée » *sont du type $Z_{k-1} = Z_k q_k + Z_{k+1}$ avec toujours $q_k > 0$, donc des valeurs naturelles non nulles* [2] *différentes de Z_n produiront des valeurs différentes de Z_0.*

1. Ce n'est qu'à la fin de la preuve que nous aurons effectivement prouver l'existence de u et v.
2. En fait, il n'est pas besoin d'imposer une condition particulière à Z_n. Voyez-vous pourquoi ?

4. Des coefficients via des algorithmes programmables

4.1. **La version « humaine » à la main.** Il n'est pas dur de coder directement la méthode humaine par descente puis remontée[3]. Voici un algorithme, peu efficace mais instructif, où $\star$ est un symbole à part, $R[-1]$ le dernier élément de la liste R et $R[-2]$ l'avant-dernier, et enfin $[x, y] + [r, s, t] \overset{\text{déf}}{=} [x, y, r, s, t]$ *(additionner des listes c'est les concaténer et donc* $R + [r]$ *est un raccourci pour* « on ajoute l'élément r à droite de la liste R »*).*

Algorithme 1 : Descente et remontée avec du papier et un crayon

Donnée : $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$ avec $a \geq b$
Résultat : $(u\,;v) \in \mathbb{Z} \times \mathbb{Z}$ tel que $au + bv = \text{pgcd}(a\,;b)$
Actions

```
# Phase de descente
# Q est une liste qui va stocker les quotients entiers q_k.
# R est une liste qui va stocker les restes r_k (rappelons que
# r_0 = a et r_1 = b).
```

$Q \leftarrow [\star]$; $R \leftarrow [a, b]$
Tant Que $R[-1] \neq 0$ **:**
  $\alpha \leftarrow R[-2]$; $\beta \leftarrow R[-1]$
  $\alpha = q\beta + r$ est la division euclidienne standard.
  $Q \leftarrow Q + [q]$; $R \leftarrow R + [r]$

```
# Phase de remontée
# Z est une liste qui va stocker les entiers tout à droite.
```

$\varepsilon \leftarrow 1$; $Z \leftarrow [1, 0]$; $c \leftarrow (-1)$
Tant Que $Q[c] \neq \star$ **:**
  $z \leftarrow Q[c] \cdot Z[-2] + Z[-1]$
  $Z \leftarrow Z + [z]$
  $\varepsilon \leftarrow (-\varepsilon)$; $c \leftarrow c - 1$

```
# On gère le signe devant le pgcd grâce à ε.
```

$u \leftarrow \varepsilon \cdot Z[-2]$; $v \leftarrow (-\varepsilon \cdot Z[-1])$
Renvoyer $(u\,;v)$

Nous avons traduit brutalement ce que l'on fait humainement mais à bien y regarder, la seule liste dont nous avons réellement besoin est Q. On peut donc proposer la variante suivante programmable qui est à la fois proche de la version de descente et remontée tout en limitant l'impact sur la mémoire.

3. Sur le lieu de téléchargement du document que vous lisez, se trouvent les fichiers `down.py` et `up.py` dans le dossier `bezout-coef-for-human/euclid2tikz`. Ces codes traduisent directement la méthode à la main par descente puis remontée.

Algorithme 2 : Descente et remontée moins mémophage

Donnée : $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$ avec $a \geq b$
Résultat : $(u\,;v) \in \mathbb{Z} \times \mathbb{Z}$ tel que $au + bv = \text{pgcd}(a\,;b)$
Actions

```
# Phase de descente
```
$Q \leftarrow [\star]$
Tant Que $b \neq 0$ **:**
  $a = qb + r$ est la division euclidienne standard.
  $Q \leftarrow Q + [q]$
  $a \leftarrow b$ **;** $b \leftarrow r$

```
# Phase de remontée
```
$u \leftarrow 1$ **;** $v \leftarrow 0$
$\varepsilon \leftarrow 1$ **;** $c \leftarrow (-1)$
Tant Que $Q[c] \neq \star$ **:**
  $temp \leftarrow Q[c]v + u$ **;** $u \leftarrow v$ **;** $v \leftarrow temp$
  $\varepsilon \leftarrow (-\varepsilon)$ **;** $c \leftarrow c - 1$
$u \leftarrow \varepsilon \cdot u$ **;** $v \leftarrow (-\varepsilon \cdot v)$
Renvoyer $(u\,;v)$

4.2. **La version « humaine » via les matrices.** Voici la version matricielle de l'algorithme de remontée et descente où de nouveau on limite l'impact sur la mémoire. La matrice R correspond au produit cumulé des matrices R_k.

Algorithme 3 : Descente et remontée via les matrices

Donnée : $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$ avec $a \geq b$
Résultat : $(u\,;v) \in \mathbb{Z} \times \mathbb{Z}$ tel que $au + bv = \text{pgcd}(a\,;b)$
Actions

$a_0 \leftarrow a$ **;** $\varepsilon \leftarrow 1$
$R \leftarrow \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Tant Que $b \neq 0$ **:**
  $a = qb + r$ est la division euclidienne standard.
  $R \leftarrow R \cdot \begin{pmatrix} q & 1 \\ 1 & 0 \end{pmatrix}$
  $a \leftarrow b$ **;** $b \leftarrow r$
  $\varepsilon \leftarrow (-\varepsilon)$
$R \leftarrow R \cdot \begin{pmatrix} a_0 & 0 \\ 0 & \varepsilon \end{pmatrix}$
$u \leftarrow R_{22}$ **;** $v \leftarrow (-R_{12})$
Renvoyer $(u\,;v)$

4.3. **Tailles des coefficients lors de la remontée.** Nous redonnons la représentation symbolique complète, ceci afin de rappeler les notations utilisées.

$$\begin{array}{ccc} & r_0 = a & Z_0 = X \\ q_1 & r_1 = b & Z_1 = Y \\ q_2 & r_2 & Z_2 \\ \vdots & \vdots & \vdots \\ q_n & r_n & Z_n = 0 \\ & 0 & 1 \end{array}$$

Représentation symbolique au complet.

Pour $k \in [\![1\,;n]\!]$, nous savons que $Z_{k-1} = q_k Z_k + Z_{k+1}$ avec $(Z_n\,;Z_{n+1}) = (0\,;1)$, et aussi que $r_{k-1} = q_k r_k + r_{k+1}$ avec $(r_n\,;r_{n+1}) = (\text{pgcd}(a\,;b)\,;0)$.

- Le cas minimal est $n = 1$ puisque $1 \overset{\text{hyp}}{\leq} b \overset{\text{hyp}}{\leq} a$ *(en fait $n = 1$ correspond au cas où $b \mid a$)*. Nous devons donc calculer au moins un quotient q_k.
- Nous avons clairement $0 \leq Z_n < r_n$.
- Comme $Z_{n-1} = q_n Z_n + Z_{n+1} = 1$, $r_{n-1} = q_n r_n + r_{n+1} = q_n r_n$ et $r_n < r_{n-1}$ par définition de la division euclidienne standard, nous avons $1 \leq Z_{n-1} < r_{n-1}$. Notons au passage que $q_n \geq 2$.
- Supposons maintenant que $n > 1$. Comme $Z_{n-2} = q_{n-1}Z_{n-1} + Z_n$, il est clair que $Z_{n-2} \geq 1$. De plus, nous avons :
$$\begin{aligned} Z_{n-2} &= q_{n-1}Z_{n-1} + Z_n \\ &< q_{n-1}r_{n-1} + r_n \\ &= r_{n-2} \end{aligned}$$
- Une récurrence descendante finie nous donne que $\forall k \in [\![0\,;n]\!]$, $0 \leq Z_k < r_k$.

D'après ce qui précède et comme de plus la suite finie $(r_k)_{0\leq k\leq n+1}$ est décroissante, nous savons que $\forall k \in [\![1\,;n]\!]$, $0 \leq Z_k < r_1 = b$ et $0 \leq Z_0 < r_0 = a$. En particulier $(u\,;v) = \pm(Z_1\,;-Z_0)$ qui est tel que $au + bv = \text{pgcd}(a\,;b)$ vérifie aussi $0 \leq |u| < b$ et $0 \leq |v| < a$.

Remarque 3. *Le résultat précédent empêche toute explosion en taille des calculs intermédiaires. Ceci est une très bonne chose !*

Remarque 4. *Il n'est pas dur de vérifier que la suite $(Z_k)_{0\leq k\leq n}$ est strictement décroisante.*

Remarque 5. *Notant $d = \text{pgcd}(a\,;b)$, nous avons en fait $0 \leq |u| < \frac{b}{2d}$ et $0 \leq |v| < \frac{a}{2d}$, et plus généralement $\forall k \in [\![1\,;n]\!]$, $0 \leq Z_k < \frac{b}{2d}$. Ceci vient des deux constatations suivantes.*

(1) Tout d'abord en notant que $r_n \leq \frac{1}{2}r_{n-1}$, nous avons $0 \leq |u| < \frac{b}{2}$ et $0 \leq |v| < \frac{a}{2}$.

(2) Posons $a' = \frac{a}{d}$ et $b' = \frac{b}{d}$. Les suites $(q'_k)_k$ et $(r'_k)_k$ associées à a' et b' sont tout simplement $(q_k)_k$ et $\left(\frac{r_k}{d}\right)_k$, la deuxième suite n'étant pas utilisée pour la phase de remontée. Pour comprendre il faut commencer par noter que si $a = bq + r$ désigne la division euclidienne standard alors $\frac{a}{d} = \frac{b}{d}q + \frac{r}{d}$ en est aussi une.

4.4. **Pas terribles...** Les algorithmes 2 et 3 vus ci-dessus sont informatiquement très maladroits. Voici pourquoi.

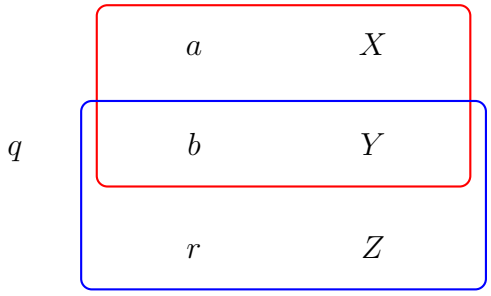
(1) L'algorithme 2 utilise une liste de taille la somme de 1 et du nombre d'étapes de l'algorithme d'Euclide pour calculer $\mathrm{pgcd}(a\,;b)$. Dans la section 7, nous verrons que ce nombre d'étapes est environ égal à 5 fois le nombre de chiffres de l'écriture décimale du plus petit des deux entiers a et b. Donc si l'on travaille avec des entiers de tailles assez grandes, la taille de la liste risque de devenir problématique sur du matériel où l'usage de la mémoire est critique *(penser aux objets connectés)*.

(2) Le problème avec l'algorithme 3 est le produit cumulé des matrices qui cache beaucoup d'opérations intermédiaires. Il serait bien de pouvoir s'en passer !

Dans la section qui suit nous allons voir que l'on peut chercher plus efficacement des coefficients de Bachet-Bézout, et ceci sans faire appel ni à des raisonnements avancés, ni à un algorithme complexe dans sa structure.

5. Un algorithme classique bien plus efficace

5.1. **On peut faire mieux !** Dans la section 3.1, nous avons vu que la clé de la réussite de l'algorithme de descente et remontée est l'égalité $aY - bX = bZ - rY$ dans la représentation ci-dessous où $a = qb + r$ est la division euclidienne standard et $X = qY + Z$.



Calculs faits dans les deux phases.

Nous avons donc exhibé un invariant et dès que l'on arrive à $r = 0$, c'est à dire à la fin de la phase de descente, nous pouvons avoir $bZ - rY = \text{pgcd}(a\,;b)$ grâce au choix $Z = 1$, et du coup en remontant les calculs nous arrivons à nos fins *(au signe près)*.

La méthode précédente est peu efficace à cause de la nécessité de mémoriser certains calculs pour la phase de remontée. Ceci est une contrainte forte ! Nous allons essayer de nous passer de cette nécessité de mémoriser des choses. Pour cela repartons de la représentation symbolique *« complète »* ci-dessous où $r_{n+1} = 0$ et $r_n = \text{pgcd}(a\,;b)$.

$$\begin{array}{ccc} & r_0 = a & Z_0 \\ q_1 & r_1 = b & Z_1 \\ q_2 & r_2 & Z_2 \\ \vdots & \vdots & \vdots \\ q_n & r_n & Z_n \\ & r_{n+1} & Z_{n+1} \end{array}$$

Représentation symbolique au complet où $r_{n+1} = 0$.

Ce qui fait fonctionner l'algorithme de descente puis remontée c'est que les r_k et les Z_k vérifient la même relation de récurrence.

(1) $r_{k+2} = r_k - q_{k+1}r_{k+1}$ car $r_k = q_{k+1}r_{k+1} + r_{k+2}$ est la division euclidienne standard.

(2) $Z_{k+2} = Z_k - q_{k+1}Z_{k+1}$ soit $Z_k = q_{k+1}Z_{k+1} + Z_{k+2}$ par définition.

Nous allons essayer de construire deux suites (u_k) et (v_k) telles que $au_k + bv_k = r_k$ car nous aurons alors la relation de Bachet-Bézout $au_n + bv_n = r_n = \text{pgcd}(a\,;b)$. Étant donné ce qui précède, il est maintenant naturel de supposer que $u_{k+2} = u_k - q_{k+1}u_{k+1}$ et $v_{k+2} = v_k - q_{k+1}v_{k+1}$.

En effet, ceci nous donne :

$$\begin{aligned} au_{k+2} + bv_{k+2} &= a(u_k - q_{k+1}u_{k+1}) + b(v_k - q_{k+1}v_{k+1}) \\ &= au_k + bv_k - q_{k+1}(au_{k+1} + bv_{k+1}) \\ &= r_k - q_{k+1}r_{k+1} \\ &= r_{k+2} \end{aligned}$$

Il nous reste à trouver les valeurs initiales. Ceci est immédiat puisque nous avons :

(1) $au_0 + bv_0 = r_0 = a$ donc $(u_0\,;v_0) = (1\,;0)$ s'impose.
(2) $au_1 + bv_1 = r_1 = b$ donc $(u_1\,;v_1) = (0\,;1)$ s'impose.

Notons que nécessairement $n \geq 1$. Nous voilà prêts à proposer un algorithme classique et efficace pour déterminer des coefficients de Bachet-Bézout.

Algorithme 4 : Classique et efficace

Donnée : $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$ avec $a \geq b$
Résultat : $(u\,;v) \in \mathbb{Z} \times \mathbb{Z}$ tel que $au + bv = \text{pgcd}(a\,;b)$
Actions
  $u' \leftarrow 1$; $u'' \leftarrow 0$
  $v' \leftarrow 0$; $v'' \leftarrow 1$
  Tant Que $b \neq 0$ **:**
    $a = qb + r$ est la division euclidienne standard.
    $temp_u \leftarrow u' - qu''$; $u' \leftarrow u''$; $u'' \leftarrow temp_u$
    $temp_v \leftarrow v' - qv''$; $v' \leftarrow v''$; $v'' \leftarrow temp_v$
  Renvoyer $(u'\,;v')$

5.2. **Tailles des coefficients de la méthode efficace.** [4]
Dans l'algorithme précédent nous avons défini les suites (u_k) et (v_k) par les relations de récurrence $u_{k+2} = u_k - q_{k+1}u_{k+1}$ et $v_{k+2} = v_k - q_{k+1}v_{k+1}$ couplées avec les conditions initiales $(u_0\,;v_0) = (1\,;0)$ et $(u_1\,;v_1) = (0\,;1)$ où les q_k sont les quotients intermédiaires de l'algorithme d'Euclide. Ces suites fournissent les coefficients de Bachet-Bézout u_n et v_n.

Reprenons une interprétation matricielle comme nous l'avions fait de façon féconde pour l'algorithme de descente puis remontée. Posant $A_k = \begin{pmatrix} u_k & v_k \\ u_{k+1} & v_{k+1} \end{pmatrix}$, nous avons $A_{k+1} = Q_k A_k$ où $Q_k = \begin{pmatrix} 0 & 1 \\ 1 & -q_k \end{pmatrix}$. Ceci nous fournit l'invariant $\det A_{k+1} = -\det A_k = \pm \det A_0 = \pm 1$ [5].

Commençons par tenter d'évaluer la taille de u_{n+1}. Par construction, notant $d = \text{pgcd}(a\,;b)$, nous avons $A_n \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r_n \\ r_{n+1} \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix}$.

Posant ensuite $a' = \frac{a}{d}$ et $b' = \frac{b}{d}$, nous avons $A_n \begin{pmatrix} a' \\ b' \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Ceci nous permet d'avoir $\begin{pmatrix} a' \\ b' \end{pmatrix} = A_n^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \pm \begin{pmatrix} v_{n+1} & -v_n \\ -u_{n+1} & u_n \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \pm \begin{pmatrix} v_{n+1} \\ -u_{n+1} \end{pmatrix}$ grâce à $\det A_n = \pm 1$. Nous avons

4. Nous reprenons, en la précisant, la preuve page 50 du livre *« Cours de calcul formel – Algorithmes fondamentaux »* de Philippe Saux Picart aux éditions Ellipses.
5. Nous venons d'établir que $u_k v_{k+1} - u_{k+1} v_k = \pm 1$, une identité qui ne coule pas de source sans utiliser des matrices.

établi que $a' = \pm v_{n+1}$ et $b' = \mp u_{n+1}$. Il nous reste à *« remonter la récurrence »* pour déduire de $u_{n+1} = \pm b'$ des majorations des valeurs des précédents u_k. Le cas des v_k sera similaire à traiter comme nous allons le constater.

Une étude informatique [6] permet de conjecturer rapidement les choses suivantes.

- $\forall k \in [\![0\,;n+1]\!]$, $u_{2p} > 0$ si $k = 2p$ est pair, $u_{2p+1} \leq 0$ si $k = 2p+1$ est impair, et même $u_{2p+1} < 0$ si $2p+1 \geq 3$.
- La suite $(|u_k|)_{1\leq k\leq n+1}$ semble être croissante. Si tel est le cas, nous aurons la majoration $|u_k| \leq b'$ pour $k \in [\![1\,;n+1]\!]$ *(en réalité, nous allons faire un peu mieux).*

Commençons par démontrer par récurrence sur $p \in \mathbb{N}$ que si $2p \leq n+1$ alors $u_{2p} > 0$, et si $2p+1 \leq n+1$ alors $u_{2p+1} \leq 0$ avec aussi $u_{2p+1} < 0$ si de plus $2p+1 \geq 3$.

- **Cas de base :** comme $(u_0\,;u_1) = (1\,;0)$, nous avons bien le début de la récurrence.
- **Hérédité :** supposons avoir $2(p+1) \leq n+1$. Comme $2p < 2p+1 \leq n$, nous avons par hypothèse de récurrence $u_{2p} \geq 0$ et $u_{2p+1} \leq 0$.

  $u_{2p+2} = u_{2p} - q_{2p+1}u_{2p+1}$ implique que $u_{2p+2} > -q_{2p+1}u_{2p+1} = |q_{2p+1}u_{2p+1}| > 0$. Nous obtenons au passage une information précise à savoir que $|u_{2p+2}| > q_{2p+1}|u_{2p+1}|$.

  Supposons ensuite avoir $2(p+1)+1 \leq n+1$. De nouveau $u_{2p} > 0$ et $u_{2p+1} \leq 0$ par hypothèse de récurrence.

  $u_{2p+3} = u_{2p+1} - q_{2p+2}u_{2p+2}$ implique que $u_{2p+3} < -q_{2p+2}u_{2p+2} \leq 0$. Nous obtenons au passage une information précise à savoir que $|u_{2p+3}| > q_{2p+2}|u_{2p+2}|$.

La preuve par récurrence précédente nous a fourni $|u_{k+1}| > q_k|u_k|$ dès que $k \in [\![1\,;n]\!]$. Ceci implique la croissance stricte de la suite $(|u_k|)_{1\leq k\leq n+1}$. De plus dans la section 4.3, nous avons vu que $q_n \geq 2$. Combiné avec $|u_{n+1}| > q_n|u_n|$, ceci nous donne $|u_n| < \frac{1}{2}|u_{n+1}| = \frac{b'}{2}$ puis $\forall k \in [\![1\,;n]\!]$, $|u_k| < \frac{b'}{2}$.

On prendra garde au cas particulier de $|u_0| = 1$. En fait, cela est sans intérêt car nous avons forcément $n \geq 1$ et donc on se soucie peu du cas particulier de $|u_0|$.

Le cas de la suite (v_k) est similaire à traiter, sans particularité pour v_0, via la propriété suivante à démontrer : $\forall p \in \mathbb{N}$, si $2p \leq n+1$ alors $v_{2p} \leq 0$ avec de plus $u_{2p} < 0$ dès que $2p \geq 2$, et si $2p+1 \leq n+1$ alors $v_{2p+1} > 0$. On obtient alors $\forall k \in [\![0\,;n]\!]$, $|v_k| < \frac{a'}{2}$.

En résumé, nous n'avons de nouveau pas d'explosion des tailles des nombres u_k et v_k. Ceci rend donc l'algorithme 4 est à la fois efficace dans sa gestion de la mémoire et peu gourmand en calculs intermédiaires.

6. Sur le lieu de téléchargement de ce document, voir le fichier `Python` ayant pour chemin relatif `bezout-coef-for-human/algo-efficient/bachetbezout.py`.

6. Une infinité de coefficients de Bachet-Bézout

Les algorithmes 2, 3 et 4 donnent chacun l'existence de coefficients de Bachet-Bézout u et v pour $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$ avec $a \geq b$ de sorte que l'on ait $au + bv = d$ où $d = \text{pgcd}(a\,;b)$.

Considérons un autre couple $(x\,;y) \in \mathbb{Z} \times \mathbb{Z}$ tel que $ax + by = d$. Par soustraction, nous avons $a(x-u) + b(y-v) = 0$ soit $a(x-u) = -b(y-v)$ puis $a'(x-u) = -b'(y-v)$ en posant $a' = \frac{a}{d}$ et $b' = \frac{b}{d}$.

Comme $\text{pgcd}(a'\,;b') = 1$, d'après le lemme de Gauss $a' \mid (y-v)$ soit $\exists k \in \mathbb{Z}$ tel que $y - v = ka'$ d'où $a'(x-u) = -b'ka'$ puis $x - u = -kb'$.

En résumé, il est nécessaire que $x = u - kb'$ et $y = v + ka'$ où $k \in \mathbb{Z}$. Cette condition étant clairement suffisante, nous savons que tous les coefficients de Bachet-Bézout sont du type $(x\,;y) = (u - k\frac{b}{d}\,;v + k\frac{a}{d})$ avec $k \in \mathbb{Z}$.

Remarque 6. *Comme pour deux valeurs consécutives de $k \in \mathbb{Z}$, les $(u - kb')$ et $(v + ka')$ associés ne différent en valeur absolue que de b' et a' respectivement, nous pouvons affirmer qu'il n'existe qu'un seul couple $(u\,;v)$ de coefficients de Bachet-Bézout vérifiant $|u| < \frac{b'}{2}$ et $|v| < \frac{a'}{2}$.*

7. Nombre d'étapes de l'algorithme d'Euclide

Tous les algorithmes vus précédemment s'appuient sur l'algorithme d'Euclide. Nous allons donc chercher à évaluer le nombre d'étapes de l'algorithme d'Euclide ce qui permettra d'estimer, sans effort, la complexité des algorithmes présentés pour déterminer des coefficients de Bachet-Bézout.

7.1. **Une première estimation.** Dans la représentation ci-dessous, $(q_k)_{1\leq k\leq n}$ est la suite des quotients et $(r_k)_{2\leq k\leq n}$ celle des restes fournis par l'algorithme d'Euclide.

$$\begin{array}{cc} & r_0 = a \\ q_1 & r_1 = b \\ q_2 & r_2 \\ \vdots & \vdots \\ q_n & r_n \neq 0 \\ & 0 \end{array}$$

L'algorithme d'Euclide au complet où $n \geq 1$ forcément.

Nous savons que $(q_k)_{1\leq k\leq n} \subseteq \mathbb{N}^*$ avec $q_n \geq 2$, et que $(r_k)_{1\leq k\leq n} \subseteq \mathbb{N}^*$ est strictement décroissante. Comme $r_k = q_{k+1}r_{k+1} + r_{k+2}$ si $1 \leq k \leq n-2$, nous avons $r_k \geq r_{k+1} + r_{k+2} > 2r_{k+2}$ d'où $r_k \geq 2r_{k+2} + 1$ dès que $1 \leq k \leq n-2$. Ceci nous donne les majorations suivantes.

- **Cas $n = 2p \geq 2$ est pair.** Comme $r_{2p} = r_n \geq 1$, nous avons :
  $$\begin{aligned} b &> r_2 \\ &\geq 2r_4 + 1 \\ &\geq 4r_6 + 2 + 1 \\ &\dots \\ &\geq 2^{p-1}r_{2p} + \sum_{i=0}^{p-2} 2^i \\ &\geq 2^{p-1} + 2^{p-1} - 1 \\ &= 2^p - 1 \end{aligned}$$
  Nous avons donc $2^p \leq b$ puis $\log(2^p) \leq \log b$ et $p \leq \frac{\log b}{\log 2}$ où log désigne le logarithme décimal. Donc $n \leq \frac{2}{\log 2} \cdot \log b$ ici.
- **Cas $n = 2p+1$ est impair.** Comme $r_{2p+1} = r_n \geq 1$, nous avons :
  $$\begin{aligned} b &= r_1 \\ &> 2r_3 + 1 \\ &\dots \\ &\geq 2^p r_{2p+1} + \sum_{i=0}^{p-1} 2^i \end{aligned}$$

$b > 2^p + 2^p - 1$
$\geq 2^{p+1} - 1$
Nous avons donc $2^{p+1} \leq b$ puis $p + 1 \leq \frac{\log b}{\log 2}$. Donc $n \leq \frac{2}{\log 2} \cdot \log b - 1$ ici.

Dans les deux cas, $n \leq \frac{2}{\log 2} \cdot \log b$. Comme $\frac{2}{\log 2} \approx 6,65$, notant d le nombre de chiffres décimaux de b, de sorte que $\log b < d$, nous avons l'estimation $n < 7d$.

En résumé, l'algorithme d'Euclide appliqué à $(a\,;b) \in \mathbb{N}^* \times \mathbb{N}^*$ avec $a \geq b$ demandera au maximum $7d - 1$ étapes où d est le nombre de chiffres décimaux de b.

Remarque 7. *Cette estimation, rapide à établir, montre que le nombre d'étapes augmente de façon logarithmique par rapport à b le plus petit des entiers naturels a et b auxquels est appliqué l'algorithme d'Euclide.*

7.2. **L'estimation de Lamé.** En fait, il est assez facile d'améliorer l'estimation précédente. Prenons le schéma suivant où les deux colonnes de droite donnent des minorants évidents des restes fournis par l'algorithme d'Euclide. Les deux colonnes de droite utilisent la construction via des divisions euclidiennes *« inversées »* avec des quotients les plus petits possibles, à savoir tous égaux à 1 *(voir la colonne tout à droite)*.

	$r_0 = a$			
q_1	$r_1 = b$	$\geq$	f_n	1
q_2	r_2	$\geq$	f_{n-1}	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
q_{n-1}	r_{n-1}	$\geq$	$f_2 = 2$	1
$q_n \geq 2$	$r_n \neq 0$	$\geq$	$f_1 = 1$	2
	0	$\geq$	$f_0 = 0$	

Estimer au mieux le nombre d'étapes de l'algorithme d'Euclide.

Pour les fans de Nicolas B. [7], voici une démonstration formelle de toutes les minorations. Tout d'abord par définition, la suite $(f_k)_{0\leq k\leq n}$ est définie par la condition initiale $(f_0\,;f_1) = (0\,;1)$ puis $f_2 = 2f_1 + f_0$ et la relation de récurrence $f_{k+2} = f_{k+1} + f_k$ pour $k \in [\![3\,;n-2]\!]$. Démontrons par récurrence sur $k \in [\![0\,;n]\!]$ que $f_k \leq r_{n+1-k}$ en nous souvenant que $n \geq 1$. L'hypothèse de récurrence sera que l'inégalité est vérifiée pour tous les indices i tels que $i \leq k$.

- *Cas de base pour $k \leq 2$.* Il est clair que $f_0 \leq r_{n+1}$ et $f_1 \leq r_n$. Ensuite $r_{n-1} = q_n r_n + r_{n+1}$ avec $q_n \geq 2$ donne $r_{n-1} \geq 2r_n + r_{n+1} \geq 2f_1 + f_0 = f_2$. Ceci achève la preuve des cas de base.

7. Alias Nicolas Bourbaki.

- *Hérédité pour* $k \in [\![3\,;n-3]\!]$. Nous avons $r_{n+1-(k+1)} = r_{n-k} = q_{n-k+1}r_{n-k+1} + r_{n-k+2}$ avec $q_n \geq 1$. Or $n-k+1 = n+1-k$ et $n-k+2 = n+1-(k-1)$ donc l'hypothèse de récurrence et $q_n \geq 1$ donnent $r_{n+1-(k+1)} \geq r_{n-k+1} + r_{n-k+2} \geq f_k + f_{k-1} = f_{k+1}$, la dernière égalité venant de $k \geq 3$. Ceci établit bien l'inégalité au rang $(k+1)$ et donc pour tous les rangs i tels que $i \leq k+1$.

Poursuivons en notant que le 1er terme f_0 de la suite f ne jouera pas un rôle particulier dans l'évaluation du nombre d'étapes. Ceci permet de remplacer la suite f par la bien connue et très classique suite de Fibonacci [8] F définie par les conditions initiales $F_0 = F_1 = 1$ et la relation de récurrence $F_{k+2} = F_{k+1} + F_k$ puisque $\forall k \in [\![1\,;n]\!]$, $F_k = f_k$.

Nous pouvons donc affirmer que $n \leq \max\left\{k \in \mathbb{N}^* \mid F_k \leq b\right\}$. Le cas où $a = f_n + f_{n-1}$ et $b = f_n$ montre que l'on peut pas espérer faire mieux ! Nous allons estimer ce maximum de deux façons.

Méthode 1. Considérons les suites non nulles du type $(q^k)_{k\in\mathbb{N}}$ telles que $q^{n+2} = q^{n+1} + q^n$. Il est facile de montrer qu'il n'y en a que de deux types, à savoir les suites $(\phi^k)_{k\in\mathbb{N}}$ et $(\psi^k)_{k\in\mathbb{N}}$ où $\psi = \frac{1-\sqrt{5}}{2}$ et $\phi = \frac{1+\sqrt{5}}{2}$, le nombre d'or, sont les deux solutions de l'équation $x^2 = x+1$. Nous avons alors les faits suivants.

(1) $F_0 = F_1 = 1 = \phi^0$
(2) $F_2 = 2 = \frac{1+\sqrt{9}}{2} > \phi$
(3) $F_3 = F_2 + F_1 > \phi^1 + \phi^0 = \phi^2$
(4) $F_4 = F_3 + F_2 > \phi^2 + \phi^1 = \phi^3$ ...

Une récurrence immédiate à faire nous donne $\forall k \in \mathbb{N}^*$, $F_k > \phi^{k-1}$ de sorte que $F_n \leq b$ implique $\phi^{n-1} < b$ puis $n < 1 + \frac{\log b}{\log \phi}$. Notant d le nombre de chiffres décimaux de b, nous avons $n < 1 + \frac{d}{\log \phi}$. Ensuite $\frac{1}{\log \phi} \approx 4,78$ donne $n < 1 + 5d$ puis $n \leq 5d$ ce qui est un peu mieux que la première estimation $n < 7d$.

Remarque 8. *On peut démontrer que* $\forall k \in \mathbb{N}$, $F_k = \frac{\phi^{k+1} - \psi^{k+1}}{\sqrt{5}}$. *Pour cela, on prouve l'existence de* $(m\,;p) \in \mathbb{R}^2$ *tel que la suite* u *de terme générale* $u_k = m\phi^k + p\psi^k$ *vérifie* $u_0 = u_1 = 1$. *Il est alors facile de conclure.*

Méthode 2. Pauvres de nous qui ne connaissons pas le nombre d'or. Que faire ? Examinons les 24 premières valeurs [9] de la suite F.

$F_0 = 1$	$F_8 = 34$	$F_{16} = 1597$
$F_1 = 1$	$F_9 = 55$	$F_{17} = 2584$
$F_2 = 2$	$F_{10} = 89$	$F_{18} = 4181$
$F_3 = 3$	$F_{11} = 144$	$F_{19} = 6765$
$F_4 = 5$	$F_{12} = 233$	$F_{20} = 10946$
$F_5 = 8$	$F_{13} = 377$	$F_{21} = 17711$
$F_6 = 13$	$F_{14} = 610$	$F_{22} = 28657$
$F_7 = 21$	$F_{15} = 987$	$F_{23} = 46368$

Un peu d'observation montre que $\forall k \in \mathbb{N}^*$, $F_{k+5} > 10F_k$ semble être vraie. Une telle inégalité a l'utilité de nous donner une information sur la taille décimale des F_k. Cette conjecture se

8. On entend souvent à tors dire que la suite de Fibonacci donne la complexité au pire de l'algorithme d'Euclide. Ce n'est pas exactement vrai à cause du tout dernier reste nul.

9. Les valeurs ont été fournies par le fichier `explore.py` disponible sur le lieu de téléchargement du document que vous lisez : voir le dossier `bezout-coef-for-human/fibo`.

consolide facilement via un programme informatique [10]. Il nous reste à la vérifier à l'aide d'un raisonnement direct.

- *1^er cas : $k = 1$.* Nous avons bien $10F_1 = 10 < 13 = F_6$.
- *2^e cas : $k \geq 2$.* Nous avons ici :

  $$\begin{aligned} F_{k+5} &= F_{k+4} + F_{k+3} \\ &= 2F_{k+3} + F_{k+2} \\ &= 3F_{k+2} + 2F_{k+1} \\ &= 5F_{k+1} + 3F_k \\ &= 8F_k + 5F_{k-1} \quad \textit{(en se souvenant que } k \geq 2\textit{)} \end{aligned}$$

  Par décroissance de la suite F et comme $k \geq 2$, nous avons $F_k = F_{k-1} + F_{k-2} \leq 2F_{k-1}$. Comme de plus $F_{k-1} > 0$ puisque $k \geq 2$, nous obtenons :

  $$\begin{aligned} F_{k+5} &= 8F_k + 4F_{k-1} + F_{k-1} \\ &> 8F_k + 4F_{k-1} \\ &\geq 8F_k + 2F_k \\ &= 10F_k \end{aligned}$$

Comme $F_1 = 1$, le résultat précédent nous donne que $\forall j \in \mathbb{N}$, $\forall i \in [\![1\,;5]\!]$, $10^j \leq F_{i+5j}$. Autrement dit, F_{i+5j} s'écrit avec au moins $(j+1)$ chiffres décimaux. Ceci se démontre via la récurrence facile suivante sur j avec i fixé.

- *Cas de base pour $j = 0$.* Nous avons bien $10^0 = 1 \leq F_i$ si $i \in [\![1\,;5]\!]$.
- *Hérédité pour $j \in \mathbb{N}$.* Supposons avoir $10^j \leq F_{i+5j}$, nous avons alors comme souhaité : $F_{i+5(j+1)} = F_{(i+5j)+5} > 10F_{i+5j} \geq 10^{j+1}$.

Notant d le nombre de chiffres décimaux de b, comme $b < 10^d$ et $F_n \leq b$, nous avons $F_n < 10^d$. Soit $n = i + 5j$ la division euclidienne de n par 5, nous avons alors $10^j < 10^d$ puis $j < d$ et $n < i + 5d \leq 5(d+1)$ d'où $n \leq 5d + 4$. C'est moins bien que le résultat de la méthode 1.

En fait, on peut améliorer l'estimation en raisonnant comme suit [11]. Si nous avions $n > 5d$, soit $n \geq 5d + 1$, alors par croissance de la suite F, nous aurions $F_n \geq F_{5d+1} \geq 10^d$ qui contredirait $F_n < 10^d$.

10. Voir le fichier `conjecture.py` dans le dossier `bezout-coef-for-human/fibo` présent sur le lieu de téléchargement du document que vous lisez.

11. L'auteur a fait le choix de laisser l'estimation un peu grossière $n \leq 5d + 4$ en raisonnant comme s'il ne connaissait pas le résultat de la méthode 1 afin de rendre la méthode 2 auto-suffisante. Le fait que l'on puisse améliorer l'estimation grossière peut se conjecturer par des expériences numériques informatiques.