Pythagore

August 24, 2023

1 Triplets pythagoriciens

Marc Lorenzi

20 août 2023

[1]:
```
import matplotlib.pyplot as plt
import random
```

1.1 1. Introduction

1.1.1 1.1 Qu'est-ce qu'un triplet pythagoricien ?

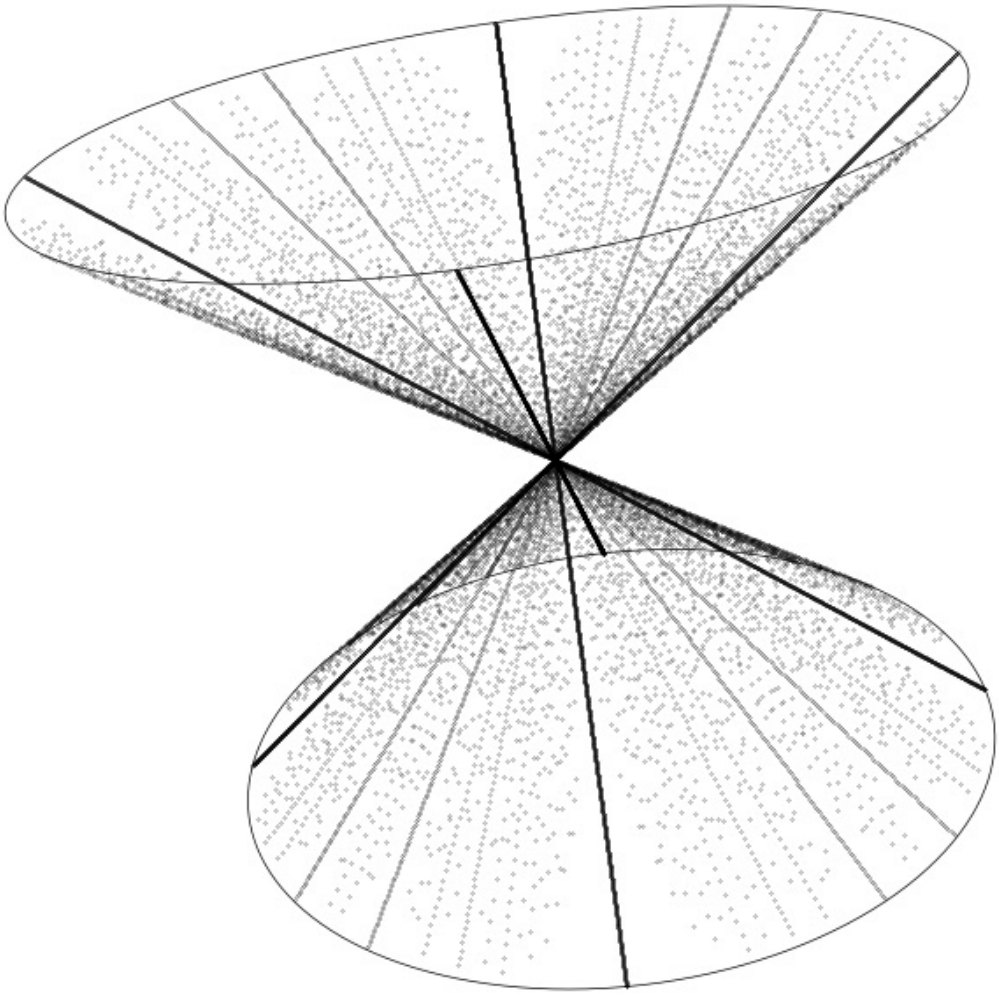
L'un des théorèmes les plus connus des mathématiques est sans doute le *théorème de Pythagore*. Celui-ci dit que si un triangle rectangle a une hypothénuse de longueur c, et que les côtés de l'angle droit sont de longueurs a et b, alors $a^2 + b^2 = c^2$. Très tôt, la question suivante s'est posée : existe-t-il des triangles rectangles dont les côtés ont des longueurs *entières* ?

Définition. Le triplet $(a, b, c) \in \mathbb{Z}^3$ est un *triplet pythagoricien* si

$$a^2 + b^2 = c^2$$

Notation. Nous noterons $\mathcal{P}$ l'ensemble des triplets pythagoriciens.

La figure ci-dessous représente les triplets pythagoriciens (a, b, c) tels que $-1000 \leq a, b, c \leq 1000$. Ces points de l'espace se trouvent sur le cône d'équation $x^2 + y^2 = z^2$.



La fonction `est_pythagoricien` prend en paramètre un triplet u d'entiers relatifs. Elle renvoie `True` si u est pythagoricien et `False` sinon.

[2]:
```
def est_pythagoricien(u):
    a, b, c = u
    return a ** 2 + b ** 2 == c ** 2
```

Par exemple, $(3, 4, 5)$ et $(4, 3, 5)$ sont des triplets pythagoriciens. En voici un autre moins évident.

[3]:
```
est_pythagoricien((12045691, 5437620, 13216141))
```

[3]:
```
True
```

Pour l'instant nous ne savons pas fabriquer de triplets pythagoriciens, alors gardons précieusement le triplet ci-dessus. Nous l'appellerons u_0.

[4]:
```
u0 = (12045691, 5437620, 13216141)
```

Dans ce notebook, nous allons commencer par explorer quelques propriétés des élémments de $\mathcal{P}$. Puis nous verrons comment, lorsqu'on possède un élément de $\mathcal{P}$, fabriquer d'autres éléments de $\mathcal{P}$. Enfin, nous verrons que notre méthode permet de fabriquer **TOUS** les éléments de $\mathcal{P}$.

1.1.2 1.2 Triplets triviaux, triplets primitifs

Définition. Un triplet pythagoricien (a, b, c) est *trivial* si a ou b est nul.

Les triplets triviaux sont les triplets $(\pm t, 0, \pm t)$ et $(0, \pm t, \pm t)$ où $t \in \mathbb{N}$. Ces triplets ne sont évidemment pas très intéressants.

Les trois propositions qui suivent sont faciles à prouver. Elles nous montrent que que si on connaît un triplet pythagoricien, il est facile d'en trouver beaucoup d'autres.

Proposition. Soit $(a, b, c) \in \mathcal{P}$. Alors, $(\pm a, \pm b, \pm c) \in \mathcal{P}$.

Notation. Nous noterons $\mathcal{P}^+$ l'ensemble des triplets pythagoriciens (a, b, c) tels que $a, b, c \in \mathbb{N}^*$.

[5]:
```
def est_pythagoricien_positif(u):
    a, b, c = u
    return est_pythagoricien(u) and a > 0 and b > 0 and c > 0
```

Proposition. Soit $(a, b, c) \in \mathcal{P}$. Soit $k \in \mathbb{Z}$. - $(b, a, c) \in \mathcal{P}$. - $(ka, kb, kc) \in \mathcal{P}$.

Proposition. Soit $(a, b, c) \in \mathcal{P}$. Soit $d \in \mathbb{Z}^*$. On suppose que d divise a, b et c. Alors,

$$\left(\frac{a}{d}, \frac{b}{d}, \frac{c}{d}\right) \in \mathcal{P}$$

Si nous prenons $d = pgcd(a, b, c)$, nous obtenons un triplet pythagoricien dont les éléments sont premiers entre-eux. À partir d'un tel triplet, il est alors facile de reconstruire tous les triplets pythagoriciens. Ceci nous suggère la définition suivante.

Définition. Le triplet $(a, b, c) \in \mathcal{P}^+$ est *primitif* si a est impair et $pgcd(a, b, c) = 1$.

La condition « a est impair » peut paraître restrictive, mais elle ne l'est pas. En voici la raison.

Soit $(a, b, c) \in \mathcal{P}^+$ tel que $pgcd(a, b, c) = 1$. On ne peut pas avoir a et b tous les deux pairs, sinon c l'est aussi. En fait, on ne peut pas non plus avoir a et b impairs. En effet, $1^2 \equiv 3^2 \equiv 1 \bmod 4$. Donc, si a et b sont impairs, alors $a^2 + b^2 \equiv 2 \bmod 4$. Or, aucun carré n'est congru à 2 modulo 4 (car $0^2 \equiv 2^2 \equiv 0 \bmod 4$). Ainsi, un et un seul des deux triplets (a, b, c) et (b, a, c) est primitif.

Retenons donc que si (a, b, c) est un triplet primitif, a est impair, b est pair et, forcément, c est impair.

Notation. Nous noterons $\mathcal{P}_0$ l'ensemble des triplets primitifs.

Il est clair qu'il existe une infinité de triplets pythagoriciens (par exemple les triplets $(3k, 4k, 5k)$ où $k \in \mathbb{N}^*$). Mais existe-t-il une infinité de triplets pythagoriciens *primitifs* ? Nous verrons que oui, et d'ici la fin de ce notebook nous saurons les créer tous.

Écrivons une fonction `est_pythagoricien_primitif` qui renvoie `Tue` si son paramètre appartient à $\mathcal{P}_0$ et `False` sinon.

La fonction `pgcd` renvoie le pgcd des deux entiers a et b.

[6]:
```
def pgcd(a, b):
    while b != 0:
        a, b = b, a % b
    return a
```

La fonction `pgcd_liste` renvoie le pgcd des entiers de la liste (ou du n-uplet) s.

[7]:
```
def pgcd_liste(s):
    d = 0
    for a in s:
        d = pgcd(d, a)
    return d
```

[8]:
```
pgcd_liste((4, 8, 6))
```

[8]:
```
2
```

[9]:
```
def est_pythagoricien_primitif(u):
    return est_pythagoricien_positif(u) and u[0] % 2 == 1 and pgcd_liste(u) == 1
```

Qu'en est-il de notre triplet pythagoricien préféré ?

[10]:
```
print(u0)
print(est_pythagoricien_primitif(u0))
```

```
(12045691, 5437620, 13216141)
True
```

Résumons nos notations.

- $\mathcal{P}$ est l'ensemble des triplets pythagoriciens.
- $\mathcal{P}^+$ est l'ensemble des triplets pythagoriciens à coefficients strictement positifs.
- $\mathcal{P}_0$ est l'ensemble des triplets pythagoriciens primitifs.

On a les inclusions strictes

$$\mathcal{P}_0 \subset \mathcal{P}^+ \subset \mathcal{P}$$

1.2 2. Algèbre linéaire

1.2.1 2.1 Une symétrie de l'espace $\mathbb{R}^3$

Considérons l'endomorphimse σ de $\mathbb{R}^3$ dont la matrice dans la base canonique est

$$S = \begin{pmatrix} -1 & -2 & 2 \\ -2 & -1 & 2 \\ -2 & -2 & 3 \end{pmatrix}$$

Nous avons pour tout $(a, b, c) \in \mathbb{R}^3$, $\sigma(a, b, c) = (a', b', c')$ où

$$\begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = \begin{pmatrix} -a - 2b + 2c \\ -2a - b + 2c \\ -2a - 2b + 3c \end{pmatrix} = \begin{pmatrix} a + k \\ b + k \\ c + k \end{pmatrix}$$

où $k = -2a - 2b + 2c$.

Proposition. σ est la symétrie par rapport au plan d'équation $x + y - z = 0$, parallèlement à la droite engendrée par $(1, 1, 1)$.

Démonstration. On vérifie par un simple calcul matriciel que $S^2 = I_3$. Ainsi, σ est une symétrie. De plus, $\sigma(1, 1, 1) = -(1, 1, 1)$. Soit enfin u appartenant au plan d'équation $x + y - z = 0$. On a donc $u = (x, y, x + y)$ où $x, y \in \mathbb{R}$. Notons $(x', y', z') = \sigma(u)$. On a

$$\begin{aligned} x' &= -x - 2y + 2z = -x - 2y + 2(x + y) = x \\ y' &= -2x - y + 2z = -2x - y + 2(x + y) = y \\ z' &= -2x - 2y + 3z = -2x - 2y + 3(x + y) = x + y = z \end{aligned}$$

Ainsi, $\sigma(u) = u$.

[11]:
```
def sigma(u):
    a, b, c = u
    return (-a - 2 * b + 2 * c, -2 * a - b + 2 * c, -2 * a - 2 * b + 3 * c)
```

Reprenons (encore !) le triplet u_0. Quelle est son image par σ ?

[12]:
```
sigma(u0)
```

[12]:
```
(3511351, -3096720, 4681801)
```

Tentons une expérience.

[13]:
```
est_pythagoricien(sigma(u0))
```

[13]:
```
True
```

Coup de chance ? Non, pas du tout.

Proposition. Soit $(a, b, c) \in \mathcal{P}$. Alors, $\sigma(a, b, c) \in \mathcal{P}$.

Démonstration. Notons $(a', b', c') = \sigma(a, b, c)$. On a $a' = a + k$, $b' = b + k$ et $c' = c + k$ où

$$k = -2a - 2b + 2c$$

De là,

$$\begin{aligned} a'^2 + b'^2 - c'^2 &= (a+k)^2 + (b+k)^2 - (c+k)^2 \\ &= a^2 + b^2 - c^2 + k(2a + 2b - 2c) + k^2 \\ &= k(2a + 2b - 2c + k) \\ &= 0 \end{aligned}$$

Nous avons même mieux que cela.

Proposition. Soit $(a, b, c) \in \mathcal{P}$. Soit $(a', b', c') = \sigma(a, b, c)$. Alors, $pgcd(a', b', c') = pgcd(a, b, c)$.

Démonstration. Soit $d \in \mathbb{Z}$. Supposons que d divise a, b et c. Alors, d divise $-a - 2b + 2c = a'$. De même, d divise b' et d divise c'. Inversement, supposons que d divise a', b' et c'. Comme σ est bijective et $\sigma^{-1} = \sigma$, le même raisonnement montre que d divise a, b et c.

Proposition. Soit $(a, b, c) \in \mathcal{P}_0$. Soit $(a', b', c') = \sigma(a, b, c)$. Alors, a' est impair et b' est pair.

Démonstration. $a' = -a + 2(-b + c)$ est de même parité que a et $b' = -b + 2(-a + c)$ est de même parité que b.

1.2.2 2.2 Itérations

Si nous prenons un triplet primitif (a, b, c) et que nous appliquons σ, nous obtenons un triplet pythagoricien dont les éléments sont premiers entre-eux, mais peuvent être négatifs. En prenant leur valeur absolue, nous obtenons donc un triplet (a', b', c') primitif … sauf si $a' = 0$ ou $b' = 0$.

[14]:
```
def sigma_abs(u):
    a1, b1, c1 = sigma(u)
    return (abs(a1), abs(b1), abs(c1))
```

Tentons une expérience. Partons de u_0 et itérons `sigma_abs`.

[15]:
```
u = u0
while u[0] != 0 and u[1] != 0 and u[2] != 0:
    print(u)
    u = sigma_abs(u)
print(u)
```

```
(12045691, 5437620, 13216141)
(3511351, 3096720, 4681801)
(341189, 755820, 829261)
(194307, 220324, 293765)
(47425, 21408, 52033)
(13825, 12192, 18433)
(1343, 2976, 3265)
(765, 868, 1157)
(187, 84, 205)
```

```
(55, 48, 73)
(5, 12, 13)
(3, 4, 5)
(1, 0, 1)
```

Nous tenons clairement quelque chose. Si nous partons d'un triplet primitif et que nous itérons, nous obtenons de nouveaux triplets primitifs. Sur notre exemple, les valeurs des coefficients décroissent strictement et on finit par aboutir au triplet pythagoricien trivial $(1, 0, 1)$. Il nous reste à montrer que ceci est tout à fait général. Commençons par régler le cas des triplets triviaux.

Proposition. Soit $(a, b, c) \in \mathcal{P}_0$. Soit $(a', b', c') = \sigma(a, b, c)$. Le triplet (a', b', c') est trivial si et seulement si $(a, b, c) = (3, 4, 5)$.

Démonstration. Supposons (a', b', c') trivial. a' est de même parité que a et a est impair, donc $a' \neq 0$. De là, $b' = 0$ et $a' = \pm c'$.

Peut-on avoir $a' = c'$? Dans ce cas

$$-a - 2b + 2c = -2a - 2b + 3c$$

et donc $a = c$, d'où $b = 0$. Contradiction. Donc, $a' = -c'$ De plus, $pgcd(a', b', c') = 1$ donc $a' = -\varepsilon$ et $c' = \varepsilon$, où $\varepsilon = \pm 1$. On a donc

$$\left\{\begin{array}{lcl} -a - 2b + 2c & = & -\varepsilon \\ -2a - b + 2c & = & 0 \\ -2a - 2b + 3c & = & \varepsilon \end{array}\right.$$

Les opérations $(2) \leftarrow (2) - 2 \times (1)$ et $(3) \leftarrow (3) - 2 \times (1)$ donnent le système équivalent

$$\left\{\begin{array}{lcl} -a - 2b + 2c & = & -\varepsilon \\ 3b - 2c & = & 2\varepsilon \\ 2b - c & = & 3\varepsilon \end{array}\right.$$

La combinaison $3 \times (3) - 2 \times (2)$ donne $c = 5\varepsilon$. Comme $c > 0$, $\varepsilon = 1$ et $c = 5$.

La combinaison $2 \times (3) - (2)$ donne $b = 4\varepsilon = 4$. Enfin,

$$a = -2b + 2c + \varepsilon = 3$$

Inversement, $\sigma(3, 4, 5) = (-1, 0, 1)$ est un triplet trivial.

Remarque. On montre facilement à partir de ce qui précède que si $(a, b, c) \in \mathcal{P}^+$ alors

- $a' = 0$ si et seulement si $(a, b, c) = k(4, 3, 5)$ où $k \in \mathbb{N}^*$.
- $b' = 0$ si et seulement si $(a, b, c) = k(3, 4, 5)$ où $k \in \mathbb{N}^*$.

1.2.3 2.3 Inégalités

Soit $(a, b, c) \in \mathcal{P}^+$. Soit $(a', b', c') = \sigma(a, b, c)$. Pour bien comprendre comment les choses se passent, il nous faut savoir à quelles conditions on a $a' < 0$ et de même pour b' et c'. Concentrons-nous pour l'instant sur a' et b' (nous montrerons plus loin que $c' > 0$).

- On a $a' < 0$ si et seulement si $2b > 2c - a$, ce qui équivaut à

$$4b^2 > (2c - a)^2$$

c'est à dire, comme $b^2 = c^2 - a^2$,

$$4c^2 - 4a^2 > 4c^2 + a^2 - 4ac$$

ou encore $4ac > 5a^2$. Comme $a > 0$, on obtient finalement la condition équivalente

$$4c > 5a$$

- De même, $b' < 0$ si et seulement si $b > 2c - 2a$ ce qui équivaut à

$$b^2 > 4(c - a)^2$$

c'est à dire

$$c^2 - a^2 > 4c^2 + 4a^2 - 8ac$$

ou encore

$$5a^2 + 3c^2 - 8ac < 0$$

Remarquons la factorisation

$$5a^2 + 3c^2 - 8ac = (a - c)(5a - 3c)$$

Comme $a < c$, on obtient finalement la condition équivalente

$$5a > 3c$$

Résumons.

- $a' < 0 \iff 5a < 4c$
- $b' < 0 \iff 5a > 3c$

On a donc

- $a' < 0$ et $b' < 0 \iff 3c < 5a < 4c$

- $a' < 0$ et $b' > 0 \iff 5a < 3c$
- $a' > 0$ et $b' < 0 \iff 5a > 4c$

Remarquons qu'il est impossible d'avoir $a' > 0$ et $b' > 0$. Enfin, nous avons vu que seuls les multiples entiers de $(3, 4, 5)$ ou $(4, 3, 5)$ pouvaient mener à $a' = 0$ ou $b' = 0$. En particulier, si $(a, b, c) \in \mathcal{P}_0$, $a' = 0$ est impossible et $b' = 0$ si et seulement si $(a, b, c) = (3, 4, 5)$.

1.2.4 2.4 Symétries, encore

Notons σ_1 et σ_2 les symétries de $\mathbb{R}^3$ définies par

$$\sigma_1(x, y, z) = (-x, y, z)$$
$$\sigma_2(x, y, z) = (x, -y, z)$$

Soit $(a, b, c) \in \mathcal{P}_0$ différent de $(3, 4, 5)$. En faisant le bilan de ce qui précède, nous avons le résultat suivant.

- Si $5a < 3c$, alors $(\sigma_1 \circ \sigma)(a, b, c) \in \mathcal{P}_0$.
- Si $3c < 5a < 4c$, alors $(\sigma_2 \circ \sigma_1 \circ \sigma)(a, b, c) \in \mathcal{P}_0$.
- Si $4c < 5a$, alors $(\sigma_2 \circ \sigma)(a, b, c) \in \mathcal{P}_0$.

Nous pouvons maintenant réécrire la fonction `sigma_abs`de façon plus explicite. Nous appellerons P cette nouvelle fonction (nous verrons pourquoi d'ici la fin du notebook).

[16]:
```
def sigma1(u): return (-u[0], u[1], u[2])
def sigma2(u): return (u[0], -u[1], u[2])
```

[17]:
```
def P(u):
    a, b, c = u
    if 5 * a < 3 * c: return sigma1(sigma(u))
    elif 5 * a < 4 * c: return sigma2(sigma1(sigma(u)))
    else: return sigma2(sigma(u))
```

Reprenons l'exemple vu pour `sigma_abs`.

[18]:
```
u = u0
while u[0] != 0 and u[1] != 0 and u[2] != 0:
    print(u)
    u = P(u)
print(u)
```

```
(12045691, 5437620, 13216141)
(3511351, 3096720, 4681801)
(341189, 755820, 829261)
(194307, 220324, 293765)
(47425, 21408, 52033)
(13825, 12192, 18433)
(1343, 2976, 3265)
```

```
(765, 868, 1157)
(187, 84, 205)
(55, 48, 73)
(5, 12, 13)
(3, 4, 5)
(1, 0, 1)
```

Que nous utilisions la fonction `sigma_abs`ou la fonction P, nous retrouvons le même résultat, ce qui est heureux.

1.3 3. Descente

1.3.1 3.1 Introduction

En mathématiques, le *principe de descente infinie* est une technique de démonstration qui fonctionne de la façon suivante. Pour montrer qu'un objet x vérifie une propriété P, on suppose que ce n'est pas le cas. Puis on crée un objet x' « de taille plus petite » que celle de x qui ne vérifie pas P et on recommence avec x'. On crée ainsi une suite d'objets x, x', x'', etc. qui sont de tailles de plus en plus petites et qui ne vérifient pas P. Si les tailles de ces objets sont des entiers naturels, on aboutit ainsi à une contradiction, car il n'existe pas de suite strictement décroissante d'entiers naturels.

Cette technique est en fait équivalent au principe de récurrence forte. C'est la raison pour laquelle elle n'est plus vraiment utilisée de nos jours. La preuve moderne consiste à prendre un entier n arbitraire et à supposer que tous les objets de taille strictement inférieure à n vérifient la propriété. Puis on montre que tous les objets de taille n la vérifient aussi.

Dans l'exemple des itérations de la fonction P sur le triplet u_0, nous voyons le principe de descente à l'oeuvre. Les itérées de P à partir de u_0 sont des triplets dont les coefficients sont des suites d'entiers naturels strictement décroissantes. On ne peut qu'aboutir à une « contradiction », c'est à dire au triplet $(1, 0, 1)$ qui n'appartient pas à $\mathcal{P}_0$.

Pour généraliser tout ceci, il nous faut prouver que si nous partons d'un triplet u quelconque, il y a bien stricte décroissance des coefficients des triplets. C'est l'objet du paragraphe suivant.

1.3.2 3.2 Inégalités, encore

Lemme. Soit $(a, b, c) \in \mathcal{P}^+$. On a

- $a < c$, $b < c$.
- $c < a + b < \sqrt{2}c$.

Démonstration.

- $c^2 = a^2 + b^2 > a^2$ donc $c > a$. De même, $c > b$.
- $c^2 = a^2 + b^2 < a^2 + b^2 + 2ab = (a + b)^2$, donc $c < a + b$.
- Remarquons que $a \neq b$, sinon $c^2 = 2a^2$, ce qui est impossible car $\sqrt{2}$ est irrationnel. De là, $0 < (a - b)^2 = a^2 + b^2 - 2ab$, donc $2ab < a^2 + b^2$. D'où

$$(a+b)^2 = a^2 + 2ab + b^2 < 2(a^2+b^2) = 2c^2$$

donc $a + b < \sqrt{2}c$.

Proposition. Soit $(a, b, c) \in \mathcal{P}^+$. Soit $(a', b', c') = \sigma(a, b, c)$. On a

- $-a < a' < a$.
- $-b < b' < b$.
- $0 < c' < c$.

Démonstration.

- $a - a' = 2(a + b - c) > 0$ donc $a' < a$. Et aussi $a + a' = 2(b - c) < 0$ donc $-a < a'$. De même, $-b < b' < b$.
- $c - c' = 2(a + b - c) > 0$ donc $c' < c$. Et aussi $c' = -2a - 2b + 3c > -2a - 2b + 2\sqrt{2}c > 0$.

1.3.3 3.3 Existence

Nous avons vu au paragraphe 2.4 que les trois fonctions $\sigma_1 \circ \sigma$, $\sigma_2 \circ \sigma_1 \circ \sigma$ et $\sigma_2 \circ \sigma$ jouent un rôle important. Notons

$$\begin{aligned} G_1 &= \sigma_1 \circ \sigma \\ G_2 &= \sigma_2 \circ \sigma_1 \circ \sigma \\ G_3 &= \sigma_2 \circ \sigma \end{aligned}$$

[19]:
```
def G1(u): return sigma1(sigma(u))
def G2(u): return sigma2(sigma1(sigma(u)))
def G3(u): return sigma2(sigma(u))
```

Soit $(a, b, c) \in \mathcal{P}_0$ différent de $(3, 4, 5)$. Nous avons montré que

- Si $5a < 3c$, alors $G_1(a, b, c) \in \mathcal{P}_0$.
- Si $3c < 5a < 4c$, alors $G_2(a, b, c) \in \mathcal{P}_0$.
- Si $4c < 5a$, alors $G_3(a, b, c) \in \mathcal{P}_0$.

En utilisant les résultats précédents nous avons de plus que si $(a', b', c') = G_i(a, b, c)$ où $i = 1, 2$ ou 3 selon les cas, alors $0 < a' < a$, $0 < b' < b$ et $0 < c' < c$, et donc $(a', b', c') \in \mathcal{P}_0$.

Comme les σ_i et σ sont des symétries, ces applications sont des bijections qui sont leur propre réciproque. Les G_i sont donc aussi bijectives. Notons, pour $i = 1, 2, 3$, $F_i = G_i^{-1}$. On a

$$\begin{aligned} F_1 &= \sigma \circ \sigma_1 \\ F_2 &= \sigma \circ \sigma_1 \circ \sigma_2 \\ F_3 &= \sigma \circ \sigma_2 \end{aligned}$$

On vérifie facilement que les matrices de F_1, F_2 et F_3 dans la base canonique de $\mathbb{R}^3$ sont

$$A_1 = \begin{pmatrix} 1 & -2 & 2 \\ 2 & -1 & 2 \\ 2 & -2 & 3 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 3 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} -1 & 2 & 2 \\ -2 & 1 & 2 \\ -2 & 2 & 3 \end{pmatrix}$$

Pour définir les fonctions F_i en Python, on peut utiliser les matrices ci-dessus … ou pas.

[20]:
```
def F1(u): return sigma(sigma1(u))
def F2(u): return sigma(sigma1(sigma2(u)))
def F3(u): return sigma(sigma2(u))
```

Le théorème suivant nous dit qu'il est possible, à partir du triplet $(3, 4, 5)$ et des fonctions F_i, d'engendrer **TOUS** les triplets pythagoriciens primitifs.

Proposition. Soit $(a, b, c) \in \mathcal{P}_0$. Il existe $k \in \mathbb{N}$ et $f_1, \dots, f_k \in \{F_1, F_2, F_3\}$ telles que

$$(a, b, c) = (f_1 \circ \dots \circ f_k)(3, 4, 5)$$

Démonstration. Faisons une récurrence sur $n = \max(a, b, c)$. La plus petite valeur possible de n est 5. Dans ce cas, $k = 0$ convient. Soit $n > 5$. Supposons la propriété vraie pour tout $m < n$. Soit $(a, b, c) \in \mathcal{P}_0$ tel que $\max(a, b, c) = n$. Par ce qui précède, il existe $i \in \{1, 2, 3\}$ tel que

$$(a', b', c') = G_i(a, b, c) \in \mathcal{P}_0$$

avec de plus $a' < a$, $b' < b$ et $c' < c$. On a donc $\max(a', b', c') < \max(a, b, c)$. Par l'hypothèse de récurrence, il existe $k \in \mathbb{N}$ et $f_2, \dots, f_k \in \{F_1, F_2, F_3\}$ telles que

$$(a', b', c') = (f_2 \circ \dots \circ f_k)(3, 4, 5)$$

De là,

$$(a, b, c) = F_i(a', b', c') = (F_i \circ f_2 \circ \dots \circ f_k)(3, 4, 5)$$

1.3.4 3.4 Unicité

Il s'avère que la décomposition vue au paragraphe précédent est unique. Montrons tout d'abord un petit lemme.

Lemme. Les ensembles $F_1(\mathcal{P}_0)$, $F_2(\mathcal{P}_0)$ et $F_3(\mathcal{P}_0)$ sont disjoints deux à deux.

Démonstration. Soit $(a,b,c) \in {\mathbb{N}^*}^3$.

- Si $(a,b,c) \in F_1(\mathcal{P}_0)$, alors, $5a < 3c$.
- Si $(a,b,c) \in F_2(\mathcal{P}_0)$, alors, $3c < 5a < 4c$.
- Si $(a,b,c) \in F_3(\mathcal{P}_0)$, alors, $4c < 5a$.

D'où le résultat.

Remarquons que le triplet $(3,4,5)$ n'appartient ni à $F_1(\mathcal{P}_0)$, ni à $F_2(\mathcal{P}_0)$, ni à $F_3(\mathcal{P}_0)$. En fait, nous avons montré que

$$\mathcal{P}_0 = \{(3,4,5)\} \cup F_1(\mathcal{P}_0) \cup F_2(\mathcal{P}_0) \cup F_3(\mathcal{P}_0)$$

et cette réunion est disjointe.

Venons-en au théorème d'unicité.

Proposition. Soient $k, \ell \in \mathbb{N}$. Soient $f_1, ..., f_k, g_1, ..., g_\ell \in \{F_1, F_2, F_3\}$. On suppose que

$$(f_1 \circ ... \circ f_k)(3,4,5) = (g_1 \circ ... \circ g_\ell)(3,4,5)$$

Alors, $k = \ell$ et pour tout $i \in [|1,k|]$, $f_i = g_i$.

Démonstration. Faisons une récurrence sur $n = k + \ell$.

Si $n = 0$, alors $k = \ell = 0$ et il n'y a rien à montrer.

Soit $n > 0$. Supposons la propriété vérifiée pour tout $m < n$. Soient $k, \ell \in \mathbb{N}$ tels que $k + \ell = n$. Soient $f_1, ..., f_k, g_1, ..., g_\ell \in \{F_1, F_2, F_3\}$. Supposons que

$$(f_1 \circ ... \circ f_k)(3,4,5) = (g_1 \circ ... \circ g_\ell)(3,4,5)$$

Tout d'abord, aucun des entiers k et ℓ n'est nul. En effet, si, par exemple, $k > 0$ et $\ell = 0$, alors $(3,4,5) \in f_1(\mathcal{P}^+)$, ce qui n'est pas le cas.

Le membre de gauche de l'égalité ci-dessus appartient à $f_1(\mathcal{P}^+)$ et le membre de droite appartient à $g_1(\mathcal{P}^+)$. Par le lemme ci-dessus, on a donc $f_1 = g_1$. En composant par f_1^{-1}, il vient

$$(f_2 \circ ... \circ f_k)(3,4,5) = (g_2 \circ ... \circ g_\ell)(3,4,5)$$

Comme $m = (k-1) + (\ell - 1) < k + \ell = n$, on peut maintenant appliquer l'hypothèse de récurrence et en déduire que $k = \ell$, $f_2 = g_2$, ..., $f_k = g_k$.

1.3.5 3.5 Bilan

Pour résumer, tout triplet primitif $u \in \mathcal{P}_0$ s'écrit de façon unique

$$u = (f_1 \circ ... \circ f_k)(3, 4, 5)$$

où $k \in \mathbb{N}$ et les f_i appartiennent à l'ensemble $\{F_1, F_2, F_3\}$.

De plus, nous savons comment, à partir du triplet u, construire de façon effective les f_i.

La fonction `decomposer` prend en paramètre un triplet primitif u. Elle renvoie un couple (fs, us) où

- $fs = [i_1, ..., i_k]$ est une liste d'entiers appartenant à $\{1, 2, 3\}$.
- $us = [u_0, ..., u_k]$ est une liste de triplets primitifs, et
  - $u_0 = (3, 4, 5)$
  - $u_1 = F_{i_k}(u_0)$
  - $u_2 = (F_{i_{k-1}} \circ F_{i_k})(u_0)$
  - $u_3 = (F_{i_{k-2}} \circ F_{i_{k-1}} \circ F_{i_k})(u_0)$
  - ...
  - $u_k = (F_{i_1} \circ ... \circ F_{i_k})(u_0)$

[21]:
```
def decomposer(u):
    fs = []
    us = [u]
    while u != (3, 4, 5):
        a, b, c = u
        if 5 * a < 3 * c:
            fs.append(1)
            u = G1(u)
        elif 5 * a < 4 * c:
            fs.append(2)
            u = G2(u)
        else:
            fs.append(3)
            u = G3(u)
        us.append(u)
    fs.reverse()
    us.reverse()
    return (fs, us)
```

[22]:
```
fs, us = decomposer(u0)
print(fs)
print(us)
```

```
[1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3]
[(3, 4, 5), (5, 12, 13), (55, 48, 73), (187, 84, 205), (765, 868, 1157), (1343,
2976, 3265), (13825, 12192, 18433), (47425, 21408, 52033), (194307, 220324,
```

```
293765), (341189, 755820, 829261), (3511351, 3096720, 4681801), (12045691,
5437620, 13216141)]
```

Notre triplet u_0 n'était donc pas choisi complètement au hasard :-).

La fonction « inverse » de `decomposer`, que nous appellerons `recomposer`, prend en paramètre une liste cs d'entiers valant 0,1,2 ou 3 et un triplet primitif u. Pour chaque élément c de cs,

- Si $c = 1$, 2 ou 3 on remplace u par $f_c(u)$.
- Si $c = 0$, on remplace u par $P(u)$.

[23]:
```
def recomposer(cs, u):
    for c in cs:
        if c == 1: u = F1(u)
        elif c == 2: u = F2(u)
        elif c == 3: u = F3(u)
        elif c == 0: u = P(u)
        else: pass
    return u
```

[24]:
```
recomposer([1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3], (3, 4, 5))
```

[24]:
```
(12045691, 5437620, 13216141)
```

La valeur 0 correspond à un retour en arrière :

[25]:
```
print(recomposer([1, 2, 3, 2, 2, 3, 1, 1], (3, 4, 5)))
print(recomposer([1, 2, 3, 2, 2, 3, 1, 1, 0, 0, 0], (3, 4, 5)))
print(recomposer([1, 2, 3, 2, 2], (3, 4, 5)))
```

```
(63879, 172480, 183929)
(4815, 4712, 6737)
(4815, 4712, 6737)
```

Nous pouvons maintenant fabriquer tous les triplets pythagoriciens primitifs, et il y a même une et une seule façon de créer chacun d'entre-eux à partir du triplet $(3, 4, 5)$. La fonction ci-dessous renvoie un triplet pythagoricien primitif aléatoire « de niveau k », où k est le nombre de fonctions f_i intervenant dans la décomposition du triplet.

[26]:
```
def triplet_pytha_aleatoire(k):
    fs = [random.randint(1, 3) for i in range(k)]
    return recomposer(fs, (3, 4, 5))
```

Voici un triplet primitif aléatoire de niveau 666. Le triplet renvoyé est évidemment différent à chaque évaluation de la cellule.

[27]:
```
u = triplet_pytha_aleatoire(666)
print(est_pythagoricien_primitif(u))
print(u)
```

```
True
(26230496089456853561882453925021179829000914194581053606785597863443098531979925684889541607730568237218728380288463789496328648202419645513871322189960838921434413353659949876843476330158973450126822398861374270293784712327670559764566189746388433559715043753963614699042847322048805622564156244282322851808337370437062242703708150137565956747431295518751048006273687343647, 2331003786338412802402724870886414505022465088899108422329940232800608441489306379467992116774511960898926486734023640709355306360338957245642905193160492308529317250960745925107859165912027080287891285117660552702810615211002330246932962096073827070401839109802851136097044832569024527358129150315225577367200923692300250472650079804620869462623899849655364162718330775469623899849655364162718330775469623, 3509126373175256524585016307683641241655489076611906310039615343993669939613044091699447794695132638166612698801515117537963104933322899901692100034890579829310119041934672346120108677434399191393255416826196190088148225383645289548858998719917286381719952786166636828540263514718065072941399350445869886189288008564743398307525471648405487736428655571244720415030276417894 5)
```

1.4 4. L'arbre de Pythagore

1.4.1 4.1 Une structure d'arbre ternaire

L'ensemble $\mathcal{P}_0$ des triplets pythagoriciens primitifs peut être, en fin de compte, modélisé par un arbre ternaire $\mathcal{T}$ dont les étiquettes des noeuds sont les éléments de $\mathcal{P}_0$.

- La racine de $\mathcal{T}$ a pour étiquette $(3, 4, 5)$.
- Pour tout noeud ν de $\mathcal{T}$ d'étiquette $u \in \mathcal{P}_0$, les trois fils de ν ont pour étiquettes $F_1(u)$, $F_2(u)$ et $F_3(u)$.

Nous allons nous attaquer au dessin explicite de cet arbre, enfin de quelques niveaux de celui-ci vu que cet arbre possède une infinité de noeuds. Notre théorème d'existence et d'unicité nous montre que chaque triplet primitif est l'étiquette d'un et un seul noeud de $\mathcal{T}$. Nous allons donc confondre, dans la suite, les noeuds de $\mathcal{T}$ avec leurs étiquettes.

Nous venons aussi de comprendre pourquoi la fonction P que nous avons définie il y a un petit moment s'appelle P : si $u \in \mathcal{P}_0$ est différent de $(3, 4, 5)$, $P(u)$ est le *père* de u dans l'arbre $\mathcal{T}$.

1.4.2 4.2 Dessiner $\mathcal{T}$

La fonction `fils` prend en paramètre un triplet primitif u et renvoie la liste de ses fils.

[28]:
```
def fils(u):
    return [F1(u), F2(u), F3(u)]
```

La fonction `arbre_aux` prend en paramètres

- Un triplet primitif u
- Un paramètre de profondeur $p \in \mathbb{N}$.
- Un flottant x qui représente l'abscisse du point d'affichage du triplet u.
- Deux flottants y_{min} et y_{max} qui représentent les ordonnées minimale et maximale du tracé.

- Un flottant $d > 0$ qui nous permettra d'ajuster certains affichages.

Cette fonction se rappelle trois fois sur les fils de u en diminuant de 1 le paramètre p. Elle termine lorsque $p = 0$. Je ne détaillerai pas plus avant son code.

[29]:
```
def arbre_aux(u, p, x, ymin, ymax, d):
    plt.text(x, (ymin + ymax) / 2, vers_chaine(u),
             fontsize=8, fontfamily='monospace',
             horizontalalignment='left', verticalalignment='center')
    if p > 0:
        uh, um, ub = fils(u)
        l = len(vers_chaine(u))
        plt.plot([x + d * l, x + 0.95], [(ymin + ymax) / 2, (5 * ymin + ymax) /␣
↪6], 'k', lw=1)
        plt.plot([x + d * l, x + 0.95], [(ymin + ymax) / 2, (ymin + ymax) / 2],␣
↪'k', lw=1)
        plt.plot([x + d * l, x + 0.95], [(ymin + ymax) / 2, (ymin + 5 * ymax) /␣
↪6], 'k', lw=1)

        arbre_aux(ub, p - 1, x + 1, ymin, (2 * ymin + ymax) / 3, d)
        arbre_aux(um, p - 1, x + 1, (2 * ymin + ymax) / 3, (ymin +  2 * ymax) /␣
↪3, d)
        arbre_aux(uh, p - 1, x + 1, (ymin +  2 * ymax) / 3, ymax, d)
```

La fonction `vers_chaine` prend en paramètre un triplet pythagoricien et renvoie sa représentation sous forme de chaîne de caractères.

[30]:
```
def vers_chaine(u):
    a, b, c = u
    return '(' + str(a) + ', ' + str(b) + ', ' + str(c) + ')'
```

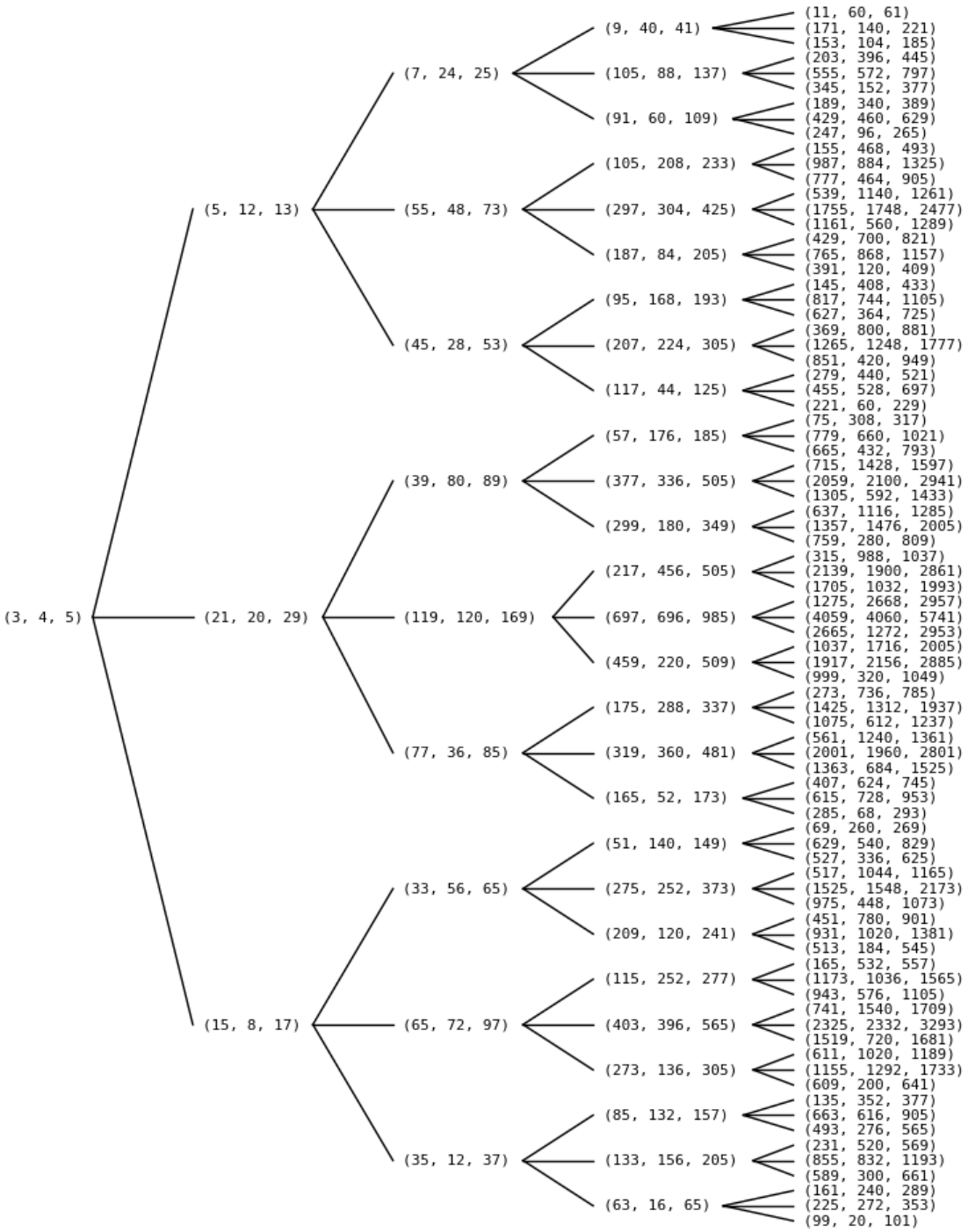
En fin, la fonction `tracer_arbre` trace tous les niveaux de l'arbre $\mathcal{T}$ jusqu'à la profondeur p.

[31]:
```
def tracer_arbre(p):
    plt.xlim(0, p + 0.1)
    plt.ylim(0, 1)
    plt.axis('off')
    arbre_aux((3, 4, 5), p, 0, 0, 1, 0.05)
```

[32]:
```
plt.rcParams['figure.figsize'] = (8, 12)
```

En affichant jusqu'à la profondeur 4, nous verrons apparaître les 121 « premiers » triplets pythagoriciens primitifs. Mieux vaut ne pas dépasser cette profondeur …

[33]:
```
tracer_arbre(4)
```



1.4.3 4.3 La décomposition d'Euclide

À vrai dire, le résultat que nous allons donner ici aurait pu être démontré sans utiliser ce qui précède, mais il est une excellente occasion de voir que l'on peut utiliser l'arbre $\mathcal{T}$ pour prouver des

théorèmes relatifs aux triplets pythagoriciens.

Notons

$$\mathcal{P}_2 = \{(m,n) \in \mathbb{N} : m > n > 0, pgcd(m,n) = 1, m \text{ et } n \text{ de parités distinctes}\}$$

Proposition. Soit $(m,n) \in \mathcal{P}_2$. Soit $(a,b,c) = (m^2 - n^2, 2mn, m^2 + n^2)$. Le triplet (a,b,c) est pythagoricien primitif.

Démonstration. Clairement, $a, b, c \in \mathbb{N}^*$. Ensuite,

$$a^2 + b^2 = (m^2 - n^2)^2 + 4m^2n^2 = (m^2 + n^2)^2 = c^2$$

Enfin, soit d un diviseur commun de a, b et c. L'entier d divise $a + c = 2m^2$ et aussi $c - a = 2n^2$, donc d divise $pgcd(2m^2, 2n^2) = 2$. Ainsi, $d = 1$ ou $d = 2$. Il reste à remarquer que a est impair, donc $d \neq 2$. Ainsi, $d = 1$ et donc $pgcd(a,b,c) = 1$. En conclusion, $(a,b,c) \in \mathcal{P}_0$.

Ce résultat admet une réciproque.

Proposition. Soit (a,b,c) un triplet pythagoricien primitif. Il existe un couple $(m,n) \in \mathcal{P}_2$ tel que

$$\begin{cases} a &= m^2 - n^2 \\ b &= 2mn \\ c &= m^2 + n^2 \end{cases}$$

Démonstration. Nous allons montrer ce résultat par récurrence sur la profondeur p de (a,b,c) dans l'arbre $\mathcal{T}$.

Si $p = 0$, alors $(a,b,c) = (3,4,5)$ et $m = 2$, $n = 1$ conviennent.

Soit $p \in \mathbb{N}$. Supposons que tous les triplets de profondeur p vérifient la propriété. Soit (a',b',c') un triplet de profondeur $p+1$. On a $(a',b',c') = F(a,b,c)$ où (a,b,c) est un noeud de profondeur p et F est l'une des trois applications F_1, F_2, F_3. Par l'hypothèse de récurrence il existe un couple $(m,n) \in \mathcal{P}_2$ tel que $a = m^2 - n^2$, $b = 2mn$ et $c = m^2 + n^2$.

- Cas 1, $F = F_1$. La matrice de F_1 dans la base canonique de $\mathbb{R}^3$ a été donnée dans le paragraphe 3.3. On a

$$\begin{aligned} a' &= a - 2b + 2c \\ &= (m^2 - n^2) - 4mn + 2(m^2 + n^2) \\ &= 3m^2 + n^2 - 4mn \\ &= (2m - n)^2 - m^2 \end{aligned}$$

$$\begin{aligned} b' &= 2a - b + 2c \\ &= 2(m^2 - n^2) - 2mn + 2(m^2 + n^2) \\ &= 4m^2 - 2mn \\ &= 2(2m - n)m \end{aligned}$$

$$\begin{aligned} c' &= 2a - 2b + 3c \\ &= 2(m^2 - n^2) - 4mn + 3(m^2 + n^2) \\ &= 5m^2 + n^2 - 4mn \\ &= (2m - n)^2 + m^2 \end{aligned}$$

Posons $m' = 2m - n$ et $n' = m$. On a facilement $(m', n') \in \mathcal{P}_2$. Le couple (m', n') vérifie donc la propriété.

- Cas 2, $F = F_2$. Les mêmes types de calculs montrent que $m' = 2m+n$ et $n' = m$ conviennent.
- Cas 3, $F = F_3$. Ici , $m' = m + 2n$ et $n' = n$ conviennent.

Cerise sur le gâteau, on a non seulement existence mais aussi unicité.

Proposition. Le couple (m, n) de la proposition précédente est uniquement déterminé.

Démonstration. Soit (a, b, c) un triplet primitif. Soit $(m, n) \in \mathcal{P}_2$. Si $a = m^2 - n^2$, $b = 2mn$ et $c = m^2 + n^2$, on a alors $c + a = 2m^2$ et $c - a = 2n^2$ et donc

$$m = \sqrt{\frac{c + a}{2}}$$

$$n = \sqrt{\frac{c - a}{2}}$$

d'où l'unicité.

Il est donc *très* facile de calculer les entiers m et n à partir des entiers a, b et c. La seule difficulté est de posséder une fonction « racine carrée entière », que voici. Cette fonction procède par dichotomie.

[34]:
```
def intsqrt(n):
    a, b = 1, n + 1
    while b - a > 1:
        c = (a + b) // 2
        if c * c <= n: a = c
        else: b = c
    return a
```

[35]:
```
def decomposition_euclide(u):
    a, b, c = u
```

```
    m = intsqrt((c + a) // 2)
    n = intsqrt((c - a) // 2)
    return (m, n)
```

Testons sur notre triplet fétiche.

[36]:
```
print(u0)
m, n = decomposition_euclide(u0)
print(m, n)
print(m ** 2 - n ** 2, 2 * m * n, m ** 2 + n ** 2)
```

```
(12045691, 5437620, 13216141)
3554 765
12045691 5437620 13216141
```

1.4.4 4.4 Chemins remarquables

Définition. Un *chemin* dans l'arbre $\mathcal{T}$ est une suite $(u_n)_{n\in\mathbb{N}}$ vérifiant $u_0 = (3,4,5)$ et, pour tout $n \in \mathbb{N}$, $u_n = P(u_{n+1})$.

On peut caractériser un chemin par un *mot infini* $\alpha_1\alpha_2\alpha_3\ldots$ où $\alpha_n \in \{H, M, B\}$, en convenant que

- si $\alpha_n = H$, alors, $u_{n+1} = F_1(u_n)$ est le fils *haut* de u_n
- si $\alpha_n = M$, alors, $u_{n+1} = F_2(u_n)$ est le fils *médian* de u_n
- si $\alpha_n = B$, alors, $u_{n+1} = F_3(u_n)$ est le fils *bas* de u_n

En observant l'arbre ci-dessus, on repère facilement trois chemins remarquables. Une preuve serait évidemment nécessaire, nous ne la ferons pas ici.

- Le chemin $HHH\ldots$ contient tous les triplets primitifs (a,b,c) tels que $c = b+1$. Ce chemin s'appelle la *séquence de Pythagore*.
- Le chemin $BBB\ldots$ contient tous les triplets primitifs (a,b,c) tels que $c = a+2$. Ce chemin s'appelle la *séquence de Platon*.
- Le chemin $MMM\ldots$ contient tous les triplets primitifs (a,b,c) tels que $b = a \pm 1$. Ce chemin s'appelle la *séquence de Fermat*.

Il serait évidemment intéressant de calculer explicitement les triplets pythagoriciens qui apparaissent dans ces chemins. C'est très facile pour les séquences de Pythagore et de Platon.

Proposition. Les termes de la séquence de Pythagore sont les triplets

$$F_1^n(3,4,5) = (2n+3, 6n^2+6n+4, 6n^2+6n+5)$$

Démonstration. C'est une récurrence sur n.

Proposition. Les termes de la séquence de Platon sont les triplets

$$F_3^n(3,4,5) = (4n^2+8n+3, 4n+4, 4n^2+8n+5)$$

Démonstration. Ici encore, c'est une récurrence sur n.

Les triplets de la séquence de Fermat sont un peu plus difficiles à caractériser. Leur valeur est reliée aux solutions de l'équation de *Pell*

$$x^2 - 2y^2 = \pm 1$$

Il est plus simple de donner les termes de cette séquence via leur décomposition d'Euclide. Citons sans preuve le résultat suivant.

Proposition. Les termes de la séquence de Fermat son les triplets

$$F_2^k(3,4,5) = (m_k^2 - n_k^2, 2m_k n_k, m_k^2 + n_k^2)$$

où

$$\begin{aligned} m_k &= \frac{1}{2\sqrt{2}}\left((1+\sqrt{2})^{k+2} - (1-\sqrt{2})^{k+2}\right) \\ n_k &= \frac{1}{2\sqrt{2}}\left((1+\sqrt{2})^{k+1} - (1-\sqrt{2})^{k+1}\right) \end{aligned}$$

Il y a bien d'autres chemins remarquables dans l'arbre $\mathcal{T}$. Pour n'en citer que deux, que dire des chemins $HMHMHM\ldots$ et $MHMHMH\ldots$? Eh bien … tous les triplets de ces chemins ont un de leurs coefficients qui est un nombre de Fibonacci.

Voici une fonction `fibonacci` qui renvoie le $n$ième nombre de Fibonacci.

[37]:
```
def fibonacci(n):
    u, v = 0, 1
    for k in range(n):
        u, v = v, u + v
    return u
```

[38]:
```
print([(n, fibonacci(n)) for n in range(12)])
```

```
[(0, 0), (1, 1), (2, 1), (3, 2), (4, 3), (5, 5), (6, 8), (7, 13), (8, 21), (9,
34), (10, 55), (11, 89)]
```

La fonction `triplet_fibonacci1` renvoie le triplet pythagoricien à l'extrémité du chemin $HMHM\ldots$ (n lettres) dans l'arbre $\mathcal{T}$.

[39]:
```
def triplet_fibonacci1(n):
    u = (3, 4, 5)
    for k in range(n):
        if k % 2 == 0: u = F1(u)
        else: u = F2(u)
    return u
```

Prenons un exemple.

[40]:
```
print(triplet_fibonacci1(10))
```

```
(5702887, 5100816, 7651225)
```

Le premier élément de ce triplet est le 34ème nombre de Fibonacci.

[41]:
```
print(fibonacci(34))
```

```
5702887
```

De même, la fonction `triplet_fibonacci2` renvoie le triplet pythagoricien à l'extrémité du chemin $MHMH\dots$ (n lettres) dans l'arbre $\mathcal{T}$.

[42]:
```
def triplet_fibonacci2(n):
    u = (3, 4, 5)
    for k in range(n):
        if k % 2 == 0: u = F2(u)
        else: u = F1(u)
    return u
```

Prenons un exemple.

[43]:
```
print(triplet_fibonacci2(10))
```

```
(4126647, 8253296, 9227465)
```

Le troisième élément de ce triplet est le 35ème nombre de Fibonacci.

[44]:
```
print(fibonacci(35))
```

```
9227465
```

1.4.5 4.5 Et après ?

Est-il possible de refaire tout ce que nous avons fait sur des problèmes plus généraux ? Parmi beaucoup de possibilités, citons deux généralisations.

- Les *quadruplets pythagoriciens* sont les quadruplets d'entiers (a, b, c, d) tels que $a^2+b^2+c^2 = d^2$. Ce que nous avons fait peut être refait avec, cette fois-ci, des symétries de $\mathbb{R}^4$. On peut engendrer tous les quadruplets pythagoriciens primitifs en partant du quadruplet $(1, 2, 2, 3)$ et en appliquant 7 symétries.
- On peut travailler avec des *formes quadratiques* quelconques, c'est à dire s'intéresser à des triplets d'entiers (a, b, c) tels que

$$\alpha a^2 + \beta b^2 + \gamma c^2 + \delta ab + \epsilon ac + \zeta bc = 0$$

où $\alpha, \dots, \zeta \in \mathbb{Z}$. Ici encore, tout ce que nous avons fait peut être généralisé.

On quitte alors le domaine des mathématiques « élémentaires » (algèbre linéaire, matrices, pgcd) qui nous a suffi à tout établir dans ce notebook …

1.5 Bibliographie

L'article de A. Hall est très court, il a été le point de départ de ce notebook.

- A. Hall (1970), Genealogy of Pythagorean Triads

L'article de Luis Teia offre une vision plus géométrique.

- L. Teia, Anatomy of the Pythagoras' Tree

Les deux articles ci-dessous peuvent être trouvés sur la page de Keith Conrad.

https://kconrad.math.uconn.edu/blurbs/

- K. Conrad, Pythagorean Triples
- K. Conrad, Pythagorean Descent

L'article de Roger Alperin offre un autre point de vue sur les triplets pythagoriciens.

- R. Alperin, The Modular Tree of Pythagoras

Les deux articles ci-dessous s'intéressent aux généralisations dont nous avons parlé (quadruplets pythagoriciens, formes quadratiques).

- J. Rushall, M. Guttierez, V. McCarty (2020), On the Complete Tree of Primitive Pythagorean Quadruples
- B. Cha, E. Nguyen, B. Tauber (2017), Quadratic Forms and Their Berggren Trees