Equação de Pell

Prof. Doherty Andrade

www.metodosnumericos.com.br

1. Números de Pell

Os números de Pell são números semelhantes aos números de Fibonacci, são gerados pela fórmula abaixo da seguinte forma:

$$P_n = 2P_{n-1} + P_{n-2}$$

com sementes $P_0 = 0$ e $P_1 = 1$.

Os primeiros números de Pell são 0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741, 13860, 33461, ...

A função a seguir, denomonada de Pell(int n), retorne P_n o n-ésimo número de Pell.

In [3]:
```
# Calcula o n-ésimo número de Pell
def Pell(n) :
    if (n <= 2) :
        return n
    return (2 * pell(n - 1) + pell(n - 2))
```

In [4]:
```
Pell(5)
```

Out[4]:
```
29
```

In [5]:
```
Pell(8)
```

Out[5]:
```
408
```

O seguinte código calcula os n primeiros números de Pell.

In [6]:
```
def pell2(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        p_prev = 0
        p_curr = 1
        for i in range(2, n + 1):
            p_next = 2 * p_curr + p_prev
            p_prev = p_curr
            p_curr = p_next
        return p_curr

# Receber entrada do usuário
try:
    N = int(input("Digite um valor para N: "))
    if N < 0:
```

```
        print("Por favor, digite um número não negativo.")
    else:
        for i in range(N + 1):
            print(f'P_{i} = {pell2(i)}')
except ValueError:
    print("Por favor, digite um número inteiro válido.")
```

```
Digite um valor para N: 10
P_0 = 0
P_1 = 1
P_2 = 2
P_3 = 5
P_4 = 12
P_5 = 29
P_6 = 70
P_7 = 169
P_8 = 408
P_9 = 985
P_10 = 2378
```

2. Matriz Geradora

Os números de Pell podem ser expressos de maneira elegante usando potências de uma matriz específica. A relação é dada pela seguinte expressão matricial:

$$\begin{bmatrix} P_n \\ P_{n-1} \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} P_1 \\ P_0 \end{bmatrix}$$

A matriz $\begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}$ é conhecida como a matriz de Pell ou matriz geradora. Elevando essa matriz à potência n, e realizando o produto, você obtém os números correspondem a P_n e P_{n-1}.

Por exemplo, para $n = 3$, a matriz resultante seria:

$$\begin{bmatrix} P_3 \\ P_2 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}^3 \begin{bmatrix} P_1 \\ P_0 \end{bmatrix}$$

Multiplicando as matrizes, você obterá os valores correspondentes de P_3 e P_2.

Esta expressão matricial oferece uma maneira elegante de calcular os números de Pell e destaca a conexão entre os números de Pell e as propriedades algébricas das matrizes.

O seguinte código utiliza esta formulação para gerar os números de Pell.

Este código solicita ao usuário que insira um valor para N e, em seguida, calcula e imprime os números de Pell correspondentes até o valor fornecido utilizando a matriz geradora e suas potencias.. Certifique-se de inserir um número inteiro não negativo para obter resultados significativos.

In [8]:
```
import numpy as np

def matrix_power(matrix, exponent):
    return np.linalg.matrix_power(matrix, exponent)

def pell_matrix(n):
    pell_matrix = np.array([[2, 1], [1, 0]])
```

```
    seed_vector = np.array([[1], [0]])

    result_matrix = matrix_power(pell_matrix, n)
    result_vector = np.dot(result_matrix, seed_vector)

    return result_vector[0][0]

try:
    N = int(input("Digite um valor para N: "))
    if N < 0:
        print("Por favor, digite um número não negativo.")
    else:
        for i in range(N + 1):
            print(f'P_{i} = {pell_matrix(i)}')
except ValueError:
    print("Por favor, digite um número inteiro válido.")
```

```
Digite um valor para N: 10
P_0 = 1
P_1 = 2
P_2 = 5
P_3 = 12
P_4 = 29
P_5 = 70
P_6 = 169
P_7 = 408
P_8 = 985
P_9 = 2378
P_10 = 5741
```

Se você deseja visulizar as potências da matriz geradora, então utilize este código abaixo.

In [9]:
```
import numpy as np

def matrix_power(matrix, exponent):
    return np.linalg.matrix_power(matrix, exponent)

def pell_matrix(n):
    pell_matrix = np.array([[2, 1], [1, 0]])
    seed_vector = np.array([[1], [0]])

    result_matrix = matrix_power(pell_matrix, n)
    result_vector = np.dot(result_matrix, seed_vector)

    print(f'Matriz para n={n}:\n{result_matrix}')

    return result_vector[0][0]

try:
    N = int(input("Digite um valor para N: "))
    if N < 0:
        print("Por favor, digite um número não negativo.")
    else:
        for i in range(N + 1):
            print(f'P_{i} = {pell_matrix(i)}')
except ValueError:
    print("Por favor, digite um número inteiro válido.")
```

```
Digite um valor para N: 10
Matriz para n=0:
[[1 0]
 [0 1]]
P_0 = 1
Matriz para n=1:
[[2 1]
 [1 0]]
P_1 = 2
Matriz para n=2:
[[5 2]
 [2 1]]
P_2 = 5
Matriz para n=3:
[[12  5]
 [ 5  2]]
P_3 = 12
Matriz para n=4:
[[29 12]
 [12  5]]
P_4 = 29
Matriz para n=5:
[[70 29]
 [29 12]]
P_5 = 70
Matriz para n=6:
[[169  70]
 [ 70  29]]
P_6 = 169
Matriz para n=7:
[[408 169]
 [169  70]]
P_7 = 408
Matriz para n=8:
[[985 408]
 [408 169]]
P_8 = 985
Matriz para n=9:
[[2378  985]
 [ 985  408]]
P_9 = 2378
Matriz para n=10:
[[5741 2378]
 [2378  985]]
P_10 = 5741
```

3. Aproximação para raiz quadrade de N.

Considere agora a equação $x^2 - 61y^2 = 1$, que apresenta solução minimal dada por (1766319049, 226153980). Isso fornece uma excelente aproximação para √61:

$$\sqrt{61} \approx \frac{x}{y} = \frac{1766319049}{226153980} \approx 7.81024967590665.$$

A relação entre a solução da equação de Pell $x^2 - Ny^2 = 1$ e a aproximação da raiz quadrada $\sqrt{N}$ é de fato dada por $\sqrt{N} = \frac{x}{y}$, para soluções não triviais.

O seguinte código determina soluções da equação de Pell

$$ax^2 - Ny^2 = 1,$$

onde N não é um quadrado perfeito e, em seguida, determina uma aproximação para $\sqrt{N}$.

In [22]:
```
from sympy.solvers.diophantine.diophantine import diop_DN
from sympy import symbols, sqrt

def solve_pell_equation(N):
    x, y = symbols('x y')
    solution = diop_DN(N, 1)
    return solution

# Entrada do usuário para o valor de N
try:
    N = int(input("Digite um valor inteiro positivo para N: "))
    if N <= 0:
        print("Por favor, digite um número inteiro positivo.")
    else:
        # Encontrar uma solução para x^2 - Ny^2 = 1
        solution = solve_pell_equation(N)

        # Calcular a aproximação para sqrt(N)
        sqrt_N_approx = sqrt(N).evalf(subs={symbols('x'): solution[0][0], symbols('

        print(f"Solução para x^2 - {N}y^2 = 1: {solution}")
        print(f"Aproximação para sqrt({N}): {sqrt_N_approx}")
except ValueError:
    print("Por favor, digite um número inteiro válido.")
```

```
Digite um valor inteiro positivo para N: 61
Solução para x^2 - 61y^2 = 1: [(1766319049, 226153980)]
Aproximação para sqrt(61): 7.81024967590665
```

4. Solução para a equação de Pell

A equação de Pell é uma equação diofantina da forma

$$x^2 - dy^2 = 1$$

onde d é um inteiro positivo não quadrátrico.

A solução fundamental para a equação de Pell é um par (x, y) de inteiros satisfazendo a equação onde x e y são minimais e positivos.

Existe sempre uma solução para esta equação: a solução trivial $(1, 0)$. Esta não é contada.

Teorema: as soluções inteiras e positivas da equação de Pell são precisamente os pares (x_n, y_n) que satisfazem

$$x_n + y_n\sqrt{d} = (x_1 + y_1\sqrt{d})^n.$$

Por exemplo, para $d = 2$, temos

$$x_1 + y_1\sqrt{2} = 3 + 2\sqrt{2}$$

$$x_2 + y_2\sqrt{2} = 17 + 12\sqrt{2}$$

$$x_3 + y_3\sqrt{2} = 99 + 70\sqrt{2}$$

$$x_4 + y_4\sqrt{2} = 577 + 408\sqrt{2}$$

Por exemplo, para $d = 3$, temos

$$x_1 + y_1\sqrt{3} = 2 + 1\sqrt{3}$$

$$x_2 + y_2\sqrt{3} = 7 + 4\sqrt{3}$$

$$x_3 + y_3\sqrt{3} = 26 + 15\sqrt{3}$$

$$x_4 + y_4\sqrt{3} = 97 + 56\sqrt{3}$$

O seguinte codigo determina as soluções da equação de Pell.

In [14]:
```
import math

def solvePell(n):
    x = int(math.sqrt(n))
    y, z, r = x, 1, x << 1
    e1, e2 = 1, 0
    f1, f2 = 0, 1
    while True:
        y = r * z - y
        z = (n - y * y) // z
        r = (x + y) // z

        e1, e2 = e2, e1 + e2 * r
        f1, f2 = f2, f1 + f2 * r

        a, b = f2 * x + e2, f2
        if a * a - n * b * b == 1:
            return a, b

for n in [2, 3, 61, 109, 181, 277]:
    x, y = solvePell(n)
    print("x^2 - %3d * y^2 = 1 para  x = %27d e y = %25d" % (n, x, y))
```

```
x^2 -   2 * y^2 = 1 para  x =                           3 e y =
2
x^2 -   3 * y^2 = 1 para  x =                           2 e y =
1
x^2 -  61 * y^2 = 1 para  x =                  1766319049 e y =                  22
6153980
x^2 - 109 * y^2 = 1 para  x =             158070671986249 e y =             1514042
4455100
x^2 - 181 * y^2 = 1 para  x =          2469645423824185801 e y =          18356729868
3461940
x^2 - 277 * y^2 = 1 para  x =        159150073798980475849 e y =        956240117387
8027020
```

Como podemos observar, o par $(1, 0)$ é sempre solução. Esta solução é chamada de solução trivial. Chamamos de solução fundamental à menor solução não trivial.

Conhecendo-se a solução fundamental (a, b), podemos determinar todas as demais soluções usando o par de fórmulas de recorrência:

$$x_{k+1} = ax_k + dby_k$$

$$y_{k+1} = bx_k + ay_k, k \geq 0.$$

que pode ser simplificada para

$$x_{k+2} = 2ax_{k+1} - x_k$$

$$y_{k+2} = 2ay_{k+1} - y_k,$$

onde $x_0 = 1$, $y_0 = 0$ e $x_1 = a$ e $y_1 = b$.

Como determinar uma solução fundamental da equação de Pell? Não existe uma fórmula fechada para determinar a solução fundamental da equação de Pell, mas existem vários algoritmos que podemos rodar para a solução fundamental.

O algortimo das frações contínuas devido a Brownker é um dos mais diretos.

Para este método precisamos inicialmente determinar a expansão em frações contínuas de $\sqrt{d}$ e paramos quando ela começa a repetir. Então, calculamos a fração convergente do primeiro ciclo excluindo o último elemento do ciclo. O numerador é x_1 e o denominador é y_1.

Por exemplo, $\sqrt{23} = [4; 1, 3, 1, 8, 1, 3, 1, 8, \cdots] = [4; \overline{1, 3, 1, 8}]$.

Então,

$$4 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1}}}$$

Ou seja,

$$\frac{24}{5} \approx \sqrt{23},$$

e assim, $(24, 5)$ é uma solução fundamental de $x^2 - 23y^2 = 1$.

Programinha usando que a solução é dada por

$$x_{k+2} = 2ax_{k+1} - x_k$$

$$y_{k+2} = 2ay_{k+1} - y_k,$$

onde $x_0 = 1$, $y_0 = 0$ e $x_1 = a$ e $y_1 = b$.

In [19]:
```
# solução de $x^2-2y^2=1$ tem solução fundamental (3,2)
a = 3
b = 2
X = [1, a]
Y = [0, b]
n = int(input("Digite um número natural maior do que ou igual a 2: "))
for k in range(2,n):
    X_k = 2*a* X[k-1] - X[k-2]
    Y_k = 2*a* Y[k-1] - Y[k-2]
    X.append(X_k)
    Y. append(Y_k)
print('X=', X)
print('Y=', Y)
```

```
Digite um número natural maior do que ou igual a 2: 5
X= [1, 3, 17, 99, 577]
Y= [0, 2, 12, 70, 408]
```

Sympy do Python tem uma função para determinar uma solução da diofantina #x^2-dy^2=1$.

In [21]:
```
#Python
from sympy.solvers.diophantine.diophantine import diop_DN
diop_DN(61, 1) # Solves equation x**2 - 2*y**2 = 1
```

Out[21]:
```
[(1766319049, 226153980)]
```

Frações Contínuas

Frações contínuas finitas

Sejam a e b inteiros com $b > 0$. Se o algoritmo de Euclides para estes inteiros produz a sequência

$$\begin{aligned} a &= q_0 b + r_0 && (1)\\ b &= q_1 r_0 + r_1 && (2)\\ r_0 &= q_1 r_1 + r_2 && (3)\\ \vdots &= \vdots && (4)\\ r_{k_0-2} &= q_{k_0-1} r_{k_0-1} + r_{k_0} && (5)\\ r_{k_0-1} &= q_{k_0} r_{k_0}, && (6) \end{aligned}$$

Então podemos escrever

$$\frac{a}{b} = q_0 + \frac{r_0}{b} = q_0 + \frac{1}{\frac{b}{r_0}} = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cdots + \cfrac{1}{q_{k_0-1} + \cfrac{1}{q_{k_0}}}}}$$

Esta última expressão é chamada de expanão em fração contínua de $\frac{a}{b}$ e representamos por $[q_0; q_1, q_2, \ldots, q_{k_0}]$ e também dizemos que $[q_0; q_1, q_2, \ldots, q_{k_0}]$ representa $\frac{a}{b}$.

Em geral, $[a_0; a_1, a_2, \ldots, a_n]$ dá uma fração contínua finita se cada a_k é um inteiro positivo exceto possivelmente a_0. Note que a expansão não é necessariamente única.

Exemplo: $$

\frac{21}{13} = 1+ \frac{8}{13}= 1+\frac{1}{1+\frac{5}{8}}

1+\frac{1}{1+\frac{1}{1+ \frac{3}{5}}}

1+\frac{1}{1+\frac{1}{1+ \frac{1}{1+ \frac{2}{3}}}} =$$

$$= 1 + \cfrac{1}{1 + \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{2}}}}}$$

$$= 1 + \cfrac{1}{1 + \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\frac{1}{1}}}}}}$$

Assim,

$$\frac{21}{13} = [1; 1, 1, 1, 1, 2] = [1; 1, 1, 1, 1, 1, 1].$$

Até agora foram consideradas apenas expansões finitas de frações contínuas, mas frações contínuas infinitas também são interessantes.

Frações contínuas infinitas tem a forma

$$[a_0; a_1, a_2, a_3, \ldots] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \frac{1}{a_3 + \ldots}}}$$

onde $a_0, a_1, a_2, \ldots$ são inteiros todos positivos exceto possivelmente a_0

Questões de convergência são pertinentes, mas serão tratadas posteriormente.

Exemplo: assuma que faz sentido que o número real α admite a seguinte fração contínua infinita $[1; 1, 1, 1, 1, \ldots]$. Determine α.

De fato, se

$$\alpha = [1; 1, 1, 1, 1, \ldots] = 1 + \cfrac{1}{1 + \cfrac{1}{1 + \frac{1}{1 + \ldots}}}$$

então,

$$\alpha = 1 + \frac{1}{\alpha}.$$

Portanto,

$$\alpha^2 - \alpha - 1 = 0.$$

Segue que $\alpha = \frac{1+\sqrt{5}}{2}$ ou $\alpha = \frac{1-\sqrt{5}}{2}$.

É claro que $\alpha > 0$ e assim, $\alpha = \frac{1+\sqrt{5}}{2}$.

Exemplo: assuma que faz sentido que o número real α admite a seguinte fração contínua infinita $[1; 2, 2, 2, 2, \ldots]$. Determine α.

De fato, se $\alpha > 0$ tem que satisfazer

$$\alpha - 1 = \frac{1}{1 + \alpha}.$$

Portanto,

$$\alpha^2 - 1 = 1.$$

Segue que $\alpha = \sqrt{2}$.

5. Mais sobre a solução para a equação de Pell

A equação de Pell é uma equação diofantina da forma

$$x^2 - dy^2 = 1$$

onde d não é um inteiro não quadrátrico.

Se $d = k^2$ fosse um inteiro quadrático, poderia ser incorporado á variável y e teríamos

$$x^2 - (ky)^2 = 1$$

o que resultaria em

$$(x - ky)(x + ky) = 1.$$

Ou resultaria em

$$y = \frac{1}{k}\sqrt{x^2 - 1},$$

que não admite solução inteira positiva além da trivial $(1, 0)$.

Note que $x^2 - dy^2 = 1$ implica que

$$\left(\frac{x}{y}\right)^2 = d + \frac{1}{y^2} \approx d.$$

Assim, soluções da equação de Pell são aproximações para $\sqrt{d}$. Por esta razão o caso $x^2 - 2y^2$ foi estudado pelos pitagóricos pois se x e y são grandes então

$$\left(\frac{x}{y}\right) \approx \sqrt{2}.$$

Teorema 1: (Lagrange- 1768) Se d é um inteiro positivo não quadrado, então a equação

$$x^2 - dy^2 = 1$$

tem infinitas soluções.

Teorema 2: Para cada $k \in Z$, existe um número finito de soluções para $x^2 - dy^2 = k$ digamos $(x_1, y_1), \ldots, (x_n, y_n)$, tal que toda solução é da forma

$$x = ax_i + dby_i$$

e

$$y = bx_i + ay_i$$

onde

$$a^2 - db^2 = 1$$

.

Por exemplo,

$$x^2 - 3y^2 = -2$$

tem soluções dadas por

$$x = a \pm 3b, \text{ e } y = b \pm a$$

onde $a^2 - 3b^2 = 1$.

A solução fundamental para a equação de Pell é um par (x, y) de inteiros satisfazendo a equação onde x e y são minimais e positivos.

Existe sempre uma solução para esta equação: a solução trivial $(1, 0)$. Esta não é contada.

Teorema: as soluções inteiras e positivas da equação de Pell são precisamente os pares (x_n, y_n) que satisfazem

$$x_n + y_n\sqrt{d} = (x_1 + y_1\sqrt{d})^n.$$

Por exemplo, para $d = 2$, temos

$$x_1 + y_1\sqrt{2} = 3 + 2\sqrt{2}$$

$$x_2 + y_2\sqrt{2} = 17 + 12\sqrt{2}$$

$$x_3 + y_3\sqrt{2} = 99 + 70\sqrt{2}$$

$$x_4 + y_4\sqrt{2} = 577 + 408\sqrt{2}$$

Por exemplo, para $d = 3$, temos

$$x_1 + y_1\sqrt{3} = 2 + 1\sqrt{3}$$

$$x_2 + y_2\sqrt{3} = 7 + 4\sqrt{3}$$

$$x_3 + y_3\sqrt{3} = 26 + 15\sqrt{3}$$

$$x_4 + y_4\sqrt{3} = 97 + 56\sqrt{3}$$

A função a seguir determina a solução fundamental de

$$x^2 - dy^2 = 1.$$

In [1]:
```
import math

def solvePell(n):
    x = int(math.sqrt(n))
    y, z, r = x, 1, x << 1
    e1, e2 = 1, 0
    f1, f2 = 0, 1
    while True:
        y = r * z - y
        z = (n - y * y) // z
        r = (x + y) // z
```

```
        e1, e2 = e2, e1 + e2 * r
        f1, f2 = f2, f1 + f2 * r

        a, b = f2 * x + e2, f2
        if a * a - n * b * b == 1:
            return a, b
```

In [2]:
```
x, y = solvePell(61)
print('x=',x,',y=',y)
```

```
x= 1766319049 ,y= 226153980
```

In [3]:
```
for n in [2,3, 5, 61, 109, 181, 277, 341]:
    x, y = solvePell(n)
    print("x^2 - %3d * y^2 = 1 para  x = %27d e y = %25d" % (n, x, y))
```

```
x^2 -   2 * y^2 = 1 para  x =                           3 e y =
2
x^2 -   3 * y^2 = 1 para  x =                           2 e y =
1
x^2 -   5 * y^2 = 1 para  x =                           9 e y =
4
x^2 -  61 * y^2 = 1 para  x =                  1766319049 e y =                  22
6153980
x^2 - 109 * y^2 = 1 para  x =              158070671986249 e y =              1514042
4455100
x^2 - 181 * y^2 = 1 para  x =           2469645423824185801 e y =          18356729868
3461940
x^2 - 277 * y^2 = 1 para  x =       1591500737989804758 49 e y =       956240117387
8027020
x^2 - 341 * y^2 = 1 para  x =                    10626551 e y =
575460
```

Como podemos observar, o par $(1, 0)$ é sempre solução. Esta solução é chamada de solução trivial. Chamamos de solução fundamental à menor solução não trivial.

Conhecendo-se a solução fundamental (a, b), podemos determinar todas as demais soluções usando o par de fórmulas de recorrência:

$$x_{k+1} = ax_k + dby_k$$

$$y_{k+1} = bx_k + ay_k, k \geq 0.$$

que pode ser simplificada para

$$x_{k+2} = 2ax_{k+1} - x_k$$

$$y_{k+2} = 2ay_{k+1} - y_k,$$

onde $x_0 = 1$, $y_0 = 0$ e $x_1 = a$ e $y_1 = b$.

Usando que $a^2 - 1 = db^2$ as fórmulas explícitas para x_ke y_k são

$$x_k = \frac{1}{2}\left(a + b\sqrt{d}\right)^k + \frac{1}{2}\left(a - b\sqrt{d}\right)^k$$

$$y_k = \frac{1}{2}\sqrt{d}\left(a + b\sqrt{d}\right)^k - \frac{1}{2}\sqrt{d}\left(a - b\sqrt{d}\right)^k$$

6. Como determinar uma solução fundamental?

Como determinar uma solução fundamental da equação de Pell? Não existe uma fórmula fechada para determinar a solução fundamental da equação de Pell, mas existem vários algoritmos que podemos rodar para a solução fundamental.

O algortimo das frações contínuas devido a Brownker é um dos mais diretos.

Para este método precisamos inicialmente determinar a expanção em frações contínuas de $\sqrt{d}$ e paramos quando ela começa a repetir. Então, calculamos a fração convergente do primeiro ciclo excluindo o último elemento do ciclo. O numerador é x_1 e o denominador é y_1.

Por exemplo, $\sqrt{23} = [4; 1, 3, 1, 8, 1, 3, 1, 8, \cdots] = [4; \overline{1, 3, 1, 8}]$.

Então,

$$4 + \frac{1}{1 + \frac{1}{1+\frac{1}{1}}}$$

Ou seja,

$$\frac{24}{5} \approx \sqrt{23},$$

e assim, $(24, 5)$ é uma solução fundamental de $x^2 - 23y^2 = 1$.

A seguir apresentamos uma função que usa que a solução é dada por

$$x_{k+2} = 2ax_{k+1} - x_k$$

$$y_{k+2} = 2ay_{k+1} - y_k,$$

onde $x_0 = 1$, $y_0 = 0$ e $x_1 = a$ e $y_1 = b$ para determinar as n-primeiras soluções de

$$x^2 - dy^2 = 1;$$

In [17]:
```
# solução de $x^2-2y^2=1$ tem solução fundamental (3,2)
a = 3
b = 2
X = [1, a]
Y = [0, b]
n = int(input("Digite um número natural maior do que ou igual a 2: "))
for k in range(2,n):
    X_k = 2*a* X[k-1] - X[k-2]
    Y_k = 2*a* Y[k-1] - Y[k-2]
    X.append(X_k)
    Y. append(Y_k)
print('X=', X)
print('Y=', Y)
for i in range(0,n):
    print('Se deu zero',(X[i],Y[i]), 'é solução:', X[i]**2- 2*Y[i]**2 -1)
# This code is contributed by Doherty Andrade.
```

```
Digite um número natural maior do que ou igual a 2: 11
X= [1, 3, 17, 99, 577, 3363, 19601, 114243, 665857, 3880899, 22619537]
Y= [0, 2, 12, 70, 408, 2378, 13860, 80782, 470832, 2744210, 15994428]
Se deu zero (1, 0) é solução: 0
Se deu zero (3, 2) é solução: 0
Se deu zero (17, 12) é solução: 0
Se deu zero (99, 70) é solução: 0
Se deu zero (577, 408) é solução: 0
Se deu zero (3363, 2378) é solução: 0
Se deu zero (19601, 13860) é solução: 0
Se deu zero (114243, 80782) é solução: 0
Se deu zero (665857, 470832) é solução: 0
Se deu zero (3880899, 2744210) é solução: 0
Se deu zero (22619537, 15994428) é solução: 0
```

In [4]:
```
#Python
from sympy.solvers.diophantine.diophantine import diop_DN
diop_DN(23, 1) # Solves equation x**2 - 23*y**2 = 1
```

Out[4]:
```
[(24, 5)]
```

In [5]:
```
#Python
from sympy.solvers.diophantine.diophantine import diop_DN
diop_DN(61, 1) # Solves equation x**2 - 61*y**2 = 1
```

Out[5]:
```
[(1766319049, 226153980)]
```

7. O pacote solvers diofantine

This module contains diophantine() and helper functions that are needed to solve certain Diophantine equations. Esta parte contém ajuda para a função diophantine() que é útil pra resolver equações diofantinas.

EXEMPLO: resolver a diofantina

$$2x + 3y = 5.$$

In [6]:
```
from sympy.solvers.diophantine import diophantine
from sympy import symbols
x, y, z = symbols("x, y, z", integer=True)
```

In [7]:
```
from sympy.solvers.diophantine.diophantine import diop_solve

diop_solve(2*x + 3*y - 5)
```

Out[7]: $(3t_0 - 5,\ 5 - 2t_0)$

Exemplo: resolver a equação de Pell

$$x^2 - 13y^2 = 1$$

In [8]:
```
from sympy.solvers.diophantine.diophantine import diop_DN
diop_DN(2, 1) # Solves equation x**2 - 2*y**2 = 1
```

Out[8]:
```
[(3, 2)]
```

In [9]:
```
from sympy.solvers.diophantine.diophantine import diop_DN
diop_DN(13, 4) # Solves equation x**2 - 13*y**2 = 4
```

Out[9]:
```
[(119, 33), (11, 3), (1298, 360)]
```

In [10]:
```
from sympy.solvers.diophantine.diophantine import diop_DN
diop_DN(23, 1) # Solves equation x**2 - 23*y**2 = 1
```

Out[10]:
```
[(24, 5)]
```

In [11]:
```
#so tem a solução trivial
from sympy.solvers.diophantine.diophantine import diop_DN
diop_DN(9, 1) # Solves equation x**2 - 9*y**2 = 1
```

Out[11]:
```
[(1, 0)]
```

In [ ]:
```

```