Método de aceptación y rechazo

X: una variable discreta que queremos generar:

$$X \in \{x_1, x_2, x_3, \dots, x_n, \dots\}$$

$$P(X = x_j) = p_j$$

Se conoce cómo generar una variable Y, discreta

$$Y \in \{x_1, x_2, \dots, x_n, \dots\}$$ y pueden ser otros más

$$P(Y = x_j) = q_j$$

Además se cumple que $\frac{p_j}{q_j} \leq c$, para todo j donde $p_j \neq 0$.

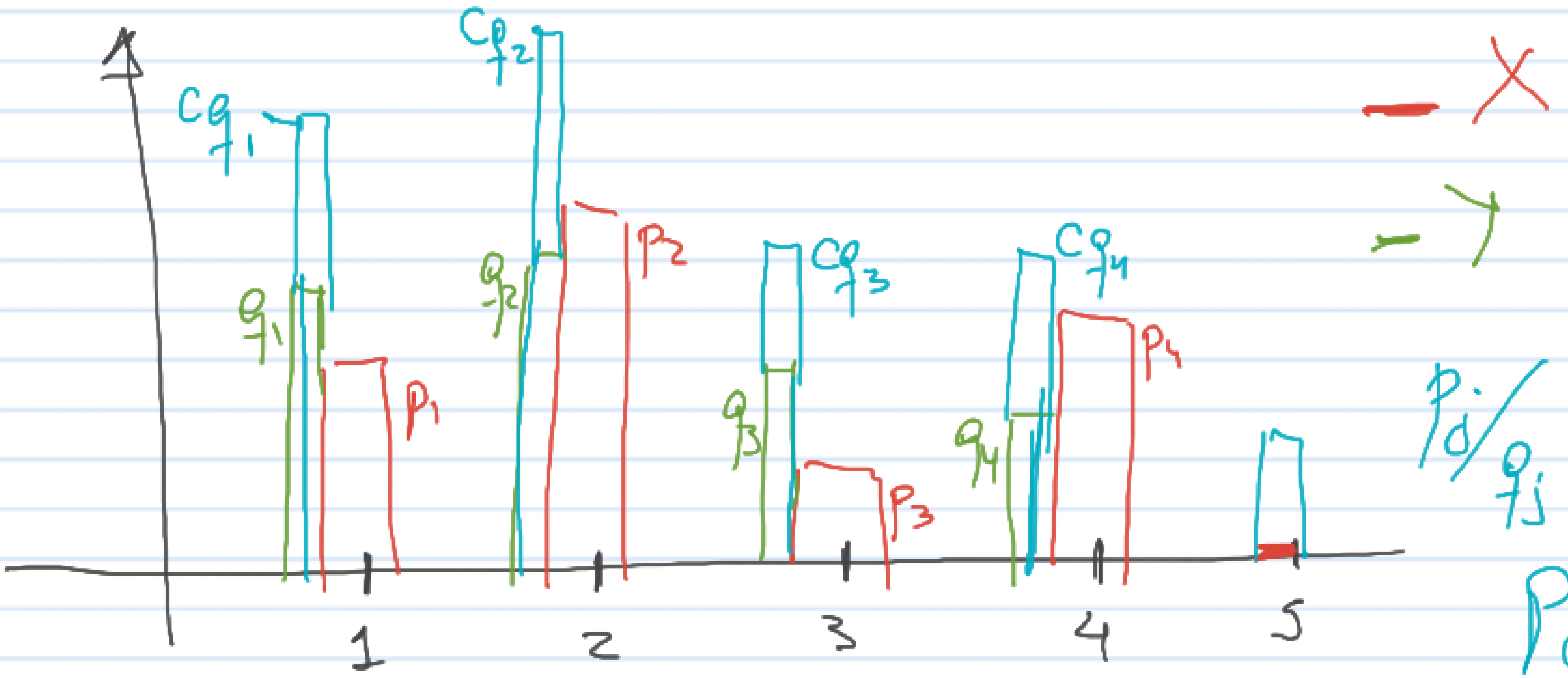
$$\underbrace{\sum_{j \geq 1} p_j}_{=1} \leq \sum_{j \geq 1} c\, q_j = c \underbrace{\sum_{j \geq 1} q_j}_{=1}$$

Luego $\boxed{c \geq 1}$

En realidad $c = 1$ implica Y y X son la misma variable.

Luego en el caso general ocurre $\boxed{c > 1}$

Método



$p_j / q_j \leq c$

$p_j \leq c \cdot q_j$

Algoritmo:

```
def AyR( ):
    while True:
        simular y
        U = random()
        if U < py / (c * qy):
            return y
```

¿Por qué funciona?

P(generar x_j) $= P$(generar x_j en la iteración 1 ó generar x_j en la iteración 2 ó … ó generar x_j en la iteración k ó …)

$= \sum_{k=1}^{\infty} P$(generar x_j en la iteración k)

P(generar x_j en la iteración k) $=$

$= P$(rechazar los k-1 primeros valores y generar x_j en la it. k y aceptarlo).

vamos cuál es la prob de aceptar un valor

= sigue en pizarra 6

$$P(\text{aceptar un valor}) = P(\text{generar } Y = x_1 \text{ y aceptarlo ó generar } Y = x_2 \text{ y aceptarlo ó } \vdots \text{ generar } Y = x_j \text{ y aceptarlo ó} \ldots)$$

$$= \sum_{i \geq 1} P\left(\text{generar } Y = x_i \text{ y } U < \frac{p_i}{c q_i}\right)$$

$$= \sum_{i \geq 1} P(\text{generar } Y = x_i) \cdot P\left(U < \frac{p_i}{c q_i}\right)$$

$$= \sum_{i \geq 1} \cancel{q_i} \cdot \frac{p_i}{c \cancel{q_i}} = \frac{1}{c} \sum_{i \geq 1} p_i$$

$$= \frac{1}{c}$$

$$= \sum_{k=1} \underbrace{\left(1 - \frac{1}{c}\right)^{k-1}}_{\text{rechazar } k-1 \text{ veces}} \underbrace{\cancel{q_j}}_{\text{generar } x_j} \cdot \underbrace{\frac{P_j}{c\,\cancel{q_j}}}_{\text{aceptar } x_j} = P_j \underbrace{\sum_{k=1}^{\infty} \frac{1}{c}\left(1-\frac{1}{c}\right)^{k-1}}_{=1}$$

$$\boxed{P(\text{generar } x_j) = P_j}$$

Ejemplo 5.3. Sea X una variable aleatoria con valores en $\{1, 2, \ldots, 10\}$ y probabilidades 0.11, 0.12, 0.09, 0.08, 0.12, 0.10, 0.09, 0.09, 0.10, 0.10. Si se generan valores con el método

Tomamos $Y \sim \mathcal{U}\{1, 10\}$ uniforme discreta;

$$q_j = 0,1 \qquad 1 \leq j \leq 10$$

$$\frac{p_j}{q_j} \leq \frac{0.12}{0.10} = 1.2$$

Podemos tomar $c = 1.2$.

$$\frac{p_j}{c\, q_j} = \frac{p_j}{1.2 \times 0,1} = \boxed{\frac{p_j}{0,12}}$$

Algoritmo

```
def varX():
    Y = int(random()*10)+1
    if random() < Py/0,12
    return Y
```

Como $\frac{1}{c} = P(\text{aceptar})$; entonces $E[\text{número de iteraciones}] = \frac{1}{1/c}$

$$= \boxed{c}$$

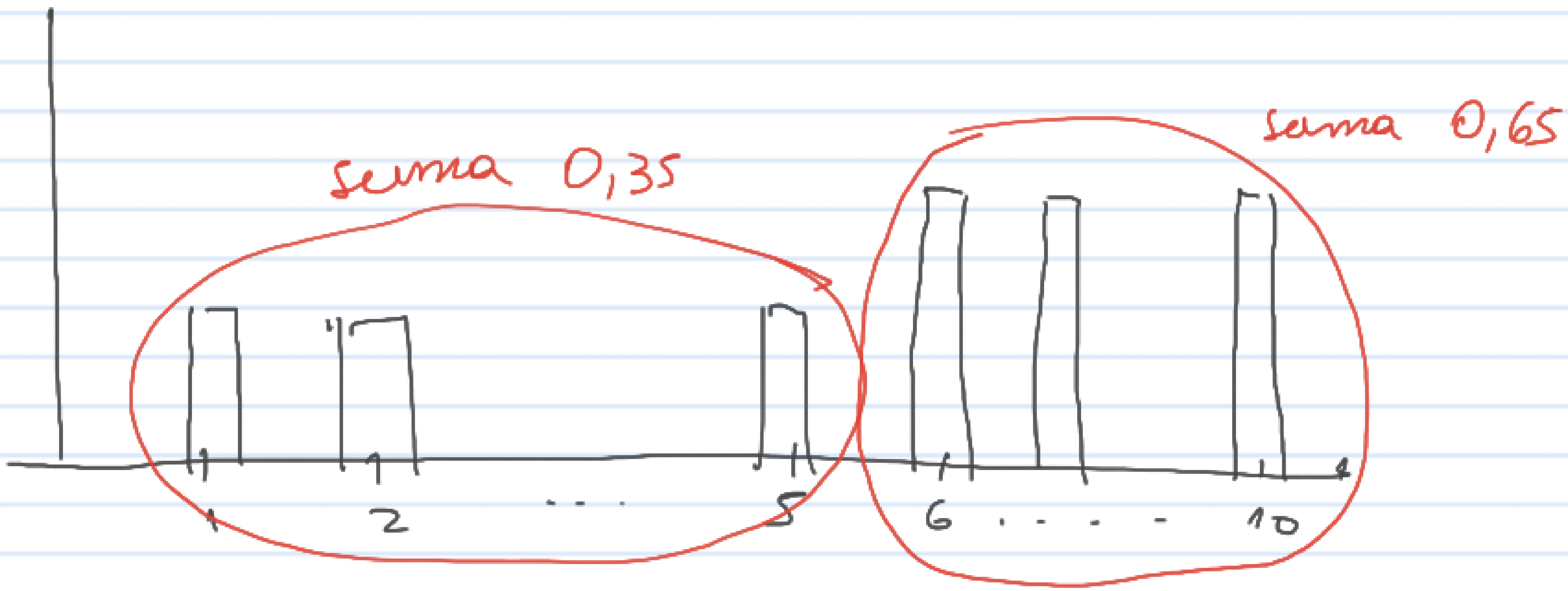
Luego cuanto menor sea c; menos iteraciones hace el algoritmo.

Método de composición

Ejemplo: $\quad X \in \{1, 2, \ldots, 10\}$

$$P(X = j) = 0{,}07 \qquad 1 \leq j \leq 5$$

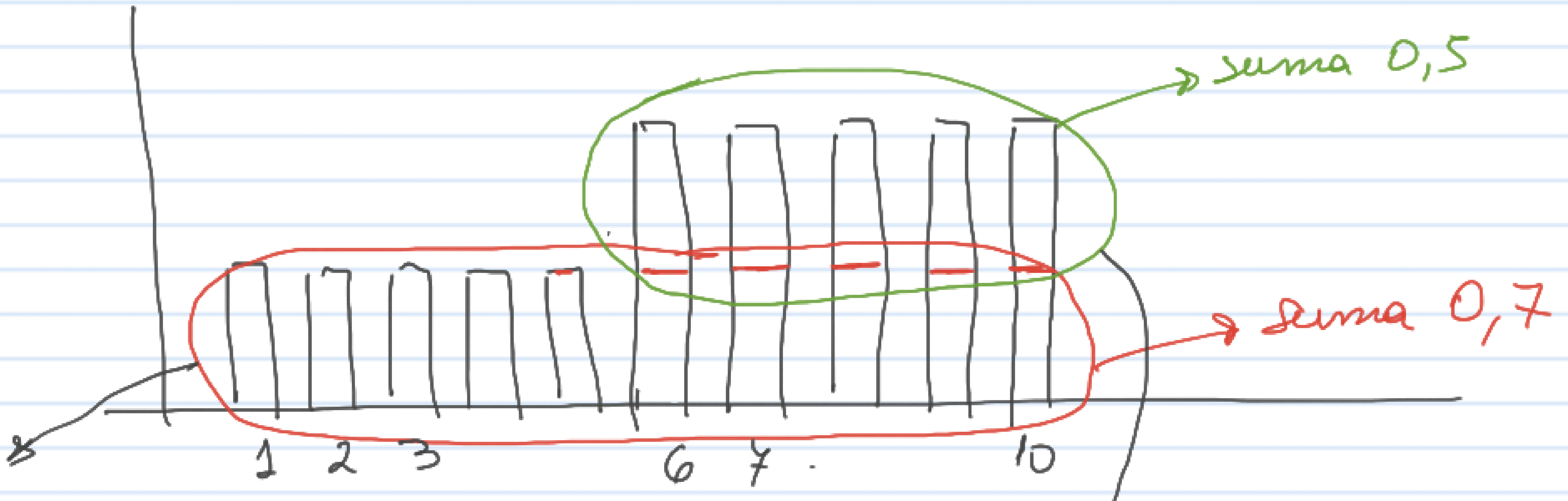
$$P(X = j) = 0{,}13 \qquad 6 \leq j \leq 10$$



Método

```
def compX ( ):
    U = random( )
    if U < 0.35 :
        return int(random( )*5) + 1
    else:
        return int(random( )*5) +6
```

Otra forma



pensar en

$X_1 \sim \mathcal{U}(\{1,10\})$

$X_2 \sim \mathcal{U}(\{6,10\})$

Algoritmo:

```
def compX( ):
    U = random( )
    if U < 0,7 :
        return int(random( )*10)+1
    else
        return int(random( )*5)+6
```

P(generar 6) =

$P(U<0,7) \cdot 0,1 +$

$P(U>0,7) \cdot 0,2$

$= 0,07 + 0,06 = 0,13$

$$P(X \leq x) = \alpha_1 \cdot P(X_1 \leq x) + \alpha_2 \cdot P(X_2 \leq x) + \dots + \alpha_n P(X_n \leq x)$$

$X_1, X_2, \dots, X_n$; $\alpha_i > 0$; $\alpha_1 + \alpha_2 + \dots + \alpha_n = 1$

<u>Algoritmo</u>:

```
def compX( )
    U=random( )
    if U < α1:
        simular X1
    if U < α1+α2:
        simular X2
    ⋮
    elif simular Xn
```

Métodos para variables que toman una cantidad finita de valores

- Método del Alias
- Método de la urna

Método de la urna:

$$X \in \{0, 1, 2, \ldots, n-1\}; \qquad P(X=j) = p_j$$

Sea M tal que $M \cdot p_j \in \mathbb{N}$; para todo j.

Se define un arreglo A de long M

0	0	0	0	1 1 ... 1	2 2 2	... $n-1$ $n-1$
$A[0]$	$A[1]$			$p_1 \times M$ veces	$p_2 \times M$ veces	$A[M-1]$

Algoritmo

```
def urnaX(A, M):
    U = random( )
    j = int(U * M)
    return A[j]
```

Ejemplo $X \in \{1, 2, 3, 4\}$;

$p_1 = 0.1 \qquad p_2 = 0.2 \qquad p_3 = 0.3 \qquad p_4 = 0,4$

1	2	2	3	3	3	4	4	4	4