Universidad Nacional de Colombia

Informe Tarea 2

Modelos y Simulaciones
Docente: Luis Gerardo Astaiza Amado

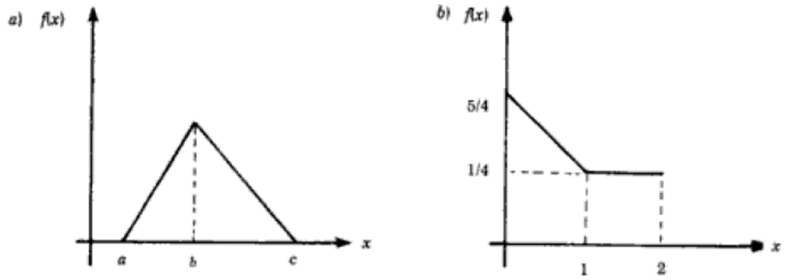
Grupo 2:

Acevedo Salinas, Juan Steban
Amaya Rodriguez, Maria Camila
Roncancio Toca, Camilo Andres
Velasquez Gomez, David Santiago

2024-1

Generación de variables aleatorias.

1. Generar por el método de la transformada inversa, números al azar que sigan las siguientes distribuciones de probabilidad:



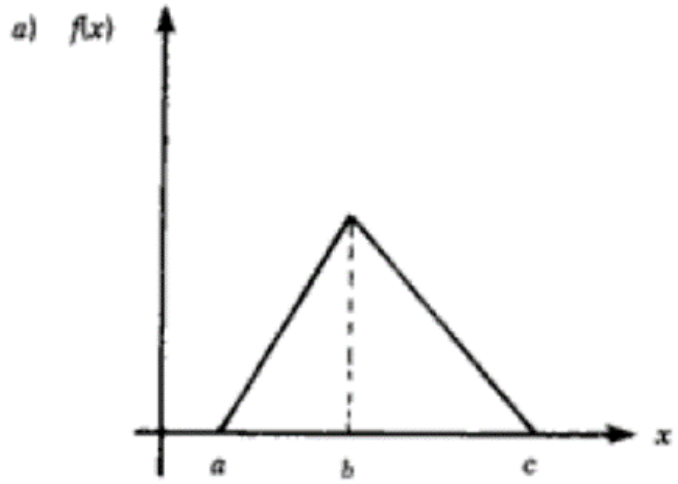
Para poder generar números aleatorios por el método de la transformada inversa, primero es necesario conceptualizar este método:

El método de la transformada inversa es una técnica utilizada en la generación de variables aleatorias que sigue una distribución específica. La idea detrás de este método es aprovechar la relación entre las distribuciones de probabilidad acumulativa (CDF) y las variables aleatorias

uniformemente distribuidas para generar variables aleatorias con la distribución deseada. Dentro de este método se establecen diversos componentes y procedimientos, descritos a continuación:

1. **Función de Distribución Acumulativa (CDF):** Comenzamos con una distribución de probabilidad específica para la cual queremos generar variables aleatorias. Esta distribución tiene una función de distribución acumulativa (CDF), denotada como $F(x)$, que mapea un valor x a la probabilidad de que la variable aleatoria sea menor o igual a x.
2. **Generación de Variables Aleatorias Uniformemente Distribuidas:** La primera etapa del método de la transformada inversa implica generar valores aleatorios uniformemente distribuidos en el intervalo [0, 1]. Estos valores aleatorios se denotan como U.
3. **Inversión de la Función de Distribución Acumulativa (CDF):** La clave del método de la transformada inversa es encontrar la inversa de la función de distribución acumulativa (CDF). Esto implica resolver $F^{-1}(U) = x$, donde U es un valor aleatorio uniformemente distribuido y x es el valor correspondiente de la variable aleatoria que queremos generar.
4. **Generación de Variables Aleatorias con la Distribución Deseada:** Una vez que se ha invertido la función de distribución acumulativa (CDF), podemos generar variables aleatorias con la distribución deseada al evaluar $F^{-1}(U)$, donde U es un valor aleatorio uniformemente distribuido.

Punto A



Dado que tenemos una distribución triangular, se define la función de densidad de probabilidad de la siguiente manera:

$$f(x) = \begin{cases} \frac{2(x-a)}{(b-a)(c-a)} & \text{si } a \leq x \leq b \\ \frac{2(c-x)}{(c-a)(c-b)} & \text{si } b < x \leq c \\ 0 & \text{en otro caso} \end{cases}$$

Una vez se obtiene la función de densidad de probabilidad se integra en ambos tramos de la función, con el objetivo de obtener la función de distribución acumulada:

- $a \leq x \leq b$

$$F(x) = \int_{-\infty}^{x} f(x)\,dx = \int_{-\infty}^{x} \frac{2(x-a)}{(b-a)(c-a)}\,dx = \frac{2(x-a)}{(b-a)(c-a)} \int_{a}^{x} (x-a)\,dx = \frac{(x-a)^2}{(b-a)(c-a)}$$

- $b \leq x \leq c$

$$F(x) = \int_{-\infty}^{x} f(x)\,dx = \frac{(b-a)}{(c-a)} + \int_{-\infty}^{x} \frac{2(c-x)}{(c-a)(c-b)}\,dx = \frac{2}{(c-a)(c-b)} \int_{a}^{x} (c-x)\,dx = 1 - \frac{(c-x)^2}{(c-a)(c-b)}$$

Ahora, reemplazamos F(x) por un Random (Rnd()) y despejamos x.

- $a \leq x \leq b$

$$F(x) = \frac{(x-a)^2}{(b-a)(c-a)} \rightarrow Rnd() = \frac{(x-a)^2}{(b-a)(c-a)}$$

$$\rightarrow (b-a)(c-a)Rnd() = (x-a)^2 \rightarrow \sqrt{(b-a)(c-a)Rnd()} = \sqrt{(x-a)^2}$$

$$\rightarrow (x-a) = \pm\sqrt{(b-a)(c-a)Rnd()} \rightarrow x = \pm\sqrt{(b-a)(c-a)Rnd()} + a$$

- $b \leq x \leq c$

$$F(x) = 1 - \frac{(c-x)^2}{(c-a)(c-b)} \rightarrow Rnd() = 1 - \frac{(c-x)^2}{(c-a)(c-b)}$$

$$\rightarrow 1 - Rnd() = \frac{(c-x)^2}{(c-a)(c-b)} \rightarrow (c-a)(c-b)(1-Rnd()) = (c-x)^2$$

$$\rightarrow \sqrt{(c-a)(c-b)(1-Rnd())} = (c-x)^2 \rightarrow \pm\sqrt{(c-a)(c-b)(1-Rnd())} = \sqrt{(c-x)^2}$$

$$\rightarrow \pm\sqrt{(c-a)(c-b)(1-Rnd())} = c - x \rightarrow \pm\sqrt{(c-a)(c-b)(1-Rnd())} + c = x$$

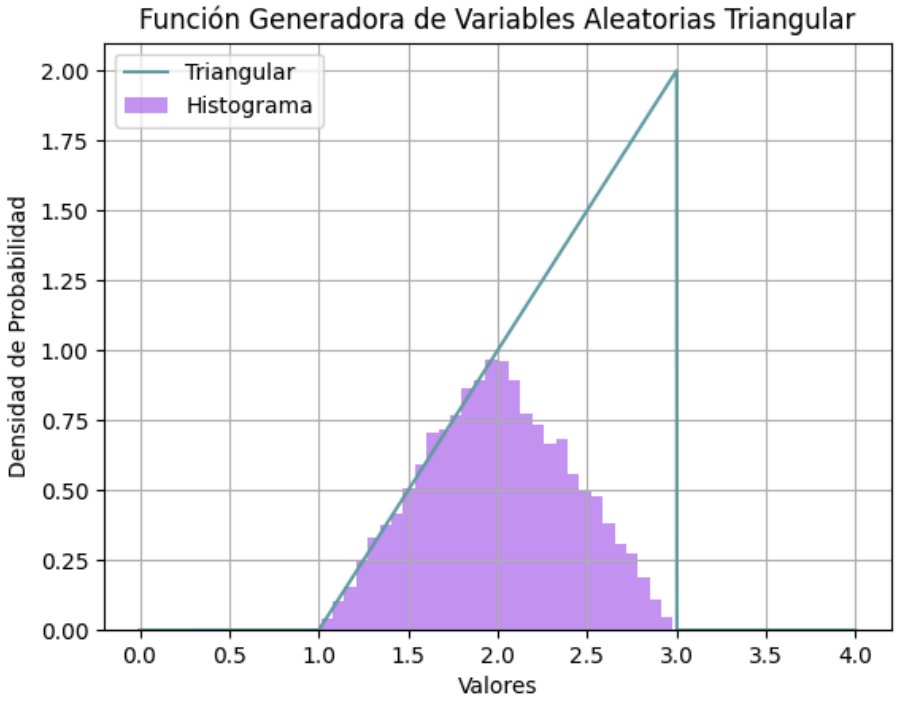
Finalmente, con el número U generamos números aleatorios dentro del intervalo [0, 1]

- Si $0 < U < (b - a)(c - a)$ entonces: $x = \sqrt{(b - a)(c - a)U} + a$
- Sino $x = -\sqrt{(c - a)(c - b)(1 - U)} + c$

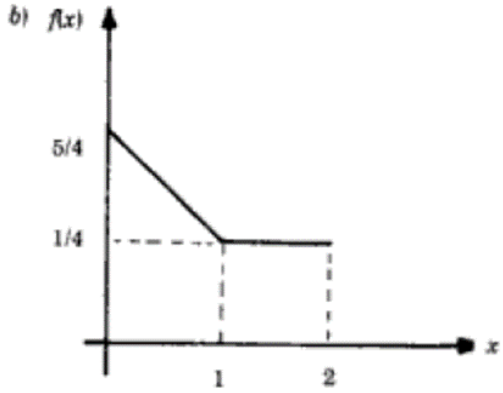
Esta información es suficiente para crear un programa en Python que permita generar 10000 números aleatorios, junto con su gráfica de densidad:

```python
import random
import math
import numpy as np
import matplotlib.pyplot as plt

# Función de densidad de probabilidad de la distribución triangular.
def triangular(x, a, b, c):

    if x < a:
        return 0
    elif a <= x < c:
        return 2*(x-a)/((b-a)*(c-a))
    elif c <= x <= b:
        return 2*(b-x)/((b-a)*(b-c))
    else:
        return 0

def aleatorios_triangular(rango_max):
    # Lista donde se almacenan los números generados
    aleatorios_generados = []
    # Valores de a, b, c y total
    a, b, c, total = 1, 2, 3, 0
    # Ciclo de 0 al número total de valores generados
    for _ in range(0, rango_max):
        U = random.random()
        if(U <= (b - a)/(c - a)):
            total = a + math.sqrt((b - a)*(c - a)*U)
        else:
            total = c - math.sqrt((c - a)*(c - b)*(1 - U))
        aleatorios_generados.append(total)

    return aleatorios_generados

# Generar variables aleatorias
x = aleatorios_triangular(10000)

x_values = np.linspace(0, 4, 10000)
y_values = [triangular(x_val, 1, 2, 3) for x_val in x_values]

# Graficar la función de densidad de probabilidad y el histograma de las variables aleatorias generadas
plt.plot(x_values, y_values, label='Triangular', color="cadetblue")
plt.hist(x, bins=30, density=True, alpha=0.5, label='Histograma', color="blueviolet")
plt.xlabel('Valores')
plt.ylabel('Densidad de Probabilidad')
plt.title('Función Generadora de Variables Aleatorias Triangular')
plt.legend()
plt.grid(True)
plt.show()
```

[2.0058479036501358, 2.0304878050280513, 1.4046296707333785, 2.297978745547471, 2.54280042998471, 1.6627344707462544, 2.5913670536132267, 2.1222801736545, 2.323849871159298, 1.9196417391980658, 1.4325928501185907, 2.3118011390310587, 2.4676243790186136, 1.3066408087905756, 1.4612748465522585, 2.140311339467047, 1.4837532246399108, 2.930292980353773, 2.4348658514295676, 2.4170288864802956, 1.1232302402454226, 2.0802092308003752, 2.3591735777234626, 2.082919169005075, 2.568571259521306, 2.136803587903154, 1.2774666754830226, 2.263993613828047, 1.3378261284098003, 2.118690187775628, 2.1076807845406322, 1.2781283777801768, 2.777210724048789, 2.252526306019562, 2.5776900588970584, 1.9436899160188836, 1.637928525155636, 1.3338427614823076, 1.4999429271357552, 2.1752233545615027, 2.072906367339791, 1.9530466253649306, 2.3383909321701513, 2.4633793383253066, 1.4230658037336392, 1.726516321157242, 2.749384470293113, 2.449195089817975, 2.800020835424366, 1.4871939266246323, 1.1104453410878286, 1.0776093528024937, 1.6935608576095469, 2.002633637391343, 2.764173450067477, 2.4711253949239994, 2.455905542725775, 1.8361540435933241, 1.815282468735839, 2.4391130579570452, 2.0326444016609235, 1.7154670326842636, 1.7668805546875295, 2.0523378688564553, 2.3345953962853527, 2.2034892846583425, 1.9429718177233655, 2.621996121126564, 2.2039028976611945, 1.2499948282095066, 2.08041517778171, 2.612999880279627, 1.6096300653421363, 1.8294469444799226, 1.7340217830780018, 2.100196889129964 7, 2.09915971164627, 1.9984586546439873, 2.19680815818282, 2.753284299295452 7, 1.9992652545961285, 1.9207170726574556, 1.5292936233798942, 1.9190688895514103116, 2.2170084815329725, 2.4767936295020565, 2.142574194042449, 2.528629498072424 7]



Punto B



Primeramente, se define la función de densidad de probabilidad de la siguiente manera:

- Para el intervalo [0, 1]:

Pendiente: $m = \frac{0.25 - 1.25}{1 - 0} = -1$

Ecuación de la recta: $\frac{5}{4} = -1 * 0 + b \rightarrow y = -x + \frac{5}{4}$

- Para el intervalo [1, 2]:

Recta: $y = b\ con\ b = \frac{1}{4}$

$$f(x) = \begin{cases} -x + \frac{5}{4} & \text{si } 0 \leq x \leq -1 \\ \frac{1}{4} & \text{si } 1 < x \leq 2 \\ 0 & \text{en otro caso} \end{cases}$$

Una vez se obtiene la función de densidad de probabilidad se integra en ambos tramos de la función, con el objetivo de obtener la función de distribución acumulada:

- Para el intervalo [0, 1]:

$$F(x) = \int_0^x f(x)\,dx = \int_0^x -x + \frac{5}{4}\,dx = -\frac{x^2}{2} + \frac{5}{4}x\Big|_0^x = -\frac{x^2}{2} + \frac{5}{4}x$$

- Para el intervalo [1, 2]:

$$F(x) = \int_1^x f(x)\,dx = \frac{3}{4} + \int_1^x \frac{1}{4}\,dx = \frac{3}{4} + \frac{1}{4}x\Big|_1^x = -\frac{x}{4} + \frac{x}{2}$$

Ahora, reemplazamos F(x) por un Random (Rnd()) y despejamos x.

- Para el intervalo [0, 1]:

$$F(x) = -\frac{x^2}{2} + \frac{5}{4}x \rightarrow Rnd() = -\frac{x^2}{2} + \frac{5}{4}x$$

$$\rightarrow -2Rnd() = x^2 - \frac{5}{2}x \rightarrow -2Rnd() + \frac{25}{16} = x^2 - \frac{5}{2}x + \frac{25}{16}$$

$$\rightarrow \sqrt{-2Rnd() + \frac{25}{16}} = \sqrt{\left(x - \frac{5}{4}\right)^2}, \rightarrow -\sqrt{-2Rnd() + \frac{25}{16}} + \frac{5}{4} = x$$

- Para el intervalo [1, 2]:

$$F(x) = \frac{x}{4} + \frac{1}{2} \rightarrow Rnd() = \frac{x}{4} + \frac{1}{2} \rightarrow 4Rnd() = x + 2 \rightarrow 4Rnd() - 2 = x$$

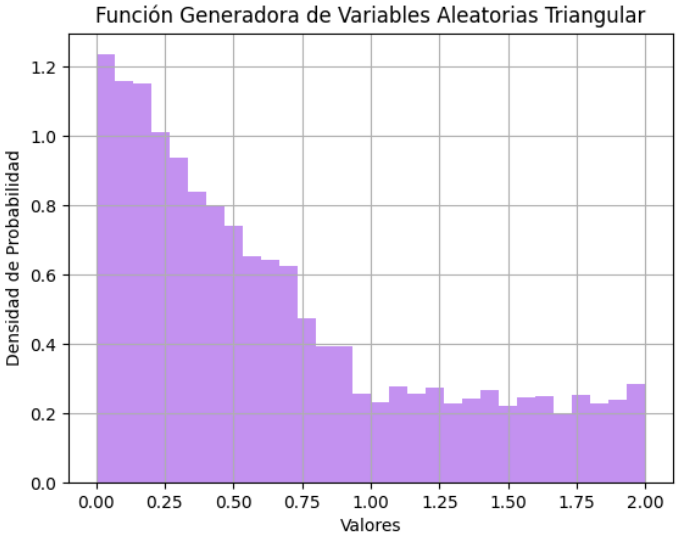
Finalmente, con el número *U* generamos números aleatorios dentro del intervalo [0, 1]

- Si $0 < U < \frac{3}{4}$ entonces: $x = -\sqrt{-2U + \frac{25}{16}} + \frac{5}{4}$
- Sino $x = 4U - 2$

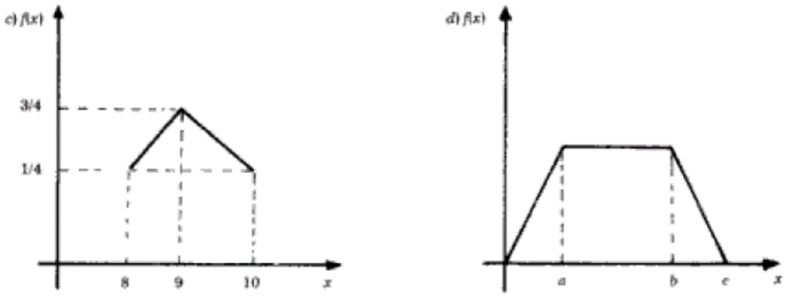
Esta información es suficiente para crear un programa en Python que permita generar 10000 números aleatorios, junto con su gráfica de densidad:

```python
import random
import math
import numpy as np
import matplotlib.pyplot as plt

def aleatorios_triangular(rango_max):
    aleatorios_generados = []

    for _ in range(rango_max):
        U = random.random()
        if U <= 3/4:
            total = (5/4) - math.sqrt(-2*U + 25/16)
        else:
            total = 4*U - 2
        aleatorios_generados.append(total)

    return aleatorios_generados

# Generar variables aleatorias
x = aleatorios_triangular(10000)
print(x)

# Graficar el histograma de las variables aleatorias generadas
plt.hist(x, bins=30, density=True, alpha=0.5, color="blueviolet")
plt.xlabel('Valores')
plt.ylabel('Densidad de Probabilidad')
plt.title('Función Generadora de Variables Aleatorias Triangular')
plt.grid(True)
plt.show()
```

```
[0.5450260457642443, 0.9207079056331707, 0.7966797922946397, 1.3244807498208258, 0.8715245162638239, 0.9068863183779067, 0.4850684142701931, 0.17056241426277396, 0.7966755772619133, 0.05058186628494998, 0.18810028561430103, 1.7381531274611048, 0.5062508926145982, 0.02233571997685346 3, 0.3339065913433900 7, 1.2095940123253457, 0.3893645725454928 4, 0.5354473850991154, 1.5019149217682366, 0.24215557616246586, 0.06471849136432994, 0.9727406946464652, 0.2499569461720980 7, 0.3028868628704836, 0.9410322287521922, 0.10351928876853234, 1.0574748916185737, 0.16734793971213913, 0.19127567706176252, 0.2835965133639373, 0.14567435000228834, 1.0480519207714272, 0.19110680797110735, 0.10583044625549176, 0.3770845950118733, 1.6140174594965675, 0.4003940658848171, 0.40013207922884897, 0.21330910085314359, 0.18154245675682223, 0.06147760606781105, 0.7793752723212273, 0.4367267470248164, 1.2706321774341132, 0.30043252514365204, 1.2373094846428927, 0.8720963473822441, 0.6219398786255357, 0.5674163986347843, 0.3752699578015739, 0.39351467544847474, 1.4815599643757338, 0.8081021901301395, 0.34543311679402466, 0.7440965609071276, 0.3061645170378091, 1.6274611260976752, 0.07019755609876088, 1.8841324462813716, 0.4135478881351795, 0.030285718253121763, 1.9597508983740441, 1.6130679842160274, 0.34814149036544184, 0.0350449130081083, 0.2320266762910108, 0.33641312439510096, 0.0700922133162507, 0.08783412738080454, 0.05777494760214408, 1.3818046289366501, 0.3191193783980 41, 0.09886216664745273, 0.07357461157149614, 0.11329917523426514, 0.388222964654234, 1.0203155057680409, 1.4778724014889852, 0.7373380103464499, 0.2586056360763954, 0.7406755775767669, 0.45658436848177797, 1.0153983990948365, 0.004793979697039319, 0.23597276292582747, 0.2884406381698683, 1.5610784793675188, 0.29713492156984667, 0.6152099058172977, 0.5066454061764257, 0.17703031829078664, 0.8502561645701122, 1.6005537351297492, 1.2794715287079481, 0.03063297559172873, 0.688658158195529, 0.8718196261232779, 1.0742385505593286, 0.08014856898045064, 0.23129474012991125]
```



2. Generar por el método de composición, números al azar que sigan las siguientes distribuciones:



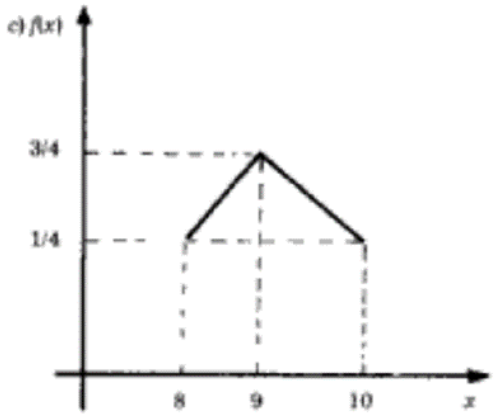
Para poder generar números aleatorios por medio del método de composición, primero es necesario conceptualizar este método:

El método de composición es una técnica alternativa utilizada en la generación de variables aleatorias que sigue una distribución específica, el cual aprovecha la relación entre varias distribuciones de probabilidad, utilizando una combinación de distribuciones de probabilidad más simples para generar una distribución más compleja. Para su implementación, se siguen los siguientes pasos:

1. **Identificar las funciones de distribución componentes y sus pesos:** En primer lugar, identificamos las funciones de distribución F_j componentes y sus respectivos pesos P_j. Cada F_j debe ser una función de distribución válida y la suma de todos los pesos debe ser igual a uno.

2. **Obtener las funciones de densidad de probabilidad componentes:** Luego, necesitamos obtener las funciones de densidad de probabilidad $f_j(x)$ correspondientes a cada función de distribución $F_j(x)$.
3. **Aplicar el método de la transformada inversa para cada función de distribución componente:** Para cada función de distribución $F_j(x)$ aplicamos el método de la transformada inversa para obtener las expresiones que nos permiten generar variables aleatorias que sigan esa distribución.
4. **Generar un número aleatorio:** Generamos un número aleatorio U_1 uniformemente distribuido en el intervalo (0,1).
5. **Seleccionar la función generadora correspondiente:** Seleccionamos la función generadora correspondiente $F_j(x)$ basada en el valor de U_1. Es decir, determinamos en qué tramo de la función compuesta cae el valor de U_1 y seleccionamos la función componente correspondiente.
6. **Generar un segundo número aleatorio:** Generamos un segundo número aleatorio U_2 uniformemente distribuido en el intervalo (0, 1) y se sustituye en la función generadora correspondiente para obtener el número aleatorio deseado.

Punto A



Se define la función de densidad de probabilidad para cada tramo, de la siguiente manera:

- Para el intervalo [8, 9] se halla la pendiente y se reemplaza en la ecuación de la recta

$$m = \frac{(0.75 - 0.25)}{(9 - 8)} = \frac{1}{2} \rightarrow \frac{1}{4} = 4 + b \rightarrow b = -\frac{15}{4}$$

$$y = \frac{1}{2}x - \frac{15}{4}$$

- Para el intervalo [9, 10] se halla la pendiente y se reemplaza en la ecuación de la recta

$$m = \frac{(0.75 - 0.25)}{(9 - 10)} = -\frac{1}{2} \rightarrow \frac{1}{4} = -5 + b \rightarrow b = \frac{21}{4}$$

$$y = -\frac{1}{2}x + \frac{21}{4}$$

$$f(x) = \begin{cases} \frac{1}{2}x - \frac{15}{4} & \text{si } 8 \leq x \leq 9 \\ -\frac{1}{2}x + \frac{21}{4} & \text{si } 9 \leq x \leq 10 \\ 0 & \text{en otro caso} \end{cases}$$

Una vez se obtiene la función de densidad de probabilidad se integra en ambos tramos de la función, con el objetivo de obtener la función de distribución acumulada:

- Para el intervalo [8, 9]:

$$F(x) = k\int_8^9 f(x)\,dx = \int_8^9 \frac{1}{2}x - \frac{15}{4}\,dx = 1 \rightarrow k\left(\frac{1}{2}\right) = 1 \rightarrow k = 2$$

así

$$f_1(x) = \begin{cases} x - \frac{15}{2} & \text{si } 8 \leq x \leq 9 \\ 0 & \text{en otro caso} \end{cases}$$

- Para el intervalo [9, 10]:

$$F(x) = k\int_9^{10} f(x)\,dx = \int_9^{10} -\frac{1}{2}x + \frac{21}{4}\,dx = 1 \rightarrow k\left(\frac{1}{2}\right) = 1 \rightarrow k = 2$$

así

$$f_2(x) = \begin{cases} -x + \frac{21}{2} & \text{si } 9 \leq x \leq 10 \\ 0 & \text{en otro caso} \end{cases}$$

Así se satisface que $f(x) = 0.5f_1(x) + 0.5f_2(x)$ y se determinan las funciones de distribución acumulada como $f_1(x) = \frac{x^2}{2} - \frac{15}{2}x + 28$ y $f_2(x) = -\frac{x^2}{2} + \frac{21}{2}x - 54$, se sustituye $f_1(x)$ y $f_2(x)$con Rnd().

- Para $f_1(x)$:

$$f_1(x) = \frac{x^2}{2} - \frac{15}{2}x + 28 \rightarrow Rnd() = \frac{x^2}{2} - \frac{15}{2}x + 28 \rightarrow 2(Rnd() - 28) = x^2 - 15x$$

$$\rightarrow 2(Rnd() - 28) + \frac{225}{4} = x^2 - 15x + \frac{225}{4} \rightarrow 2(Rnd() - 28) + \frac{225}{4} = \left(x - \frac{15}{2}\right)^2$$

$$\rightarrow \pm\sqrt{2(Rnd() - 28) + \frac{225}{4}} = \left(x - \frac{15}{2}\right) \rightarrow +\sqrt{2(Rnd() - 28) + \frac{225}{4}} + \frac{15}{2} = x$$

- Para $f_2(x)$:

$$f_2(x) = -\frac{x^2}{2} + \frac{21}{2}x - 54 \rightarrow Rnd() = -\frac{x^2}{2} + \frac{21}{2}x - 54 \rightarrow -2(Rnd() + 54) = x^2 - 21x$$

$$\rightarrow -2(Rnd() + 54) + \frac{441}{54} = x^2 - 21x + \frac{441}{4} \rightarrow -2(Rnd() + 54) + \frac{441}{4} = \left(x - \frac{21}{2}\right)^2$$

$$\rightarrow \pm\sqrt{2(Rnd() + 54) + \frac{441}{4}} = \left(x - \frac{21}{2}\right) \rightarrow +\sqrt{2(Rnd() + 54) + \frac{441}{4}} + \frac{21}{2} = x$$

Se genera un número aleatorio U_1 y U_2:

- Para $0 \leq U_1 \leq \frac{1}{2}$

$$+\sqrt{2(U_2 - 28) + \frac{225}{4}} + \frac{15}{2} = x$$

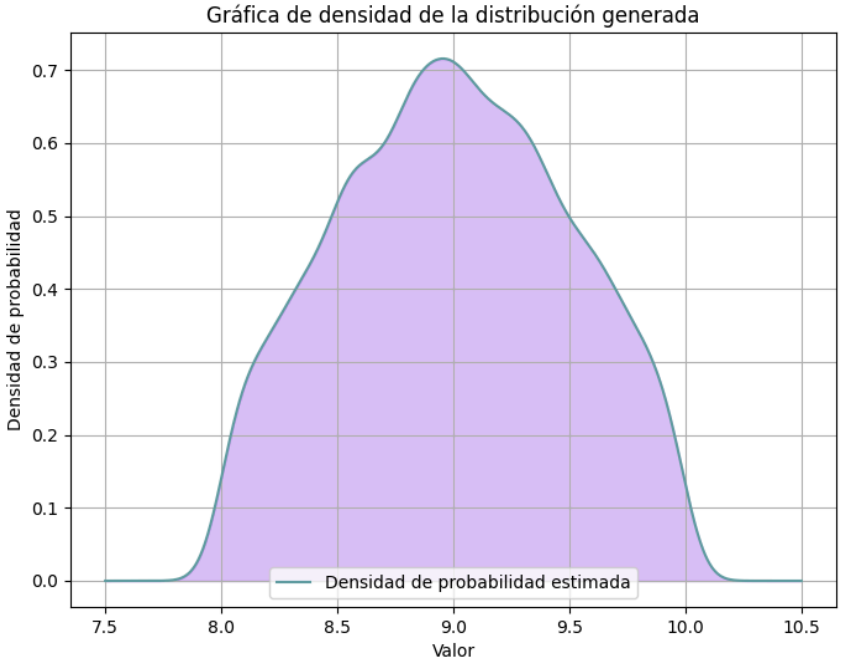
Sino:

$$-\sqrt{-2(U_2 + 54) + \frac{441}{4}} + \frac{21}{2} = x$$

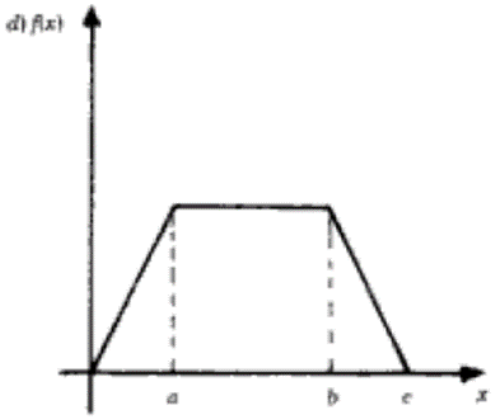
Esta información es suficiente para crear un programa en Python que permita generar 10000 números aleatorios, junto con su gráfica de densidad:

```python
import random, math
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

def aleatorios_composicion(rango_max):
    aleatorios_generados = []
    for _ in range(0, rango_max):
        total = 0
        U1 = random.random()
        U2 = random.random()
        if (U1 <= 1/2):
            total = (15/2) + math.sqrt(2*(U2 - 28) + 225/4)
        else:
            total = (21/2) - math.sqrt(-2*(U2 +54) + 441/4)

        aleatorios_generados.append(total)
    return aleatorios_generados

# Generar datos aleatorios
datos = aleatorios_composicion(10000)

# Gráfica de densidad de probabilidad
kde = gaussian_kde(datos)
x_vals = np.linspace(7.5, 10.5, 1000)
y_vals = kde(x_vals)
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label='Densidad de probabilidad estimada', color='cadetblue')
plt.fill_between(x_vals, y_vals, color='blueviolet', alpha=0.3)
plt.title('Gráfica de densidad de la distribución generada')
plt.xlabel('Valor')
plt.ylabel('Densidad de probabilidad')
plt.legend()
plt.grid(True)
plt.show()
```

```
[9.997388765020082, 8.738400898728763, 8.893960446678047, 8.95882422170871, 9.583280191633735, 8.863330073236133, 8.847624732120986, 9.427633883784223, 8.02729323816873, 8.240317196960993, 9.385250669805059, 9.202637589887061, 8.780870909651348, 9.702722782022462, 9.329257901242558, 8.980791877391626, 9.848219926259237, 8.602848174098407, 9.188307819360443, 8.760570107040067, 9.41365183639984, 8.297868329887011, 9.094552711113948, 9.668791778315423, 9.11311060402621, 8.16636438983484, 8.823191498498637, 9.02606301409932, 9.944181588902183, 9.664157472559726, 8.691552887475781, 8.582781731624836, 8.443107564018382, 8.853404881826425, 9.708808991622293, 9.039115144237803, 9.34086983002122, 9.942300503748367, 8.291598682286544, 8.990363268507904, 8.350402138113473, 9.129727479472006, 8.389052407653578, 9.311343616844086, 9.80271730010419, 8.633276339010548, 8.876384880047347, 9.132738936364232, 8.354614023830301, 8.508905831453895, 9.151292938431093, 9.067333459595796, 9.961783319684153, 8.832716634690755, 8.517123307707324, 9.55361542440535, 9.659287454138664, 9.157706655006827, 9.075044546011046, 8.879097722177159, 8.220769051002303, 9.927393957086394, 9.616434294544007, 9.63486360432086, 8.461557796877491, 9.942146625283486, 9.335580042293829, 9.718254956477564, 8.87798793551277, 9.092094145800843, 9.509804990835667, 9.068005539731526, 8.971380267097851, 8.97290477802311, 9.864417841589674, 9.1806474405637, 9.175727345035794, 8.804790081671653, 9.175751011759038, 9.148426918728234, 9.601473787815774, 9.989893394139719, 9.083570546937093, 9.512772078410972, 9.038854218229993, 9.084605300864848, 9.68751848349539, 9.29109263942723, 9.111055460728144, 9.02197041658435, 9.290429177106082, 8.774181952182056, 8.765665370210247, 8.229578522117038, 9.12364365850846, 9.196366014440231, 9.122743018848686, 9.047032134953842, 8.554294152107655, 8.797419357876535]
```



Punto B



Se define la función de densidad de probabilidad para cada tramo, de la siguiente manera:

- Para el intervalo [0, a]

$$y = \frac{2x}{a(c+b-a)}$$

- Para el intervalo [a, b]

$$y = \frac{2}{(c+b-a)}$$

- Para el intervalo [b, c]

$$y = -\frac{2x}{(c-b)(c+b-a)} + \frac{2c}{(c-b)(c+b-a)} = \frac{2x}{(c+b-a)} * \frac{(c-x)}{(c-b)}$$

Obtenemos la siguiente función de densidad donde $k = \frac{2}{c+b-a}$

$$f(x) = \begin{cases} 0 & \text{si } x \leq 0 \\ k\frac{x}{a} & \text{si } 0 < x < a \\ k & \text{si } a \leq x \leq b \\ k\frac{(c-x)}{(c-b)} & \text{si } b < x \leq c \\ 0 & \text{si } x > c \end{cases}$$

Una vez se obtiene la función de densidad de probabilidad se integra en los tramos de la función, con el objetivo de hallar m y obtener la función de distribución acumulada:

- Para el intervalo [0, a]

$$F_2(x) = m\int_0^a f(x)\,dx = 1 \to m\int_0^a \frac{kx}{a}\,dx = 1 \to \frac{mak}{2} = 1 \to m = \frac{2}{ak}$$

así

$$f_2(x) = \begin{cases} \frac{2x}{a^2} & \text{si } 0 < x \leq a \\ 0 & \text{en otro caso} \end{cases}$$

- Para el intervalo [a, b]

$$F_3(x) = m\int_a^b f(x)\,dx = 1 \to m\int_a^b k\,dx = 1 \to m(bk-ak) = 1 \to m = \frac{1}{k(b-a)}$$

así

$$f_3(x) = \begin{cases} \frac{1}{b-a} & \text{si } a < x \leq b \\ 0 & \text{en otro caso} \end{cases}$$

- Para el intervalo [b, c]

$$F_4(x) = m\int_b^c f(x)\,dx = 1 \rightarrow m\int_b^c k\frac{c-x}{c-b}\,dx = 1 \rightarrow \frac{mk(c-b)}{2} = 1 \rightarrow m = \frac{2}{k(c-b)}$$

así

$$f_4(x) = \begin{cases} \frac{2(c-b)}{(c-b)^2} & \text{si } b < x \leq c \\ 0 & \text{en otro caso} \end{cases}$$

Así se satisface que $f(x) = \frac{ak}{2}f_2(x) + k(b-a)f_3(x) + \frac{k(c-b)}{2}f_a(x)$ y se determinan las funciones de distribución acumulada como $f_2(x) = \frac{x^2}{a^2}$, $f_3(x) = \frac{x-a}{b-a}$ y $f_3(x) = \frac{2cx - x^2 - 2bc + b^2}{(c-b)^2}$, se sustituye $f_2(x)$, $f_3(x)$ y $f_4(x)$ con Rnd() y se despeja x.

- Para $f_2(x)$:

$$f_2(x) = \frac{x^2}{a^2} \rightarrow Rnd() = \frac{x^2}{a^2} \rightarrow a^2 Rnd() = x^2 \rightarrow \pm\sqrt{a^2 Rnd()}$$

- Para $f_3(x)$:

$$f_3(x) = \frac{x-a}{b-a} \rightarrow Rnd() = \frac{x-a}{b-a} \rightarrow (b-a)Rnd() = x - a \rightarrow (b-a)Rnd() + a = x$$

- Para $f_4(x)$:

$$f_4(x) = \frac{2cx - x^2 - 2bc + b^2}{(c-b)^2} \rightarrow Rnd() = \frac{2cx - x^2 - 2bc + b^2}{(c-b)^2} \rightarrow (c-b)^2 Rnd() = 2cx - x^2 - 2bc + b^2$$

$$\rightarrow -(c-b)^2 Rnd() - 2bc + b^2 = -2cx + x^2 \rightarrow -(c-b)^2 Rnd() - 2bc + b^2 + \frac{c}{4} = x^2 - 2cx + c^2$$

$$\rightarrow -(c-b)^2 Rnd() - 2bc + b^2 + c^2 = (x - c^2) \rightarrow -\sqrt{-(c-b)^2 Rnd() - 2bc + b^2 + c^2} + c = x$$

Se genera un número aleatorio U_1 y U_2:

- Para $0 \leq U_1 \leq \frac{ak}{2}$

$$\sqrt{a^2 U_2} = x$$

- Para $\frac{ak}{2}U_1 \leq k(b-a) + \frac{ak}{2}$

$$(b-a)U_2 + a = x$$

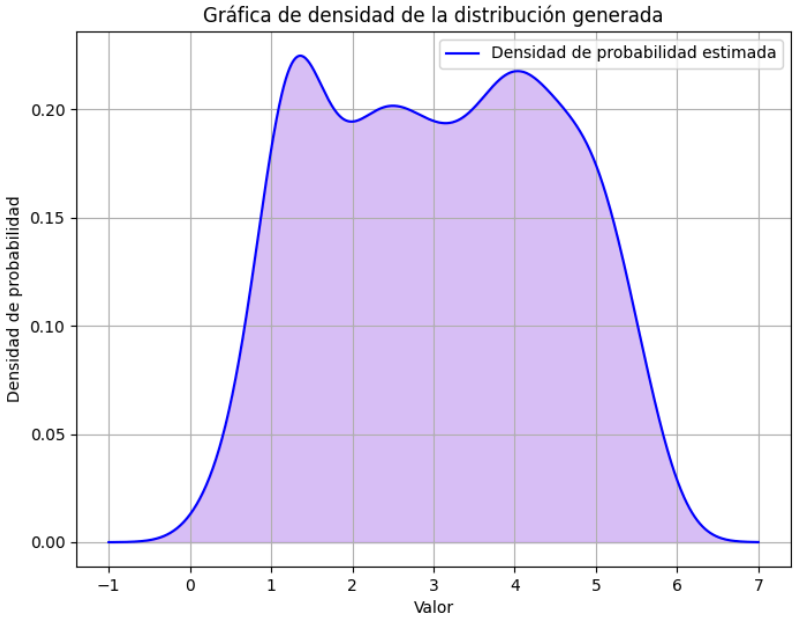
Sino:

$$-\sqrt{-(c-b)^2 Rnd() - 2bc + b^2 + c^2} + c = x$$

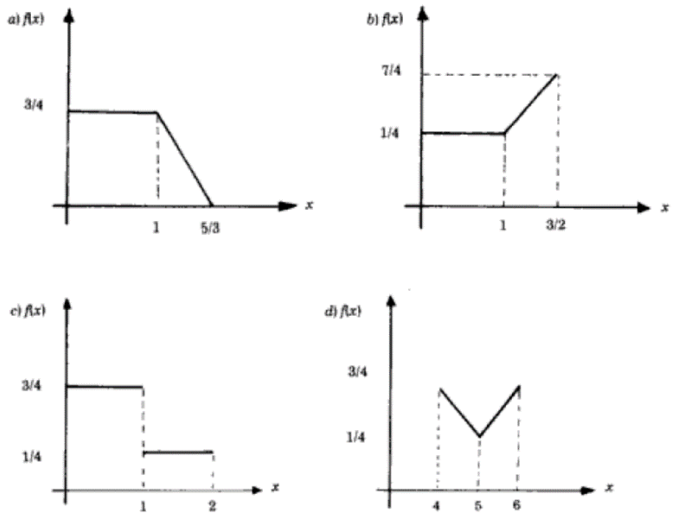
Esta información es suficiente para crear un programa en Python que permita generar 10000 números aleatorios, junto con su gráfica de densidad:

```python
import random
import math
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import gaussian_kde

def aleatorios_composicion(rango_max):
    aleatorios_generados = []
    a, b, c = 1, 5, 6
    k = 2 / (c + b - a)
    for _ in range(0, rango_max):
        total = 0
        U1 = random.random()
        U2 = random.random()
        if U1 <= (a * k) / 2:
            total = a * math.sqrt(2 * U2)
        elif U1 < k * (b - a) + (a * k) / 2:
            total = (b - a) * U2 + a
        else:
            total = c - math.sqrt((c * c) - 2 * b * c + b * b - U2 * (c * c) + 2 * U2 * c * b - U2 * b * b)

        aleatorios_generados.append(total)
    return aleatorios_generados

# Generar datos aleatorios
datos = aleatorios_composicion(100)
print(datos)

# Graficar la densidad de probabilidad estimada
kde = gaussian_kde(datos)
x_vals = np.linspace(-1, 7, 1000)
y_vals = kde(x_vals)
plt.figure(figsize=(8, 6))
plt.plot(x_vals, y_vals, label='Densidad de probabilidad estimada', color='blue')
plt.fill_between(x_vals, y_vals, color='blueviolet', alpha=0.3)
plt.title('Gráfica de densidad de la distribución generada')
plt.xlabel('Valor')
plt.ylabel('Densidad de probabilidad')
plt.legend()
plt.grid(True)
plt.show()
```

```
[3.319768546242594, 2.3548770485510757, 0.7550565872932885, 2.0297036931678845, 3.90905051268104, 2.177324765762332, 2.2031785332855556, 3.5221379370475776, 1.343854857440139, 4.073987809701082, 1.7371081051745576, 1.081104209569292, 1.202602004379092, 2.854599097238437, 2.009036809125323, 3.1215911177323044, 1.4024288468121644, 1.1704556678100977, 5.211688616621551, 0.5551331002100043, 4.482978120165733, 3.5696256780183786, 3.6605818619688955, 4.764630132367065, 4.498568842588027, 4.777413122419872, 3.810064866573008 6, 3.386360193107672, 2.628681912735837, 3.035431306542437, 1.2537595546660822, 2.412203811738082, 2.5138173674045436, 5.552741049305691, 1.1842830416003405, 1.9089494656491008, 1.3798400307399525, 3.1601048686084536, 4.949184990731655, 2.829815896035336, 2.6928780357976456, 1.7670065719218995, 3.604357457203674, 4.753081532164039, 5.460092236458812, 1.3147992605350944, 1.2837914057207855, 3.024458939778832, 4.199212647758318, 1.7692991291006996, 1.7340853786363981, 4.87553287663169, 4.980768563574158, 3.0408352165277934, 3.4988664434395838, 3.2678340737282703, 2.90469920216385, 3.599222552090855, 1.1337394344665643, 2.59586135889211, 4.173584683326206, 2.953712157897175, 1.5469155680039726, 2.4311474579709595, 3.9461441837143547, 3.2995279950296137, 0.8396423148971898, 3.960377528317077, 4.792031656688402, 1.8712836490351479, 1.8038689023033583, 3.4081947391470093, 1.0457729284469344, 1.7195325088834537, 1.6885447431909486, 3.587664874421303, 1.9080087742199399, 1.3011711708856295, 2.9298429499479974, 3.2999952147927885, 3.0777271391513312, 1.1538115740174464, 2.3602585909864198, 0.734038759703037, 2.979435144852344, 2.5614039992346886, 3.031974065589336, 5.266100948816354, 1.2431609129912657, 2.855132158665812, 1.7841798088204093, 4.844482164385022, 5.133624731997454, 2.8096009801799835, 2.8271709080241205, 0.8350146769210821, 1.1752465067848588, 1.195335696890015, 5.015980540940106, 0.39720606361643146]
```



3. Generar por el método de rechazo, números al azar que sigan las siguientes distribuciones de probabilidad:



El método del rechazo consiste en una técnica que requiere de los siguientes pasos para desarrollarse:

1. Identificar las funciones de distribución componentes
2. Obtener las expresiones matemáticas de las funciones identificadas
3. Generar dos números aleatorios R1 y R2.
4. Determinar el valor de un x dado por:

$$x = a + (b - a)R1$$

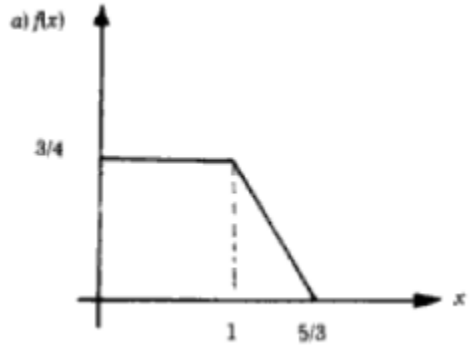
Donde a es primer valor que toma la función en su primer punto de la y b siendo el último.

5. Una vez obtenido x, se reemplaza en la respectiva función.
6. Se verifica si:

$$R2 < f(x)/M$$

Siendo M el valor máximo que puede tomar la función. De cumplirse la anterior condición se considera que el valor aleatorio es aceptado, en caso contrario el número es rechazado y vuelve a ejecutarse desde el paso 3.

a) Dada la siguiente distribución de probabilidad:



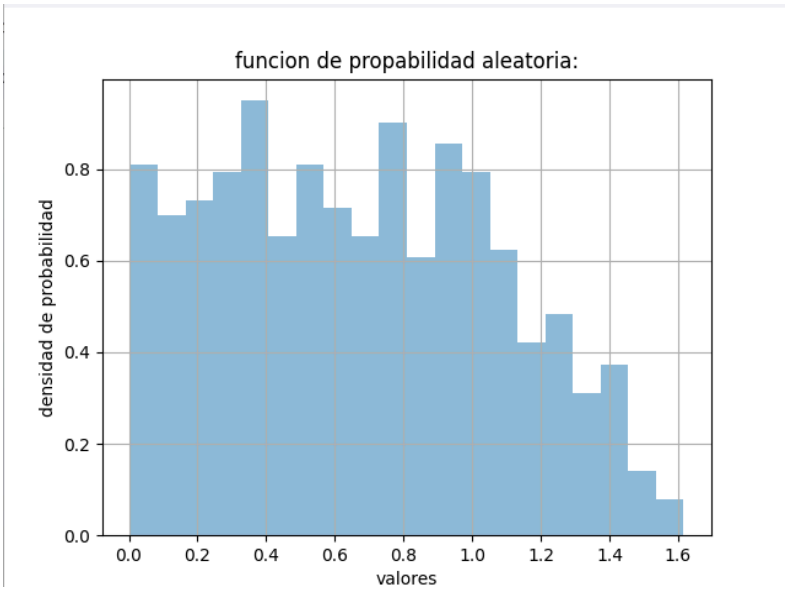
Se deduce que su respectiva función es:

$$f(x) = 3/4 \quad si\, 0 < x < 1$$
$$f(x) = -(9/8)x + 15/8 \quad si\, 1 < x < 5/3$$

Estas se ven implementadas dentro de la técnica del rechazo mediante el siguiente código en python:

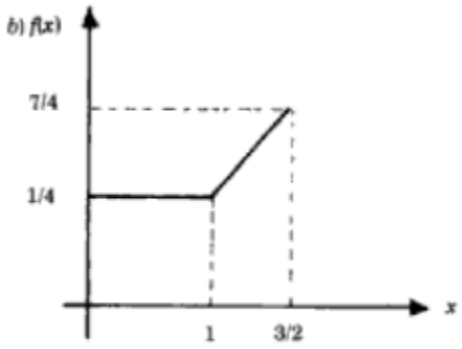
```python
import random
import math
import numpy as np
import matplotlib.pyplot as plt

def aleatorios1(rangoMax):
    randomGenerados = []
    a = 0
    b=1
    c=5/3
    M=3/4
    for _ in range(rangoMax):
        R1 = random.random()
        R2 = random.random()
        U = a+(c-a)*R1
        if U < b:
            y = 1/4
            randomGenerados.append(U)
        else:
            y= -(9*U/8)+(15/8)
            if R2 <= y / M:
                randomGenerados.append(U)
    return randomGenerados

x = aleatorios1(1000)
print(x)
plt.hist(x,bins=20,density=True,alpha=0.5)
plt.xlabel('valores')
plt.ylabel('densidad de probabilidad')
plt.title('funcion de propabilidad aleatoria:')
plt.grid(True)
plt.show()
```

Una vez ejecutado el código se obtiene la siguiente distribución para los números aleatorios generados:



Es posible notar que la distribución adquiere valores cercanos a ¾ o 0,75 del 0 al 1, mientras que estos empiezan a bajar su densidad a partir de este punto hasta el 5/3 o 1,66. Muestra un comportamiento similar al de la gráfica fácilmente observable.

b) Dada la siguiente distribución:



Deducimos que su función es:

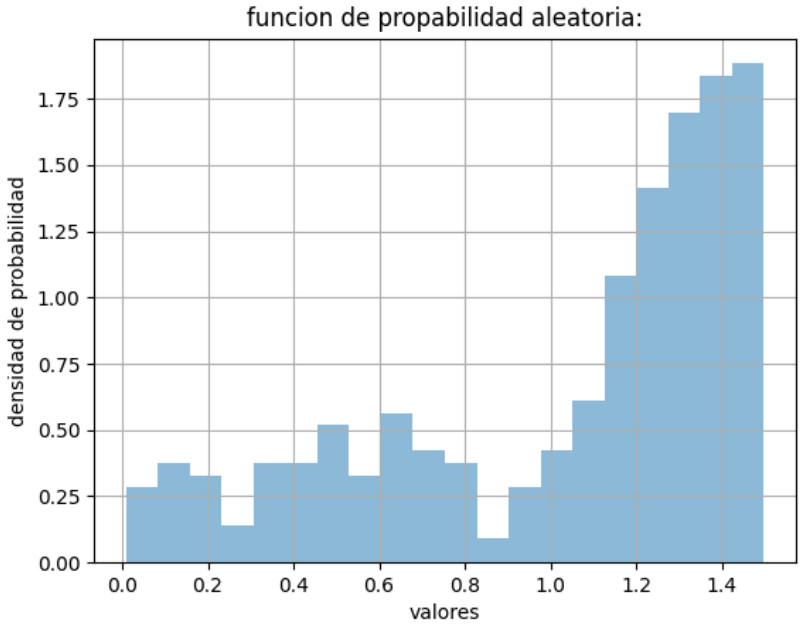
$$f(x) = 1/4 \quad si: 0 < x < 1$$

$$f(x) = 3x - 11/4 \quad si: 1 < x < 3/2$$

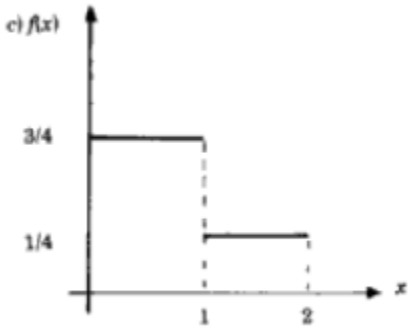
Implementamos esto dentro del siguiente código:

```
import random
import math
import numpy as np
import matplotlib.pyplot as plt

def aleatorios1(rangoMax):
    randomGenerados = []
    a = 0
    b=1
    c=3/2
    M=7/4
    for _ in range(rangoMax):
        R1 = random.random()
        R2 = random.random()
        U = a+(c-a)*R1
        if U < b:
            y = 1/4
        else:
            y= (3*U)-(11/4)
        if R2 <= y / M:
            randomGenerados.append(U)

    return randomGenerados
x = aleatorios1(1000)
print(x)
plt.hist(x,bins=20,density=True,alpha=0.5)
plt.xlabel('valores')
plt.ylabel('densidad de probabilidad')
plt.title('funcion de propabilidad aleatoria:')
plt.grid(True)
plt.show()
```

Y se obtiene la siguiente gráfica de densidad:



c) Dada la siguiente distribución:



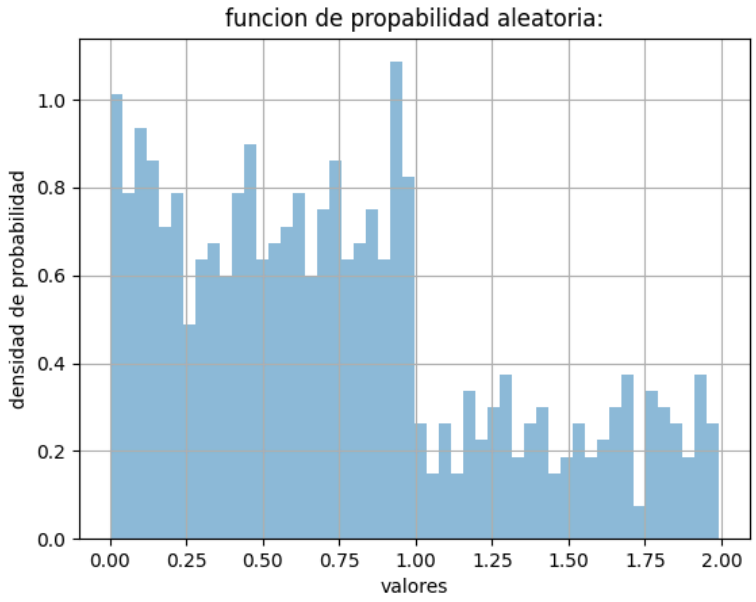
Se deduce que la función es:

$$f(x) = 3/4 \quad si\ 0 < x < 1$$
$$f(x) = 1/4 \quad si\ 1 < x < 2$$

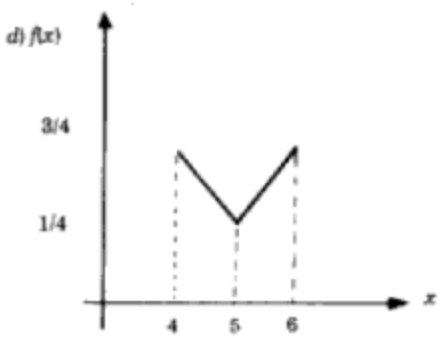
Implementada en el siguiente código:

```python
import random
import math
import numpy as np
import matplotlib.pyplot as plt

def aleatorios1(rangoMax):
    randomGenerados = []
    a = 0
    b=1
    c=2
    M=3/4
    for _ in range(rangoMax):
        R1 = random.random()
        R2 = random.random()
        U = a+(c-a)*R1
        if U < b:
            y = 3/4
        else:
            y= 1/4
        if R2 <= y / M:
            randomGenerados.append(U)

    return randomGenerados
x = aleatorios1(1000)
print(x)
plt.hist(x,bins=50,density=True,alpha=0.5)
plt.xlabel('valores')
plt.ylabel('densidad de probabilidad')
plt.title('funcion de propabilidad aleatoria:')
plt.grid(True)
plt.show()
```

Se obtiene la siguiente distribución:



d) Finalmente para la última distribución:



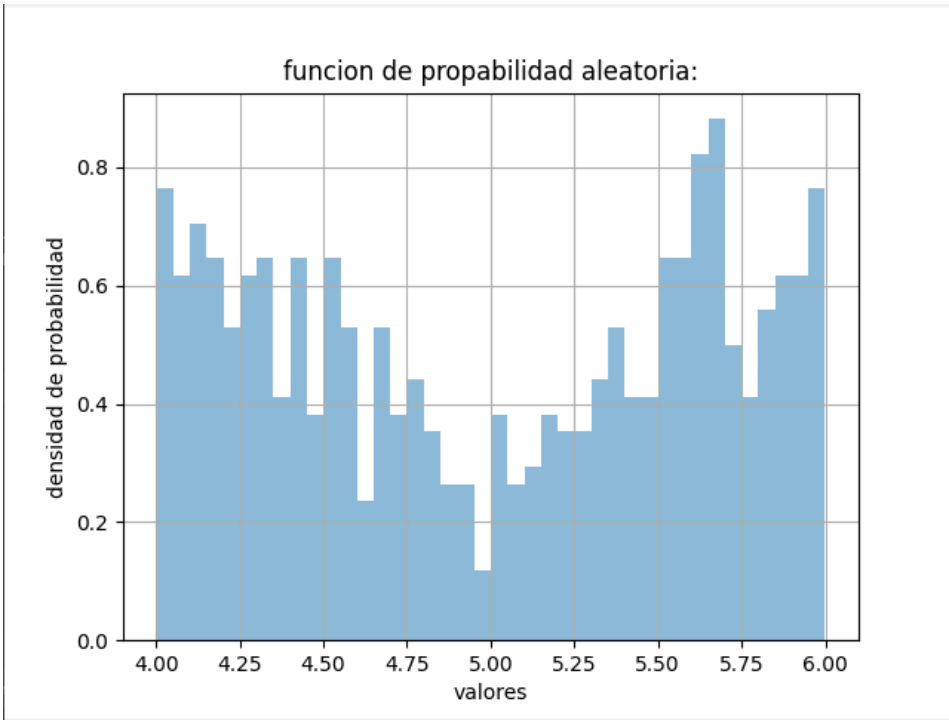
Se deduce que la función es:

$$f(x) = -(1/2)x + 11/4 \quad si: \ 4 < x < 5$$
$$f(x) = (1/2)x - 9/4 \quad si: \ 5 < x < 6$$

Implementado en codigo:

```
import random
import math
import numpy as np
import matplotlib.pyplot as plt

def aleatorios1(rangoMax):
    randomGenerados = []
    a = 4
    b=5
    c=6
    M=3/4
    for _ in range(rangoMax):
        R1 = random.random()
        R2 = random.random()
        U = a+(c-a)*R1
        if U < b:
            y = -(1/2)*U+(11/4)
        else:
            y= (1/2)*U-(9/4)
        if R2 <= y / M:
            randomGenerados.append(U)

    return randomGenerados
x = aleatorios1(1000)
print(x)
plt.hist(x,bins=40,density=True,alpha=0.5)
plt.xlabel('valores')
plt.ylabel('densidad de probabilidad')
plt.title('funcion de propabilidad aleatoria')
plt.grid(True)
plt.show()
```

Se obtiene la siguiente gráfica de densidad:



4. Diseñe los algoritmos de transformada inversa, composición, y rechazo aceptación para generar valores de variables aleatorias de cada una de las siguientes funciones de

(a)

$$f(x) = \begin{cases} \frac{3x^2}{2} & \text{if } -1 \le x \le 1 \\ 0 & \text{otherwise} \end{cases}$$

(b) For $0 < a < \frac{1}{2}$,

$$f(x) = \begin{cases} 0 & \text{if } x \le 0 \\ \frac{x}{a(1-a)} & \text{if } 0 \le x \le a \\ \frac{1}{1-a} & \text{if } a \le x \le 1-a \\ \frac{1-x}{a(1-a)} & \text{if } 1-a \le x \le 1 \\ 0 & \text{if } 1 \le x \end{cases}$$

densidad. Discuta cuál algoritmo es preferible para cada densidad (Primero grafique las densidades).

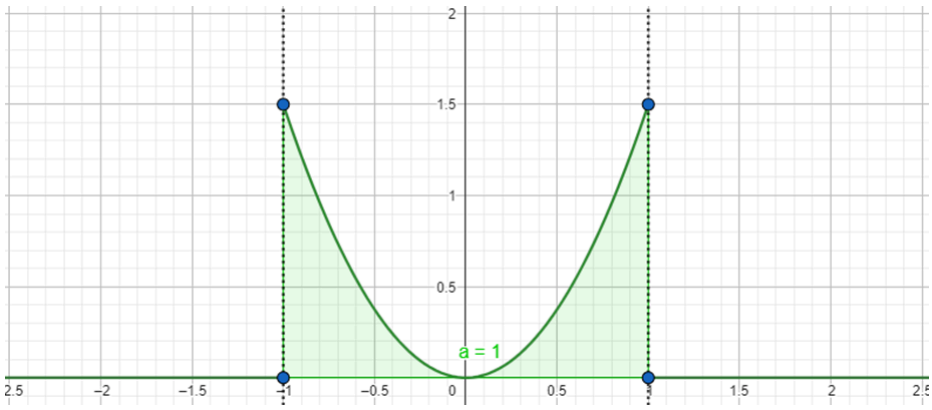
Solución: Los siguientes algoritmos son una versión generalizada para la generación de números aleatorios por medio de los métodos de la transformada inversa, composición y rechazo aceptación.

Algoritmos.py > acceptance_rejection_method

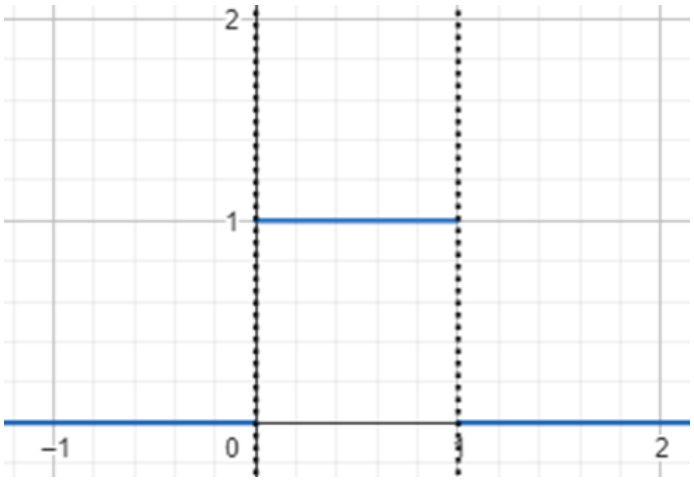
```python
import numpy as np

def inverse_transform_sampling(n_samples, inv_cdf):
    u = np.random.uniform(size=n_samples)

    samples = inv_cdf(u)

    return samples

def composition_method(n_samples, dist1, dist2, p1, size=1):
    b = np.random.binomial(n=1, p=p1, size=n_samples)

    samples1 = dist1.rvs(size=n_samples)
    samples2 = dist2.rvs(size=n_samples)

    samples = np.where(b, samples1, samples2)

    return samples

def acceptance_rejection_method(n_samples, target_pdf, proposal_pdf, proposal_rvs, c):
    samples = []

    while len(samples) < n_samples:
        y = proposal_rvs()

        u = np.random.uniform()

        if u <= target_pdf(y) / (c * proposal_pdf(y)):
            samples.append(y)

    return np.array(samples)
```

Las gráficas que representan las densidades son:

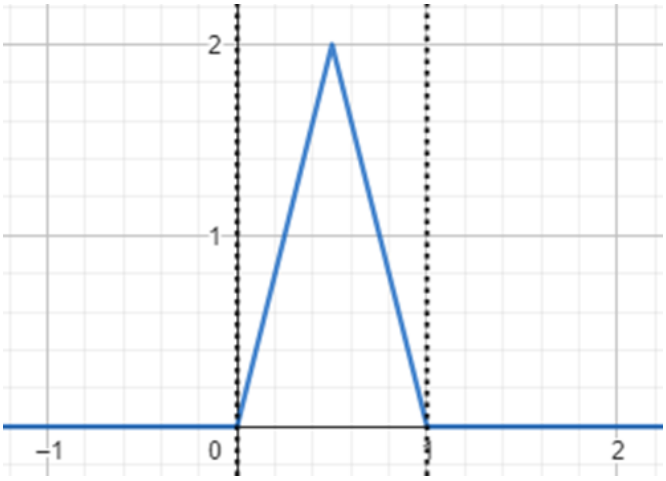
a)



b) Cuando 0 < a:



Cuando a < ½:

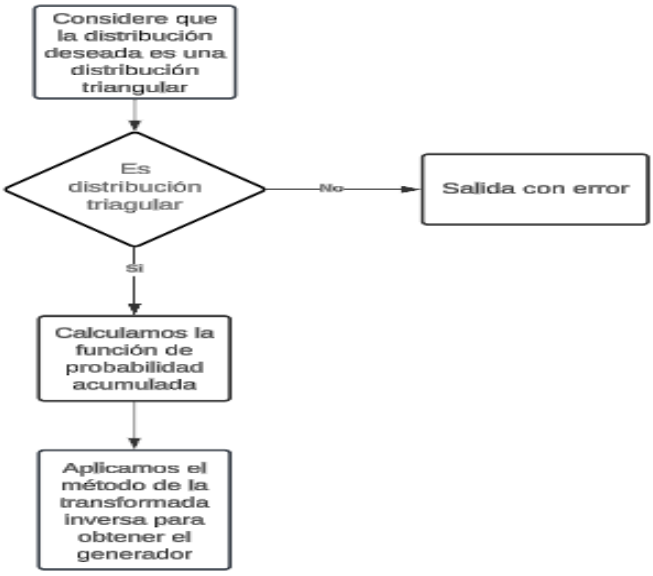


5. Desarrollar un esquema general para generar valores aleatorios para la distribución triangular con pdf:

$$f(x)=\begin{cases}\frac{1}{2}(x-2), & 2\le x\le 3\\ \frac{1}{2}\left(2-\frac{x}{3}\right), & 3<x\le 6\\ 0, & \text{otherwise}\end{cases}$$

Genere 10 valores de la variable aleatoria, evalué la media muestral , y compare con la media de la distribución.

Solución: El esquema que podemos extraer del uso de la distribución triangular de manera generalizada para la generación de valores aleatorios.



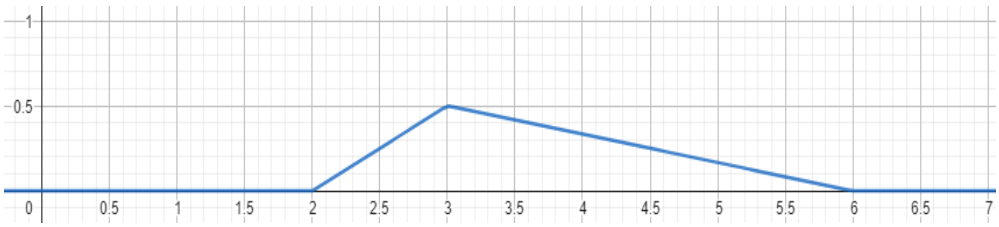
De igual forma podemos obtener una serie de fórmulas generales para el cálculo del generador de número aleatorios de forma que encontramos: la forma general para la distribución triangular y función de probabilidad acumulada respectivamente.

$$f(x)=\begin{cases}\frac{2(x-a)}{(b-a)(c-a)} & a\le x<b\\ \frac{2(c-x)}{(c-b)(c-a)} & b\le x\le c\\ 0 & otherwise\end{cases}\qquad F(x)=\begin{cases}\frac{(x-a)^2}{(b-a)(c-a)} & a\le x<b\\ \frac{b-a}{c-a}+\frac{(x-b)^2}{(c-b)(c-a)} & b\le x\le c\end{cases}$$

Y de igual manera tenemos una estructura generalizada para la creación del generador de números aleatorios usando la transformada inversa.

$$X = \begin{cases} a + \sqrt{(b-a)(c-a)R} & 0 \le R < \dfrac{b-a}{c-a} \\ b + \sqrt{[(c-a)R - (b-a)](c-b)} & \dfrac{b-a}{c-a} \le R \le 1 \end{cases}$$

Para finalizar y calcular 10 valores por medio de una hoja de cálculo, primeramente vamos a ver su comportamiento gráfico.



Con esto podemos obtener los siguientes valores:

1	Distribución Triangular				
2				Pivote	
3	Valor minimo	2	c-a/b-a		0,25
4	Valor promedio	3			
5	Valor maximo	6			
6					
7	Numero Aleatorio	Numero Distribución Triangular			
8	0,37043519	4,413097596			
9	0,282400746	4,305775394			
10	0,642093267	4,803493865			
11	0,639218908	4,798698885			
12	0,901752523	5,373110927			
13	0,562295024	4,676814487			
14	0,971266502	5,660980837			
15	0,459838937	4,530086991			
16	0,589745676	4,718978026			
17	0,707713783	4,918729975			

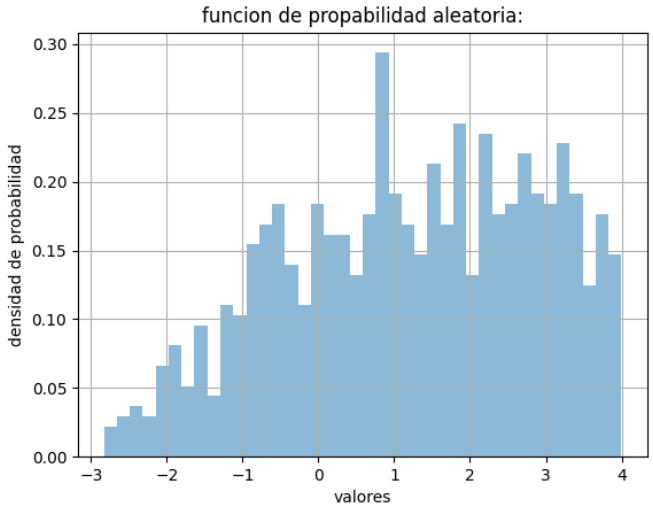
6. Desarrolle un generador para una distribución triangular con rango (1,10) y una media de 4.
7. Dada la siguiente cdf para una variable aleatoria continua con rango desde – 3 a 4, desarrolle un generador para la variable:

$$F(x) = \begin{cases} 0, & x \le -3 \\ \dfrac{1}{2} + \dfrac{x}{6}, & -3 < x \le 0 \\ \dfrac{1}{2} + \dfrac{x^2}{32}, & 0 < x \le 4 \\ 1, & x > 4 \end{cases}$$

Utilizando el método del rechazo explicado anteriormente en el numeral 3 e implementando dentro del siguiente código:

```python
import random
import math
import numpy as np
import matplotlib.pyplot as plt

def aleatorios1(rangoMax):
    randomGenerados = []
    a = -3
    b=0
    c=4
    M=1/2
    for _ in range(rangoMax):
        R1 = random.random()
        R2 = random.random()
        U = a+(c-a)*R1
        if U < b:
            y = (1/2)+(U/6)
        else:
            y= (1/2)+((U*U)/32)
        if R2 <= y / M:
            randomGenerados.append(U)

    return randomGenerados
x = aleatorios1(1000)
print(x)
plt.hist(x,bins=40,density=True,alpha=0.5)
plt.xlabel('valores')
plt.ylabel('densidad de probabilidad')
plt.title('funcion de propabilidad aleatoria:')
plt.grid(True)
plt.show()
```

Se genera la siguiente gráfica de densidad:



8. Dada la cdf $F(x) = \frac{x^4}{16}$ en $0 \leq x \leq 2$, desarrollar un generador para esta distribución.
9. Dada la pdf $F(x) = \frac{x^2}{9}$ en $0 \leq x \leq 3$, desarrolle un generador para esta distribución.

Solución punto 8 y 9

En el desarrollo de los puntos 8 y 9 se tomó el mismo marco teórico debido a que se basan en la misma teoría, en el punto 9 se agrega un paso adicional, es decir para el punto 8 se omite realizar la integración de la PDF para convertirla en una CDF, solamente se deriva la CDF para obtener la PDF y poder comprar con los números aleatorios generados.

1. **Marco teórico del método de la transformada inversa para generar números aleatorios**

   El método de la transformada inversa es una técnica general para generar números aleatorios que siguen una distribución de probabilidad específica. Se basa en la relación entre la función de distribución de probabilidad (PDF) y la función de distribución acumulada (CDF) de una variable aleatoria.

   1. Definiciones:

   - Variable aleatoria: Una variable que puede tomar diferentes valores con diferentes probabilidades.
   - Función de densidad de probabilidad (PDF): La función que describe la probabilidad de que una variable aleatoria tome un valor específico.
   - Función de distribución acumulada (CDF): La probabilidad de que una variable aleatoria sea menor o igual que un valor específico.

   2. Relación entre PDF y CDF:

   La CDF se puede obtener a partir de la PDF mediante la siguiente integral:

   $$F(X) = \int_{-\infty}^{x} f(t)dt$$

   donde:

   - F(x) es la CDF
   - f(x) es la PDF
   - x es el valor de la variable aleatoria

   3. Generación de números aleatorios:

   El método de la transformada inversa se basa en la siguiente idea:

- Si U es una variable aleatoria uniforme en el intervalo [0, 1], entonces F(U) tendrá la misma distribución que la variable aleatoria X con PDF f(x).

En otras palabras, para generar un número aleatorio con la distribución f(x), podemos seguir estos pasos:

1. Generar un número aleatorio uniforme U en el intervalo [0, 1].
2. Calcular F(U).
3. El valor F(U) es un número aleatorio con la distribución f(x).

4. Ejemplo:

Supongamos que queremos generar números aleatorios con la distribución exponencial con tasa λ. La PDF de esta distribución es:

$$f(x) = \lambda e^{-\lambda x}$$

La CDF de esta distribución es:

$$F(x) = 1 - e^{-\lambda x}$$

Para generar un número aleatorio con esta distribución, podemos seguir estos pasos:

1. Generar un número aleatorio uniforme U en el intervalo [0, 1].
2. Calcular $F(U) = 1 - e^{-\lambda U}$
3. El valor F(U) es un número aleatorio con la distribución exponencial con tasa λ.

5. Ventajas y desventajas:

Ventajas:

- El método de la transformada inversa es un método general que se puede aplicar a cualquier distribución de probabilidad continua.
- Es un método relativamente sencillo de implementar.

Desventajas:

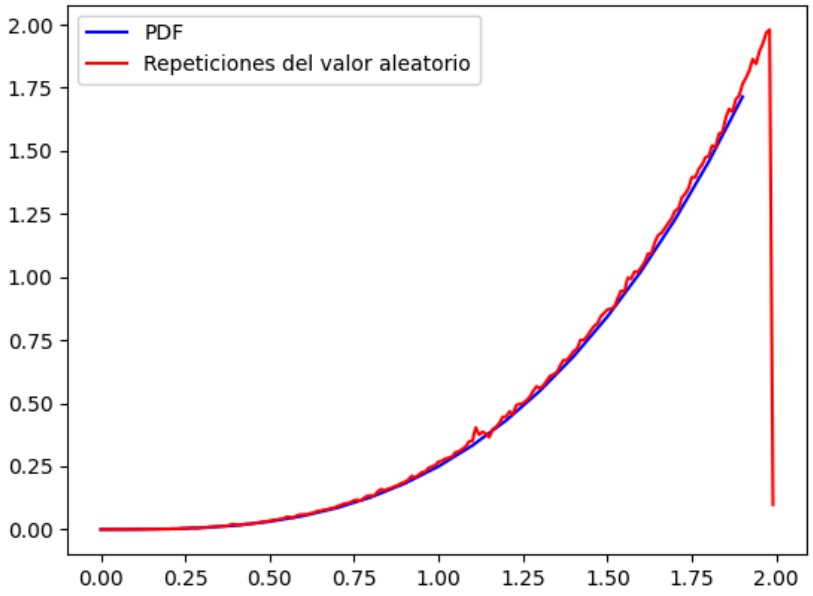
- En algunos casos, la CDF puede no tener una expresión analítica simple para su inversa.
- El método puede ser computacionalmente costoso para algunas distribuciones de probabilidad.

2. Algoritmo para el caso planteado

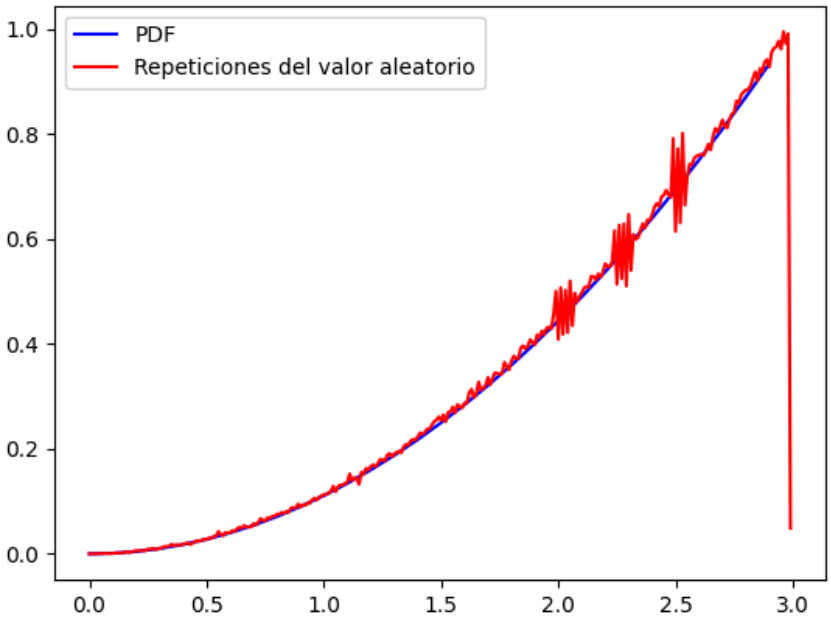
En este caso el algoritmo fue desarrollado una IDE online llamado Replit, esto permitiendo mantener el mismo entorno de ejecución del programa independiente de la computadora usada para este fin, para acceder se puede hacer por medio de este enlace.

3. Gráfica de PDF con números aleatorios generados

- **Punto 8**



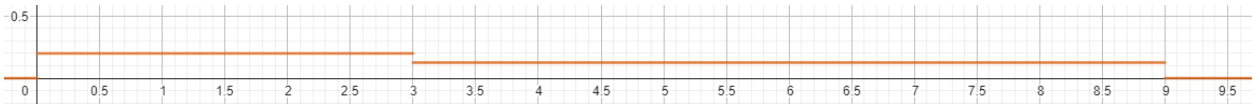
- **Punto 9**



$$F(x) = \begin{cases} \frac{1}{5}, & 0 < x \leq 3 \\ \frac{1}{8}, & 3 < x \leq 9 \\ 0, & \text{De lo contratio} \end{cases}$$

10. Desarrolle el generador para la variable aleatoria con pdf dado por:

    Primeramente identificamos esta función de distribución uniforme, por lo cual ya sabemos qué método debemos usar para la generación de números aleatorios.



```
33  def uniform_inv_cdf(x):
34      return x
35
36  n_samples = 1000
37  samples = inverse_transform_sampling(n_samples, uniform_inv_cdf)
38
39  print(samples)
```

Problemas asignados a cada uno de los grupos.

1. Una cadena hotelera tiene dos buses para recoger y dejar personas en un aeropuerto local y dos hoteles separados. Los buses viajan desde el aeropuerto al hotel 1, luego al hotel 2, y regresan al aeropuerto para continuar con este patrón. El tiempo de viaje entre cada lugar sigue una distribución normal con una media de 20 y una desviación estándar de 2 minutos. El tiempo de llegada de los viajeros desde sus vuelos se distribuye exponencialmente con una media de 2.5 minutos. Cincuenta por ciento de las personas se bajan en el primer hotel, y el bus recoge personas de este hotel que desean ir al aeropuerto. El otro cincuenta por ciento de las personas se baja en el segundo hotel, y el bus recoge nuevamente personas. En el aeropuerto, todo el mundo se baja. En ambos hoteles las personas llegan al paradero del bus para ir al aeropuerto con tiempos entre llegadas exponenciales con media de 5 minutos. Simular el sistema donde el primer bus sale del aeropuerto al iniciar la simulación y el segundo sale del aeropuerto 30 minutos después del primero. Determine la cantidad de asientos requeridos en ambos buses tal que cualquier persona esperando pueda ser recogida.

   Solución:

   Definir:

   a. **Parámetros de entrada**
      - sim_clock: Tiempo actual de la simulación.
      - end_time: Tiempo a cumplirse para dar por terminada la simulación.
      - número de veces que se ejecuta la simulación.
   b. **Variables del modelado**
      - event_type: tipo de evento (12 en total).
      - num_events: cantidad de eventos posibles dentro de la simulación.
   c. **Descripción del evento y tipo de evento**

Evento	Descripción	Tipo

arrivalViajero, arrivalPasajeroH1, arrivalPasajeroH2	Evento en el que un pasajero o viajero llega a su respectiva estación de espera de autobús y aumenta en uno la fila.	1
llegadaH1busA, llegadaH2busA, llegadaH1busB, llegadaH2busB, salidaAeroPbusA, salidaAeroPbusB, salidaAeroPbusA, salidaAeroPbusB	Salida desde una de las estaciones hasta la siguiente, se realizan las respectivas descargas y abordaje de pasajeros y se evalúa la capacidad máxima requerida.	2
sim_end_event	Fin de la simulación, cuando se cumple el tiempo de simulación indicado.	3

d. Listas y sus atributos

- resultados: Lista con el registro de todas las capacidades maximas requeridas en las N simulaciones realizadas.
- time_next_event: Lista con el tiempo en el que ocurrirá el próximo evento de cada uno de los posibles eventos a ocurrir.

e. Contadores y/o acumuladores

- colaH1
- colaH2
- colaAirport

Todas hacen referencia a la cola de su respectivo lugar.

f. Medidas de desempeño

- capacidadMáxima: Cantidad requerida para mover a todos los pasajeros en cada viaje.
- máximo: capacidad máxima requerida mas alta registrada.

g. Subprogramas y propósito

- main(): Se inicia una simulación, se programan los eventos iniciales y se define cuando se dará por terminada la simulación, se mantiene un bucle

que ejecuta el próximo evento mientras no se haya cumplido el tiempo de ejecucion , retorna la capacidad máxima que fue empleada en esta simulación.

- timing(): Recorre la lista de time_next_event para determinar el próximo evento a ejecutar.
- máximo(): Compara la actual capacidad máxima obtenida y determina si es más alta que la actualmente almacenada, de darse el caso se guarda como la nueva capacidad máxima mas alta registrada.

Diagramas de flujo:

Programa Principal:

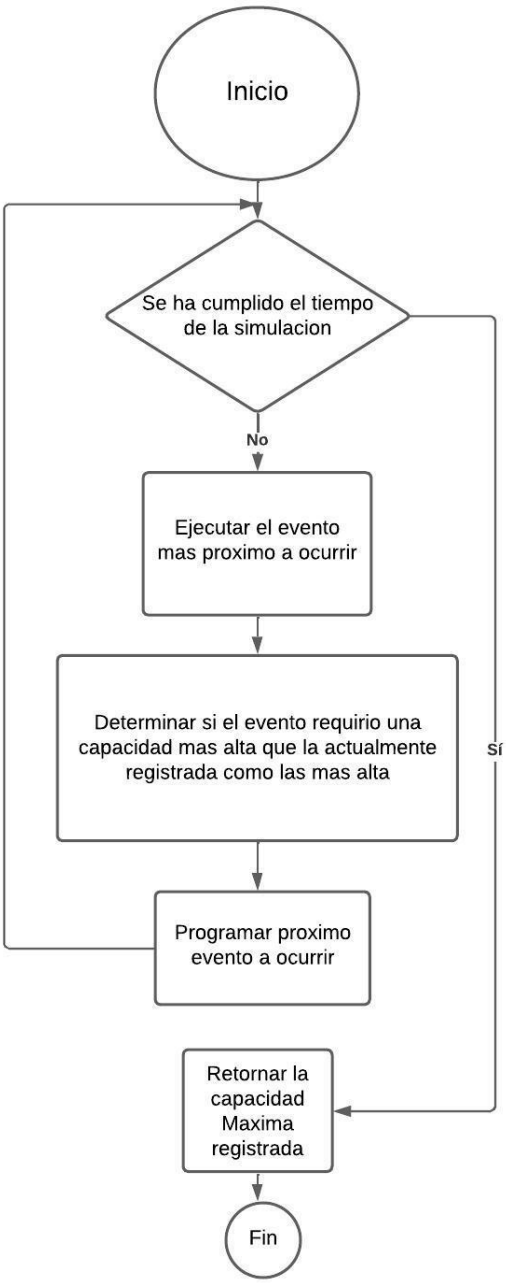
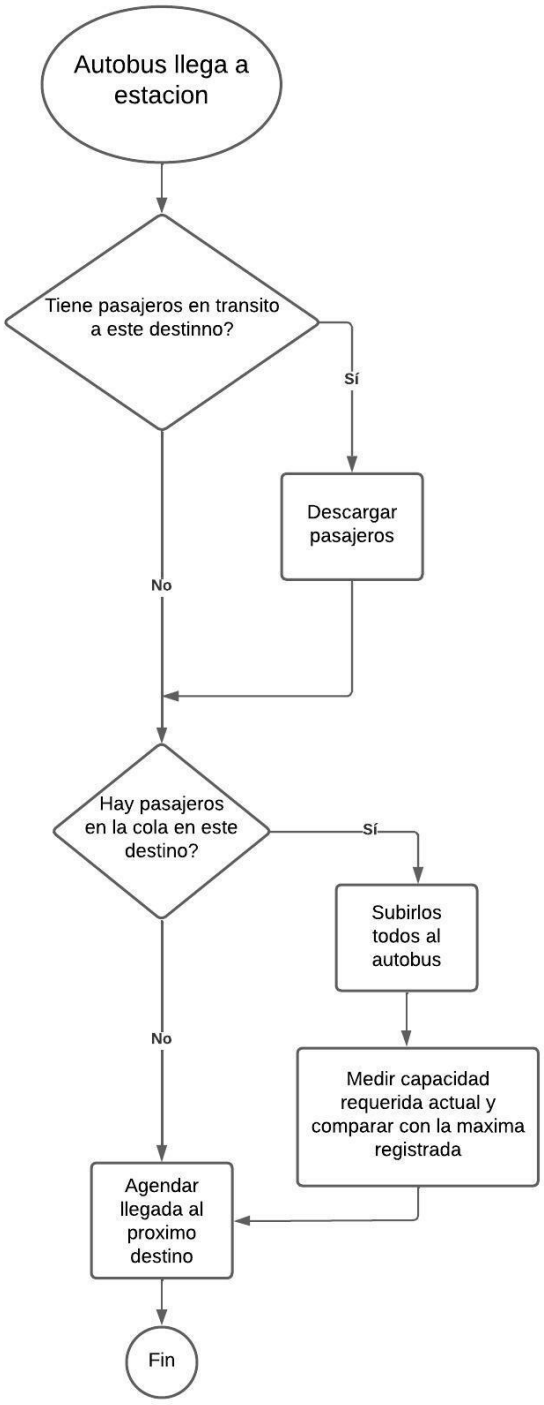


Diagrama generalizado para la llegada de un autobús a su estación:



Análisis de resultados:

Después de ejecutar la simulación 1000 veces con un tiempo de equivalente a 24 horas en cada una, el valor de capacidad máxima más alto que se obtuvo fue 51, por lo tanto la cantidad de asientos requeridos en cada buses de modo que siempre se pueda recoger a todos los pasajeros será de 51 dentro de estos parámetros.

Planteamiento de alguna mejora:

Dado que la diferencia de tiempos de salida de ambos autobuses es algo corta, generalmente el autobús que más capacidad requiere será siempre el autobús A dado que por ejemplo al volver a iniciar la ruta desde el aeropuerto este recoge a todos los pasajeros que hayan llegado a la fila por lo que en este caso particular sería óptimo subir a una menor cantidad de pasajeros (la mitad de la fila por ejemplo) en vez de a todos los pasajeros ya que al momento de llegar el bus B este solo tendrá que subir los pasajeros que hayan llegado dentro de los 30 minutos de margen de diferencia más el tiempo variable que se dio a lo largo del ciclo.

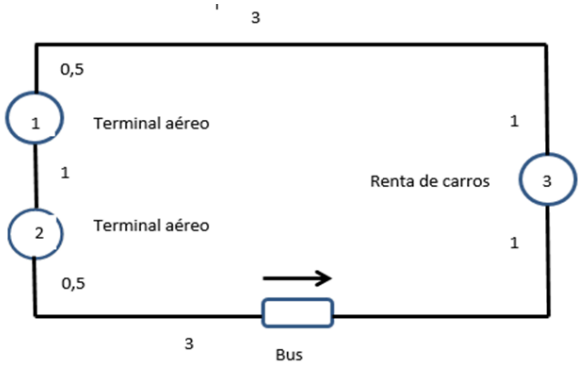
2. Considere un sistema de renta de carros como se presenta en la figura, con todas las distancias en millas. Las personas llegan a la localización i (donde i=1, 2, 3) con tiempos entre llegadas exponenciales independientes a las tasas respectivas de 14,10 y 24 por hora. Cada lugar tiene una cola FIFO con capacidad ilimitada. Hay un bus con capacidad de 20 personas y una velocidad de 30 millas/hora. El bus inicialmente está en la localización 3 (renta de carros), y sale inmediatamente en dirección contraria a las agujas del reloj. Todas las personas que llegan a un terminal desean ir a la renta de carros. Todas las personas en el lugar de renta de carros desean ir a los terminales 1 y 2 con probabilidades respectivas de 0.583 y 0.417. Cuando un bus arriba a un lugar, se aplica las siguientes reglas:
    a. Primero se descarga a las personas en una forma FIFO. El tiempo de descarga de una persona se distribuye uniformemente entre 16 y 24 segundos.
    b. Luego las personas se cargan al bus hasta su capacidad, con un tiempo de carga por persona que se distribuye uniformemente entre 15 y 25 segundos.
    c. El bus siempre permanece 5 minutos en cada lugar. Si ninguna carga o descarga está en proceso después de cinco minutos, el bus saldrá inmediatamente.

    Correr el simulador por 80 horas y obtener estadísticas sobre:

    a. Número promedio y máximo en cada cola.
    b. Demora promedio y máxima en cada cola.
    c. Número promedio y máximo en cada bus.
    d. Tiempo promedio, máximo y mínimo que el bus está detenido en cada lugar.
    e. Tiempo promedio, máximo y mínimo para que el bus realice un ciclo (desde la partida de la renta de carros a la siguiente de tal partida).

f. Tiempo promedio, máximo y mínimo que una persona está en el sistema por lugar de arribo.

Use los siguientes stream de números aleatorios: i, tiempo entre llegadas al lugar i (donde



i= 1, 2,3); 4, tiempos de descarga; 5, tiempos de carga; 6, determinar el lugar de destino de una llegada en la renta de carros.

Solución
SIMULACIÓN (Enlace al código)

1. **Definiciones**

   a) **Parámetros de entrada:**

   Para la simulación se requieren los siguiente parámetros de entrada:

   - Tiempo de simulación
   - Tasa de llegada de personas
   - Capacidad bus
   - Velocidad bus
   - Ubicación bus
   - Distancia en millas (Ordenado de izquierda a derecha en sentido contrario de las manecillas del reloj empezando por la ubicación inicial del bus)
   - Probabilidades de terminales destino
   - Tiempo de descarga del bus
   - Tiempo de carga del bus
   - Tiempo de permanencia del bus

   b) **Variables de modelado:**

   Para la simulación se usan las siguientes variables de modelado:

- Lista de de colas de los tres lugares
- Lista de eventos
- Lista con una lista con el número de personas en la cola en cada cambio de cantidad en las tres colas
- Lista de retrasos por persona en cada fila
- Cantidad máxima de personas en cada cola
- Retraso máximo en cada cola
- Tiempo minimo y maximo de espera del bus
- Lista de tiempos de cada una de las vueltas completadas por el bus
- Tiempo máximo de una vuelta del bus
- Tiempo mínimo de una vuelta del bus
- Lista de tiempo de estancia de cada persona en cada uno de los sitios de arribo

c) Descripción de evento y tipo de evento:

En el manejo de esta simulación se construyeron 7 eventos

1. **Evento tipo 1 Llegada de un pasajero al sitio 1**
   En este evento se describen y programan los eventos posteriores a la llegada de un pasajero a la terminal número 1.

2. **Evento tipo 2 Llegada de un pasajero al sitio 2**
   En este evento se describen y programan los eventos posteriores a la llegada de un pasajero a la terminal número 2.

3. **Evento tipo 3 Llegada de un pasajero al sitio 3**
   En este evento se describen y programan los eventos posteriores a la llegada de un pasajero a la terminal número 1.

4. **Evento tipo 4 Llegada del autobús a un sitio**
   Este evento describe lo que ocurre cuando llega una autobús a cada uno de sus sitios, programando así la bajada o subida del primer pasajero en ese sitio y también se programa la salida preliminar el bus de ese sitio.

5. **Evento tipo 5 Bajada de un pasajero del autobús**
   En este evento se programan las bajadas de los pasajeros con la demora de cada uno, donde se verifica que pasajeros se van a bajar en la estación en la que está detenido el autobús.

6. **Evento tipo 6 Subida de un pasajero al autobús**
   Este evento describe la subida de un pasajero y su demora al autobús, donde se verifica que la cola del sitio donde está detenido el bus no esté vacía y se suban las personas, para programar la siguiente subida en caso de la cola no quedar vacía.

7. **Evento tipo 7 Salida del autobús de un sitio**
   Este evento describe la salida de un autobús de alguno de los sitios, esto después de verificar que no hay nadie en la cola para subirse y que ha transcurrido el tiempo mínimo de permanencia en la estación del autobús.

d) Listas y sus atributos:

- **Lista de eventos:** Esta lista determina el tiempo que falta para que ocurra cada uno de los eventos descritos en el punto anterior.
- **Ocupación del bus:** Esta lista guarda la cantidad de pasajeros del bus cada vez que esta es modificada, cuando se sube o baja un pasajero.
- **Longitud de cada cola:** Esta lista guarda la longitud de cada una de las colas en tiempo real.
- **Pasajeros del sitio 3:** Esta lista indica cuántos pasajeros que están en el bus van hacia la terminal 1 y 2 respectivamente.

e) Contadores y/o acumuladores:

- **Demoras del bus:** Esta lista cuenta de cada una de las demoras del bus en cada una de sus estaciones.
- **Demora en las colas:** Esta lista cuenta cada una de las demoras de los pasajeros en cada uno de los sitios.
- Histórico de longitud de las colas
- **Longitudes de las colas:** Esta lista guarda las longitudes de las colas en cada instante que se modifica, es decir con la salida y llegada de un pasajero.
- **Demora máxima en las colas:** Esta variable cuenta la demora máxima en cada una de las colas
- **Demora del bus:** Esta variable acumula una lista de demoras del bus en cada uno de los sitios donde llega
- **Duración del ciclo:** Esta lista acumula la demora del bus en cada uno de los ciclos que hace.
- **Duración de cada pasajero:** Esta variable cuenta el tiempo que estuvo cada uno de los pasajeros desde que llegó a alguno de los sitios hasta que se bajó del bus.
- Cantidad de pasajeros demorados: Esta variable acumula la cantidad de pasajeros que han tenido demora así sea 0 en cada uno de los sitios.

f) Medidas de desempeño:

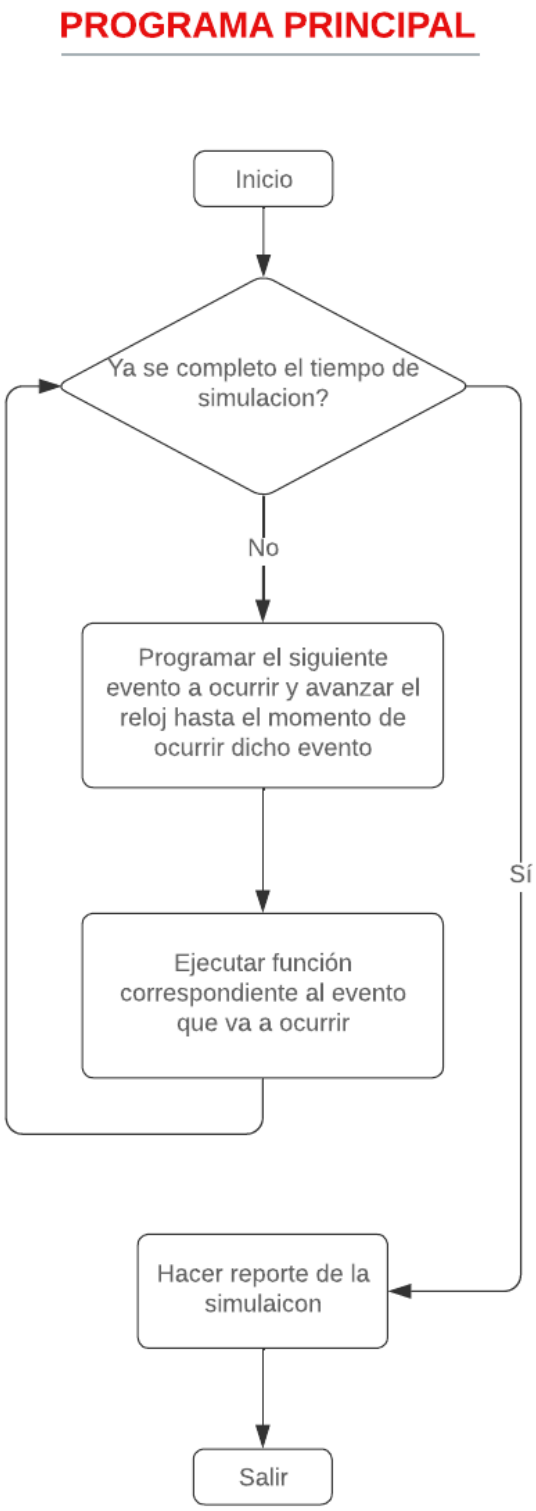
- **Demora promedio y máxima en cada cola:** Se usa para saber que tan eficiente en cuestiones de tiempo es el llegar a una de los sitios y dirigirse a otro.
- **Demora promedio, máxima y mínima del bus en cada una de las estaciones:** Esto mide la cantidad de tiempo que gasta el bus en las estaciones, puede usarse para saber que tanto demora el bus quieto en cada estación.
- **Demora en dar un ciclo entero:** Esta mide qué tan eficiente es el sistema basándose en las distancias.
- **Número de personas en cada cola:** Esta medida se toma para saber qué tantas personas se acumulan en las filas de cada sitio.
- **Número de personas en el bus:** Esta medida informa que tan lleno suele ir el bus o que tan vacío.

g) Subprogramas y propósito:

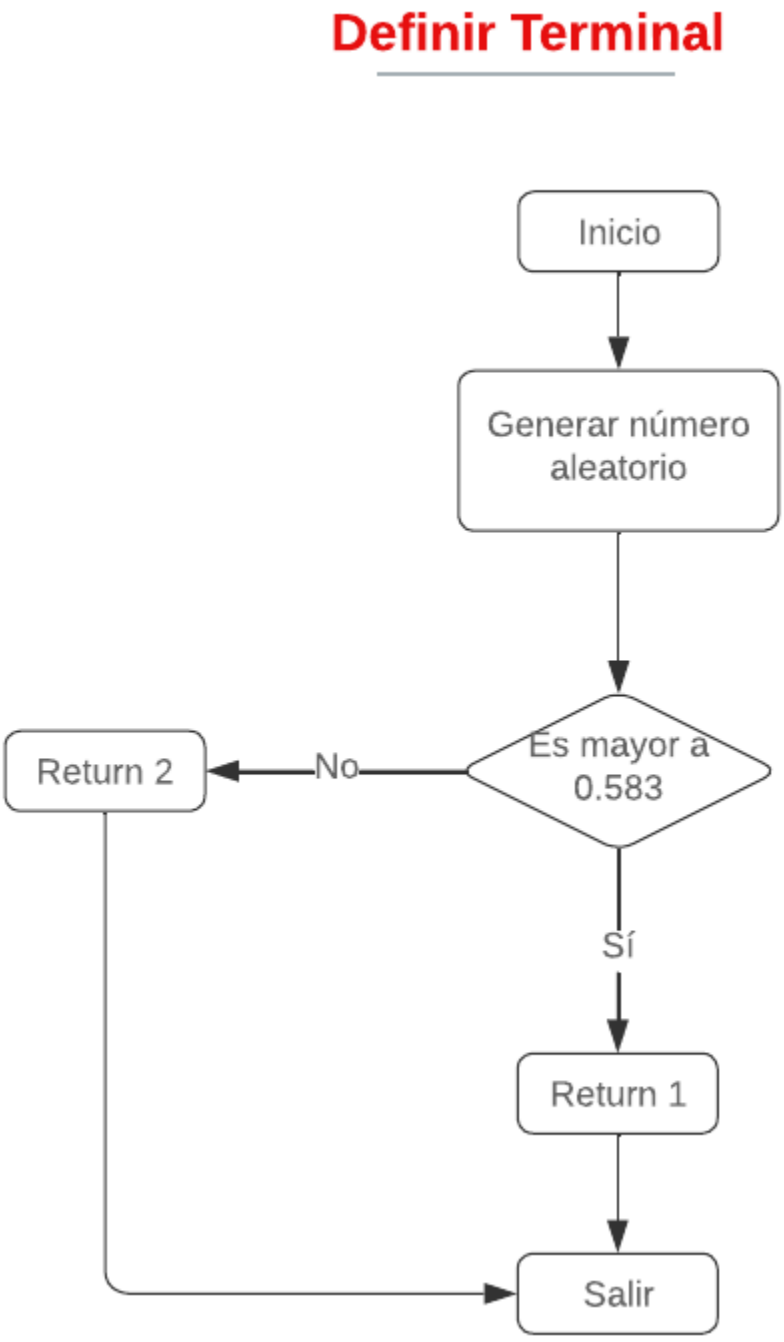
- **Establecer terminal:** Esta función establece a qué terminal va cada uno de los pasajeros que se subió en el sitio de renta de vehículos.
- **Avance de reloj:** Se encarga de avanzar el reloj de la simulación para todos los eventos en general, lo avanza de acuerdo al tiempo del siguiente evento.
- **Timing:** Actualiza la variable que indica cuál será el tipo del siguiente evento.
- **Obtener siguiente evento:** Devuelve el número del evento que debe ocurrir de acuerdo al que siga en orden cronológico.
- **Iniciar simulación:** Inicia la simulación con los parámetros iniciales y ejecutando el evento que se le indique.
- **Reporte:** Toma las variables estadísticas y acumuladores para calcular y presentar los resultados finales de la simulación.

2. Diagramas de flujo

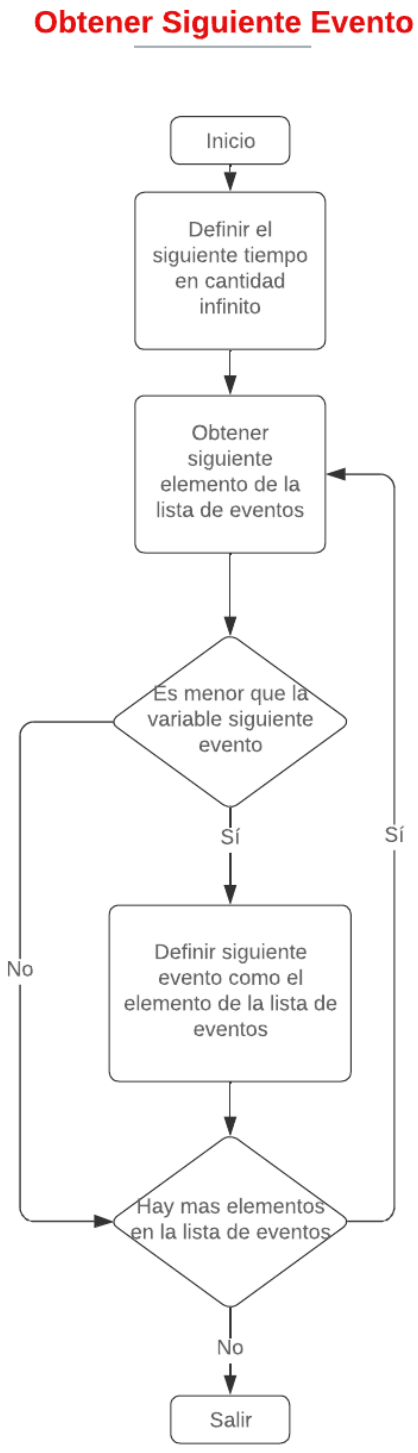
Programa principal



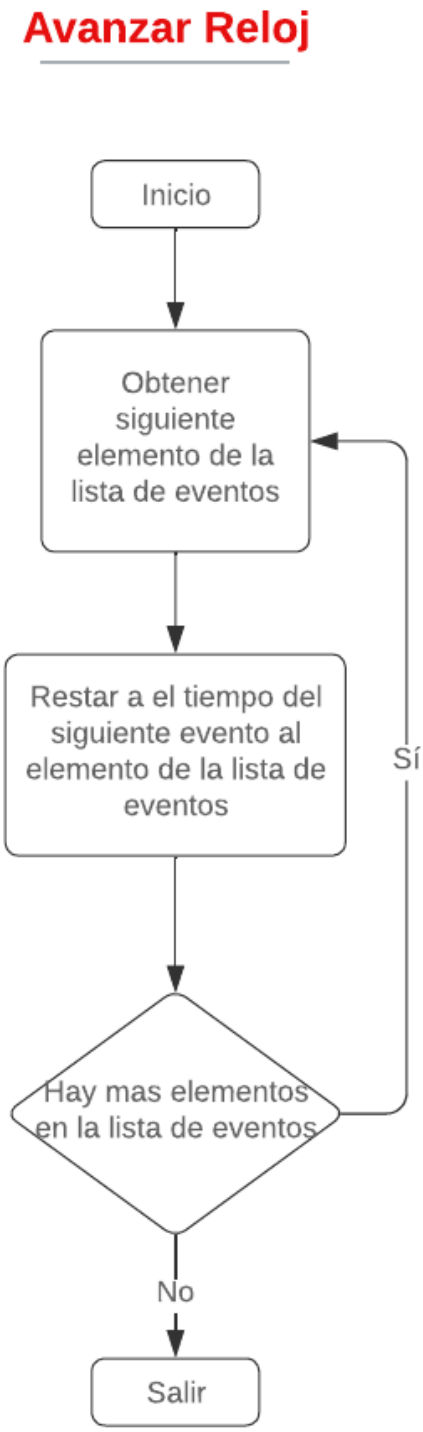
Decidir terminal



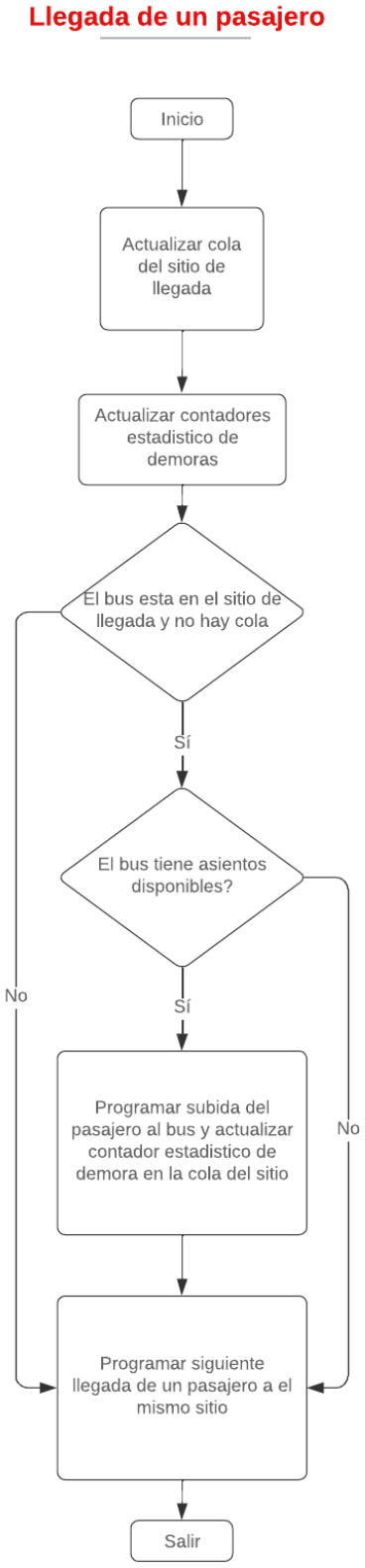
Obtener siguiente evento



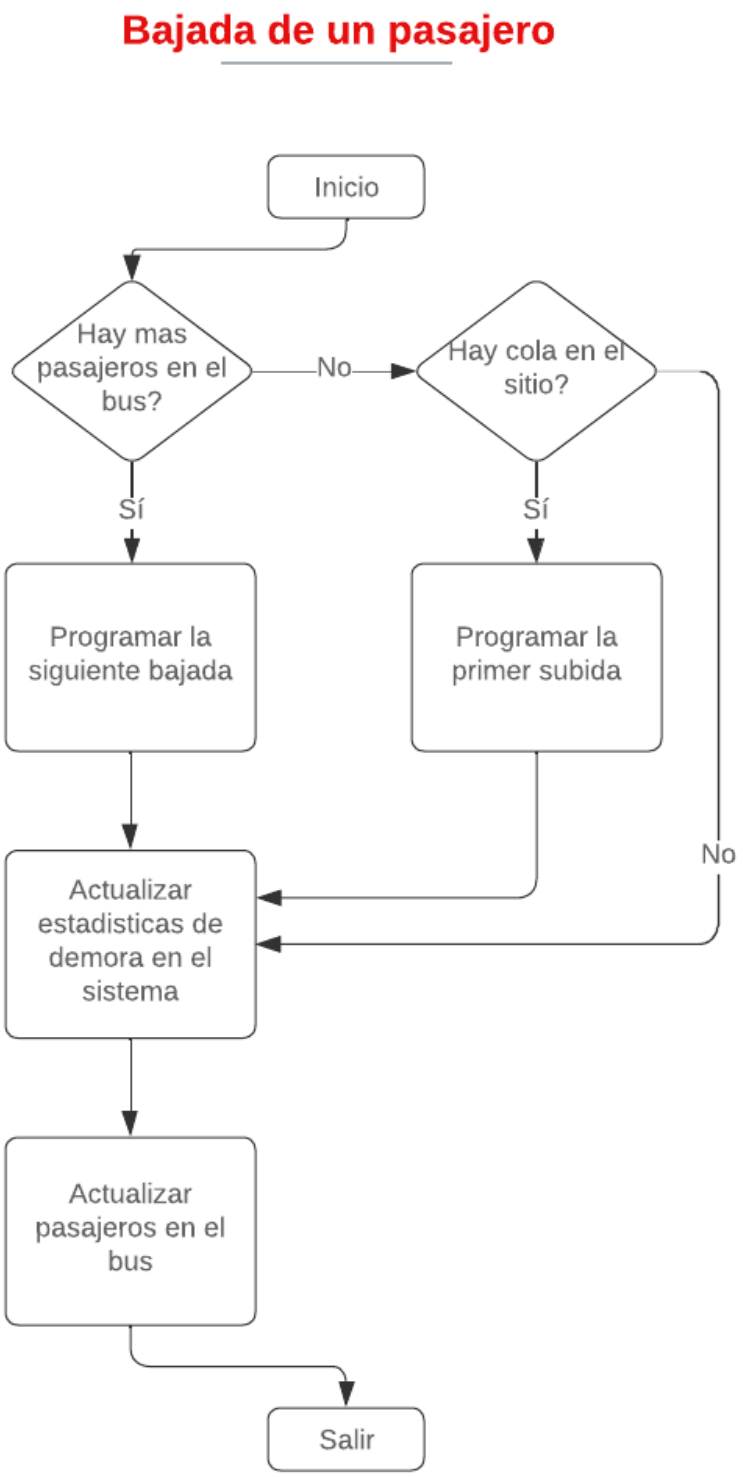
Avanzar reloj



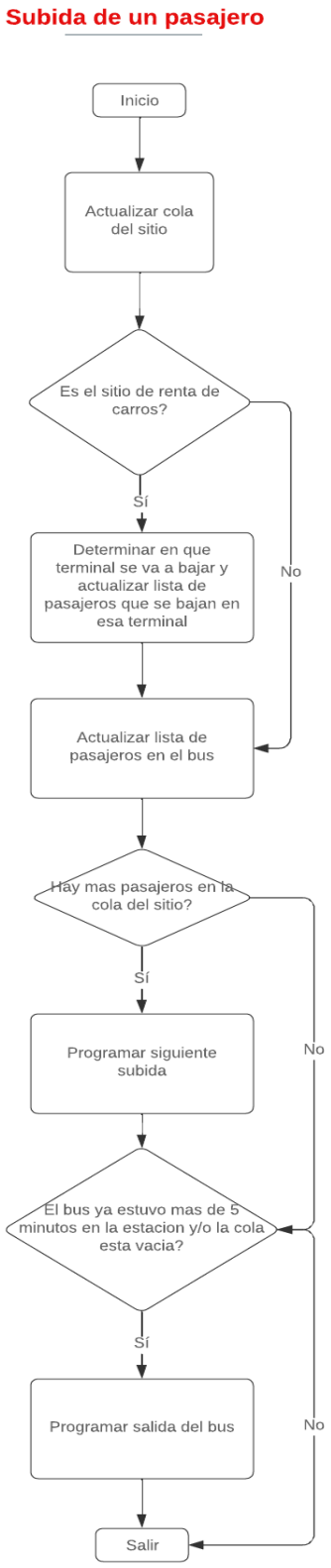
Llegada de pasajero



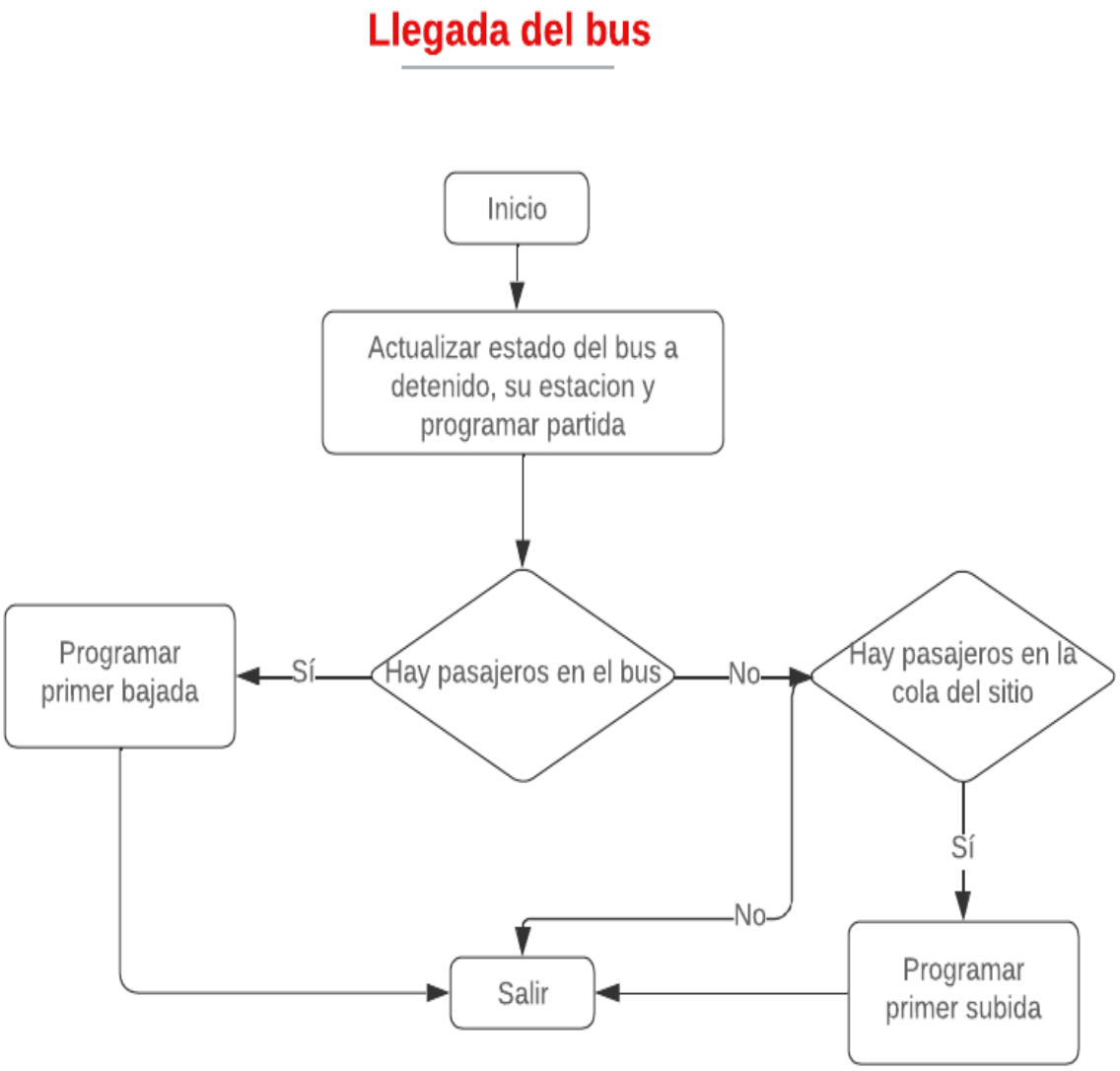
Bajada de un pasajero



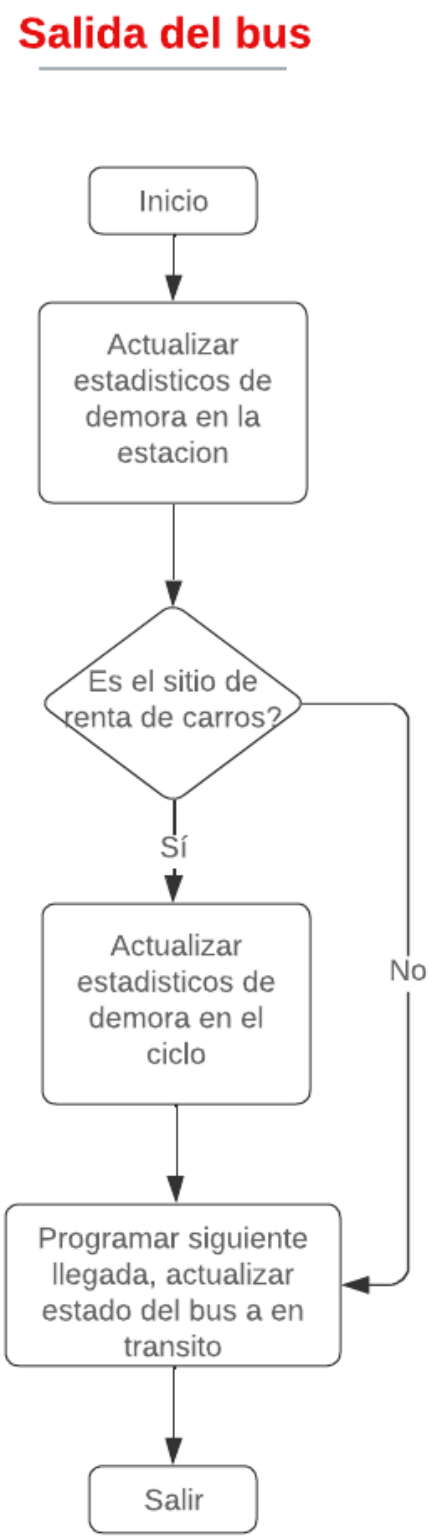
Subida de un pasajero



Llegada del bus



Salida del bus



3. Análisis de resultados

Al ejecutar la simulación se evidencia que la demora de las personas en la cola de cada estación en promedio es mucho menor que la demora máxima, lo cual implica que mayor mente las personas no demoran tanto en sus respectivas colas. La cantidad de personas refleja un comportamiento similar, el número máximo de personas está bastante alejado de la cantidad promedio de personas en cada una de las filas, lo cual indica una longitud de fila baja usualmente. Por los dos puntos anteriores el bus se queda unos minutos más que su demora mínima en cada estación en promedio, lo cual indica que la espera mínima está bastante bien ajustada para poder hacer eficiente el sistema, este aspecto se cumplen para las terminales 1 y 3, pero para el sitio de renta de vehículos el bus suele estar detenido más cerca al tiempo máximo que al tiempo mínimo, esto se ve debido a que el número de pasajeros promedio es mayor en la cola de este sitio que en la cola de los otros dos sitios, y debido a que los pasajeros de ambas terminales se bajan aquí toma más tiempo, ya que todos los pasajeros de este sitio se dividen en ambas terminales lo cual disminuye el número de pasajeros que bajan y por ende hace más eficiente el tiempo de demora del bus en las terminales.

En contraste con lo anotado anteriormente se puede observar que el tiempo que demora una persona en el sistema que llega al sitio de renta es menor al de una persona que llega a cualquiera de las otras terminales, siguiendo el hilo de lo analizado anteriormente se puede notar que cuando se demora mas el bus en la estación de renta esta espera se suma a las personas de las terminales, luego estas mismas deben esperar lo que tarde el bus en ambas terminales para finalmente dirigirse al sitio de renta.

Haciendo un balance entre estos aspectos se puede decir que el tiempo de demora es similar entre las estaciones, como se evidencia en la simulación, aunque ligeramente mayor para las personas que llegan a alguna de las terminales, con énfasis en la terminal 1.

Con respecto al bus se puede ver cómo en promedio no está cerca de la capacidad máxima de pasajeros, lo cual indica que suele recoger a todas las personas en cada estación y poder descongestionar las colas sin necesidad de otro bus. También por la demora en cada estación se suele acercar el tiempo de máxima demora en dar un ciclo el autobús, sin embargo con la demora mínima la diferencia no es más de diez minutos entonces no es una diferencia tan grande.

En conclusión el sistema funciona bien de la manera que está planteado, sin embargo en los momentos de máxima capacidad, demora máxima y filas con mayor número de personas sería de practicidad agregar una cuarta con menor cantidad de pasajeros por hora, para desaturar el bus en esos momentos, sin embargo sería más óptimo agregar

un segundo bus en esos momentos donde se alcanza la capacidad máxima del bus y tiempos de demora demasiado alejados del tiempo promedio.

4. **Modificación planteada**

Este sistema funciona de una manera eficiente, sin embargo los tiempo de duración en el sistema puede alcanzar las dos horas, lo cual para un sistema tan pequeño no es óptimo, para solucionar esto se pueden implementar varias soluciones, agregar una parada intermedia que tenga una menor frecuencia de llegada de pasajeros para desaturar en momentos de alta demanda, junto con un segundo bus que trabaja paralelamente con el primero, esto se puede acompañar de mejoras físicas con mayor capacidad del bus, y una mejora mecánica que permita al bus recorrer las distancias en un tiempo menor, estas mejoras pueden ser escalables usando solamente el aumento de la cantidad de buses para mover más rápidamente mayor cantidad de personas.

3. Un elevador en una planta manufacturera transporta exactamente 400 kilogramos de material. Hay tres clases de material, los cuales llegan en cajas de peso conocido. Estos materiales y sus distribuciones de tiempo entre llegadas son dados a continuación:

Material	Peso (Kilogramos)	Tiempos entre llegadas (min)
A	200	5 ± 2 UNIFORME
B	100	6 (CONSTANTE)
C	50	P(2) =0.33 P(3)=0.67

El elevador toma 1 minuto llegar al segundo piso, 2 minutos descargar, y 1 minuto regresar al primer piso. El elevador no abandona el primer piso hasta que tenga la carga completa. Simular 8 horas de operación del sistema.

Solución

1. **Definir:**
    a. **Parámetros de entrada**
        i. clock = Tiempo de duración de la simulación en minutos. (int)
        ii. arrival_time = Tiempo de llegada de los materiales. (float)
        iii. max_weight = Peso máximo dentro del elevador. (int)
        iv. num_material = Número de materiales. (int)
        v. material_weight = Arreglo con los pesos de cada material. (array int)

vi. mean_interattival = Diccionario con las distribuciones de tiempo de llegada de cada material (dict int)

b. Variables de modelado

i. material_type = Tipo de material (A = 1, B = 2, C = 3)
ii. total_arrivals = Arreglo con el total de llegadas para cada material (array int)
iii. total_departures = Arreglo con el total de salidas para cada material (array int)
iv. arrival_time = Matriz con los tiempos de llegada de los materiales (array 2d float)
v. curr_weight = Peso actual dentro del elevador (int)
vi. box_weight = Peso de la caja en un instante dentro del elevador (int)

c. Descripción del evento y tipo del evento

Evento	Descripción	Tipo
arrival_event	Llegada de un tipo de material al elevador	1
departure_event	Salida del elevador	2
sim_end_event	Fin de la simulación	3

d. Listas y sus atributos

Listas	Atributos	ID
queue_list	1. Tipo de material 2. Peso del material 3. Tiempo de llegada a la cola	1-3
server_list	1. Tipo de material que llegó 2. Peso del material que llegó 3. Tiempo de llegada de materiales	4
events_list	1. Lista de eventos de SIMPY	25

e. Contadores y/o acumuladores

i. curr_time = Tiempo actual en la simulación (float)
ii. total_arrrivals = Número total de llegadas de un material (int)
iii. total_outputs = Número total de salidas de un material (int)

f. Medidas de desempeño

i. transit_time = Promedio del tiempo de tránsito (float)

ii. delay_time = Promedio del tiempo de demora (float)

g. Subprogramas y propósito

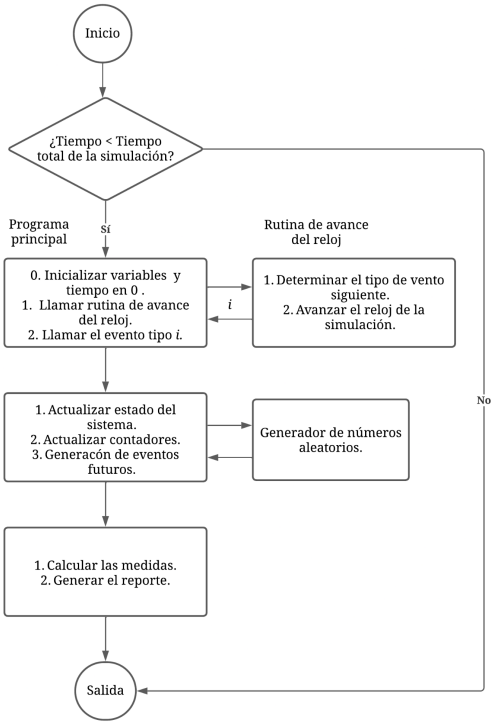
i. main() = Inicializa la simulación a partir de la lectura de los parámetros de entrada, también se programan las llegadas iniciales y se establece el fin de la simulación según lo requerido.

ii. arrival() = Programa la próxima llegada de un material aleatorio y evalúa si se puede colocar en la cola del ascensor. Si es posible, verifica que el elevador esté lleno para transportarlo hacia arriba y prepararlo para bajar, lo que sucederá después de descargar el material.

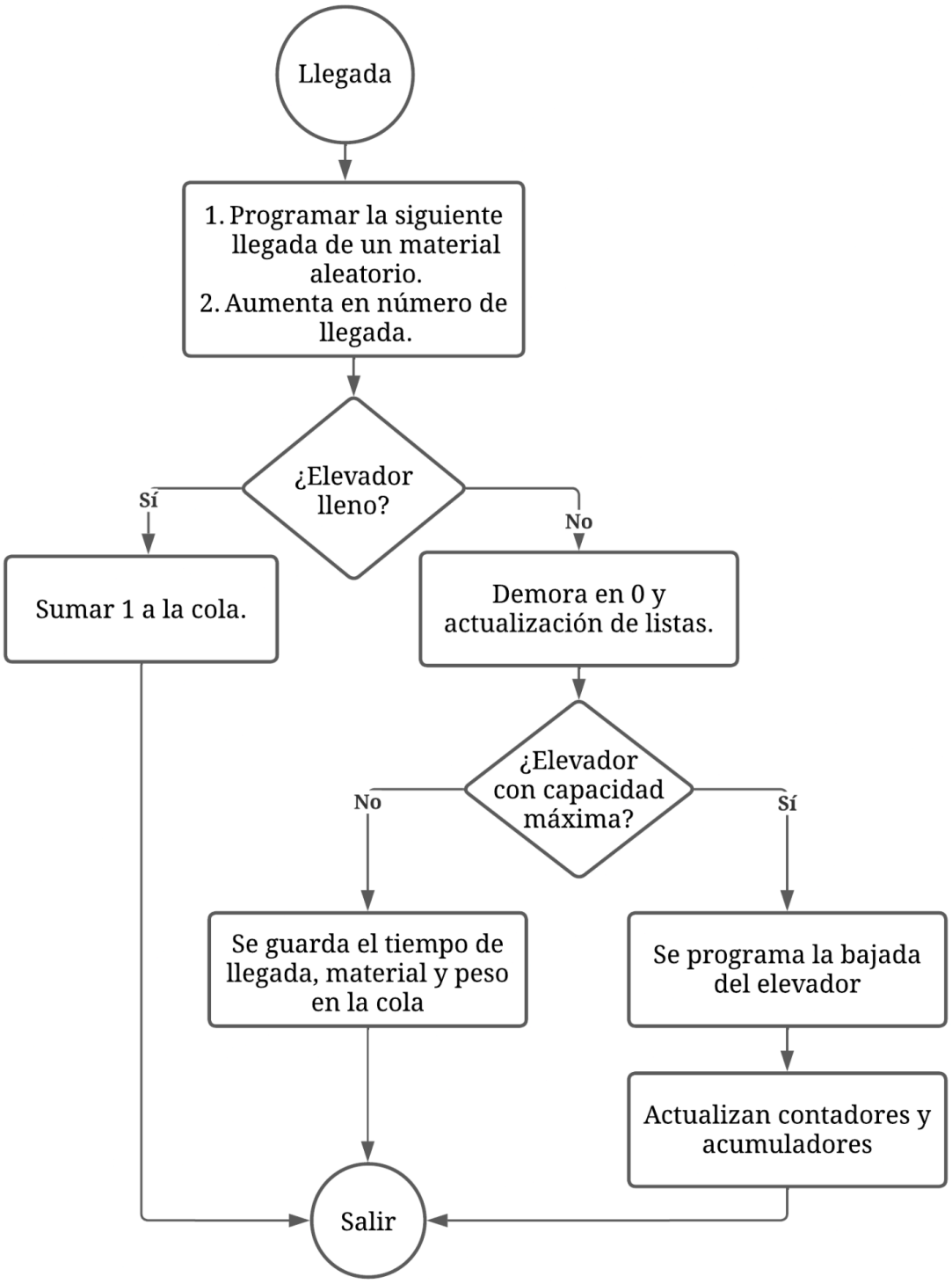
iii. output() = Simula el viaje del elevador y actualiza contadores estadísticos.

2. Diagramas de flujo del programa principal y de cada rutina

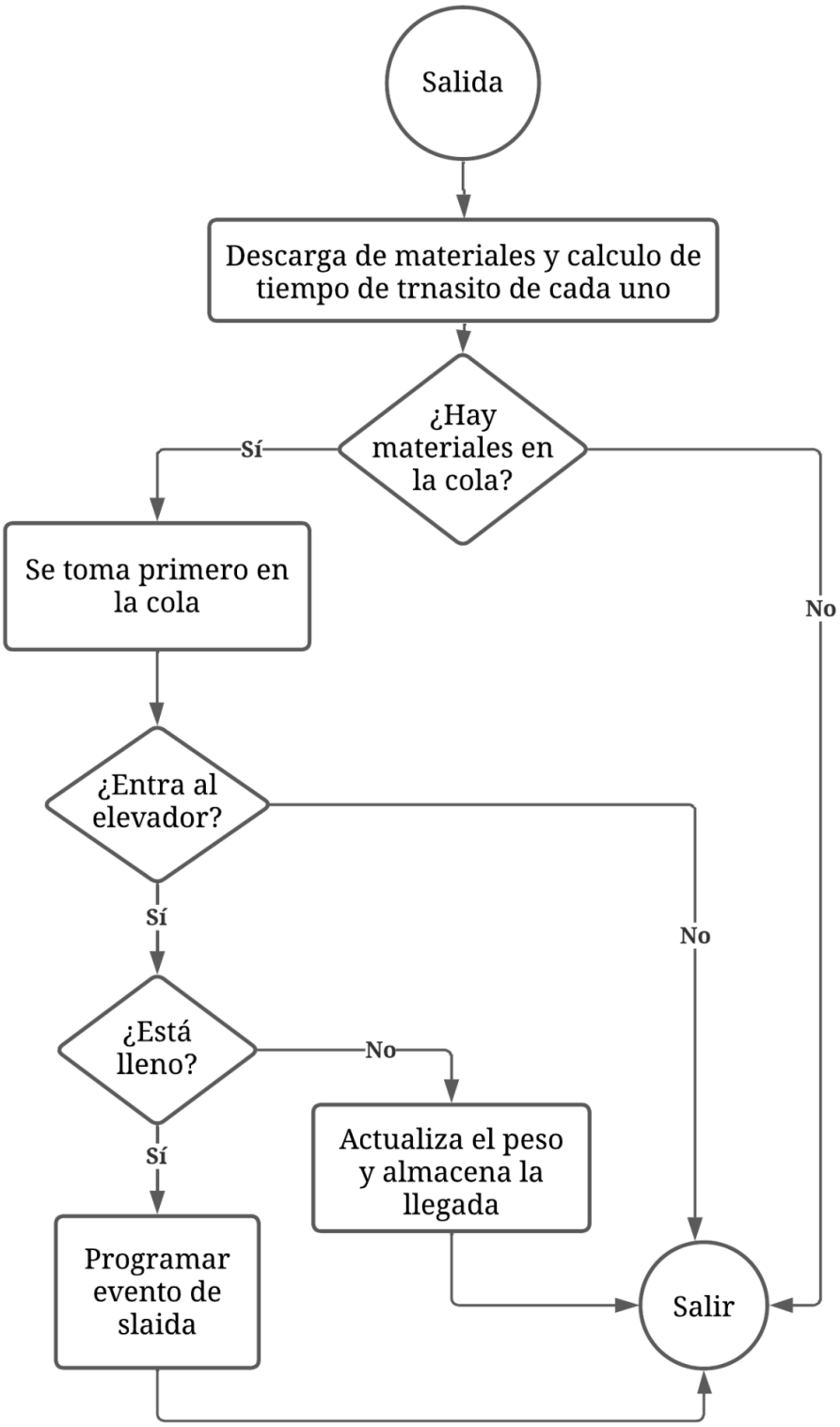
a. Diagrama del programa principal



b. Diagrama del evento de llegada



c. Diagrama del evento de salida



3. Análisis de resultados

Con base en estos resultados se pueden responder las siguientes preguntas:

Material	Total de llegadas	Total de salidas	Tiempo promedio de tránsito	Tiempo promedio de espera
1	30	30	10.175	4.961
2	33	33	7.668	3.024
3	58	57	4.695	2.599

a. ¿Cuál es el tiempo promedio de tránsito para una caja de material A (Tiempo desde su llegada hasta su descarga)?

   Se obtiene que el tiempo promedio de tránsito para una caja de material A fue de 10.175 minutos, es decir, 10 minutos y 5 segundos.

b. ¿Cuál es el tiempo promedio de espera para una caja de material B?

   El tiempo promedio de espera para una caja de material B fue de 3.024 minutos.

c. ¿Cuántas cajas de material C realizan el viaje en 1 hora?

   Al cambiar el parámetro de 480 minutos a 60 minutos, se obtienen un total de 8 cajas tipo C que realizan el viaje.

d. ¿Cómo se afectan las respuestas anteriores si las cajas de material C llegan con distribución expo (6)?

   Al cambiar la distribución con la que llegan las cajas tipo C a una expo (6), se incrementa el tiempo promedio de tránsito para el material A de 10 minutos y 5 segundos a 11 minutos y 37 segundos, además se incrementa el tiempo promedio de espera del material B de 3.024 minutos a 3.25 minutos. Finalmente, incrementó el número de cajas de material C que hicieron el viaje en 1.

4. Modificación que mejora el desempeño del sistema

Para una mejora en el desempeño del sistema se hicieron cambios en las distribuciones de tiempo entre llegadas de los materiales, además de ello también se redujeron los tiempos de descarga y transporte del elevador, estos cambios aumentaron la cantidad de cajas que realizaron el viaje, especialmente las del material C, además de reducir el tiempo espera para las cajas de los tres materiales, tal como se muestra a continuación:

Material	Total de llegadas	Total de salidas	Tiempo promedio de tránsito	Tiempo promedio de espera
1	32	32	8.957	3.549
2	36	36	6.534	2.326
3	68	68	3.498	1.337

4. Dos máquinas, A y B se encuentran a 50 m una de otra. Se cuenta con un montacargas para transportar piezas de A a B. El montacargas arranca hacia B cuando se juntan N piezas. En B se descarga y regresa vacío a A. El tiempo de recorrido está distribuido normalmente μ=3 minutos y σ =5 segundos. La máquina A produce 54 piezas/hora con una distribución Poisson. Si el tiempo para cargar y descargar las piezas es de 10 segundos. Calcule el valor de N para minimizar el tiempo de espera por pieza en A.

Solución:

a. *Parámetros de entrada:*
mean_machine_A = Cantidad de piezas que produce la máquina A en un minuto.
mean_transport = Tiempo medio de transporte.
end_sim = Tiempo para el fin de la simulación.

b. *Variables del modelamiento:*
status_forklift = Estado del montacargas.
N = Piezas que se deben juntar para transportar.

Stream	
1	Piezas
2	Transportar

Sampst número variable	Significado
1	Espera Maquina A

c. *Descripción del evento y tipo de evento.*

Evento	Tipo
Creación piezas	1
Transporte	2
Fin de la simulación	3

d. *Listas y sus atributos.*

Lista	Atributo_1	Atributo_2
Piezas	-	-
List_event	Event_time	Event_type

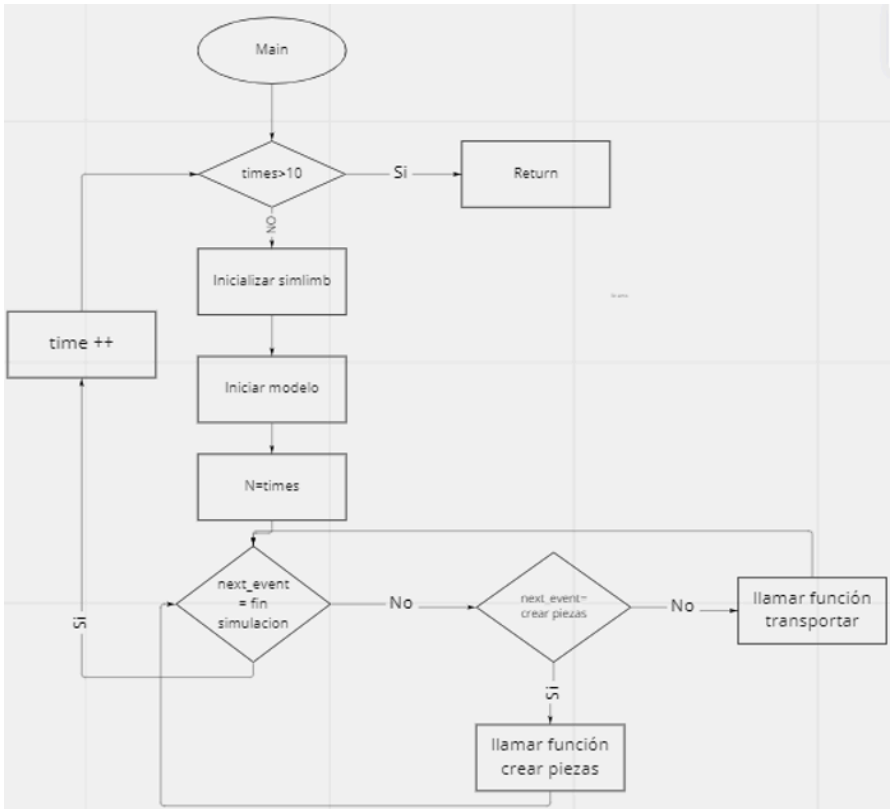
e. *Contadores y acumuladores.*
sum_parts = Cantidad de piezas producidas en A sin transportar.

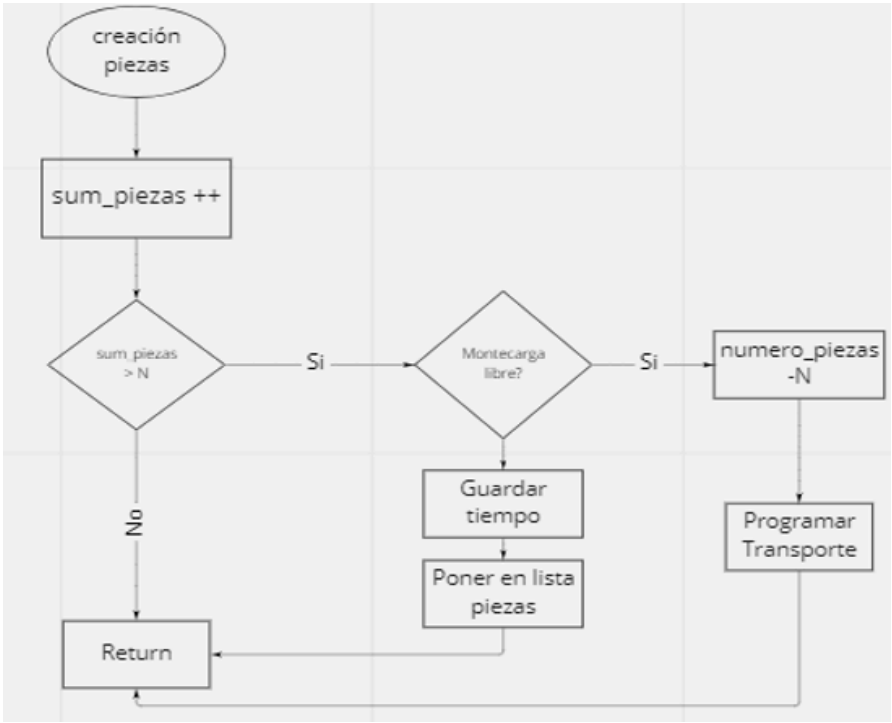
total_parts = Cantidad total de piezas.

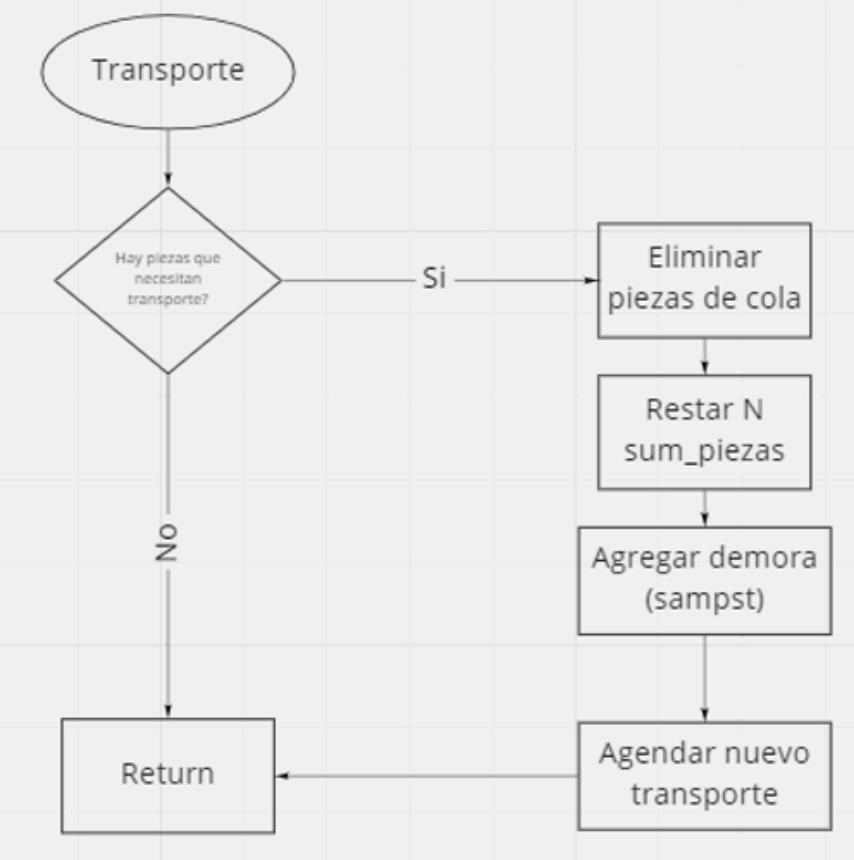
f. Medidas de desempeño.

sampst_delay = Almacenamos el tiempo se tuvo que esperar en la máquina A.

g. Diagramas







i. Resultados

```
-------------------------------------------------
REPORTE INICIAL DE LA VARIABLES QUE SE ESTAN USANDO PARA LA SIMULACIÓN
-------------------------------------------------
Tiempo limite de la simulación: 2000
Numero N de partes para transporte: 10
Tiempo promedio de producción de piezas: 0.9 minutos/piezas
-------------------------------------------------
REPORTE DE LA SIMUACIÓN
-------------------------------------------------
Retraso en el transporte: 0.030567149197850085
Viajes realizados: 223
Piezas creadas: 2248
```

```
-------------------------------------------------
REPORTE INICIAL DE LA VARIABLES QUE SE ESTAN USANDO PARA LA SIMULACIÓN
-------------------------------------------------
Tiempo limite de la simulación: 200
Numero N de partes para transporte: 10
Tiempo promedio de producción de piezas: 0.9 minutos/piezas
-------------------------------------------------
REPORTE DE LA SIMUACIÓN
-------------------------------------------------
Retraso en el transporte: 0.0
Viajes realizados: 20
Piezas creadas: 209
```