Jesús Antonio Soto Velázquez - 1570031
Luis Ariel Cano Castillo - 1445956

Manual Técnico y Operativo

Programa de Simulación del Sistema Metrorrey

Manual Operativo

Hay dos modos de operación de la simulación: la primera es una única ejecución de la simulación desde el reloj iniciando en 0 y terminando el 7200 segundos. En cada evento se hace una impresión en pantalla.
Para su uso, ejecutar:

```
$python simulacionmetro.py
```

A continuación, se le pedirá que introduzca el valor del tiempo de envíos de trenes a la primera estación. Este valor debe ser mayor a 210 por las restricciones a las que está sujeto el sistema.

El segundo modo de operación está diseñado para la experimentación. El resultado es un archivo donde se registran las salidas de 10 iteraciones utilizando los mismos parámetros.
Para su uso, ejecutar:

```
$python iteraciones.py
```

De igual manera, se le pedirá que introduzca el tiempo de envío de trenes. Una vez terminada la ejecución, podrá verse el archivo con los datos generados.

Manual Técnico

El programa está dividido en 2 clases y diversas funciones. La clase más relevante es la de Tren, donde se guarda el estado del tren. Se crea una instancia del tren cada vez que ocurre el evento de envío de tren.

```
class Tren:
    def __init__(self, aidi):
        self.clientes = [0 for _ in xrange(12)]
        self.capacidad = [40, 32, 24, 24, 32, 40]
        for i in range(0, len(self.capacidad)):
            self.capacidad.append(self.capacidad[i])
        self.estacion = -1
        self.movimiento = None
```

```
        self.key = aidi
```

Entre las funciones de mayor utilidad, están aquellas que generan los tiempos de llegadas y de servicio:

```
def TLL(lambd):
    return random.expovariate(lambd)*(23/lambd)

def servicio(mean, sigma):
    return random.gauss(mean, sigma)
```

Hay varios datos de entrada a considerar, y algunos de ellos pueden ser modificados para observar su efecto en el sistema. Estos son:

```
    #### Parametros de Simulacion ####
    reloj = 0
    limite = 7200
    intervalo = inter
    recuperacion = rec
    ################################################

    #### Variables Endogenas ####
    TET = 0
    NCLL = 0
    NCE = [[0]for _ in xrange(3)]
    TEM = 0
    ################################

    #### Datos del Sistema ####
    TT = [84, 66] # tiempo de transportacion
    unidades=11

    ############################

    #### Parametros de las variables aleatorias ####
    lambd = 2.94
    mean = 18.2
    sigma = 2.71

    ################################################
```

El ciclo donde el reloj va avanzando hasta llegar al límite es el siguiente:

```
while reloj <= limite:
    delta = min([TLL1, TLL2, TLL3, TS, TLT, TLE, TRT])
    reloj+= delta
    TET = incrementTET(TET, delta, cola)

    TLL1 -= delta
    TLL2 -= delta
    TLL3 -= delta
    TLE -= delta
    TLT -= delta
    TS -= delta

    if TLL1==0:
        NCLL+=1
        cola[0][cola[0].index(min(cola[0]))]+=1
        TLL1 = TLL(lambd)

    if TLL2==0:
        NCLL+=1
        cola[1][cola[1].index(min(cola[1]))]+=1
        TLL2 = TLL(lambd)

    if TLL3==0:
        NCLL+=1
        cola[2][cola[2].index(min(cola[2]))]+=1
        TLL3 = TLL(lambd)

    if TLT==0:
        if unidades > 0:
            tren.estacion=0
            tren.movimiento = False
            unidades-=1
            TLT = intervalo
            TS = servicio(mean, sigma)
            flag = True
        else:
            TLT = TRT+1

    if TLE==0:
        tren.estacion+=1
        tren.movimiento = False
        TS = servicio(mean, sigma)
```

```
        TLE = top

    if TS==0:
        for vagon in range(0, len(tren.capacidad)):
            capacidad = tren.capacidad[vagon]
            abordo = tren.clientes[vagon]
            cupo = capacidad - abordo
            esperando = cola[tren.estacion][vagon]
            if cupo >= esperando:
                tren.clientes[vagon]+= esperando
                cola[tren.estacion][vagon] -= esperando
            else:
                suben = esperando - cupo
                tren.clientes[vagon]+= suben
                cola[tren.estacion][vagon] -= suben
                olvidados += cola[tren.estacion][vagon]
        NCE[tren.estacion].append(olvidados)
        olvidados = 0
        if tren.estacion==0 or tren.estacion==1:
            TLE = TT[tren.estacion]
            tren.movimiento = True
        else:
            tren = Tren(sistema.deployed)
            sistema.deployed+=1
        TS = top

    TRT -= delta
    if TRT==0:
        unidades+=1
        TRT=recuperacion
```

En el código anterior, cada uno de las sentencias IF definen qué hacer en caso de que suceda el evento. Esto puede ser: subir pasajeros, avanzar a la siguiente estación, añadir usuarios a las colas, sumar el número de unidades disponibles, etc.

Por último, la variable `v = True` está usada para indicar si la simulación va a ser *verbose*, y así imprimir el estado del sistema en cada salto del reloj.