CAPITULO V SIMULACIÓN

5.3 Simulación de Inventarios

5.3.1 Sistema de Inventario

El sistema de inventario es un conjunto de políticas, modelos, componentes (productos, compras, pedidos, costos, órdenes de compra, cantidad de pedido, tiempos de pedido…), que organizados e interrelacionados nos permitan optimizar el costo de mantener el inventario para que la empresa pueda obtener beneficios reales y otros como el **"GoodWill"** o prestigio de la empresa no tangible.

Beneficis de utilizar sistemas de inventarios

- Reducir los costos de almacenamiento para evitar inventarios innecesariamente grandes puede mejorar la competitividad de cualquier empresa.
- Algunas compañías han sido las pioneras en la introducción de sistemas de inventarios "justo a tiempo", donde el sistema hace hincapié, en la planificación para que los productos necesarios lleguen justa a tiempo, logrando grandes ahorros mediante la reducción de inventarios a un mínimo.

Pasos para una administración efectiva de inventarios

- Formular un modelo matemático que describa el comportamiento del sistema de inventario
- Elaborar una política optima de inventarios a partir de este modelo
- Utilizar la simulación en computador para mantener el registro de los niveles de inventario
- A partir de los registros simulados, utilizar la política óptima de inventarios para saber cuánto y cuándo conviene reabastecer.

5.3.2 Modelo de Inventario

Determinístico con revisión continua, Modelo de lote Económico EOQ (economic order quantity)

La situación de inventarios más común que enfrentan los fabricantes, distribuidores, y comerciantes es que los niveles de inventarios se reducen con el tiempo y después se reabastecen con la llegada de nuevas unidades. Se supone que los artículos bajo suposición saldrán en forma continua a una tasa constante conocida **(D=demanda)**, el inventario se reabastece **(al producir u ordenar)** un lote de tamaño fijo **(Q= unidades)** que llegan juntas en un tiempo determinado.

DESCRIPCIÓN FORMAL DEL MODELO

Objetivo: Se trata en determinar con que frecuencia y en que cantidad se debe reabastecer el inventario de manera que se minimice la suma de los costos por unidad de tiempo.

Categoría: Simbólico, Analítico, No lineal, General, Determinístico.

Componentes (Modelo de un distribuidor)

```
**Operativos**
    Costo de Compra
    Costo de Ordenar el pedido
    Costo de Mantener el pedido (Almacenamiento)
    Costo por faltante
    Con faltante
    Sin faltante
**Fluentes**
    Materiales, productos, ventas, compras, pedidos
```

Variables

**Exógenas: **
Demanda: de un articulo a tasa constante
Costo de ordenar: cada vez que se realiza un pedido
Costo de Mantenimiento: Dinero invertido en el almacenamiento del inventario (hasta que se vende o se usa)
Costo del Producto: precio por unidad de articulo
Variables de decisión (Sistema) : Tamaño de pedido (Cuánto), Tiempo entre pedidos (Cuándo)
**Endógenas: ** Costo Mínimo total, Costo Anual de Mantener el inventario, Costo anual de hacer pedidos.

Variables utilizadas

Exógenas:
D = Demanda
Co = Costo de ordenar
Ch = Costo de Mantenimiento
P = Costo del Producto: precio por unidad de articulo
Estado:
Q = Cantidad optima de pedido
Endógenas
MO(Q)=Costo Mínimo total, N= #pedidos, T=tiempo entre pedido, ChT= Costo anual de mantener, CoT = Costo anual de ordenar

ITERACCIÓN ENTRE COMPONENTES (RELACIONES MATEMATICAS)

Número anual de pedidos: $N = D/Q$

Inventario promedio anual: $Q/2$

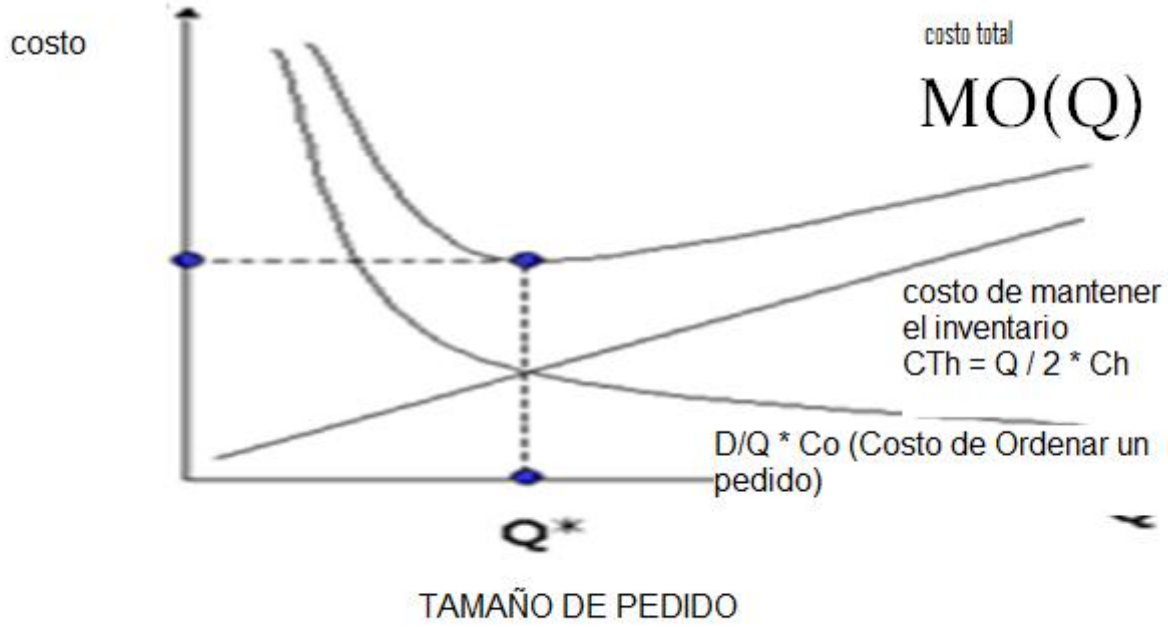
Costo anual de mantener el inventario: $ChT = Q/2 * Ch$

Costo anual de Hacer los pedidos: $CoT = D/Q * Co$

Costo total de inventario $MO(Q) = ChT + CoT$

Tiempo entre pedidos: $T = Q/D$

Gráfico del Modelo



Como podemos observar en el gráfico que más adelante lo vamos a demostrar con se genera, tenemos tres líneas la primera que esta encima del grafico es la curva resultante **MO(Q)** que es la resultante de la fusión de las dos curvas: el costo de ordenar **CoT** y una línea recta del costo de mantener el inventario **ChT**, y en el eje de la parte inferior **TAMAÑO DEL PEDIDO** tenemos la línea de **Q** la cantidad optima del pedido que es donde coinciden el mínimo de **MO(Q)** y la intsersección de las dos líneas de CoT y ChT.

Cálculo de Q* Optima

Como vemos en la figura es necesario calcular **Q*** para lo cual procedemos a igular los dos costos **CoT = ChT**, que es justamente donde se cruzan en el gráfico y además es el punto mínimo de curva del costo total MO(Q).

Resolvemos matemáticamente de la siguiente forma:

$$CoT = ChT$$

$$D/Q * Co = Q/2 * Ch$$

Despejamos Q

$$Q^2 = 2 * D * Co/Ch$$

$$Q = \sqrt{(2 * D * Co/Ch)}$$

A esta ecuación es la que llamamos **Q Optimo** que nos permite determinar la cantidad optima de pedido al mínimo costo.

Ejemplo 1

Westside Auto compra directamente del proveedor un componente que usa en la manufactura de generadores para automóviles. La operación de producción del generador de Westside, la cual trabaja a una tasa constante, requerirá mil componentes por mes a lo largo del año (12000 unidades anuales). Suponga que los costos de ordenar son 25 por pedido, el costo unitario es 2.50 por componentes y los costos de mantener anuales del 20%. WestSide labora 250 días por año y un tiempo de espera es de 5 días, además para la simulación el tiempo de entrega es de acuerdo a la distribución de tiempo en literal e. Responda las siguientes preguntas sobre la política de inventario:

a. ¿Cuál es la EOQ para esta componente?
b. ¿Cuál es el punto de reorden?
c. ¿Cuál es el tiempo de ciclo?
d. ¿Cuáles son los costos totales por pedir y mantener inventario asociados a EOQ recomendado?
e. Simule el sistema con un inventario inicial de 1000 unidades, nivel de pedido 400, ; distribución de la demanda: {(500, 0,25), (450, 040), (400, 0,20), (350, 0,15)} ; distribución del tiempo de entrega: {(2,0.15),(3,0.35),(4,0.50}

DATOS MODELO EOQ	Valores
Demanda anual	D = 12,000.00
Costo de Ordenar	Co = 25.00
Costo de Mantenimiento	Ch = 20% de 2.50 = 0.50
Costo Unitario = P	P = 2.50
Días Habiles del año	Diasano = 250
Tiempo de Espera en días	Tespera = 5
Resultados EOQ	
a. ¿Cuál es la EOQ para esta componente?	Q = 1095.45
b. ¿Cuál es el punto de reorden?	R = 54
¿Numero de pedidos del año?	N = D / Q = 12000 / 1095.45 = 10.95 approx 11 perídos
c. ¿Cuál es el tiempo de ciclo?	T = Días / N = 250 / 10.95 = 22.82
d. ¿Cuáles son los costos totales: Cot	CoT = N *Co* = *10.95* 25.00 = 273.86
ChT	ChT = Q / 2 *Ch* = *1095.44 / 2* 0.5 = 273.86

DATOS MODELO EOQ	Valores
MO(Q)	MO(Q) = CoT + ChT = 273.86 + 273.86 = 547.72

In [1]:
```
#### programación del Modelo EOQ
```

In [2]:
```
#Programa para calcular el modelo EOQ
from math import sqrt
import pandas as pd
import numpy as np

# D = Demanda
# Co = Costo de ordenar
# Ch = Costo de Mantenimiento
# P = Costo del Producto: precio por unidad de articulo
# Q = Cantidad optima de pedido
# MO(Q)=Costo Mínimo total
# N= númeor de pedidos
# T=tiempo entre pedido
# ChT= Costo anual de mantener
# CoT = Costo anual de ordenar
# MO(Q) = Costo total de Inventario

D = 12000.00
Co = 25.00
Ch = 0.50
P = 2.50
Tespera = 5
DiasAno = 250

Q = round(sqrt(((2*Co*D)/Ch)),2)
N = round(D / Q,2)
R = round((D / DiasAno) * Tespera,2)
T = round(DiasAno / N,2)
CoT = N * Co
ChT = round(Q / 2 * Ch,2)
MOQ = round(CoT + ChT,2)
CTT = round(P * D + MOQ,2)

print("Cantidad Optima de Pedito Q = ",Q)
print("Costo total de Ordenar CoT =", CoT)
print("Costo total de Mantener Inventario ChT =", ChT)
print("Costo Total de Ordenar y Mantener Inventario MO(O)", MOQ)
print("Costo Total del Sistema de Inventario CTT", CTT)
print("Número total de pedidos",N)
print("Punto de reorden = R",R)
print("Tiempo de Pedido",T)
```

```
Cantidad Optima de Pedito Q =  1095.45
Costo total de Ordenar CoT = 273.75
Costo total de Mantener Inventario ChT = 273.86
Costo Total de Ordenar y Mantener Inventario MO(O) 547.61
Costo Total del Sistema de Inventario CTT 30547.61
Número total de pedidos 10.95
Punto de reorden = R 240.0
Tiempo de Pedido 22.83
```

In [5]:
```
# Programa para generar el gráfico de costo mínimo
indice = ['Q','Costo_ordenar','Costo_Mantenimiento','Costo_total','Diferencia_Costo_Total']
# Generamos una lista ordenada de valores de Q
from pandas import DataFrame
import numpy as np
import matplotlib as plt

periodo = np.arange(1,19)

def genera_lista(Q):
    n=18
    Q_Lista = []
    i=1
    Qi = Q
    Q_Lista.append(Qi)
    for i in range(1,9):
        Qi = Qi - 60
        Q_Lista.append(Qi)

    Qi = Q
    for i in range(9, n):
        Qi = Qi +  60
        Q_Lista.append(Qi)

    return Q_Lista

Lista= genera_lista(Q)
Lista.sort()


dfQ = DataFrame(index=periodo, columns=indice).fillna(0)

dfQ['Q'] = Lista
#dfQ

for period in periodo:
    dfQ['Costo_ordenar'][period] = D * Co / dfQ['Q'][period]
    dfQ['Costo_Mantenimiento'][period] = dfQ['Q'][period] * Ch / 2
    dfQ['Costo_total'][period] = dfQ['Costo_ordenar'][period] + dfQ['Costo_Mantenimiento'][period]
    dfQ['Diferencia_Costo_Total'][period] = dfQ['Costo_total'][period] -  MOQ
dfQ
```

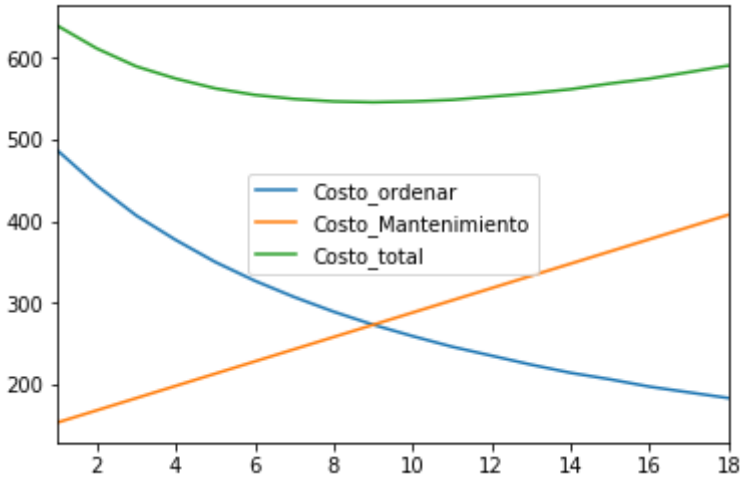
```
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:37: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:38: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
C:\ProgramData\Anaconda3\lib\site-packages\ipykernel_launcher.py:40: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

Out[5]:

	Q	Costo_ordenar	Costo_Mantenimiento	Costo_total	Diferencia_Costo_Total
1	615.45	487	153	640	92
2	675.45	444	168	612	64
3	735.45	407	183	590	42
4	795.45	377	198	575	27
5	855.45	350	213	563	15
6	915.45	327	228	555	7
7	975.45	307	243	550	2
8	1035.45	289	258	547	0
9	1095.45	273	273	546	-1
10	1155.45	259	288	547	0
11	1215.45	246	303	549	1
12	1275.45	235	318	553	5
13	1335.45	224	333	557	9
14	1395.45	214	348	562	14
15	1455.45	206	363	569	21
16	1515.45	197	378	575	27
17	1575.45	190	393	583	35
18	1635.45	183	408	591	43

In [6]:
```
dfG = dfQ.loc[:,'Costo_ordenar':'Costo_total']
dfG
dfG.plot()
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x23032c4da58>



In [ ]:

In [7]:
```
import numpy as np
from pandas import DataFrame

def make_data(product, policy, periods):
    """ Returns dataframe with the details of the inventory simulation.
    Keyword arguments:
    product -- Product object
    policy -- dict that contains the policy name and parameters. For example:
        policy = {'method': "Qs",
                  'param1': 20000,
                  'param2': 10000
                  }
    periods -- numbers of periods of the simulation
    """

    periods += 1
    # Create zero-filled Dataframe
    period_lst = np.arange(periods)  # index

    # Abbreviations
    # INV_INICIAL: INV_NETO_INICIALtial inventory position
    # INV_NETO_INICIAL: INV_NETO_INICIALtial net inventory
    # D: Demand
    # INV_FINAL: Final inventory position
    # INV_FINAL_NETO: Final net inventory
    # LS: Lost sales
    # AVG: Average inventory
    # ORD: order quantity
    # LT: lead time
    header = ['INV_INICIAL','INV_NETO_INICIAL','DEMANDA', 'INV_FINAL',
              'INV_FINAL_NETO', 'VENTAS_PERDIDAS', 'INV_PROMEDIO', 'CANT_ORDENAR', 'TIEMPO_LLEGADA']

    df = DataFrame(index=period_lst, columns=header).fillna(0)
    # Create a list that will store each period order
    order_l = [Order(quantity=0, lead_time=0)
               for x in range(periods)]

    # Fill DataFrame
    for period in period_lst:
        if period == 0:
            df['INV_INICIAL'][period] = product.initial_inventory
```

```
            df['INV_NETO_INICIAL'][period] = product.initial_inventory
            df['INV_FINAL'][period] = product.initial_inventory
            df['INV_FINAL_NETO'][period] = product.initial_inventory

        if period >= 1:
            df['INV_INICIAL'][period] = df['INV_FINAL'][period - 1] + order_l[period - 1].quantity
            df['INV_NETO_INICIAL'][period] = df['INV_FINAL_NETO'][period - 1] + pending_order(order_l, period)

            #demand = int(product.demand())
            demand = 20

            # We can't have negative demand
            if demand > 0:
                df['DEMANDA'][period] = demand
            else:
                df['DEMANDA'][period] = 0

            # We can't have negative INV_INICIAL
            if df['INV_INICIAL'][period] - df['DEMANDA'][period] < 0:
                df['INV_FINAL'][period] = 0
            else:
                df['INV_FINAL'][period] = df['INV_INICIAL'][period] - df['DEMANDA'][period]

            order_l[period].quantity, order_l[period].lead_time = placeorder(product, df['INV_FINAL'][period], policy,
period)

            df['INV_FINAL_NETO'][period] = df['INV_NETO_INICIAL'][period] - df['DEMANDA'][period]
            if df['INV_FINAL_NETO'][period] < 0:
                df['VENTAS_PERDIDAS'][period] = abs(df['INV_FINAL_NETO'][period])
                df['INV_FINAL_NETO'][period] = 0
            else:
                df['VENTAS_PERDIDAS'][period] = 0
            df['INV_PROMEDIO'][period] = (df['INV_NETO_INICIAL'][period] + df['INV_FINAL_NETO'][period]) / 2.0
            df['CANT_ORDENAR'][period] = order_l[period].quantity
            df['TIEMPO_LLEGADA'][period] = order_l[period].lead_time

    return df

def pending_order(order_l, period):
    """Return the order that arrives in actual period"""
    indices = [i for i, order in enumerate(order_l) if order.quantity]
    sum = 0
```

```
    for i in indices:
        if period - (i + order_l[i].lead_time + 1) == 0:
            sum += order_l[i].quantity
    return sum

def demand(self):
        if self.demand_dist == "Constant":
            return self.demand_p1
        elif self.demand_dist == "Normal":
            return make_distribution(
                np.random.normal,
                self.demand_p1,
                self.demand_p2)()
        elif self.demand_dist == "Triangular":
            return make_distribution(
                np.random_triangular,
                self.demand_p1,
                self.demand_p2,
                self.demand_p3
                )()

def lead_time(self):
        if self.leadtime_dist == "Constant":
            return self.leadtime_p1
        elif self.leadtime_dist == "Normal":
            return make_distribution(
                np.random.normal,
                self.leadtime_p1,
                self.leadtime_p2)()
        if self.leadtime_dist == "Triangular":
            return make_distribution(
                np.random.triangular,
                self.leadtime_p1,
                self.leadtime_p2,
                self.leadtime_p3
                )()

def __repr__(self):
        return '<Product %r>' % self.name

def placeorder(product, final_inv_pos, policy, period):
    """Place the order according the inventory policy:
```

```
        Keywords arguments:
        product -- object Product
        final_inv_pos -- final inventory position of period
        policy -- chosen policy Qs or RS
        period -- actual period
        Return:
        quantity to order
        lead time
    """

    #lead_time = int(product.lead_time())
    lead_time = 3
    # Qs = if we hit the reorder point s, order Q units
    if policy['method'] == 'Qs' and \
            final_inv_pos <= policy['param2']:
        return policy['param1'], lead_time
    # RS = if we hit the review period R and the reorder point S, order: (S -
    # final inventory pos)
    elif policy['method'] == 'RS' and \
        period % policy['param1'] == 0 and \
            final_inv_pos <= policy['param2']:
        return policy['param2'] - final_inv_pos, lead_time
    else:
        return 0, 0


politica = {'method': "Qs",
                 'param1': 50,
                 'param2': 20
                 }


class Order(object):
    """Object that stores basic data of an order"""

    def __init__(self, quantity, lead_time):
        self.quantity = quantity
        self.lead_time = lead_time

class product(object):
    def __init__ (self,name,price,order_cost,initial_inventory,demand_dist,demand_p1,
                  demand_p2,demand_p3,leadtime_dist,leadtime_p1,leadtime_p2,leadtime_p3):
```

```
        self.name=name
        self.price=price
        self.order_cost=order_cost
        self.initial_inventory=initial_inventory
        self.demand_dist=demand_dist
        self.demand_p1=demand_p1
        self.demand_p2=demand_p2
        self.demand_p3=demand_p3
        self.leadtime_dist=leadtime_dist
        self.leadtime_p1=leadtime_p1
        self.leadtime_p2=leadtime_p2
        self.leadtime_p3=leadtime_p3


producto = product("Mesa", 18.0,20.0,100,"Constant",80.0,0.0,0.0,"Constant",1.0,0.0,0.0)
print(product)
df = make_data(producto, politica, 52)
df
```

```
<class '__main__.product'>
```

Out[7]:

	INV_INICIAL	INV_NETO_INICIAL	DEMANDA	INV_FINAL	INV_FINAL_NETO	VENTAS_PERDIDAS	INV_PROMEDIO	CANT_ORDENAR	TIEMPO_LI
0	100	100	0	100	100	0	0	0	
1	100	100	20	80	80	0	90	0	
2	80	80	20	60	60	0	70	0	
3	60	60	20	40	40	0	50	0	
4	40	40	20	20	20	0	30	50	
5	70	20	20	50	0	0	10	0	
6	50	0	20	30	0	20	0	0	
7	30	0	20	10	0	20	0	50	
8	60	50	20	40	30	0	40	0	
9	40	30	20	20	10	0	20	50	
10	70	10	20	50	0	10	5	0	
11	50	50	20	30	30	0	40	0	
12	30	30	20	10	10	0	20	50	
13	60	60	20	40	40	0	50	0	
14	40	40	20	20	20	0	30	50	
15	70	20	20	50	0	0	10	0	
16	50	50	20	30	30	0	40	0	
17	30	30	20	10	10	0	20	50	
18	60	60	20	40	40	0	50	0	
19	40	40	20	20	20	0	30	50	
20	70	20	20	50	0	0	10	0	
21	50	50	20	30	30	0	40	0	
22	30	30	20	10	10	0	20	50	
23	60	60	20	40	40	0	50	0	
24	40	40	20	20	20	0	30	50	

	INV_INICIAL	INV_NETO_INICIAL	DEMANDA	INV_FINAL	INV_FINAL_NETO	VENTAS_PERDIDAS	INV_PROMEDIO	CANT_ORDENAR	TIEMPO_LI
25	70	20	20	50	0	0	10	0	
26	50	50	20	30	30	0	40	0	
27	30	30	20	10	10	0	20	50	
28	60	60	20	40	40	0	50	0	
29	40	40	20	20	20	0	30	50	
30	70	20	20	50	0	0	10	0	
31	50	50	20	30	30	0	40	0	
32	30	30	20	10	10	0	20	50	
33	60	60	20	40	40	0	50	0	
34	40	40	20	20	20	0	30	50	
35	70	20	20	50	0	0	10	0	
36	50	50	20	30	30	0	40	0	
37	30	30	20	10	10	0	20	50	
38	60	60	20	40	40	0	50	0	
39	40	40	20	20	20	0	30	50	
40	70	20	20	50	0	0	10	0	
41	50	50	20	30	30	0	40	0	
42	30	30	20	10	10	0	20	50	
43	60	60	20	40	40	0	50	0	
44	40	40	20	20	20	0	30	50	
45	70	20	20	50	0	0	10	0	
46	50	50	20	30	30	0	40	0	
47	30	30	20	10	10	0	20	50	
48	60	60	20	40	40	0	50	0	
49	40	40	20	20	20	0	30	50	
50	70	20	20	50	0	0	10	0	

	INV_INICIAL	INV_NETO_INICIAL	DEMANDA	INV_FINAL	INV_FINAL_NETO	VENTAS_PERDIDAS	INV_PROMEDIO	CANT_ORDENAR	TIEMPO_LI
51	50	50	20	30	30	0	40	0	
52	30	30	20	10	10	0	20	50	

In [ ]:

In [9]:
```
df.values
```

Out[9]:
```
array([[100, 100,   0, 100, 100,   0,   0,   0,   0],
       [100, 100,  20,  80,  80,   0,  90,   0,   0],
       [ 80,  80,  20,  60,  60,   0,  70,   0,   0],
       [ 60,  60,  20,  40,  40,   0,  50,   0,   0],
       [ 40,  40,  20,  20,  20,   0,  30,  50,   3],
       [ 70,  20,  20,  50,   0,   0,  10,   0,   0],
       [ 50,   0,  20,  30,   0,  20,   0,   0,   0],
       [ 30,   0,  20,  10,   0,  20,   0,  50,   3],
       [ 60,  50,  20,  40,  30,   0,  40,   0,   0],
       [ 40,  30,  20,  20,  10,   0,  20,  50,   3],
       [ 70,  10,  20,  50,   0,  10,   5,   0,   0],
       [ 50,  50,  20,  30,  30,   0,  40,   0,   0],
       [ 30,  30,  20,  10,  10,   0,  20,  50,   3],
       [ 60,  60,  20,  40,  40,   0,  50,   0,   0],
       [ 40,  40,  20,  20,  20,   0,  30,  50,   3],
       [ 70,  20,  20,  50,   0,   0,  10,   0,   0],
       [ 50,  50,  20,  30,  30,   0,  40,   0,   0],
       [ 30,  30,  20,  10,  10,   0,  20,  50,   3],
       [ 60,  60,  20,  40,  40,   0,  50,   0,   0],
       [ 40,  40,  20,  20,  20,   0,  30,  50,   3],
       [ 70,  20,  20,  50,   0,   0,  10,   0,   0],
       [ 50,  50,  20,  30,  30,   0,  40,   0,   0],
       [ 30,  30,  20,  10,  10,   0,  20,  50,   3],
       [ 60,  60,  20,  40,  40,   0,  50,   0,   0],
       [ 40,  40,  20,  20,  20,   0,  30,  50,   3],
       [ 70,  20,  20,  50,   0,   0,  10,   0,   0],
       [ 50,  50,  20,  30,  30,   0,  40,   0,   0],
       [ 30,  30,  20,  10,  10,   0,  20,  50,   3],
       [ 60,  60,  20,  40,  40,   0,  50,   0,   0],
       [ 40,  40,  20,  20,  20,   0,  30,  50,   3],
       [ 70,  20,  20,  50,   0,   0,  10,   0,   0],
       [ 50,  50,  20,  30,  30,   0,  40,   0,   0],
       [ 30,  30,  20,  10,  10,   0,  20,  50,   3],
       [ 60,  60,  20,  40,  40,   0,  50,   0,   0],
       [ 40,  40,  20,  20,  20,   0,  30,  50,   3],
       [ 70,  20,  20,  50,   0,   0,  10,   0,   0],
       [ 50,  50,  20,  30,  30,   0,  40,   0,   0],
       [ 30,  30,  20,  10,  10,   0,  20,  50,   3],
       [ 60,  60,  20,  40,  40,   0,  50,   0,   0],
       [ 40,  40,  20,  20,  20,   0,  30,  50,   3],
       [ 70,  20,  20,  50,   0,   0,  10,   0,   0],
```

```
[ 50,  50,  20,  30,  30,   0,  40,   0,   0],
[ 30,  30,  20,  10,  10,   0,  20,  50,   3],
[ 60,  60,  20,  40,  40,   0,  50,   0,   0],
[ 40,  40,  20,  20,  20,   0,  30,  50,   3],
[ 70,  20,  20,  50,   0,   0,  10,   0,   0],
[ 50,  50,  20,  30,  30,   0,  40,   0,   0],
[ 30,  30,  20,  10,  10,   0,  20,  50,   3],
[ 60,  60,  20,  40,  40,   0,  50,   0,   0],
[ 40,  40,  20,  20,  20,   0,  30,  50,   3],
[ 70,  20,  20,  50,   0,   0,  10,   0,   0],
[ 50,  50,  20,  30,  30,   0,  40,   0,   0],
[ 30,  30,  20,  10,  10,   0,  20,  50,   3]], dtype=int64)
```

In [ ]: