Se buscar encontrar la eficiencia de la generación de números pseudo-aletorios a través de los métodos de cuadrados medios y congruencia lineal, para ello se debe seguir el siguiente proceso:

In [ ]:

```
#1. A traves de la misma api generar una semilla diferente.
#2. Encontrar el numero de iteraciones hasta que se repita uno de sus datos.
#3. Generar 100 simulaciones con diferentes semillas.
#4. Generar un histograma con el resultado obtenidos por cada método.
#5. Agregar sus conclusiones, opiniones y recomendaciones
```

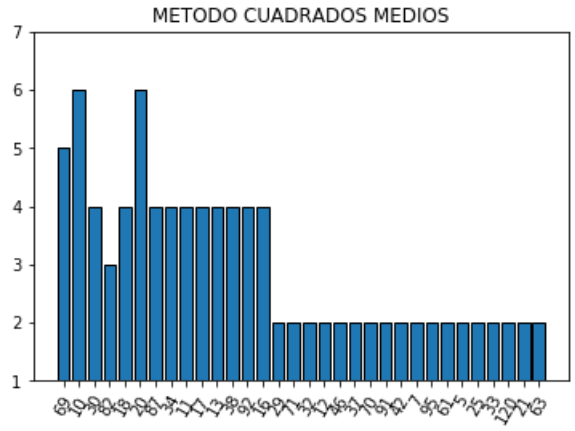
Desarrollo

Con cuadrados Medios

In [54]:

```
from pandas import DataFrame
%matplotlib inline
from matplotlib import pyplot as plt
semilla=9678
digitos=4
iteraciones=100
total_iteraciones=[]
inicio="1"
cont_val=[]
cont_datos = []
cont_int = []
datos = total_iteraciones
cont=[]

for i in range(0,iteraciones):
    valor_ui=[]
    control=True
    cont_it=1
    while control:
        semillacuadrada=semilla*semilla
        stringXN=str(semillacuadrada)
        cantidad=len(stringXN)
        d1=int((int((cantidad/2))-int((digitos/2))))
        d2=int((int((cantidad/2))+int((digitos/2))))
        nuevaSemilla=stringXN[int(d1):int(d2)]
        for j in range(0,digitos):
            inicio=inicio+"0"
        calculosemilla=float(nuevaSemilla)/float(inicio)
        semilla=int(nuevaSemilla)
        if(cont_it==1):
            valor_ui.append(nuevaSemilla)
            cont_it=cont_it+1
        else:
            for aux in range(0,len(valor_ui)):
                if(valor_ui[aux]==nuevaSemilla):
                    total_iteraciones.append(cont_it)
                    semilla=int(valor_ui[0])+100-5
                    control=False
            valor_ui.append(nuevaSemilla)
            cont_it=cont_it+1
def grafico(valor_list,valor):

    cont_val=0
    for i in valor:
        for j in valor_list:
            if i == j:
                cont_val=cont_val+1
        cont.append(cont_val)
```

```
        cont_val=0
    return cont

for x in datos:
    del cont_val[:]
    if x not in cont_int:
        cont_int.append(x)
    else:
        if x not in cont_datos:
            cont_datos.append(x)

metodo=grafico(datos,cont_datos)
limit_x = cont_datos
limit_y = cont
plt.bar(range(len(cont_datos)), limit_y, edgecolor='black')
plt.xticks(range(len(cont_datos)), limit_x, rotation=60)
plt.title("METODO CUADRADOS MEDIOS")
plt.ylim(min(limit_y)-1, max(limit_y)+1)
plt.show()
print("TOTAL QUE SE REPITIERON EN LAS",iteraciones,"ITERACIONES FUERON",len(cont_datos))
tabla_result={'Valor Iteraciones':total_iteraciones}
dataframe=DataFrame(tabla_result)
dataframe
```




```
TOTAL QUE SE REPITIERON EN LAS 100 ITERACIONES FUERON 32
```

Out[54]:

	Valor Iteraciones
0	97
1	35
2	82
3	56
4	23
...	...
95	120
96	21
97	16
98	63
99	69

100 rows × 1 columns

Desarrollo

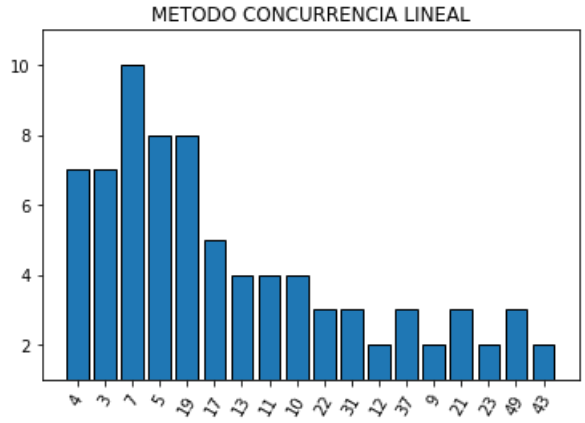
Con Concurrencia Lineal

In [56]:

```
from pandas import DataFrame
%matplotlib inline
from matplotlib import pyplot as plt
valor_a=1
valor_b=2
semilla=3398
iteraciones=100
valor_m=3
total_iteraciones_c=[]
cont_val=[]
cont_datos = []
cont_int = []
datos = total_iteraciones_c
cont=[]


for j in range(0,iteraciones):
    valor_ui_C=[]
    xn=0
    ui=0.0
    aux=0
    control= True
    cont_iter=1
    xn=semilla
    while control:
        aux=((valor_a*xn)+valor_b)%(valor_m)
        ui=aux/valor_m
        xn=aux
        if (cont_iter==1):
            valor_ui_C.append(ui)
            cont_iter=cont_iter+1
        else:
            for x in range(0,len(valor_ui_C)):
                if (valor_ui_C[x]==ui):
                    total_iteraciones_c.append(cont_iter)
                    semilla=int(xn)
                    valor_a=valor_a+2
                    valor_b=valor_b+3
                    valor_m=valor_m+1
                    control=False
            valor_ui_C.append(ui)
            cont_iter=cont_iter+1

def grafico(valor_list,valor):

    cont_val=0
    for i in valor:
        for j in valor_list:
            if i == j:
                cont_val=cont_val+1
        cont.append(cont_val)
        cont_val=0
    return cont

for x in datos:
    del cont_val[:]
    if x not in cont_int:
        cont_int.append(x)
    else:
        if x not in cont_datos:
            cont_datos.append(x)

metodo=grafico(datos,cont_datos)
print(len(cont_datos))
limit_x = cont_datos
limit_y = cont
plt.bar(range(len(cont_datos)), limit_y, edgecolor='black')
plt.xticks(range(len(cont_datos)), limit_x, rotation=60)
plt.title("METODO CONCURRENCIA LINEAL")
plt.ylim(min(limit_y)-1, max(limit_y)+1)
plt.show()
print("TOTAL QUE SE REPITIERON EN LAS",iteraciones," ITERACIONES FUERON",len(cont_datos))
tabla_result={'Valor Iteraciones':total_iteraciones_c}
dataframe=DataFrame(tabla_result)
```

```
dataframe
```

18



TOTAL QUE SE REPITIERON EN LAS 100 ITERACIONES FUERON 18

Out[56]:

	Valor Iteraciones
0	4
1	3
2	2
3	7
4	4
...	...
95	43
96	10
97	4
98	51
99	49

100 rows × 1 columns

Conclusiones

La implementacion de numeros pseudoaleatorios es un que permite resolver problemas complejos con el uso de la simulacion para el presente ejercicio relizado se partio desde una semilla antes comprobada donde los valors resultantes sean precisos

Opinion

En ambos metodos la generacion de numeros es aleatorio aun asi el proceso de la semilla siempre tendra a ser la variable mas importante solo el proceso e diferente para generar los numeros

Recomendaciones

El trabajar con numeros pseudoaleatorios puede llegar a ser un proceso complicado ya que todo dependera de la funcionalidad que se pretenda emplear para generar los numeros por lo cual es mejor realizar una comprobacion de como esta trabajando la semilla dentro del proceso de cada modelo para generar bien los datos