Universidad Politecnica Salesiana

Nombre: Jessica Ñauta.

Asigantura: Simulación.

Metodo Cuadrado Medio

In [28]:
```
from collections import Counter
from collections import defaultdict
import random
import psutil
import numpy as np
import pandas as pd
import math
import collections
import matplotlib.pyplot as plt
```

In [29]:
```
valores =[74731897457]
arreglorn=[]
def get_pos(digs):
    val1 =0
    val2 =0
    if digs%2 !=0:
        val1 = int(digs/2)
        val2 = int(digs/2)+1
    else:
        val1 = int(digs/2)
        val2 = int(digs/2)
    return val1,val2

def calcular_num(iters, val, digs):
    x0_semilla = int(val)
    aum = get_pos(digs)
    print("ITERACIÓN", "Xn", "Xn*Xn", "Longitud","Ui","Rn")
    for i in range(iters):
        xn2= x0_semilla**2
        lon = len(str(xn2))
        ui = str(xn2)[int(lon/2)-aum[0]:int(lon/2)+aum[1]]
        rn = int(ui)/10**digs
        arreglorn.append(rn)
        #df=pd.DataFrame({"Xn":x0_semilla, "Xn*Xn":xn2 ,"Longitud":lon, "UI ":ui, "R
        print(i, " ", x0_semilla," ",xn2, " ", lon, " ",ui, " ", rn)
        x0_semilla=int(ui)
    print("   ")

iters = int(input("Iteraciones: "))
digs = int(input("Ingrese el digito: "))
for i in valores:
    print("i: ", i)
    calcular_num(iters, i, digs)
    print("  ")
```

```
Iteraciones: 100
Ingrese el digito: 7
i:  74731897457
ITERACIÓN Xn Xn*Xn Longitud Ui Rn
0   74731897457   5584856497523563066849   22   9752356   0.9752356
1   9752356   95108447550736   14   8447550   0.844755
2   8447550   71361101002500   14   1101002   0.1101002
```

3	1101002	1212205404004	13	2205404	0.2205404
4	2205404	4863806803216	13	3806803	0.3806803
5	3806803	14491749080809	14	1749080	0.174908
6	1749080	3059280846400	13	9280846	0.9280846
7	9280846	86134102475716	14	4102475	0.4102475
8	4102475	16830301125625	14	0301125	0.0301125
9	301125	90676265625	11	6762656	0.6762656
10	6762656	45733516174336	14	3516174	0.3516174
11	3516174	12363479598276	14	3479598	0.3479598
12	3479598	12107602241604	14	7602241	0.7602241
13	7602241	57794068222081	14	4068222	0.4068222
14	4068222	16550430241284	14	0430241	0.0430241
15	430241	185107318081	12	1073180	0.107318
16	1073180	1151715312400	13	1715312	0.1715312
17	1715312	2942295257344	13	2295257	0.2295257
18	2295257	5268204696049	13	8204696	0.8204696
19	8204696	67317036452416	14	7036452	0.7036452
20	7036452	49511656748304	14	1656748	0.1656748
21	1656748	2744813935504	13	4813935	0.4813935
22	4813935	23173970184225	14	3970184	0.3970184
23	3970184	15762360993856	14	2360993	0.2360993
24	2360993	5574287946049	13	4287946	0.4287946
25	4287946	18386480898916	14	6480898	0.6480898
26	6480898	42002038886404	14	2038886	0.2038886
27	2038886	4157056120996	13	7056120	0.705612
28	7056120	49788829454400	14	8829454	0.8829454
29	8829454	77959257938116	14	9257938	0.9257938
30	9257938	85709416011844	14	9416011	0.9416011
31	9416011	88661263152121	14	1263152	0.1263152
32	1263152	1595552975104	13	5552975	0.5552975
33	5552975	30835531350625	14	5531350	0.553135
34	5531350	30595832822500	14	5832822	0.5832822
35	5832822	34021812483684	14	1812483	0.1812483
36	1812483	3285094625289	13	5094625	0.5094625
37	5094625	25955203890625	14	5203890	0.520389
38	5203890	27080471132100	14	0471132	0.0471132
39	471132	221965361424	12	9653614	0.9653614
40	9653614	93192263260996	14	2263260	0.226326
41	2263260	5122345827600	13	2345827	0.2345827
42	2345827	5502904313929	13	2904313	0.2904313
43	2904313	8435034001969	13	5034001	0.5034001
44	5034001	25341166068001	14	1166068	0.1166068
45	1166068	1359714580624	13	9714580	0.971458
46	9714580	94373064576400	14	3064576	0.3064576
47	3064576	9391626059776	13	1626059	0.1626059
48	1626059	2644067871481	13	4067871	0.4067871
49	4067871	16547574472641	14	7574472	0.7574472
50	7574472	57372626078784	14	2626078	0.2626078
51	2626078	6896285662084	13	6285662	0.6285662
52	6285662	39509546778244	14	9546778	0.9546778
53	9546778	91140970181284	14	0970181	0.0970181
54	970181	941251172761	12	2511727	0.2511727
55	2511727	6308772522529	13	8772522	0.8772522
56	8772522	76957142240484	14	7142240	0.714224
57	7142240	51011592217600	14	1592217	0.1592217
58	1592217	2535154975089	13	5154975	0.5154975
59	5154975	26573767250625	14	3767250	0.376725
60	3767250	14192172562500	14	2172562	0.2172562
61	2172562	4720025643844	13	0025643	0.0025643
62	25643	657563449	9	5756344	0.5756344
63	5756344	33135496246336	14	5496246	0.5496246
64	5496246	30208720092516	14	8720092	0.8720092
65	8720092	76040004488464	14	0004488	0.0004488
66	4488	20142144	8	0142144	0.0142144
67	142144	20204916736	11	2049167	0.2049167
68	2049167	4199085393889	13	9085393	0.9085393
69	9085393	82544365964449	14	4365964	0.4365964
70	4365964	19061641649296	14	1641649	0.1641649
71	1641649	2695011439201	13	5011439	0.5011439

72	5011439	25114520850721	14	4520850	0.452085
73	4520850	20438084722500	14	8084722	0.8084722
74	8084722	65362729817284	14	2729817	0.2729817
75	2729817	7451900853489	13	1900853	0.1900853
76	1900853	3613242127609	13	3242127	0.3242127
77	3242127	10511387484129	14	1387484	0.1387484
78	1387484	1925111850256	13	5111850	0.511185
79	5111850	26131010422500	14	1010422	0.1010422
80	1010422	1020952618084	13	0952618	0.0952618
81	952618	907481053924	12	4810539	0.4810539
82	4810539	23141285470521	14	1285470	0.128547
83	1285470	1652433120900	13	2433120	0.243312
84	2433120	5920072934400	13	0072934	0.0072934
85	72934	5319368356	10	1936835	0.1936835
86	1936835	3751329817225	13	1329817	0.1329817
87	1329817	1768413253489	13	8413253	0.8413253
88	8413253	70782826042009	14	2826042	0.2826042
89	2826042	7986513385764	13	6513385	0.6513385
90	6513385	42424184158225	14	4184158	0.4184158
91	4184158	17507178168964	14	7178168	0.7178168
92	7178168	51526095836224	14	6095836	0.6095836
93	6095836	37159216538896	14	9216538	0.9216538
94	9216538	84944572705444	14	4572705	0.4572705
95	4572705	20909631017025	14	9631017	0.9631017
96	9631017	92756488454289	14	6488454	0.6488454
97	6488454	42100035310116	14	0035310	0.003531
98	35310	1246796100	10	4679610	0.467961
99	4679610	21898749752100	14	8749752	0.8749752

In [24]:
```
n= int (math.sqrt(len(arreglorn)))
def clasificarNumeros(n,arregloRn):
    grupos = []
    inicio=0.00
    a=0
    b=1
    ranNumeros= {}

    for i in range(n+1):
        grupos.append(round(inicio,2))
        inicio=inicio+(1/n)

    for i in range(len(grupos)-1):
        valInferior=grupos[a]
        valSuperior=grupos[b]
        ranNumeros.update({str(valInferior)+","+str(valSuperior):[]})
        for i in arregloRn:
            if i==0.00:
                if i>=valInferior and i<= valSuperior:
                    ranNumeros[str(valInferior)+","+str(valSuperior)].append(i)
            else:
                if i>valInferior and i<= valSuperior:
                    ranNumeros[str(valInferior)+","+str(valSuperior)].append(i)
        a=b
        b=a+1
    return ranNumeros
```

In [26]:
```
def chi_cuadrado(n,arreglorn):
    ei = []
    oi = []
    to = []
    for i in list(n.keys()):
        ei.append(i)
        oi.append(n[i])
        to.append((len(n) - n[i]) ** 2 / len(n))
```

```
    d = {'Ei': ei, 'Oi': oi, "(Oi - Ei)²/Ei": to}
    df = pd.DataFrame(data=d)
    total = df['(Oi - Ei)²/Ei'].sum()
    validacion = total < arreglorn
    return df, total, validacion
```

In [32]:

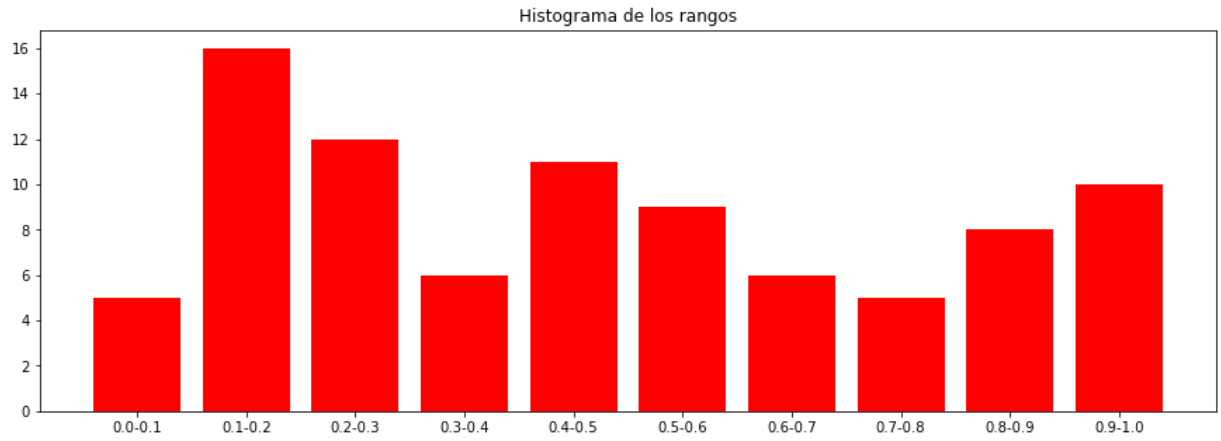
```
def plot_histrograma(dic):
    plt.figure(figsize=(15, 5))
    keys = dic.keys()
    values = dic.values()
    plt.bar(keys, values, color="red")
    plt.title("Histograma de los rangos")
    plt.show()

semilla = 74731897457
cantidad = 100
digito = 7
valor = 16.9

lista = cuadrados_medios(cantidad, semilla, digito)
dic = cantidad_lista(lista)
plot_histrograma(dic)

df, total, val = chi_cuadrado(dic, valor)
sumaOi=sumaOi
if sumaOi<=16.9:
    print('La distribución uniforme se acepta')
else:
    print('La distribución uniforme no se acepta')

print("Chi Cuadrado")
df
```




```
Total de (Oi - Ei)²/Ei 12.799999999999999
La distribución uniforme se acepta
Chi Cuadrado
```

Out[32]:

	Ei	Oi	(Oi – Ei)²/Ei
0	0.0-0.1	5	2.5
1	0.1-0.2	16	3.6
2	0.2-0.3	12	0.4
3	0.3-0.4	6	1.6
4	0.4-0.5	11	0.1
5	0.5-0.6	9	0.1
6	0.6-0.7	6	1.6

	Ei	Oi	(Oi – Ei)²/Ei
7	0.7-0.8	5	2.5
8	0.8-0.9	8	0.4
9	0.9-1.0	10	0.0

Metodo congruencia lineal

In [2]:
```
def m_congruencias_lineales(x, a, b, m):

    periodo = 0
    bandera = 0
    cont = 0
    xant = 0
    print("")
    print("Metodo Congruencias Lineales")
    print("  n  ", " Xo ", "     Un ", "  Xn+1")
    while(bandera != x):
        if (periodo == 0):
            bandera = x
        xant=x
        x = (a * x + b) % m
        print(" ", cont,"   ", xant,"  ", round(xant/m,4),"   ", x)
        periodo = periodo + 1
        cont=cont+1

def main():
    x = int(input("Introduce Xo: "))
    a = int(input("Introduce a: "))
    b = int(input("Introduce b: "))
    m = int(input("Introduce m: "))
    m_congruencias_lineales(x,a,b,m)

if __name__ == "__main__":
    main()
```

```
Introduce Xo: 7
Introduce a: 74731897457
Introduce b: 37747318974
Introduce m: 19

Metodo Congruencias Lineales
  n    Xo       Un    Xn+1
  0     7    0.3684     17
  1     17    0.8947     16
  2     16    0.8421     18
  3     18    0.9474     14
  4     14    0.7368     3
  5     3    0.1579     6
  6     6    0.3158     0
  7     0    0.0     12
  8     12    0.6316     7
```

In [3]:
```
def chi_cuadrado(n,arreglorn):
    ei = []
    oi = []
    to = []
    for i in list(n.keys()):
        ei.append(i)
        oi.append(n[i])
        to.append((len(n) - n[i]) ** 2 / len(n))
    d = {'Ei': ei, 'Oi': oi, "(Oi – Ei)²/Ei": to}
```

```
    df = pd.DataFrame(data=d)
    total = df['(Oi - Ei)²/Ei'].sum()
    validacion = total < arreglorn
    return df, total, validacion
```

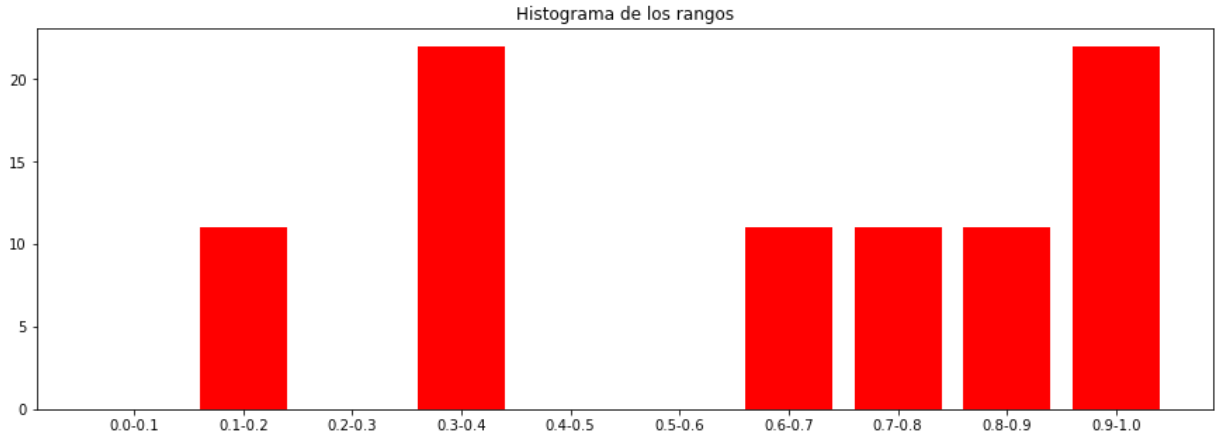
In [27]:

```
def plot_histrograma(dic):
    plt.figure(figsize=(15, 5))
    keys = dic.keys()
    values = dic.values()
    plt.bar(keys, values, color="red")
    plt.title("Histograma de los rangos")
    plt.show()

semilla = 74731897457
cantidad = 100
digito = 7
valor = 16.9
a=74731897457
c=37747318974
M=19

lista = congruencia(semilla, cantidad,a,c,M,digito)
dic = cantidad_lista(lista)
dic = cantidad_lista(lista)
plot_histrograma(dic)
df, total, val = chi_cuadrado(dic, valor)
sumaOi=sumaOi
if sumaOi<=16.9:
    print('La distribución uniforme se acepta')
else:
    print('La distribución uniforme no se acepta')

print("Chi Cuadrado")
df
```




```
La distribución uniforme no se acepta
Chi Cuadrado
```

Out[27]:

	Ei	Oi	(Oi – Ei)²/Ei
0	0.0-0.1	0	10.0
1	0.1-0.2	11	0.1
2	0.2-0.3	0	10.0
3	0.3-0.4	22	14.4
4	0.4-0.5	0	10.0
5	0.5-0.6	0	10.0
6	0.6-0.7	11	0.1

	Ei	Oi	$(Oi - Ei)^2/Ei$
7	0.7-0.8	11	0.1
8	0.8-0.9	11	0.1
9	0.9-1.0	22	14.4

In [ ]: