1 Flowchart

The probability of a=10, b=5, and c=1 is one in a thousand

In [18]:

```
import random
num=0
while 1 :
    list=[1,2,3,4,5,6,7,8,9,10]
    a=random.choice(list)
    b=random.choice(list)
    c=random.choice(list)

    if a==10 and b==5 and c==1 :
        print(a+b-10*c)
        print("总次数:",num)
        break
    else:
        if a>b:
            if b>c:
                print(a+b-10*c)
                num=num+1
            else:
                if a>c:
                    print(a+c-10*b)
                    num = num + 1
                else:
                    print(a+c-10*b)
                    num = num + 1
        else:
            if b>c:
                if a>c:
                    print(a+c-10*b)
                    num = num + 1
                else:
                    print(a+c-10*b)
                    num = num + 1
            else:
                print(b+c-10*a)
                num=num+1
```

```
-91
-72
-16
-7
-91
-86
-69
-83
-38
-24
-52
-24
-91
-89
-92
-15
5
5
总次数: 473
```

2 Continuous celing function

Here we tested the output from one to ten

In [3]:
```
list=[1,2,3,4,5,6,7,8,9,10]
def fx(x):
    if x==1:
        return 1
    else:
        return fx(round(x/3))+2*x
for i in list:
    print(fx(i))
```

```
1
5
7
9
15
17
19
23
25
27
```

3 Dice rolling

Here since we have not yet thought of a better way, the slowest method is used to solve this problem, which takes a very long run time and is very computationally large

```
In [4]: import random
        list=[1,2,3,4,5,6]
        number_of_way=[]
        def Find_number_of_ways(num):
            count = 0
            for i in list:
                for j in list:
                    for k in list:
                        for l in list:
                            for m in list:
                                for n in list:
                                    for o in list:
                                        for p in list:
                                            for q in list:
                                                for r in list:
                                                    if i + j + k + l + m + n + o + p + q + r == num:
                                                        count = count + 1
            return count
        for i in range(10,61):
            num=Find_number_of_ways(i)
            print(num)
            number_of_way.append(num)
        index=9
        max_value=0
        for num in number_of_way:
            if (num > max_value):
                max_value = num
                index = index + 1

        print('Maximum value:', max_value," index:",index)
```

```
1535040
1151370
831204
576565
383470
243925
147940
85228
46420
23760
11340
4995
2002
715
220
55
10
1
Maximum value: 4395456  index: 35
```

4 Dynamic programming

4.1

```
In [24]: import random
         def Random_integer(N):
             array = [0,1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
             array1=[]
             for i in range(0,N):
                 array1.append(random.choice(array))
             return array1
         print(Random_integer(10))
```

```
[7, 1, 1, 10, 6, 0, 10, 8, 5, 0]
```

4.2

```
In [26]: def Sum_averages(originArray):
             list=[]
             result = [[]]
             sum = 0
             sumSub = 0
             for num in originArray:#Getting inspiration from https://blog.csdn.net/weixin_43509127/articl
                 for element in result[:]:
                     x = element[:]
                     x.append(num)
                     result.append(x)
                     sum = 0
                     for y in range(0, len(x)):
                         sum = sum + x[y]
                     sumSub = sumSub + sum / len(x)
             result.pop(0)
             return sumSub
         print(Sum_averages([1,2,3,4,7,8,9,10]))
```

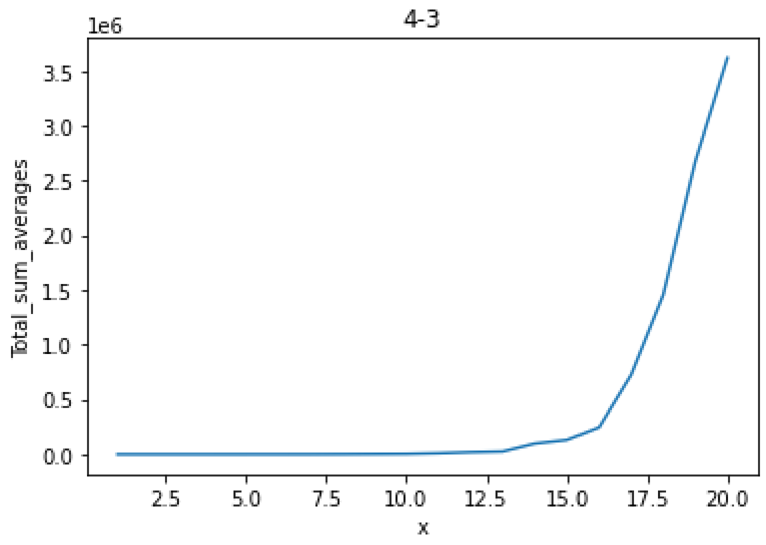
```
1402.5000000000002
```

4.3

In this question, because the algorithm used is not good enough, the calculation time is very long, and the program always fails to run successfully when x=100, so only 20 is calculated in the end

In [10]:
```
# import numpy as np
list1=[]
Total_sum_averages=[]
for x in range(1,21):
    list1.append(x)
    Total_sum_averages.append(Sum_averages(Random_integer(x)))
print(list1,Total_sum_averages)
x=np.array(list1)
y=np.array(Total_sum_averages)
x.reshape(1,20)
y.reshape(1,20)
plt.plot(x,y)
plt.xlabel('x')
plt.ylabel('Total_sum_averages')
plt.title('4-3')
plt.show()
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20] [4.0, 12.0, 39.66666666
6666664, 48.75, 210.8, 283.5, 508.00000000000017, 1306.8749999999998, 2782.1111111111113, 4296.
599999999999, 10048.90909090911, 19451.250000000004, 25833.15384615383, 98298.0000000004, 13106
8.00000000023, 245756.24999999945, 724745.5294117604, 1456350.0000000033, 2676623.1052631666, 3
617583.749999854]
```



5 Path counting

5.1

```
In [21]: def Matrix(n,m):
             list=[[]]
             choice=[0,1]
             for i in range(0,n):
                 array=[]
                 for j in range(0,m):
                     if i==0 and j == 0:
                         array.append(1)
                     elif i==n-1 and j==m-1:
                         array.append(1)
                     else:
                         array.append(random.choice(choice))

                 list.append(array)
             list.pop(0)
             return list
         print(Matrix(10,10))
```

```
[[1, 1, 1, 0, 1, 0, 0, 1, 0, 0], [0, 0, 0, 0, 1, 0, 0, 0, 1, 0], [1, 0, 0, 1, 0, 0, 0, 1, 1,
0], [1, 0, 1, 0, 0, 0, 1, 1, 0, 0], [0, 0, 0, 0, 0, 0, 1, 0, 1, 0], [0, 0, 1, 0, 1, 1, 1, 1, 0,
0], [1, 0, 1, 0, 1, 1, 0, 1, 1, 0], [0, 0, 0, 1, 0, 1, 1, 0, 0, 0], [0, 1, 1, 0, 0, 1, 0, 1, 1,
0], [1, 1, 0, 1, 0, 0, 1, 1, 0, 1]]
```

5.2

```
In [13]: path = 0
         def countPath(Matrix,x,y,endx,endy):
             global path
             if(x == endx and y == endy):
                 path += 1
                 return path
             if(y != endy and Matrix[x][y+1] != 0 ):
                 y=y+1
                 return countPath(Matrix,x,y,endx,endy)

             if(x != endx and Matrix[x+1][y] != 0 ):
                 x=x+1
                 return countPath(Matrix,x ,y,endx,endy)
             if(x==endx and Matrix[x][y+1] == 0):
                 return 0
             if (y == endy and Matrix[x+1][y ] == 0):
                 return 0
             if (Matrix[x][y+1] == 0 or Matrix[x+1][y]==0):
                 return 0
```

5.3

```
In [19]: for j in range(0,1000):
             for i in range(0, 1000):
                 MAT = Matrix(10, 8)
                 countPath(MAT, 0, 0, 9, 7)
         print("mean path:",path/1000)
```

```
mean path: 6.078
```

```
In [ ]:
```