Cryptography
Assignment 2.

NAME: SNEHA MAGANAHALLI
ROLL NO: CS21M522.

1. problem 8-2.

Solution:

$x_{i+1} = a x_i \pmod m$

$\left[\begin{array}{c}\text{longest}\\ \text{possible}\\ \text{period}\\ P\end{array}\right] = \frac{m}{4}$

x_0 should be odd

Multiplier of a is given by

$a = 3 + 8k$

$a = 5 + 8K$

for $k = 0, 1, 2 \ldots$

Formula

Given Equation.

$x_{n+1} = a x_n \bmod 2^q$

a) $\left[\begin{array}{c}\text{longest possible}\\ \text{period}\end{array}\right] = \frac{m}{4} = \frac{2^q}{4} = \frac{2^q}{2^2} = 2^{q-2} = 2^2 = \underline{4}$

b) What is a?

$a = 3+8k$ or $a = 5+8k$

for $k=0$

$a = 3+8(0)$ or $a = 5+8\cdot(0)$

$a = 3$ or $a = 5$

for $k=1$

$a = 3+8(1)$ or $a = 5+8(1)$

$a = 11$ $a = 13$

∴ a must be 3, 5, 11, 13

c) x_0 should be odd always ---- By the formula.

(2) 8.4 problem:

Solution:

$x_{n+1} = (6x_n) \bmod 13$

If seed $x_0 = 1$ the sequence obtained are:

1, 6, 10, 8, 9, 2, 12, 7, 3, 5, 4, 11, 1, ... (1)

$x_{n+1} = (7x_n) \bmod 13$

If seed $x_0 = 1$, the sequence obtained are

1, 7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2, 1 ---- (2)

4 2 1

we are able to see the consistent change as 4, 2, 1 i.e., powers of 2.

∴ This is not suitable.

∴ (1) appears more Random & it is the best.

(5) Problem 8.5
Calculate the value of π, & also check the random number generator of the system library.

Solution:

Let
Let x, y be 2 random numbers

$$\left[\begin{array}{l}\text{probability of} \\ \gcd(x,y) = 1\end{array}\right] P = \frac{6}{\pi^2} \quad \text{——(1)}$$

To calculate the system library's random number generator we have to calculate the probability of $g(x,y) = 1$

$$\left[\begin{array}{l}\text{probability of} \\ \gcd(x,y) = 1 \\ \text{of system} \\ \text{Library}\end{array}\right] P_{lib} = \frac{6}{\pi^2} \quad \text{- By (1)}$$

$$P_{lib} \cdot \pi^2 = 6$$

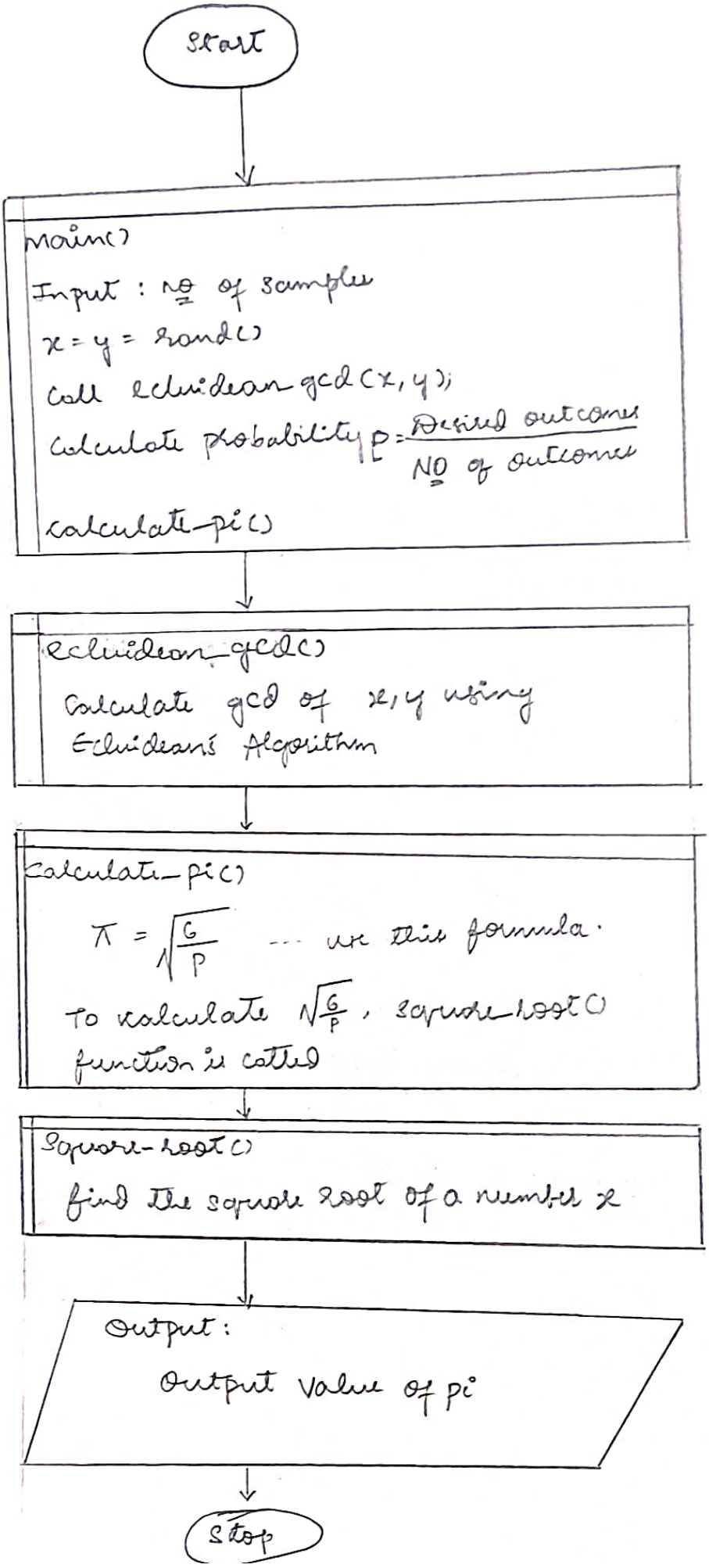
$$\pi^2 = \frac{6}{P_{lib}}$$

$$\pi = \sqrt{\frac{6}{P_{lib}}}$$

If our system library has a very good random number generator we will get $\pi = 3.142$ exactly, else it will be approximately equal to 3.142.

PTO

Flow chart


Start
main()
Input : No of samples
x = y = rand()
Call Eclideon_gcd(x, y);
Calculate probability p = Desired outcomes / No of outcomes
calculate_pi()
Eclideon_gcd()
Calculate gcd of x, y using Eclideans Algorithm
calculate_pi()
$\pi = \sqrt{\frac{6}{p}}$ ... use this formula.
To calculate $\sqrt{\frac{6}{p}}$, square_root() function is called
Square_root()
find the square root of a number x
Output :
Output Value of pi
Stop

Algorithm:

```
unsigned int eclidean_gcd(x, y)
{
        if(x<y)
            swap(x, y)

        while(x%.y!=0)
        {
            x+=y
            y=x-y
            x=x-y
            y%=x
        }
        return y
}
Squareroot(x)
{
   Return sqrt(x)
}
calculate_pi( probability)
{
    temp← 6/probability
    return Squareroot(temp)
}

//This is the start of the function.

main()
{
    input: no of samples.
    x= y= random numbers
    for( i=0  to  no of samples)
    {
            res← eclideon_gcd(x, y)
            if(res ==1)
                count++
    }
    probability← count / No of samples
    pi ← calculate_pi()
    output PI
}
```

(1) Problem 8.6:

RC4 has below 2 steps out of 3 steps.

i)
```
for (i=0 to 255)
{
    s[i] ← i
    k[i] ← key [i mod keylength]
}
```

ii)
```
j = 0
for (i=0 to 255)
{
    j = (j + s[i] + k[i]) mod 256
    swap(s[i], s[j])
}
```

S[i] values initially after step 1.

0 1 2 3 .... 255.

Even after step ii) if S[i] should have the same values then

k[0] = k[1] = 0

k[2] = 255

k[3] = 254

∴ K[i] values are

0, 0, 255, 254, 253.

By step 2,

$S[j] = S[j + S[i] + k[i]] \bmod 256$

$S[0] = S[0 + 0 + \underline{0}] = 0$

↓ If we have 0 then only we get S[0] = 0

∴ k[0] = 0

$S[1] = S[0 + 1 + \underline{0}] = \underline{1}$

↓ If we have 0 here we get S[] = 1

$S[2] = S[0 + 2 + \underline{255}] \bmod 256 = 2$

↓ If we have 255 here we get S[2] = 2.

(5) Problem 8.7 :

Solution :

a) How many bits are used to store the internal state S, When all possible permutations of S are present.

By step (2) of RC4

```
j=0
for ( i=0   to 255 )
{
    j = (j + S[i] + K[i]) Mod 256.
    swap(S[i], S[j])

}
```

To know the internal state & store it we need to know i, j, S[i]

To represent i we need 8 Bits
To represent j we need 8 bits.
To store S[i] we need 8 × 256 bits.... ∵ we have 0...255 values for S[i]

∴ 8+8+(8×256)

= 2064 bits

(b) How much information is represented by a state?

Since we have all the permutations possible for S[i] and S[i] can be 0...255 i.e. 256 values.

∴ (All permutation) = n! = 256!

i and j 256 values independently.

i.e; $(256 \times 256) = 256^2$

$\therefore$ [No of states] $= 256! \times 256^2 = 2^{1700}$

$\therefore$ we need 1700 bits to represent & store the internal state

6. Problem 8.8 :

Short description of question

i) Alice-Bob communicate, use RC4, use same key everytime, use 128 bits key.

ii) Choose V = [Random 80 bit]

c = RC4 (V || k) $\oplus$ m.

iii) send V || C

Solution :-

a) How Bob will recover message (m) from V||c using k?

Since V|| C is concatenation of V and c bits.

$\therefore$ Bob will get V, by taking the I 80 bits from V||c.

Since Bob is using RC4, he knows key k.

$\therefore$ M (msg) = RC4 (V||c) $\oplus$ k.

(b) The adversary has several values of $(V_1 \| C_1)$ $(V_2 \| C_2)$, can adversary determine same key was used?

YES.

Example:

Adversary has $(V_1 \| C_1)$, $(V_2 \| C_2)$ i.e;

He has $E(A)$, $E(B)$ ---

Where A, B are plain text
$E(A)$ → Encryption of plain text A.

$E(A) = A \oplus C$

$E(B) = B \oplus C$

Adversary will do

$E(A) \oplus E(B)$

$= (A \oplus C) \oplus (B \oplus C)$

$= A \oplus B$

$x \oplus x = 0$

∴ He Got plain text

(c) Approximately how many messages can Alice expect to send before the same key stream will be used twice.

Solution:

Birthday paradox formula

Given a random variable that is an integer with uniform distribution between 1 and n and a selection of k instances ($k \leq n, k$), that there is atleast 1 duplicate.

$= \sqrt{n}$

[Since key is fixed, the key stream varies with the random 80 bit] = $\sqrt{n}$ — Birthday paradox

$= \sqrt{2^{(80)}}$

$= 2^{40}$ messages

[with 80 bit we can have 2^{80} combinations]

(8) What is the lifetime of key? or
i.e; No of messages that can be encrypted using k?

The key should be changed before we send 2^{40} messages. So that $[2^{140}+1]^{th}$ message will use the new key.

—— X ——