<u>SOLUTIONS FOR CS6530 ASSIGNMENT - 2</u>

Prepared by **Gobinath Periyasamy**

Roll No: **CS21M501**

Contents

Problem 8.2 1
Problem 8.4 3
Problem 8.5 4
Problem 8.6 6
Problem 8.7 7
Problem 8.8 8

Problem 8.2

8.2

a. What is the maximum period obtainable from the following generator? $X_{n+1} = (aX_n) \bmod 2^4$

b. What should be the value of a?

c. What restrictions are required on the seed?

<u>*Explanation*</u>:

The sequence of random numbers {Xn} is obtained via the following iterative equation: $X_{n+1} = (aX_n) \bmod 2^4$

a the multiplier $0 < a < m$

X_0 the starting value, or seed $0 <= X_0 < m$

For given generator equation we try to get the sequence of numbers from the seed using the $X_{n+1} = (aX_n) \bmod 2^4$

Since mod is 2^4, we loop through to find mod of 16.

a the multiplier $0 < a < 16$

X_0 the starting value, or seed $0 <= X_0 < 16$

Also, since *c value is not given, it is 0.*

```
m = 2**4
c = 0
for tot_x in range(m):
    a = [i for i in range(1,m)]
    for j in a:
        x = tot_x
        out_list = [x]
        prev_x = x
        print("x =",x, "a =",j)
        for i in range(0,m):
                x1 = (j*prev_x)%m
                prev_x = x1
                if (prev_x in out_list):
                    break
                out_list.append(x1)
        print("generated values = ",out_list, "period = ",len(out_list))
        print("\n")
```

Generated Sequence for each X seed can be given as below:

Linear_congruential_generators.txt <<please refer the attachment-2 >>

Solution:

a. What is the maximum period obtainable from the following generator?

$X_{n+1} = (aX_n) \bmod 2^4$

So, the maximum period obtainable is 5 including the seed from the sequence we generated.

b. What should be the value of a?

From above generated sequence, maximum period occurs for the 'a value' of 2, 6, 10, 14. So a can be 2,6,10,14.

c. What restrictions are required on the seed?

We can see that the period is less for the even numbers of X and hence, *seed X should be* ***odd***.

Problem 8.4

8.4

With the linear congruential algorithm, a choice of parameters that provides a full period does not necessarily provide a good randomization. For example, consider the following two generators:

$X_{n+1} = (11X_n) \bmod 13$

$X_{n+1} = (2X_n) \bmod 13$

Write out the two sequences to show that both are full period. Which one appears more random to you?

Solution:

When we run the Pseudorandom Number Generator for $X_{n+1} = (11X_n) \bmod 13$

We get

```
x = 1 a = 6

generated values = [1, 6, 10, 8, 9, 2, 12, 7, 3, 5, 4, 11] period =
12
```

When we run the Pseudorandom Number Generator for $X_{n+1} = (2X_n) \bmod 13$

We get

```
x = 1 a = 7

generated values = [1, 7, 10, 5, 9, 11, 12, 6, 3, 8, 4, 2] period =
12
```

Though the full period is same, the second half of the second sequence for x =1, a =7 seems to be more predictable since each value is half of the previous one.

So, the second sequence seems to be less random than the first sequence.

Problem 8.5

8.5

In any use of pseudorandom numbers, whether for encryption, simulation, or statistical design, it is dangerous to trust blindly the random number generator that happens to be available in your computer's system library. [PARK88] found that many contemporary textbooks and programming packages make use of flawed algorithms for pseudorandom number generation. This exercise will enable you to test your system.

The test is based on a theorem attributed to Ernesto Cesaro (see [KNUT98] for a proof), which states the following: Given two randomly chosen integers, x and y, the probability that gcd(x, y) = 1 is 6/pi^2. Use this theorem in a program to determine statistically the value of p. The main program should call three subprograms: the random number generator from the system library to generate the random integers; a subprogram to calculate the greatest common divisor of two integers using Euclid's Algorithm; and a subprogram that calculates square roots. If these latter two programs are not available, you will have to write them as well. The main program should loop through a large number of random numbers to give an estimate of the aforementioned probability. From this, it is a simple matter to solve for your estimate of p.

If the result is close to 3.14, congratulations! If not, then the result is probably low, usually a value of around 2.7. Why would such an inferior result be obtained?

Explanation:

The program is

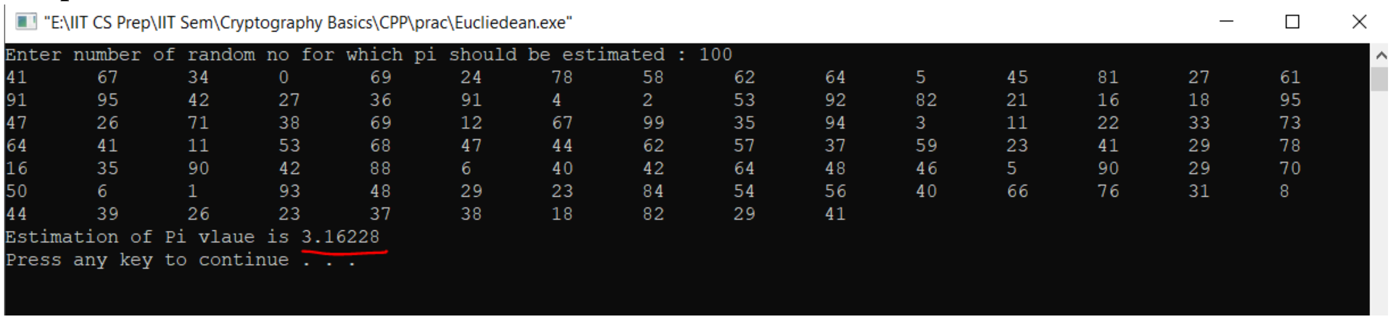
Euc_gcd_pi.cpp

<<Please refer the attachment - 3>>

```
// ******************************
// * Assignment-2 Number 8.5
// ******************************


#include <stdio.h>
#include <stdlib.h>
#include<time.h>
#include <bits/stdc++.h>
using namespace std;

// Function to return
// gcd of a and b
int Euc_gcd(int a, int b)
{
    if (a == 0)
        return b;
    return Euc_gcd(b % a, a);
}

// Main Program
int main()
{
    int total;

// ************************************************
// * Generate random numbers using rand() function
// ************************************************

    cout << "Enter number of random no for which pi should be
estimated : ";
    cin >> total;
    int r_num[total];
    for(int i = 0; i<total; i++)
        r_num[i] = (rand()%total)+1;
    for(int i = 0; i<total; i++)
        cout << r_num[i] << '\t';
    cout << endl;

    int sum  = 0,temp = 0;
    for(int i = 0; i<total; i+= 2)
    {
        temp = Euc_gcd(r_num[i], r_num[i+1]);
        if (temp == 1)
```

```
            sum++;
    }

// *******************************************
// * Estimation of PI value from 6/(PI^2)
// *******************************************

    float prob_get = (6*(total/2))/sum;
    float PI =  sqrt(prob_get);
    cout<<"Estimation of PI vlaue is "
        << PI <<endl;
    system("pause");

    return 0;

}
```

Output screen shot for 100 numbers:



```
"E:\IIT CS Prep\IIT Sem\Cryptography Basics\CPP\prac\Eucliedean.exe"
Enter number of random no for which pi should be estimated : 100
41      67      34      0       69      24      78      58      62      64      5       45      81      27      61
91      95      42      27      36      91      4       2       53      92      82      21      16      18      95
47      26      71      38      69      12      67      99      35      94      3       11      22      33      73
64      41      11      53      68      47      44      62      57      37      59      23      41      29      78
16      35      90      42      88      6       40      42      64      48      46      5       90      29      70
50      6       1       93      48      29      23      84      54      56      40      66      76      31      8
44      39      26      23      37      38      18      82      29      41
Estimation of Pi vlaue is 3.16228
Press any key to continue . . .
```

The **estimation of pi value is 3.16228** for 100 random values which is around 3.14. So the probability of random numbers is gcd(x, y) = 1.

If the value will be less than 3.14 around 2.7 means the generated random numbers do not have more gcd(x, y) = 1. If the two numbers have no common divisor other than 1, they exhibit more randomness.

Problem 8.6

8.6

What RC4 key value will leave S unchanged during initialization? That is, after the initial permutation of S, the entries of S will be equal to the values from 0 through 255 in ascending order.

Explanation:

When we use key of length 255 bytes and if the first two bytes of the key are zero, which means k[0] = 0 and k[1] = 0.

So, since k[0] = k[1], the entries would be left unchanged.

And then, we will have:

k[2] = 255, k[3] = 254, … k[255]= 2.

Problem 8.7
8.7

RC4 has a secret internal state which is a permutation of all the possible values of the vector S and the two indices i and j.

a. Using a straightforward scheme to store the internal state, how many bits are used?

<u>*Solution*</u>:

To store the internal state i, j, and S, 8 + 8 + (256 * 8) = ***2064 bits*** would be required.

b. Suppose we think of it from the point of view of how much information is represented by the state. In that case, we need to determine how many different states there are, then take the log to base 2 to find out how many bits of information this represents. Using this approach, how many bits would be needed to represent the state?

<u>*Solution*</u>:

The information is represented by the state [256! * 256^2] = 2^{1700} (approx). Hence, ***bits of 1700*** would be needed to represent the state.

Problem 8.8
8.8

Alice and Bob agree to communicate privately via email using a scheme based on

RC4, but they want to avoid using a new secret key for each transmission. Alice and Bob privately agree on a 128-bit key k. To encrypt a message m, consisting of a string of bits, the following procedure is used.

1. Choose a random 64-bit value v

2. Generate the ciphertext c = RC4(v || k) ⊕ m

3. Send the bit string (v || c)

a. Suppose Alice uses this procedure to send a message m to Bob. Describe how Bob can recover the message m from (v || c) using k.

Solution:

Bob knows that the cipher is c = RC4(v || k) ⊕ m

The first 80 bits of (v || c) can give Bob the initialization vector, v.

Bob already knows the key k.

Since v, c, k are known, Bob should generate bits using the RC4 with the v and k that the message can be recovered as **m = RC4(v || k) ⊕ c**

b. If an adversary observes several values (v1 || c1), (v2 || c2), c transmitted between Alice and Bob, how can he/she determine when the same key stream has been used to encrypt two messages?

Solution:

Cipher message would be (v || c) so that he can derive the v.

Let each cycle can be denoted as i and j ,c_i = RC4(v || k) ⊕ m_i and c_j would be RC4(v || k) ⊕ m_j

So, if adversary monitors the more and more ciphers, he can find the similarities in the cipher for the v, since Alice and Bob are using same key.

Therefore,

$c_i \oplus c_j = RC4(v \parallel k) \oplus m_i \oplus RC4(v \parallel k) \oplus m_j$

$c_i \oplus c_j = m_i \oplus m_j$

$m_i = (c_i \oplus c_j) \oplus m_j$

$m_j = (c_i \oplus c_j) \oplus m_i$

So, finding those similarities makes the messages m_j and m_i to be more vulnerable and can be decrypted as we given above.

c. Approximately how many messages can Alice expect to send before the same key stream will be used twice? Use the result from the birthday paradox described in Appendix U.

<u>Solution</u>:

We know the probability of at least one student has same birthday from birthday paradox is

$1 – (\text{days}! / ((\text{days} - \text{people}!) * (\text{days}^{\text{people}})))$

Since the key is fixed, the key stream varies with the choice of the 80-bit v, which is selected randomly. Thus, Total length of our stream 80 + 128 = 208

For 80 bits, $1 – (208! / ((208 - 80!) * (208^{80})))$ gives 0.99999997756169 of probability that the two stream can be same. So, after approximately messages with the 2^{80} bits are sent, for the same v, and hence the same key stream should be used more than once.

d. What does this imply about the lifetime of the key k (i.e., the number of messages that can be encrypted using k)?

<u>Solution</u>:

Within sending the 2^{40} messages the key value k has to be changed by Alice

.