Homework Assignment # 1

Miguel Gomez U1318856

2024-02-02 11:07:53

Contents

1 Homework 1

The extended euclidean algorithm is a spin on the usual algo that allows for us to split the number into two intermediate values. one that is equal to the number we are dividing in the algo multiplied by a number, and another that is the divisor multiplied by the quotient. This expression below is the one we end up with:

1.1 expression

$$g = gcd(a, b)$$

$$\exists\ s, t \mid s \cdot a + t \cdot b = g$$

```
start=$(date +%s.%N)
Singular sing/hw1_b.sing | grep -v -e "\*\* loaded\|\*\* library"
end=$(date +%s.%N)
echo "Execution Time: $(echo "$end - $start" | bc) seconds"
```

1.2 output of hw1_b results

```
                     SINGULAR                                 /  Development
 A Computer Algebra System for Polynomial Computations       /   version 4.2.1
                                                           0<
 by: W. Decker, G.-M. Greuel, G. Pfister, H. Schoenemann     \   May 2021
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \  Debian 1:4.2.1-p3+ds-1
// ** but for functionality you may wish to change to the new
// ** format. Please refer to the manual for further information.
The example computed GCD of 24 and 16 is:
8
// ** redefining r (ring r = integer, (x), lp;) hw1_b.sing:21
The computed myintGCD of 24 is: 8
The computed myEuclid of 24 is: 8
The computed myExtendedEuclid of the numbers is:

GCD(24,16) = 8
s = 1
t = -1

The computed GCD of the list of numbers for problem 1-b is:
10
Auf Wiedersehen.
Execution Time: .028753389 seconds
```

1.3 Pseudocode for the Euclidean algo

Algorithm 1 Euclidean Algorithm

```
procedure myExtendedEuclid($a$, $b$)
    $R1 \leftarrow a$
    $R2 \leftarrow b$ while $R2 \neq 0$ do

      $Q \leftarrow (R1/R2)$
    $r \leftarrow R1 - Q \times R2$
    $R1 \leftarrow R2$
    $R2 \leftarrow r$

    return $r$
end procedure
```

1.4 Pseudocode for the Euclidean algo

Algorithm 2 Extended Euclidean Algorithm

```
procedure myExtendedEuclid($a$, $b$)
    $R1 \leftarrow a$
    $R2 \leftarrow b$
    $S1 \leftarrow 1$
    $S2 \leftarrow 0$
    $T1 \leftarrow 0$
    $T2 \leftarrow 1$ while $R2 > 0$ do

     $Q \leftarrow \text{floor}(R1/R2)$
    $r \leftarrow R1 - Q \times R2$
    $R1 \leftarrow R2$
    $R2 \leftarrow r$
    $s \leftarrow S1 - Q \times S2$
    $S1 \leftarrow S2$
    $S2 \leftarrow s$
    $t \leftarrow T1 - Q \times T2$
    $T1 \leftarrow T2$
    $T2 \leftarrow t$

    print "GCD(", $a$, ",", $b$, ") = ", $S1 \times a + T1 \times b$
    print "s = ", $S1$
    print "t = ", $T1$
    $L \leftarrow \text{list}()$
    $L \leftarrow \text{list}(S1 \times a + T1 \times b, S1, T1)$
    return $L$
end procedure
```

1.5 Identify whether the integers 38 and 7 have multiplicative inverses in $\mathbf{Z_{180}}$

Since the number p we are working with is even, we will not have multiplicative inverses for even numbers. Therefore we only need to find the inverse for the one we can, for 7.

$$a \in \mathcal{Z}_{180}\ ,\ a^{-1} \in \mathcal{Z}_{180} \text{ if } gcd(a, 180) = 1$$

```
start=$(date +%s.%N)
Singular sing/hw1_c.sing | grep -v -e "\*\* loaded\|\*\* library"
end=$(date +%s.%N)
echo "Execution Time: $(echo "$end - $start" | bc) seconds"
```

1.6 output of hw1_c results

```
                      SINGULAR                                 /  Development
 A Computer Algebra System for Polynomial Computations       /   version 4.2.1
                                                            0<
 by: W. Decker, G.-M. Greuel, G. Pfister, H. Schoenemann     \   May 2021
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \  Debian 1:4.2.1-p3+ds-1
// ** but for functionality you may wish to change to the new
// ** format. Please refer to the manual for further information.
The computed myintGCD of 7 is:
1
The computed myintGCD of 38 is:
2

GCD(7,180) = 1
s = -77
t = 3

The inverse of 7 modulo 180 is 103

GCD(38,180) = 2
s = 19
t = -4

38 has no inverse modulo 180
Auf Wiedersehen.
Execution Time: .022770642 seconds
```

Since the gcd for the expression comes out to 1, the inverse exists and is printed out. Since 38 is even, no inverse is possible modulo 180.

2 Problem 2

Solving linear diophantine equations using linear congruences.

2.1 a) solving LC $4x \equiv 4$ mod 6

Solving this is easiest with a table of the results we would get by plugging in any values for x from the set mod 6.

x	$4x \mod 6$	Congruent to 4?
0	$4 \cdot 0 \mod 6 = 0$	No
1	$4 \cdot 1 \mod 6 = 4$	Yes
2	$4 \cdot 2 \mod 6 = 2$	No
3	$4 \cdot 3 \mod 6 = 0$	No
4	$4 \cdot 4 \mod 6 = 4$	Yes
5	$4 \cdot 5 \mod 6 = 2$	No

$\therefore$ we have exactly two solutions which are congruent for this problem.

2.2 Solving as an LDE instead

using the expression $4x \equiv 4$ mod 6, we can transform the expression into the following:

$$\begin{aligned} 4x &\equiv 4 \text{ mod } 6 \\ 6| \; 4x &- 4 \\ 6k &= 4(x-1) \\ 3k &= 2(x-1) \end{aligned}$$

Now we find values of x that would allow the expression to be integer valued when $x \in \{0..5\}$. In general, the solutions will be the same as they were before giving us just two possible solutions to the expression. Using the following:

$$\begin{aligned} x &= 1 \\ 3k &= 2(1-1) = 0 \\ k &= 0 \\ x &= 4 \\ 3k &= 2(4-1) = 6 \\ k &= 2 \end{aligned}$$

3 Problem 3

We must find a solution to the affine cipher and obtain the keys $[k_1, k_2]$. We can set this up in singular to solve for the key vector using the inverse matrix algorithm that utilizes the transformation to a reduced row eschelon form of the matrix.

```
start=$(date +%s.%N)
Singular sing/hw3.sing | grep -v -e \
    "\*\* loaded\|\*\* library\|\*\* redefining"
end=$(date +%s.%N)
echo "Execution Time: $(echo "$end - $start" | bc) seconds"
```

3.1 output of hw3 results

```
                     SINGULAR                                 /  Development
 A Computer Algebra System for Polynomial Computations       /   version 4.2.1
                                                           0<
 by: W. Decker, G.-M. Greuel, G. Pfister, H. Schoenemann     \   May 2021
FB Mathematik der Universitaet, D-67653 Kaiserslautern        \  Debian 1:4.2.1-p3+ds-1
print matrix A
18,1,
19,1
print matrix B
4,
19
Determinant of A:
printing det(A)
25
gcd(det(A), 26) is:
1
inverse of A exists
inverse of A:
inv_A[1,1]=25
inv_A[1,2]=1
inv_A[2,1]=19
inv_A[2,2]=8
check of inv_A*A = I:
_[1,1]=1
_[1,2]=0
_[2,1]=0
_[2,2]=1
Solutions for x = :
_[1,1]=15
_[2,1]=20
Auf Wiedersehen.
Execution Time: .053407240 seconds
```

Shown above is the results and we see that the key $[k_1, k_2]$ is $[15, 20]$.

4 Problem 4 - Hill Cypher

Considering the Hill Cypher, we are restricted to using only 8 letters. Therefore we must do everything modulo 8. Getting the encryption $\mathbf{C} = \mathbf{P} \cdot \mathbf{K}$, and the decryption $\mathbf{P} = \mathbf{C} \cdot \mathbf{K}^{-1}$ will be done as follows:

$$\mathbf{K} = \begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

a) Set up the problem as a system of linear congruences to identify $\mathbf{K}$.

b) Is the given matrix $\mathbf{P}$ invertible? Is the given matrix $\mathbf{C}$ invertible? In other words, can we We apply the encryption algorithm to the plaintext, character by character: compute the key as $\mathbf{C} \cdot \mathbf{P}^{1} = \mathbf{K}$?

c) Does there exist a unique (one and only one) key matrix $\mathbf{K}$ that satisfies these constraints? If not, how many distinct matrices $\mathbf{K}$ can be used for this cipher?

d) Based on the above analysis, explain whether the above system is secure to a known-plaintext or a chosen-plaintext attack? [Note: A known-plaintext attack is one where some $(\mathbf{P}, \mathbf{C})$ pairs are known to Eve. A chosen-plaintext attack is similar to the known-plaintext one, except that the $(\mathbf{P}, \mathbf{C})$ pairs are chosen by the attacker herself.]