CSC301 HW2

Alex Zhang

Jan 2023

Question 1

Let algorithm computing x^y that makes at most $O(n)$ calls to integer multiplication called EXPO. The pseudocode is following below.

```
function z = EXPO(x,y)
    if y = 0 then return 1
    z = EXPO(x, math.floor(y/2))
    z = z*z
    if y is even then
        return z
    else
        return z*x
    end if
end function
```

Proof:

Base Case: $y = 0$
The algorithm then returns 1, which is true because $x^0 = 1$ for all x.

Induction: Assume $y = n - 1$ is true, we are going to prove that $y = n$ for $z = x^y$ is also true.
The recursion will get $z' = x^{\lfloor y/2 \rfloor}$.

Case 1: $y = n$ is even
The EXPO returns z. If y is even, then $\lfloor y/2 \rfloor = y/2$, which

$$z' = x^{y/2}$$

squaring both sides,

$$(z')^2 = x^y$$

So $z = (z')^2$ when y is even. Just as what EXPO will return.

Case 2: $y = n$ is odd
Then EXPO returns $z \cdot x$. In this case $\lfloor y/2 \rfloor = (y-1)/2$, which

$$z' = x^{y-1/2}$$

squaring both sides and multipling by x,

$$(z')^2 \cdot x = x^y$$

In this case $(z')^2 = z$. So $(z')^2 \cdot x = z \cdot x$. This matched the EXPO returning value. ■

1. Integer multiplication

   Since we assume that x, y are all n-bits numbers, so in each recursion, y will chunk the last bit so there are total n times recursion calls. In each call, it will perform a integer multiplication on z it self, and sometimes possible calling multiplication for x and z. So overall, there are $O(n)$ integer multiplication calls for the whole recursion process.

2. Time Complexity

   We have showed that there are total $O(n)$ calls to integer multiplication, and for multiplication itself, th time Complexity will be $O(n^2)$, which overall, the time complexity for EXPO will be $O(n^3)$

Question 2

(a)

If a number $1 \leq x < p$ for a given prime p is its own multiplicative inverse, then

$$x^2 \equiv 1 \bmod p$$

By definition, for some integer k:

$$x^2 - 1 = pk$$
$$(x+1)(x-1) = pk$$

We have two case for $k = 0$ and $k \neq 0$.

Case 1: $k \neq 0$
Since $1 \leq x < p$ and x is relative prime to p, so either $x + 1 = p$ or $x - 1 = p$. However, the second case will not happen since $x - 1 < p - 1$. $x = p - 1$ is one number that is its own multiplicative inverse.

Case 2: $k = 0$
This time either $x - 1 = 0$ or $x + 1 = 0$, but $x + 1 \geq 2$ with the given condition so $x - 1 = 0$ and $x = 1$.
There are two numbers 1 and $p - 1$ which are their own inverse.

(b)

$(p-1)! = (p-1)(p-2)(p-3)\ldots 2 \cdot 1$, so

$$(p-1)! \equiv -1 \bmod p$$

$$(p-1)(p-2)(p-3)\ldots 2 \cdot 1 \equiv -1 \bmod p$$

$$-1 \cdot (p-2)(p-3)\ldots 2 \cdot 1 \equiv -1 \bmod p$$

We need to show that $(p-2)(p-3)\ldots 2 \equiv 1 \bmod p$.
Since for prime p, every number $1 \leq x < p$ is invertible modulo p, we have

$$ax \equiv 1 \bmod p$$

Taking $a \bmod p$, $a \in \{1, 2, \ldots, p-2, p-1\}$ and since 1 and $p-1$ are their own inverse, so for each x, there is going to be a multiplicative inverse $a \in \{2, \ldots, p-2\}$. Therefore, we can pairing up every element in $\{2, \ldots, p-2\}$ to make a multiplicative inverse respect to p.

$$-1 \cdot (p-2)(p-3)\ldots 2 \cdot 1 \equiv -1 \bmod p$$

will become

$$-1 \cdot 1 \cdot 1 \ldots 1 \cdot 1 \equiv -1 \bmod p$$

which is true.■

(c)

Suppose N is a composite number, then there will be a factor $b > 1$ and $b \leq N-1$. This indicates that d divides $(N-1)!$, meaning that d does not divide $(N-1)!+1$, same as

$$(N-1)! \not\equiv -1 \bmod N$$

So if N is not prime, then $(N-1)! \not\equiv -1 \bmod N$. ■

(d)

I think the problem is that Wilson's Theorem takes more time on doing the primality test. For example, the time complexity when N is a n-bits number will be $O(2^n n^2)$ because the the maximum times of integer multiplication will be $2^{n-1}-1$ and each multiplication itself is $O(n^2)$. Whereas Fermat's Little Theorem only takes like $O(n^3)$ and maximum of $O(n^4)$ for doing n times of check.

Question 3

(a)

Based on the question, we know that p is prime, and for encryption integer e, and e is relative prime to $s = p - 1$. The following pseduocode is the method used for recovering x for $0 \leq x < p$.

```
function Decrypt(y,e,p)
    (d,a,b) = EXTEuclid(e,p-1)
    x = MODEXP(y,d,p)
    return x
end function
```

Proof:

Case 1: $x \not\equiv 0 \bmod p$
Since in this method, Extended Euclid's algorithm is used for getting integer d where $de \equiv 1 \bmod s$. This means $de - 1 = ks$ for some inetger k.

$$x^{de} = x^{1+ks} = x \cdot x^{(p-1)k}$$

By Fermat's Little Theorem, if $x \not\equiv 0 \bmod p$, then

$$x^{ks} \equiv x^{(p-1)k} \equiv 1^k \equiv 1 \bmod p$$

Therefore

$$x^{de} \equiv x \cdot x^{ks} \equiv 1 \cdot x \equiv x \bmod p$$

Case 2: $x \equiv 0 \bmod p$
Since for this RSA-like scheme, $0 \leq x < p$, then only possible number for x is 0. This case the method will also be true since

$$0^{de} \equiv 0 \bmod p$$

The correctness of this RSA-like scheme has benn proved. ■

Bit Complexity

- Alice generates one n-bits prime p
- Alice computes $s = p - 1$
- Alice chooses e relatively prime to s
- Eve knows e and p and computes s and $d \equiv e^{-1} \bmod s$
- Eve computes $x \equiv y^d \bmod p$

When p is a n-bits prime number, the time complexity for this method is $O(n^3)$ overall. In step four, Eve will use this method and use Extended Euclidea's Algorithm to get decryption integer d which takes $O(n^3)$. In step five, Eve then using the modular exponentiation to recover the original message x, which takes $O(n^3)$. Overall, this method will take polynomial time to recover x.

(b)

With given s and public key e, we can use the Extended Euclidea's Algorithm to get the multiplicative inverse of e where

$$de \equiv 1 \bmod s$$

The time complexity for Extended Euclidea's Algorithm is $O(n^3)$ and this is polynomial time.