

Discrete Math

Functions

A)Problem Statement :

Implementation to different function related to Discrete math :

- 1) Fast Modular Exponentiation
- 2) Chinese Remainder Theorem (CRT) Applications
- 3) Prime Number Generator
- 4) Extended Euclidean Theorem

B)Data Structure Used:

Array of long integers .

C) Algorithms Used :

1) Fast Modular Exponentiation :

a- Iterative :foo (a,b)

Complexity : $O(\log_2(n))$ (n: max number of bits of(b,a))

```
while b > 0
    if b mod 2
        ans = (ans * a) mod m;
        b--;
    else
        a = (a*a)%m;
        b /= 2;
return ans
```

b- Recursive :foo (a,b)

Complexity : $O(\log_2(n))$ (n: max number of bits of(b,a))

```
if b == 0
    return 1;
if b mod 2
    return (a * foo((a * a) mod m, (b - 1) / 2)) mod m;
else
    return foo((a * a) mod m, b / 2) mod m;
```

c- Naïve 1 :

Complexity : $O(b)$, has a big chance in getting memory overflow if $c*a > \text{int/long size range}$

```
c = 1
for i = 1 to b
    c = c * a
c = c mod m
return c
```

d- Naïve 2 :

Complexity : $O(b)$ but avoids memory overflow for some range and consumes more time in computing (mod m)

```
c = 1
for i = 1 to b
    c = (c * a) mod m
return c
```

2) Extended Euclidean Theorem : foo(a,b)

Complexity : $O(\log_2(n : \text{max number of bits (a,b)}))$

This was translated to a code

```
q = a / b
r = a % b
d = r;
a = b;
b = r;
while r != 0
    s = s2;
    s2 = s1 - s2 * q;
    t = t2;
    t2 = t1 - t2 * q;
    t1 = t;
    d = r;
    r = a % b;
    q = a / b;
    a = b;
    b = r;
return d , s2 , t2;
```

3) Chinese Remainder Theorem : foo(a[], m[])

Complexity : $O(n * \log_2(\text{max number of bits (m}_i, M_k)))$

// arrays represents functions of form $X = a_i \pmod{m_i}$

```
M =  $\prod_{k=0}^n m_k$ 
For ( i=0 , i < n )
    M_k = M/m_i
    Y_i =  $M_k^{-1} \% m_i$  // Modular inverse to M_k
    x += (a_i * M_k * Y_i)
return x%M
```

4)Prime Number Generator : foo()

Complexity : $O(n * \log_2(\log_2(n)))$

```
bool b[] = false ;  
for (int i = 2; i * i <= n; i++) //Assuming no primes before 2  
if (!b[i])  
    primes.add(i);  
    for (int j = i * i; j <= n; j += i)  
        b[j] = true;  
return primes[random(primes.size)];
```

D)Assumptions :

- a- In Extended Euclidian Theorem , S will always be multiplied to $\max(a,b)$, T will always be multiplied to $\min(a,b)$.
 - b- Only the CRTOperation function will print on its own , the rest you should print the outcome on yourself in the main
- N.B : Already made some print statements in the main

E)Design decision :

- a- In CRT , I use Extended Euclidean Theorem to get inverse to $(M_i \% m_i)$ using the fact
If $\gcd(a,b) = 1$, then $[1 = a*S + b*T]$ where T would be the inverse of $(b^{-1}) \% a$

F)Sample Runs :

1) Modular Expontiation :

Test was taken from Sheet 4.6 Ex. 25

(not Sure we should use 0003 or 0300 in the last one ,
but the answer is correct XD)

```
2015^17 (% 3233) = 2545 From Iterative
1114^17 (% 3233) = 2757 From Recursive
0003^17 (% 3233) = 1211 From Iterative
```

U P L O A D
20 15 11 14 0 3

The encrypted message is: 2545 2757 1211

2) Extended Euclidean Theorem :

Test was taken from Sheet 4.3 Ex. 42

EGCD(356 , 252)

From Our code :

$$4 = (356*17) + (252*-24)$$

From an Online Calculator : [Link](#) to try yourself

First integer
356

Second integer
252

CALCULATE

Greatest Common Divisor
4

Coefficient for bigger integer
17

Coefficient for smaller integer
-24

3) Chinese Remainder Theorem :

Test was taken from Book Section 4.4 pg 280

(At least the sum , The multiply operation was proved Correct by calculating it by Win Calculator)

```
mods =new long [] {99,98,97,95};  
e.CRTOperation(mods , 123684,413456);
```

```
C = A+B = 123684 + 413456 = 537140 (Mod 89403930)  
D = A*B = 123684 * 413456 = 88247874 (Mod 89403930)
```


413,456 as (32, 92, 42, 16)

To find the sum of 123,684 and 413,456, we work with these 4-tuples instead of these two integers directly. We add the 4-tuples componentwise and reduce each component with respect to the appropriate modulus. This yields

$$\begin{aligned}(33, 8, 9, 89) + (32, 92, 42, 16) \\&= (65 \bmod 99, 100 \bmod 98, 51 \bmod 97, 105 \bmod 95) \\&= (65, 2, 51, 10).\end{aligned}$$

To find the sum, that is, the integer represented by (65, 2, 51, 10), we need to solve the system of congruences

$$\begin{aligned}x &\equiv 65 \pmod{99}, \\x &\equiv 2 \pmod{98}, \\x &\equiv 51 \pmod{97}, \\x &\equiv 10 \pmod{95}.\end{aligned}$$

It can be shown (see Exercise 53) that 537,140 is the unique nonnegative solution of this system less than 89,403,930. Consequently, 537,140 is the sum. Note that it is only when we have to recover the integer represented by (65, 2, 51, 10) that we have to do arithmetic with integers larger than 100. 

4) Prime Generator :

```
Your Random number is 2539  
Your Random number is 3001  
Your Random number is 1553
```

They're Randomly Generated I swear :D