RSA Public-key Cryptosystem

Documentation I

Mahmoud Elsayed Mohammed Nasr Eldeen

2017170554

Section ‘15’

Mahmoud Saeed Mohammed Hasan

2017170395

Section ‘15’

Mahmoud Emad Abdellatif

2017170402

Section ‘15’

Rofayda Abdelmaksoud

2016170532

Section ‘7’

RSA Public-key Cryptosystem

The source code of the big integer with add, sub and multiplication function is self-effort.

In our project we have three main functions for now the Addittion, the Subtraction and the multiplication function, these functions are used for adding, subtracting and multiplying big integers, and here you will find the analysis of each function.

Add function:

```
public string Addtion(string first_number, string second_number)
```

This line initializing the add function and takes two parameters types of string so we can deal with big integers.

```
string tmp = "";
```

In this line the statement stores the addition result.

```
if (first_number.Length > second_number.Length)
```

In this line the "if condition" checks if the number length of the first number is bigger than the length of the second number.

```
for (int i = 0; i < first_number.Length - second_number.Length; i++)
    tmp += '0';
second_number = tmp + second_number;
```

In this lines the "for loop" will iterate while the result of subtraction the second number length form the first number length greater than the zero it will increment the smaller number(2^{nd}). The "tmp" statement will be incremented every time the loop iterate and will be saved, then it will be added to the second number.

The runtime for this peace of code:

```
if (first_number.Length > second_number.Length)
{
    for (int i = 0; i < first_number.Length - second_number.Length; i++)
        tmp += '0';
    second_number = tmp + second_number;
}
```

Is:

T(N) = O(Head) * O(IF Body) SO

IF Body contains for loop

O(Head) = O(1)

O(IF Body) = #Iterations * O(For Body) = N * O(1) = O(N).

THEN T(N) = O(N).

……..

```
else if (first_number.Length < second_number.Length)
```

In this line this "else if" to check if the length of the first number is less than the second.

```
for (int i = 0; i < second_number.Length - first_number.Length; i++)
    tmp += '0';
first_number = tmp + first_number;
```

This loop will iterate while the result of subtracting the first from the second is more than the zero, it smaller number (1st) will increment, the "tmp" statement will increase every time the loop iterate then it will be added and saved to and in the first number.

The runtime for this peace of code:

```
else if (first_number.Length < second_number.Length)
{
    for (int i = 0; i < second_number.Length - first_number.Length; i++)
        tmp += '0';
    first_number = tmp + first_number;
}
```

Is:

T (N) = O(Head) * O(IF Body) SO

IF Body contains for loop

O (Head) = O (1)

(IF Body) = #Iterations * O (For Body) = N * O (1) = O (N).

THEN T (N) = O (N)

……..

```
int[] add_result = new int[first_number.Length + 1];
```

In this line the add result will store the result in array of integers then convert it to string.

……..

```
BigInteger number_1 = new BigInteger(first_number);
BigInteger number_2 = new BigInteger(second_number);
```

In this lines we are Making objects for each number the 1st and the 2nd number.

……..

```
for (int j = first_number.Length - 1; j >= 0; j--)
{
    add_result[j + 1] += number_1.bigint[j] + number_2.bigint[j];
    if (add_result[j + 1] >= 10)
    {
        int mod = add_result[j + 1] % 10;
        int div = add_result[j + 1] / 10;
        add_result[j + 1] = mod;
        add_result[j] = div;
    }
}
```

In this lines the "for loop'' is to iterate on adding the two big integers and checking on them with "if condition" this condition checks if the result of the addition is more than 10, the statements in the "if condition" we are storing the add result mod 10, and the division of the fun "add_result" to be added to next digit of result. To handle the overflow.

And the runtime is:

T (N) = #Iterations * O (Body).

#Iterations = N

O (Body) --> is about statements and If

For statement --> O (1)

For IF O (IF) = O (Head) * O (Body) [all as statements so O (1)]

THEN T (N) = N * O (1) = O (N).

……..

```
string ruslt = String.Join("", add_result.Select(index => index.ToString()).ToArray());
```

This statement converts the array of integers to string then return it.

It works like "for loop" it take all the indexes of int array and concatenate them with the other elements in the array, **it takes O (N).**

……..

```
for (int i = 0; i < ruslt.Length; i++)
{
    if (ruslt[i] != '0')
    {
        return ruslt.Substring(i);
    }
}
return "0";
```

In this lines the “for loop” iterates on the results and checks with “if condition” to make sure that results doesn’t equal to zero.

And the runtime is:

T (N) = #Iterations * O (Body).

#Iterations = N

O (Body) all is about statements so O (1)

THEN T (N) = N * O (1) = O (N).

The range of the Execution time for the add function is (0 milliseconds to 16millisecond).

Sub function:

```
public string Subtraction(string first_number, string second_number)
```

This line initiates the subtraction function which takes two parameters type of a string that takes big integers.

```
string tmp = "";
```

In this line this “statement” stores the subtraction result.

```
if (first_number.Length > second_number.Length)
```

In this line the “if condition” checks if the number length of the first number is bigger than the length of the second number.

```
for (int i = 0; i < first_number.Length - second_number.Length; i++)
    tmp += '0';
second_number = tmp + second_number;
```

This “for loop” will iterate while the result of subtraction the second number length from the first number length is greater than zero, then it will increment the smaller (2^{nd}) number, and will be added to the second number.

The runtime in this peace of code:

```
if (first_number.Length > second_number.Length)
{
    for (int i = 0; i < first_number.Length - second_number.Length; i++)
        tmp += '0';
    second_number = tmp + second_number;
}
```

Is:

T (N) = O (Head) * O (IF Body) SO

IF Body contains for loop

O (Head) = O (1)

O (IF Body) = #Iterations * O (For Body) = N * O (1) = O (N).

So T (N) = O (N).

……..

```
else if (first_number.Length < second_number.Length)
```

In this "else if" condition checks if the number length of the second number is greater than the length of the first number.

```
{
    for (int i = 0; i < second_number.Length - first_number.Length; i++)
        tmp += '0';
    first_number = tmp + first_number;
}
```

This "for loop" will iterate while the result of subtraction the first number length from the second number length is greater than zero, then it will increment the smaller (1^{st}) number, and will be added to the first number.

The runtime in this peace of code :

```
else if (first_number.Length < second_number.Length)
{
    for (int i = 0; i < second_number.Length - first_number.Length; i++)
        tmp += '0';
    first_number = tmp + first_number;
}
```

Is:

T (N) = O (Head) * O (IF Body) SO

IF Body contains for loop

O (Head) = O (1)

O (IF Body) = #Iterations * O (For Body) = N * O (1) = O(N).

So T (N) = O (N).

……..

```
for (int i = 0; i < first_number.Length; i++)
{
    if (first_number[i] < second_number[i])
    {
        tmp = first_number;
        first_number = second_number;
        second_number = tmp;
        break;
    }
    else if (first_number[i] == second_number[i])
    {
        continue;
    }
    else
        break;
}
```

In this lines the "for loop" will iterate while the length of the number is more than zero, and checks with "if condition" if the subtracted number is greater than the other number then swap the two numbers to make sure that the subtracted is less than the other number then continue, else break.

```
int[] sub_result = new int[first_number.Length];
```

In this line sub result will store the result in array of integers then convert it to string.

```
BigInteger number_1 = new BigInteger(first_number);
BigInteger number_2 = new BigInteger(second_number);
```

In this lines we are Making objects for each number the 1st and the 2nd number.

```
for (int i = first_number.Length - 1; i >= 0; i--)
{
    if (number_1.bigint[i] < number_2.bigint[i] && i != 0)
    {
        number_1.bigint[i] = number_1.bigint[i] + 10;
        number_1.bigint[i - 1]--;
    }
    sub_result[i] = number_1.bigint[i] - number_2.bigint[i];
}
```

In this lines the “for loop” iterates while the length of the first number minus one is greater than or equal zero, and checks with “if condition” if the second number is bigger than the first then borrow 10 from next digit of first number, and subtract 1 from the same digit you borrowed from then continue. This for handling the ‘borrow’.

And the runtime is:

T (N) = #Iterations * O (Body).

#Iterations = N

O (Body) all is about statements so O (1)

So T (N) = N * O (1) = O(N).

……

```
string ruslt = String.Join("", sub_result.Select(index => index.ToString()).ToArray());
```

This line converts the array of integers to string.

It works like “for loop” so it take all indexes of int array and concatenate them with the other elements in the array, **it’s also O (N).**

……

```
for (int i = 0; i < ruslt.Length; i++)
{
    if (ruslt[i] != '0')
        return ruslt.Substring(i);
}
return "0";
```

This loop removes the zeros on the left when the result contain them, so the substring here substitute the number two parts part for the zero which neglected and the number part (result).

And the runtime is:

T(N) = #Iterations * O(Body).

#Iterations = N

O(Body) --> all is about statements so O(1)

THEN --> T(N) = N * O(1) = O(N).

The range of the Execution time for the sub function is (0 milliseconds to 15 millisecond)

Multiplication function:

```
public string multiplication(string first_number, string second_number)
```

This line initiates the multiply function which takes two parameters type of string because we are dealing with big integers to multiply them.

```
int cutpos;
```

This line initiates cutpos which stores the results after dividing the number.

```
string tmp = "";
```

In this line this "statement" stores the multiplication result.

```
if (first_number.Length > second_number.Length)
```

In this line the "if condition" checks if the number length of the first number is bigger than the length of the second number.

```
for (int i = 0; i < first_number.Length - second_number.Length; i++)
    tmp += '0';
second_number = tmp + second_number;
```

This "for loop" will iterate while the result of subtraction the second number length from the first number length is greater than zero, then it will increment the smaller (2^{nd}) number, and will be added to the second number.

In this peace of code:

```
if (first_number.Length > second_number.Length)
{
    for (int i = 0; i < first_number.Length - second_number.Length; i++)
        tmp += '0';
    second_number = tmp + second_number;
}
```

The runtime is:

T (N) = O (Head) * O (IF Body) SO

O (Head) = O (1)

O (IF Body) = all is about if statements so O (1).

So T (N) = O (N).

……..

```
else if (first_number.Length < second_number.Length)
```

In this "else if" condition checks if the number length of the second number is greater than the length of the first number.

```
for (int i = 0; i < second_number.Length - first_number.Length; i++)
    tmp += '0';
first_number = tmp + first_number;
```

This "for loop" will iterate while the result of subtraction the first number length from the second number length is greater than zero, then it will increment the smaller (1st) number, and will be added to the first number.

In this peace of code:

```
else if (first_number.Length < second_number.Length)
{
    for (int i = 0; i < second_number.Length - first_number.Length; i++)
        tmp += '0';
    first_number = tmp + first_number;
}
```

The runtime is:

T (N) = O (Else Body) SO

Else body contain Min fun which compare between two numbers and return the minimum one so its' O (1)

O (Else Body) = O (1)

So T (N) = O (N).

……..

```
if (first_number.Length == 1 && second_number.Length == 1)

 return (Int64.Parse(first_number[0].ToString()) * Int64.Parse(second_number[0].ToString())).ToString();
```

This "if condition" checks if the lengths of the numbers are equally and equals to one, it will return the array converted to string, its used as a base case to get out of the recursion function.

```
else
```

In the “else” kartsuba algorithm will be applied.

```
if (first_number.Length % 2 == 0)
{
    cutpos = first_number.Length / 2;
}
else
{
    cutpos = first_number.Length / 2 + 1;
}
```

In this “if and else” conditions we are handling the numbers to make it suitable for Kartsuba algorithm, if the number is even it must be substituted to two equally lengths, else if the number is odd it must be substituted to two equally lengths plus 1, then it will be useable.

The runtime is:

T(N) = O(Else Body) SO

O(Else Body) = O(1)

THEN

T(N) = O(N).

The range of the Execution time for the multiplication function is (100 milliseconds to 125 millisecond).