METODOLOGIA DE LA PROGRAMACIÓ

Eric Pi Esteban

David Domènech Vilà

2015-2016

1-Especificació de l'algorisme

En aquesta pràctica ens demanen el disseny i la implementació d'un algorisme per poder desxifrar la clau d'un sistema de criptografia de clau pública.

L'usuari disposa de dues claus, privada i pública. Amb aquest algorisme l'usuari podrà desxifrar els missatges.

Per fer-ho utilitzarem exponenciacions modular, de la següent forma: x^y mod N, on x, y i N usualment són nombres molt grans.

2- Algorisme recursiu lineal

a. Especificació de la funció

El resultat final de la funció ha de ser el mòdul de dos números. El problema es que aquest números seran molt grans, i per això no es pot calcular el número directament, hem de buscar alguna manera de anar disminuint el número cada cop que cridem a la funció.

Observem que haurem de crear dos casos depenen si el número Y (numero elevat) es parell o imparell.

b. Analisis per casos

En aquest apartat analitzarem els diferents casos que posseeix la funció.

Cas directe Y=0 ➔F:=1

Cas recursiu Y>0 ➔Si Y % 2 = 0 ➔F:=(mod (X, Y , N)^2) % N

Si Y % 2 ≠ 0 ➔F:= (X%N) * mod (X,Y-,N) % N

c. Construcció de l'algorisme

Després de acabar el disseny lineal ja podem implementar la funció recursiva lineal.

La funció lineal recursiva es la següent:

Funció mod(X, Y, N:nat) retorna F:nat

{Pre: (X>=) i (y>=0)}

[

Y = 0 ➔F:= 1

Y > 0 ➔[

Si Y % 2 = 0 ➔F:=(mod (X, Y , N)^2) % N

Si Y % 2 ≠ 0 ➔F:= (X%N) * mod (X,Y-,N) % N

]

{Post: F=X^Y % N }

d. Verificació

La verificació d'aquest algorisme ens servirà per comprovar que el disseny del algorisme es correcte.

0-Analisis per a tots els casos:

Q(x) ➔d(x) U ¬d(x)

X>O ➔Y=0 U Y>O

1-Verificació del cas directe

Q(x) /\ d(x) ➔R(X,h(x))

x>0 /\ Y=0 ➔1=X^Y % N

1=X^0 % N

1=1 % N

1=1

2-Verificació del cas recursiu

-Comprovació de la pre-condició

Q(x) /\ ¬d(x) ➔Q(s(x))

X>0 /\ Y>0 ➔X>0

-Pas d'inducció correcte

Q(X) /\ ¬d(x) /\ R(x,f(x)) ➔R(x,c(x,v))

X>0 /\ Y>0 /\ F=X^Y/2 % N ➔F= X^Y % N

-Si Y % N = 0

F^2 % N = X^Y % N

(X^Y/2 % N) ^2 % N = X^Y % N

(X^Y/2 ^2 % N) % N = X^Y % N

(X^2Y/2 % N) % N = X^Y % N

(X^Y % N) % N = X^Y % N

X^Y % N = X^Y % N

-Si Y % N ≠ 0

((X % N) * (X^Y-1 % N)) % N = X^Y % N

(X*X^Y-1 % N) % N = X^Y % N

(X^Y % N) % N = X^Y % N

X^Y % N = X^Y % N

3-Funció cota (demostrar que la funció es decrementa)

T(s(x)) < t(x)

Y/2 < Y

Y-1 < Y

e. Estudi del cost de l'algorisme

```
int recursiu_lineal (x,y,N)

{
   long int f,funcio,funcio2;
   funcio2=0;
   funcio=0;
   f=0;

   if (y==0)              O(1)
   {
     f=1;
      }
     else if (y>0)
   {

      if(y%2==0)           O(n)
      {
         funcio=(pow (x,(y/2)));

         funcio= funcio%N;

         funcio=pow(funcio%N,2);

         funcio=funcio%N;

         f=funcio;

      }
      else {         O(n)

      funcio=x%N;

         funcio2=(pow(x,y-1));

         funcio2=funcio2%N;

         funcio=(funcio*funcio2)%N;

         f=funcio;

      }
   }
   return f;
```

REGLA DE LA SUMA max(O(n), O(n))= O(n)

3- Algorisme recursiu final

a) Transformació recursiu lineal a final mitjançant desplegat-plegat

Utilitzarem a funció Immersió per aconseguit transformar de recursiu lineal a final.

Si Y % 2 = 0 → [mod(X, Y/2, N)^2] % N

Si Y % 2 ≠ 0 → [(X%N) * mod(X, Y-1, N)] % N

Funció mod(X, Y, Z, T, H, I) ret g:nat

[

T=0 → g:=1

T>0 → [

T mod 2 = 0 → g_mod(E1, E2, E3, E4, E5, E6)

T mod 2 ≠ 0 → g_mod(E7, E8, E9, E10, E11, E12)

]

Return g;

A continuació aplicarem la tècnica del plegat-desplegat:

Si t mod 2 = 0

g_mod(E1, E2, E3, E4, E5, E6) = (E1 * mod (E2, E3, E4) ^ E5) % E6

DESPLEGAT → [(E1 % E4) * mod(E2, E3/2, E4)^2 ^E5)] % E6

PLEGAT → [(E1% E4) * mod(E2, E3/2, E4)**2E5] % E6

G_mod(E1, E2, E3, E4, E5, E6)=[(X % T) mod(Y, Z/2, T) **2H] % N

Si t mod 2 ≠ 0

g_mod(E7, E8, E9, E10, E11, E12)= (E7 * mod (E8, E9, E10) ^ E11) % E12)

DESPLEGAT➔(E7 % E) *(E7 % E8)mod(E8, E9-1, E10) * 2 * E11)]% N]

PLEGAT ➔(E7 % E)^2 * mod(E8, E9-1, E10) * 2E11)] % N

G_mod(E7, E8, E9, E10, E11, E12)=[(X % T)^2*mod(Y,Z-1,T) ^2H]% I

B) Codi de la funció recursiva final

FUNCIÓ RECURSIVA FINAL:

```
Funció mod(  X, Y, Z, T, I: nat) ret g:nat
        {Pre: Y>=0}
        [
         Y = 0 →G:=1
         Y > 0 →[
                        Y % 2 = 0 →<X, Y, Z, T, H, I> := ( X%T, Y, Z/2, T, 2H, I)
                        Y % 2 = 0  →<X, Y, Z, T, H, I> := ( (X%T)^2, Y, Z-1, T, 2H, I)
                    ]
         ]
        {Post:  g  =Y^Z-1 % T )
        Return g;
```

4- Algorisme iteratiu

A) <u>Transformació de recursiu final a iteratiu</u>

La funció iterativa es la següent:

Funció g_mod_it (X, Y, Z, T, H, I) ret g_it

{Pre: Y>=0}

[

Y>0 ➔[

Si Y % 2 = 0 ➔<(X, Y, Z, T, H, I)> := (X%T, Y, Z/2, T, 2H, I)

Si Y % 2 ≠ 0 ➔<(X, Y, Z, T, H, I)> := ((X%T)^2, Y, Z-1, T, 2H, I)

]

]

G_IT:=1

{Pre: ((G_it= Y^Z % T) i (Y^Z % T < T)) }

Return g_it;

B) Estudi del cos de l'algorisme iteratiu

```
int recursiu_iteratiu (x,y,N)

{
   long int f,funcio,maxim,cont,funcio2,z;
   funcio2=1;
   funcio=1;
   f=0;
   z=0;
   cont=0;

   maxim=pow(2,10);

   if (y==0)
   {

      f=1;          O(1)

   }

   else if (y>0)
         {

            if(y%2==0)
               {
                  while(cont<y){     Regla del producte: O(n*n)=O(n 2 )

                  funcio=funcio*x;

                  if (funcio>N){

                  funcio=funcio%N;

                  }
                  cont++;
                  }


                  f=funcio;
               }
            else {                        O(n)
```

```
        x=x%N;
        y=y-1;

        z=pow(x,y);

        z=z%N;

        x=x*z;

        funcio=x%N;

        f=funcio;

    }
}
```

regla de la suma max(O(n^2), O(n))=O(n^2)

5-Implementació en Java dels codis dissenyats en els apartats anteriors

```
funció recursiu_lineal (x,y,N) es
f,funcio,funcio2:enter;
funcio2:=0;
funcio:=0;
f:=0;
si (y=0) llavors
      f:=1;
sino si (y>0) llavors
        si(y%2=0) llavors
                funcio:=(elevar (x,(y/2)));
                funcio:= funcio%N;
                funcio:=elevar(funcio,2);
                funcio:=funcio%N;
                f:=funcio;
     sino {
        funcio:=x%N;
        funcio2=(elevar(x,y-1));
        funcio2=funcio2%N;
        funcio=(funcio*funcio2)%N;
        f=funcio;
     fsi
  fsi
Return f;
```

```
ffuncio
```

Algorisme recursiu_lineal es

```
x,y,N, f:enter;
escriure("Introdueix el valor de x");
llegir(x);
escriure("Introdueix el valor de y");
llegir(y);
escriure("Introdueix el valor de N");
llegir(N);
escriure("Els valors de x y z:",x,y,N);
f:=recursiu_lineal(x,y,N);
escriure("El resultat final es:”,f);
falgorisme
```

```
Algorisme recursiu_final es
 x,y,N,resultat:enter;
 escriure("Introdueix el valor de x”);
 llegir(x);
 escriure("Introdueix el valor de y");
 llegir("%d",&y);
 escriure("Introdueix el valor de N\n");
 llegir("%d",&N);
 escriure("Els valors de x y z: %d %d %d \n",x,y,N);
 resultat=recursiu_final(x,y,N);
 escriure("El resultat final es:",resultat);
 falgorisme

 funcio recursiu_final(int x,int y,int N) es

 retorna recursiu_final_imm(x,y,N,1);
 ffuncio

 funcio recursiu_final_imm(int x,int y,int N,int resultat) es
 int f,funcio,funcio2,Y;
 funcio2=0;
 funcio=0;
 f=0;
 Y=y;
 si (y=0) llavors
 resultat=1;
 fsi
```

```
sino si (y>0) llavors
        si(y%2=0) llavors
                resultat:=(x%N);
                Y:=y/2;
                funcio2:=elevar(x,Y);
                funcio2:=funcio2%N;
                resultat:=resultat*funcio2;
                resultat:=resultat%N;
        sino
                funcio=x%N;
                funcio2=(pow(x,Y-1));
                funcio2=funcio2%N;

                resultat=(funcio*funcio2)%N;
                resultat=resultat%N;
        fsi
fsi
retorna resultat;
ffuncio
```

```
funcio recursiu  iteratiu (x,y,N) es
f,funcio,maxim,cont,funcio2,z:enter;
funcio2:=1;
funcio:=1;
f=0;
z:=0;
cont:=0;
maxim:=elevar(2,10);
si (y=0) llavors
        f:=1;
sino si (y>0) llavors
        si (y%2=0)
                mentre(cont<y) fer
                        funcio=funcio*x;

                si (funcio>N) llavors
                                funcio=funcio%N;
                        fsi
                cont++;
                        f=funcio;
                fmentre
           sino
                        x=x%N;
                y=y-1;
                         z=pow(x,y);
                        z=z%N;
                        x=x*z;
```

```
                    funcio=x%N;
                    f=funcio;
                fsi
fsi
retorna f;

Algorisme recursiu_iteratiu es
x,y,N, f:enters;
escriure("Introdueix el valor de x\n");
llegir(x);
escriure("Introdueix el valor de y\n");
llegir(y);
escriure("Introdueix el valor de N\n");
llegir(N);
escriure("Els valors de x y z:”,x,y,N);
f:=recursiu_iteratiu(x,y,N);
escriure("El resultat final es",f);
falgorisme
```

6-Joc de proves

L'objectiu del joc de proves es assegurar que els algorismes realitzats compleixen la seva funció i no contenen errors. Per tant haurem de provar que en les diferentes opcions que presenta l'algorisme no hi hagi respostes errònies.

Algorisme recursiu lineal

Casos	Valor de X	Valor de Y	valor de N	Resultat esperat?	Resultat obtingut
1	2	10	5	4	4
2	3	10	5	4	4
3	4	10	5	1	1
4	2	11	13	7	7
5	3	11	13	9	9
6	4	11	13	10	10
7	2	12	23	2	2
8	3	12	23	3	3
9	4	12	23	4	4
10	2	0	11	1	1
11	11	0	23	1	1
12	23	0	10	1	1

Algorisme recursiu iteratiu

Casos	Valor de X	Valor de Y	valor de N	Resultat esperat?	Resultat obtingut
1	2	10	5	4	4
2	3	10	5	4	4
3	4	10	5	1	1
4	2	11	13	7	7
5	3	11	13	9	9
6	4	11	13	10	10
7	2	12	23	2	2
8	3	12	23	3	3
9	4	12	23	4	4
10	2	0	11	1	1
11	11	0	23	1	1
12	23	0	10	1	1

Algorisme recursiu final

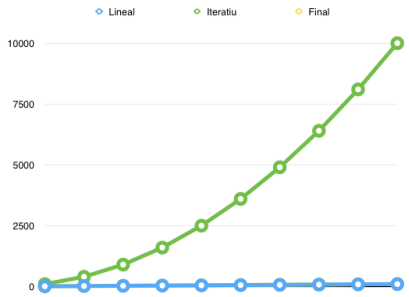
Casos	Valor de X	Valor de Y	valor de N	Resultat esperat?	Resultat obtingut
1	2	10	5	4	4
2	3	10	5	4	4
3	4	10	5	1	1
4	2	11	13	7	7
5	3	11	13	9	9
6	4	11	13	10	10
7	2	12	23	2	13
8	3	12	23	3	2
9	4	12	23	4	8
10	2	0	11	1	1
11	11	0	23	1	1
12	23	0	10	1	1

7-Estudi empíric del cost

a). Gràfica que representi el cost obtingut per diferentsexecucions del programa amb diferents jocs de dades.

taula de costos

Lineal	10	20	30	40	50	60	70	80	90	100
Final	10	20	30	40	50	60	70	80	90	100
Iteratiu	100	400	900	1600	2500	3600	4900	6400	8100	10000



b).Comentari de les dades i dels resultats obtinguts.

L'algorisme iteratiu serà el més costós, pero també el més ràpid en executar-se. Tant l'algorisme final com l'algorisme lineal tindran el mateix cost.