

Nombre d'itérations nécessaires pour retrouver
l'image initiale
dans la transformation de photomaton
version 2

manuel.luque27@gmail.com

23 juillet 2019

Dans l'article "*Images brouillées, images retrouvées*" du magazine "*Pour la science*", N° 242, de décembre 1997, Jean-Paul Delahaye donne la méthode permettant de déterminer le temps de retour, c'est à dire le nombre d'itérations nécessaires pour retrouver l'image initiale.

Si l'image est un rectangle de dimensions $2m \times 2n$ pixels, le temps de retour est le plus petit commun multiple :

- du plus petit p tel que $2^p - 1$ est divisible par $2m - 1$
- et du plus petit q tel que $2^q - 1$ est divisible par $2n - 1$.

Cette méthode utilise les packages xintgcd, fp et xfp. 3 autres versions sont proposées.

- avec les packages xlop, fp et xfp
- calcul en C
- calcul en postscript.

La macro s'écrit : `\tempsretour{m}{n}`, par exemple `\tempsretour{150}{128}` :
nombre d'itérations = 264

Version en C :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int gcd(int x, int y);

int lcm(int x, int y);
```

```

int iter(int m);

int main(int argc, char*argv[]){
if (argc<3) return EXIT_FAILURE;
int x=atoi(argv[1]);
int y=atoi(argv[2]);
    printf("\n %d\n", lcm(iter(x), iter(y)));
return EXIT_SUCCESS;
}

int gcd(int x, int y) {
return y == 0 ? x : gcd(y, x % y);
}

int lcm(int x, int y){
return x * y / gcd(x, y);
}

int iter(int m) {
int dpn=4%(2*m-1);
int n=2;
while(dpn !=1) {
++n;
dpn=2*dpn%(2*m-1);
}
return n;
}

```

La version en postscript :

```

% plus grand commun diviseur
% https://groups.google.com/forum/#!forum/comp.lang.postscript
/gcd {
    % given m, n on the stack
    dup 3 1 roll mod % leaves n, m mod n on the stack
    dup 0 eq {
        pop % leaves n on the stack when done
    } {
        gcd % find gcd of the new pair
    } ifelse
} bind def

%250 360 gcd =
% plus petit commun multiple
/lcm {
/n exch def /m exch def
n m mul
n m gcd

```

```

div floor cvi
} def

% l'image a pour dimensions 2m*2n
% plus petit naturel q tel que
% 2^q-1 est divisible par 2m-1

/ppn {
2 dict begin
/m exch def
/dpn 4 2 m mul 1 sub cvi mod def
/n 2 def
{
dpn 1 ne {/n n 1 add def /dpn 2 dpn mul cvi 2 m mul 1 sub cvi mod def}{exit} ifelse
} loop
n
end
} def
% exemple : 50 ppn =
/tempsretour {
2 dict begin
/n exch def
/m exch def
m ppn
n ppn
lcm
end
} def

% m n tempsretour
% m et n entiers
150 128 tempsretour =

```