Un algorithme est dit <u>récursif</u> s'il s'appelle lui même

<u>Exemple</u> :

$$n! = \prod_{i=1}^{n} i = 1 \times 2 \times \dots \times (n-1) \times n$$
$$= n \times (n-1)!$$

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n \times (n-1)! & \text{sinon} \end{cases}$$

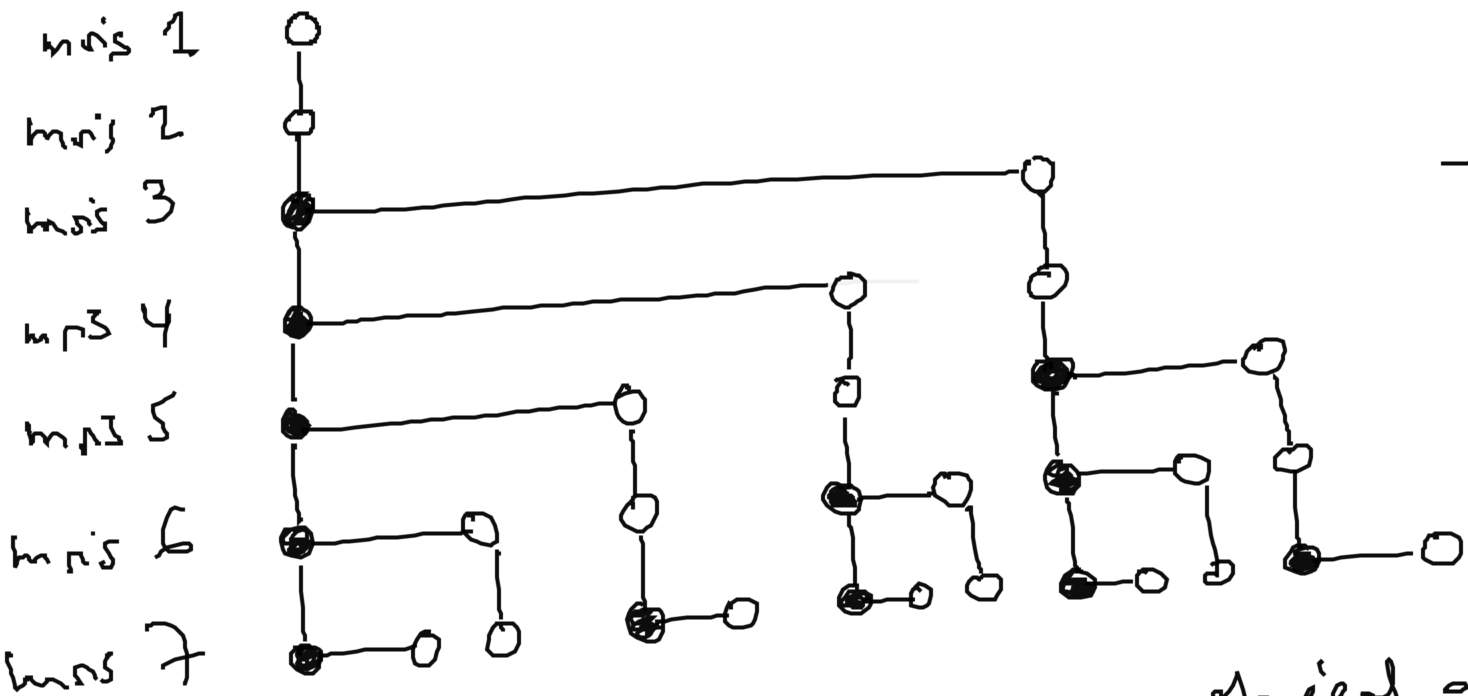
```
int factriel (int n) {
    if (n==0) || (n==1)
        return 1;
    else
        return n* factoriel (n-1);
}
```

<u>n=5</u>

$$\overbrace{\text{factriel}(5)}^{120} \;\to\; 5 * \overbrace{\underbrace{\text{factoriel}(4)}}^{24} \;\to\; 4 * \overbrace{\underbrace{\text{factoriel}(3)}}^{6} \;\to\; 3 * \overbrace{\underbrace{\text{factoriel}(2)}}^{2} \;\to\; 2 * \overbrace{\text{factoriel}(1)}^{1}$$

Séquence 2

Ex 2.1



○ : couple de lapin jeune

● : couple de lapin adulte

— : de croissance

13 couple

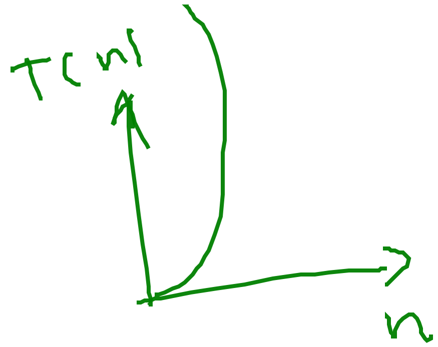
le nbr de couple de lapins au mois n est égal au nbr de couple au mois (n-1) auquel on ajoute le nbr de couple au mois (n-2)

U_n = nbr de couples au mois n

$$U_n = \begin{cases} U_0 = U_1 = 1 \\ U_{n-1} + U_{n-2} \end{cases}$$

c'est la suite de Fibonacci

Ex 2.2.



a)

```
int C (int n, int k) {
    if (k==0) || (k==n)
        return 1;
    else if (k>n)
        return 0;
    else
        return C(n-1, k-1) + C(n-1, k);
}
```

b)

$n = 4$, $k = 2$

C(4,2) = 6

return C(3,1) + C(3,2)

$\boxed{O(2^n)}$

3: C(3,1)
return C(2,0) + C(2,1)

1: C(2,0)
return 1;

2: C(2,1)
return C(1,0) + C(1,1)

1: C(1,0) 1: C(1,1)

3: C(3,2)
return C(2,1) + C(2,2)

2: C(2,1)
return C(1,0) + C(1,1)

1: C(1,0) 1: C(1,1)

1: C(2,2)
return 1

Séquence 3

Ex 3.1

a/ H (N, Source, dest, temp)

si (N = 1) ALORS

Déplacer le plateau de la source à la dest.

Sinon

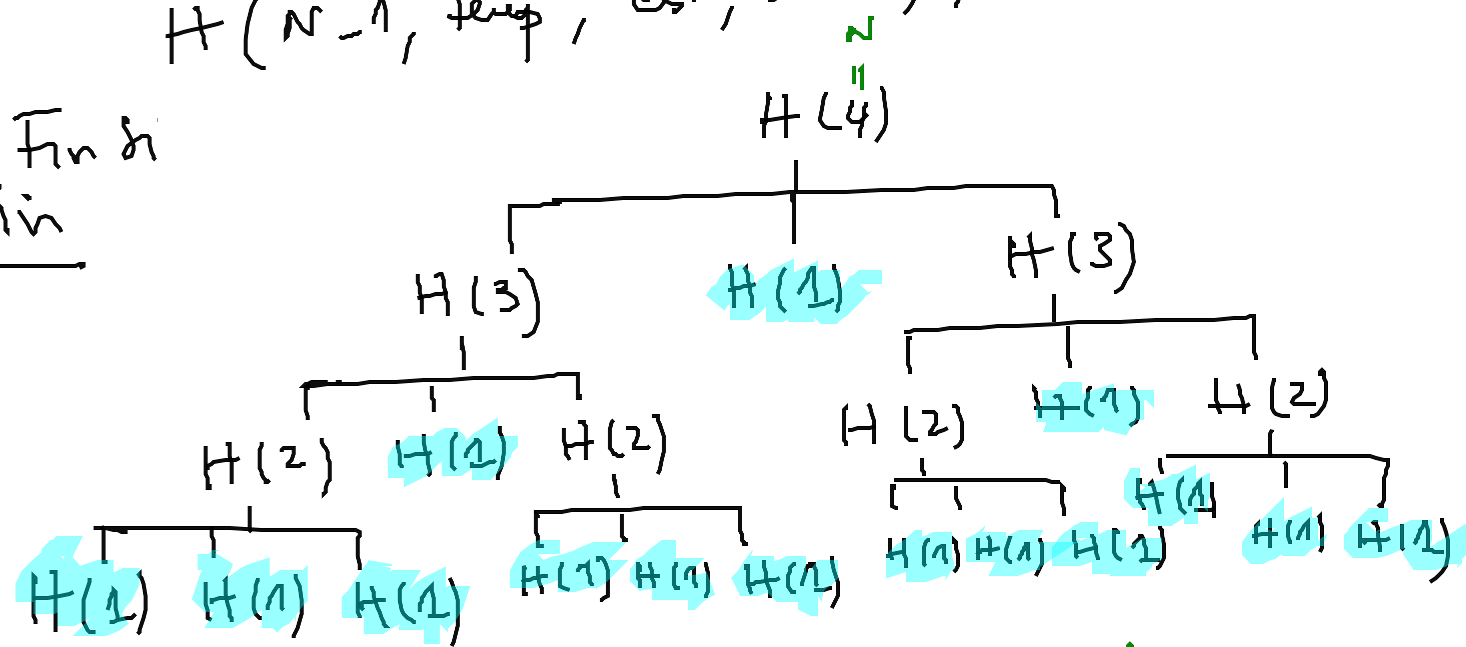
H (N-1, source, temp, dest) ;

H (1, Source, Dest, temp) ;

H (N-1, temp, dest, source) ;

Fin si

Fin



⟹ 15 déplacements pour H(4).

c°/ pour 4 plateaux il faut $15 = 2^4 - 1$ déplacements
il faut $2^n - 1$ déplacements pour n plateaux

d°/ $D(n) =$ nbre de déplacement nécessaire pour n plateaux

- $D(1) = 2^1 - 1 = 1$ vraie
- On suppose que la relation est vraie pour n
  $D(n) = 2^n - 1$
- Montrons que la relation est vraie pour $(n+1)$

$$D(n+1) = D(n) + D(1) + D(n)$$
$$= 2 \cdot \underbrace{D(n)} + 1$$
$$= 2 \times (2^n - 1) + 1$$
$$= 2^{n+1} - 2 + 1$$
$$\boxed{D(n+1) = 2^{n+1} - 1}$$

$\underline{O(2^n)}$

$\longrightarrow$ cette relation est vraie $\forall n$

Ex 3.2

A

1	2	3	4	5	6	7	8	9
15	21	35	42	53	76	119	133	150

Value = 76

First last

BinSearch(A, 1, 9, 76)

↳ BinSearch(A, 6, 9, 76)

↳ BinSearch(A, 6, 6, 76)

↳ 6

$\frac{1+9}{2} = 5$

$\frac{6+9}{2} = 7$

$\frac{6+6}{2} = 6$

```
int BinSearch (int A[], int first, int last, int value) {
    int milieu = (First + last) / 2 ;
    if (last == first) && (A[First] != value)
        return -1 ;

    else {

    if (A[milieu] == value)
        return milieu ;
    else if (A[milieu] > value)
        return BinSearch (A, first, milieu - 1, value);
    else /* A[milieu] < value */
        return BinSearch (A, milieu + 1, last, value);

    }
}
```

$O(2^n)$

1	2	3	4	5	6	7	8	9
15	21	35	42	53	76	118	133	138

last = First

value ≠ A[First]

Value = 144

⇒ return (-1)