

# Pouvoir d'expression des équations récurrentes

- Les tours de Hanoi
- La suite de Fibonacci
- Le triangle de Pascal
- Equations récurrentes et programmes

# Les tours de Hanoi

Pour déplacer une tour de  $n + 1$  éléments,

# Les tours de Hanoi

Pour déplacer une tour de  $n + 1$  éléments,  
on déplace les  $n$  premiers sur le plot intermédiaire

# Les tours de Hanoi

Pour déplacer une tour de  $n + 1$  éléments,  
on déplace les  $n$  premiers sur le plot intermédiaire  
on pose le dernier sur le plot final,

# Les tours de Hanoi

Pour déplacer une tour de  $n + 1$  éléments,  
on déplace les  $n$  premiers sur le plot intermédiaire  
on pose le dernier sur le plot final,  
et on déplace les  $n$  premiers sur celui-là

# Les tours de Hanoi

Pour déplacer une tour de  $n + 1$  éléments,  
on déplace les  $n$  premiers sur le plot intermédiaire  
on pose le dernier sur le plot final,  
et on déplace les  $n$  premiers sur celui-là

$$H_{n+1} = 2H_n + 1$$
$$H_1 = 1$$

# La suite de Fibonacci

$$f(n + 2) = f(n + 1) + f(n)$$

$$f(0) = 1$$

$$f(1) = 1$$

# Le triangle de Pascal

Pour faire une combinaison de  $p + 1$  objets parmi  $n + 1$ ,

# Le triangle de Pascal

Pour faire une combinaison de  $p + 1$  objets parmi  $n + 1$ ,  
on prend une combinaison de  $p$  objets parmi  $n$  et on y ajoute un nouvel objet,

# Le triangle de Pascal

Pour faire une combinaison de  $p + 1$  objets parmi  $n + 1$ ,  
on prend une combinaison de  $p$  objets parmi  $n$  et on y ajoute un nouvel objet,  
ou on prend une combinaison de  $p + 1$  objets parmi  $n$  et on n'ajoute pas le nouvel objet :

# Le triangle de Pascal

Pour faire une combinaison de  $p + 1$  objets parmi  $n + 1$ ,  
on prend une combinaison de  $p$  objets parmi  $n$  et on y ajoute un nouvel objet,  
ou on prend une combinaison de  $p + 1$  objets parmi  $n$  et on n'ajoute pas le nouvel objet :

$$C_{n+1}^{p+1} = C_n^p + C_n^{p+1}$$

# Equations récurrentes et programmes

Forme canonique d'un système d'équations récurrentes :

$$\begin{aligned} X(-1) & \\ X(n+1) &= F(X(n), U(n+1)) \\ Y(n+1) &= G(X(n), U(n+1)) \end{aligned}$$

avec :

- $X$ , état du système
- $U$ , entrées
- $Y$ , sorties
- $F$ , fonction de transition,
- $G$ , fonction d'observation,

# Programme associé

```
class system {  
private :  
  
    state x ;  
  
public :  
  
    system(state xinit) {x = xinit; }  
  
    output compute(input u) {  
        output y = g(x, u);  
        x = f(x, u);  
        return y;  
    }  
}
```

## Exemple : programmer Hanoi

```
class hanoi {
private :

    int val ;

public :

    hanoi() { val = 0 ; }

    void reset() { val = 0 ; }

    int compute() {
        val = 2 * val + 1;
        return val;
    }
};
```

## Exemple : exécuter Hanoi

```
#include <stdio.h>
#include ``Hanoi.hh``

main () {
    hanoi h;
    int n = 0 ;

    while (getchar() == '\n')
        printf ("hanoi( %d ) = %d \n", ++n, h.compute());
}
```

# Résultats

```
rateau% ./hanoi
```

```
hanoi( 1 ) = 1
```

```
hanoi( 2 ) = 3
```

```
hanoi( 3 ) = 7
```

```
hanoi( 4 ) = 15
```

```
hanoi( 5 ) = 31
```

```
hanoi( 6 ) = 63
```

```
hanoi( 7 ) = 127
```

```
q
```

```
rateau%
```

# Résultats

Systemes d'équations récurrentes

$\approx$

Programmes objets