Récursivité

I3 - Algorithmique et programmation

Nicolas Delestre

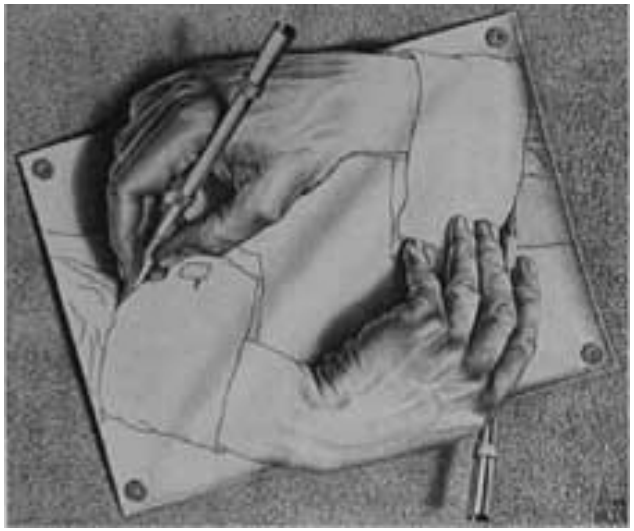
Plan

Architecture des Systèmes d'Information

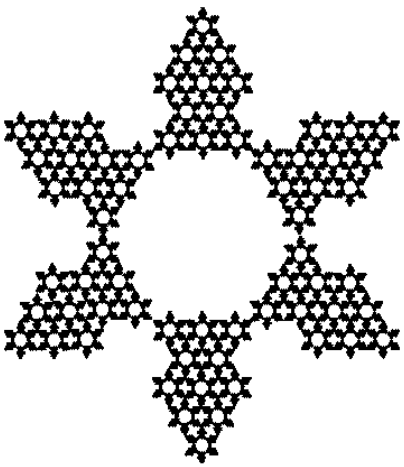
Introduction

Définition

Une entité est récursive lorsqu'on l'utilise pour la définir



Drawing Hands, Escher (1948)



http://www.russie.net/russie/art_matriochka.htm

Exemples 1 / 2

Factorielle

$$\begin{cases} 0! = 1! = 1 \\ n! = n(n-1)! \end{cases}$$

Suite de fibonacci

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(n) = F(n-1) + F(n-2), n > 1 \end{cases}$$

Poupée russe

Une poupée russe est

- une poupée “pleine”
- une poupée “vide” contenant une poupée russe

Exemples 2 / 2

Factorielle

```
fonction fact (n : Naturel) : Naturel
debut
    si n=0 ou n=1 alors
        retourner 1
    sinon
        retourner n*fact(n-1)
    finsi
fin
```

Récursivité terminale

Définition

L'appel récursif est la dernière instruction et elle est isolée

plus(a,b)

```
fonction plus (a,b : Naturel) : natuel
debut
    si b=0 alors
        retourner a
    sinon
        retourner plus(a+1,b-1)
    finsi
fin
```

plus(4,2)=plus(5,1)=plus(6,0)=6

Récursivité non terminale

Définition

L'appel récursif n'est pas la dernière instruction et/ou elle n'est pas isolée (fait partie d'une expression)

plus(a,b)

```
fonction plus (a,b : Naturel) : natuel
debut
    si b=0 alors
        retourner a
    sinon
        retourner 1+plus(a,b-1)
    finsi
fin
```

plus(4,2)=1+plus(4,1)=1+1+plus(4,0)=1+1+4=6

Méthode

Pour écrire un algorithme récursif il faut analyser le problème pour :

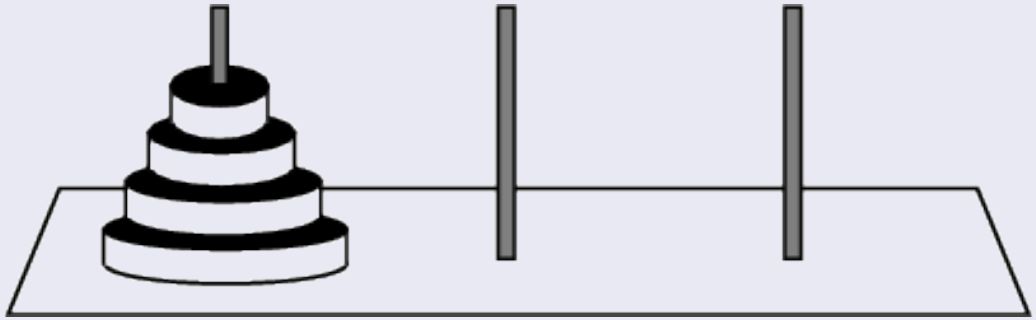
- identifer le ou les cas particuliers
- identifier le cas général qui effectue la récursion

Surtout

Lorsque l'on écrit un algorithme récursif, lors de l'appel récursif, on se positionne en tant qu'utilisateur de l'algorithme : on considère donc que le problème est résolu

Les tours de Hanoï 1 / 4

Présentation



Les tours de hanoï est un jeu solitaire dont l'objectif est de déplacer les disques qui se trouvent sur une tour (par exemple ici la première tour, celle la plus à gauche) vers une autre tour (par exemple la dernière, celle la plus à droite) en suivant les règles suivantes :

- on ne peut déplacer que le disque se trouvant au sommet d'une tour ;
- on ne peut déplacer qu'un seul disque à la fois ;
- un disque ne peut pas être posé sur un disque plus petit.

Les tours de Hanoï 2 / 4

Opérations disponibles

procédure dépilerTour **(E/S** t : TourDeHanoi,**S** d : Disque**)**
procédure empilerTour **(E/S** t : TourDeHanoi,**E** d : Disque**)**

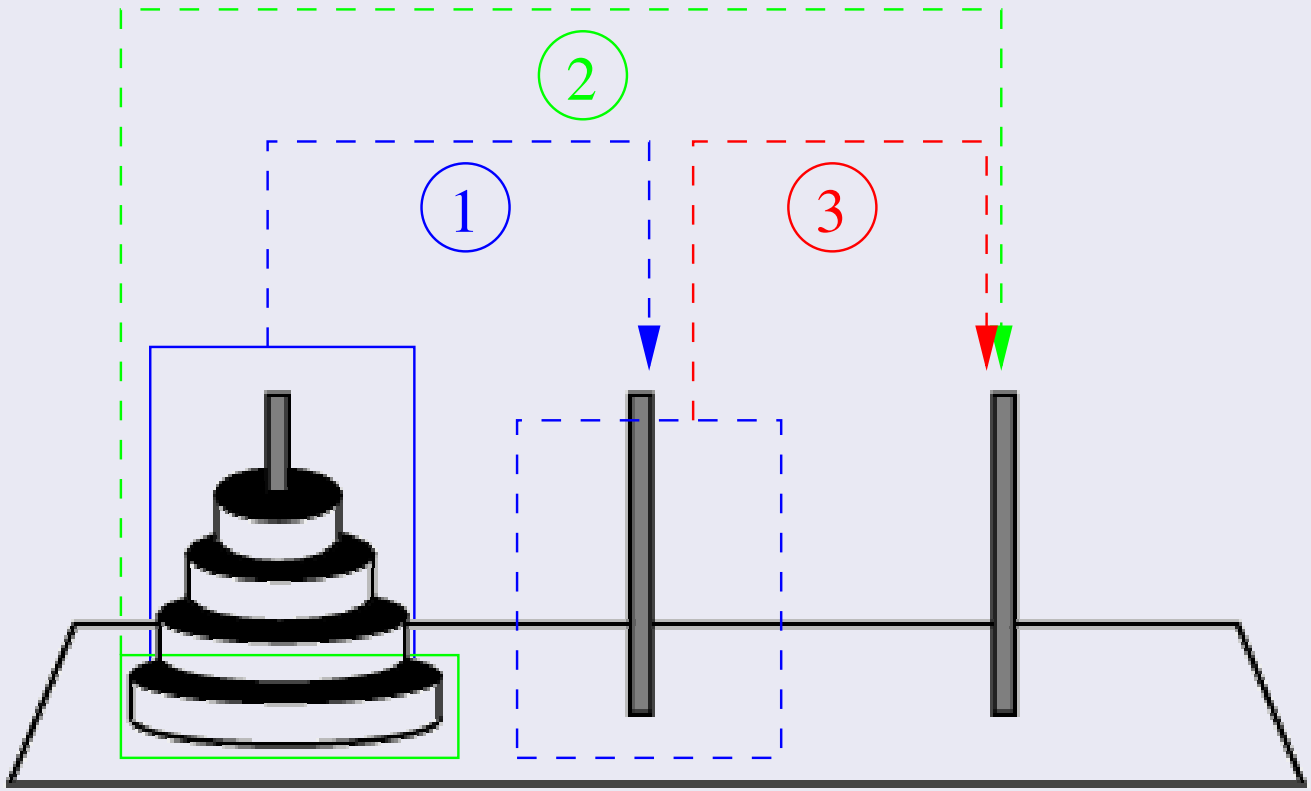
Objectif

procédure resoudreToursDeHanoi **(E** nbDisquesADeplacer : **Naturel**,
E/S source, destination, intermediaire : TourDeHanoi**)**

Architecture des Systèmes d'Information

Les tours de Hanoï 3 / 4

Analyse du problème



Les tours de Hanoï 4 / 4

Solution

procédure resoudreToursDeHanoi **(E** nbDisquesADeplacer : **Naturel**, **E/S** source, destination, intermediaire : TourDeHanoi**)**

Déclaration d : Disque

debut

si nbDisquesADeplacer>0 **alors**

resoudreToursDeHanoi(nbDisquesADeplacer-1, source, intermediaire, destination)

depiler(source,d)

empiler(destination,d)

resoudreToursDeHanoi(nbDisquesADeplacer-1, intermediaire, destination, source)

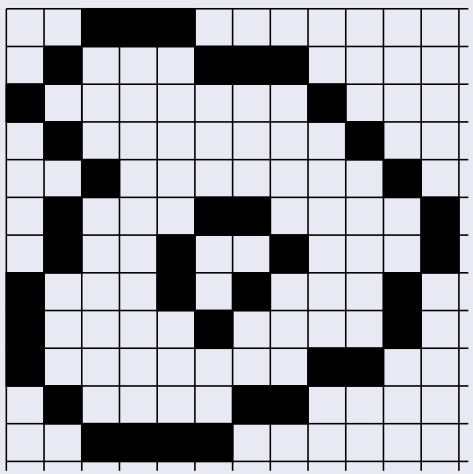
finsi

fin

Remplir une zone graphique 1 / 5

Présentation

- Un écran graphique est un quadrillage
- Chaque intersection de ce quadrillage est un `pixel` qui peut être colorisé



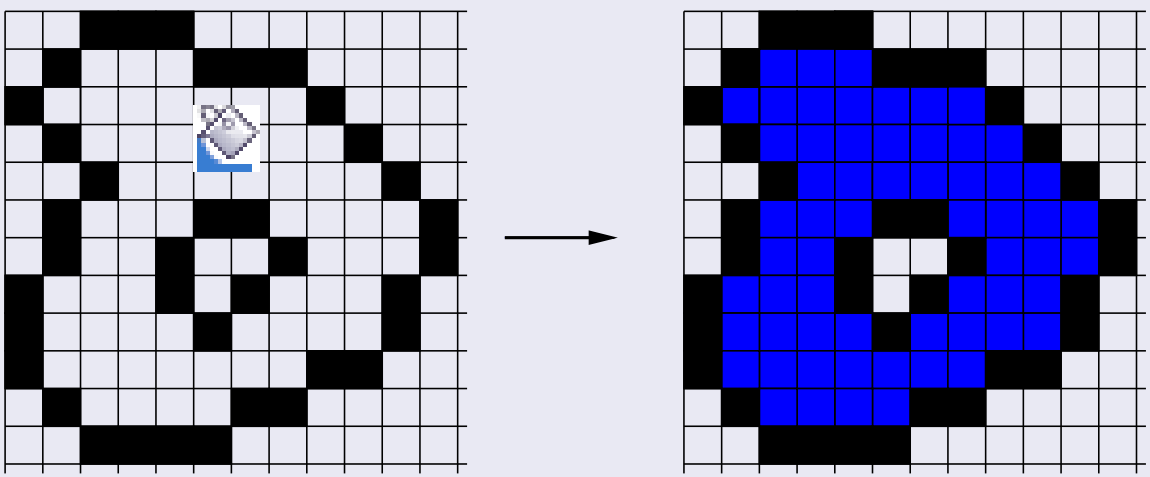
Opérations disponibles

- **procédure** fixerCouleurPixel **(E/S** e : Ecran,**E** x,y : **Naturel**, c : Couleur**)**
- **fonction** obtenirCouleurPixel **(**e : Ecran,x,y : **Naturel) :** Couleur

Remplir une zone graphique 2 / 5

Objectif

- Proposer le corps de la procédure suivante qui permet de remplir une zone
  - **procédure** remplir **(E/S** e : Ecran,**E** x,y : **Naturel**, ancienneCouleur, nouvelleCouleur : Couleur**)**



Remplir une zone graphique 3 / 5

Analyse du problème

- Remplir une zone consiste à **changer** la couleur de certains pixels en commençant par celui qui est donné :
  - Si le pixel de coordonnée (x, y) est d'une couleur différente de *ancienneCouleur*
    - **Ne rien faire**
  - Si le pixel de coordonnée (x, y) est de la même couleur que *ancienneCouleur*
    - Changer la couleur de ce pixel
    - Tenter de changer la couleur (**remplir**) des points qui se trouvent autour

Remplir une zone graphique 4 / 5

Solution

procédure remplir **(E/S** e : Ecran,**E** x,y : **Naturel**, ancienneCouleur, nouvelleCouleur : Couleur**)**
debut
    si obtenirCouleurPixel(e,x,y)=ancienneCouleur **alors**
        fixerCouleurPixel(e,x,y,nouvelleCouleur)
        remplir(e,x,y-1,ancienneCouleur,nouvelleCouleur)
        remplir(e,x,y+1,ancienneCouleur,nouvelleCouleur)
        remplir(e,x-1,y,ancienneCouleur,nouvelleCouleur)
        remplir(e,x+1,y,ancienneCouleur,nouvelleCouleur)
    finsi
fin

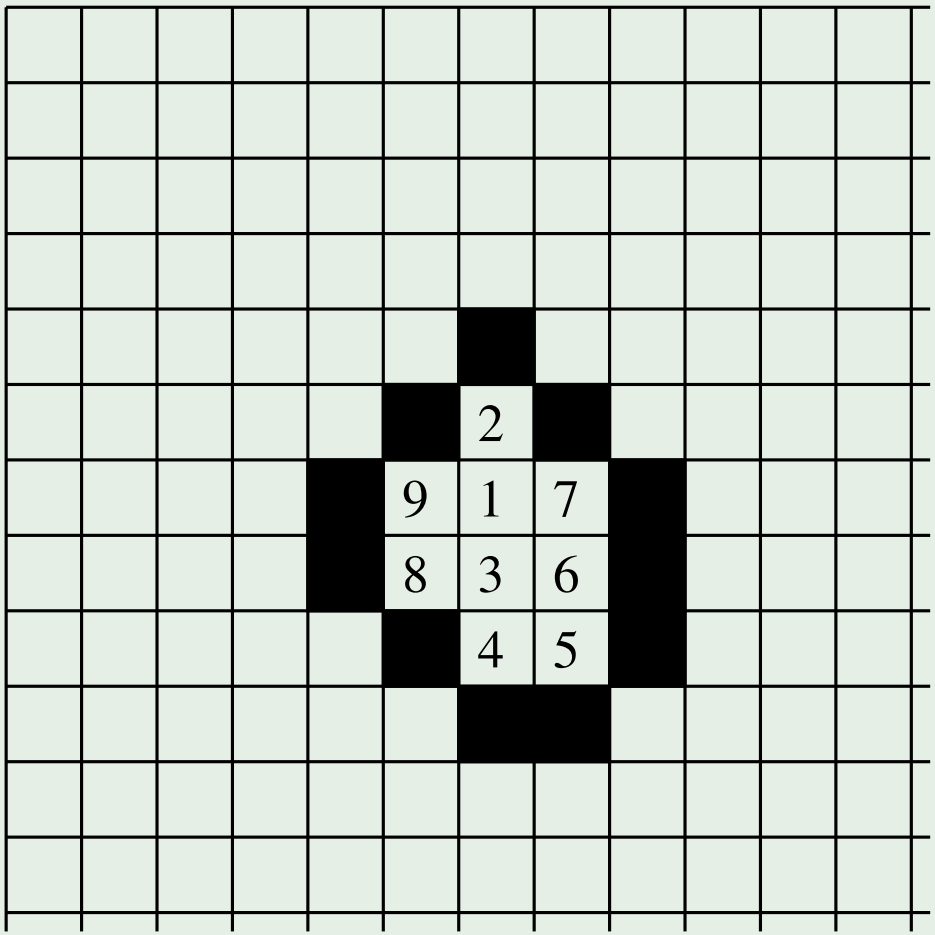
Note

On pourrait améliorer l'algorithme en vérifiant qu'on ne "sort" pas de l'écran

Remplir une zone graphique 5 / 5

Exemple d'ordre de changements de couleur des pixels

On considère que le (0,0) est en haut à gauche :



Évaluation d'une expression arithmétique 1 / 5

Présentation

- Il y a plusieurs façon de noter une expression arithmétique
  - **Infixe** : *2*(3+5)*
  - **Préfixe (ou notation polonaise)** : ** 2 + 3 5*
  - **Postfixe (ou notation polonaise inversée)** : *3 5 + 2 **
- En s'inspirant de la notation préfixe on peut représenter une expression aritmétique à l'aide d'un tableau tel que la *ième* case peut contenir :
  - un nombre
  - un opérateur et dans ce cas les opérandes sont à la position $2i$ et $2i + 1$

*	2	+			3	5

Évaluation d'une expression arithmétique 2 / 5

Objectif

- En supposant que l'on ait :
  - **Type** Expression = **Tableau[**1..MAX**] de** Terme
  - **Type** Operateur = {addition,soustraction,multiplication,division}
  - **fonction** estUneOperation **(**t : Terme**) : Booleen**
  - **fonction** obtenirOperation **(**t : Terme**) :** Operateur
  - **fonction** obtenirNombre **(**t : Terme**) : Reel**
- Donner le corps de la fonction suivante qui calcule la valeur d'une expression que l'on sait correctement formée :
  - **fonction** evaluer **(**e : Expression**) : Reel**

Évaluation d'une expression arithmétique 3 / 5

Analyse du problème

- Une **expression arithmétique** est :
  - soit un nombre,
  - soit une opération composée d'un opérateur et de deux opérandes qui sont des **expressions arithmétiques**
- **Évaluer** une expression arithmétique :
  - Si c'est un nombre, sa valeur est la valeur de ce nombre
  - Si c'est une opération, sa valeur est le calcul de cette opération en **évaluant** ses deux opérandes
- Évaluer une expression arithmétique revient donc à évaluer le contenu de la première case du tableau

Évaluation d'une expression arithmétique 4 / 5

Solution

```
fonction evaluer (e : Expression) : Reel
debut
    retourner evaluerRecursivement(e,1)
fin
```

Évaluation d'une expression arithmétique 5 / 5

Solution

```
fonction evaluerRecursivement (e : Expression, indice : Naturel) : Reel
    Déclaration  v1,v2 : Reel
debut
    si non estUneOperation(e[indice]) alors
        retourner obtenirNombre(e[indice])
    sinon
        v1 ← evaluerRecursivement(e,2*indice)
        v2 ← evaluerRecursivement(e,2*indice+1)
        cas où obtenirOperation(e[indice]) vaut
            addition : retourner v1+v2
            soustraction : retourner v1-v2
            multiplication : retourner v1*v2
            division : retourner v1/v2
        fincas
    finsi
fin
```

Conclusion...

En conclusion

- Les algorithmes récursifs sont simples (c'est simplement une autre façon de penser)
- Les algortihmes récursifs permettent de résoudre des problèmes complexes
- Il existe deux types de récursivités :
  - terminale, qui algorithmiquement peuvent être transformée en algorithme non récursif
  - non terminale
- Les algorithmes récursifs sont le plus souvent plus gourmands en ressource que leurs équivalents itératifs

Architecture des Systèmes d'Information