IREM Clermont-Ferrand
Année 2010-2011
Stage d'Algorithmique

Philippe Lac
Malika More

Exemples d'algorithmes récursifs

Les programmes sont disponibles dans l'archive associée.
Par exemple, `algo1Rtrous.sce` *désigne la traduction de l'algorithme 1 sous* `SCILAB`*, en version récursive à compléter.*
Ainsi lorsque la mention `trous` *n'est pas présente, il s'agit alors de la version complète, et lorsque la mention* `R` *n'est pas présente, il s'agit d'une version itérative.*

1 Des exercices sur les suites

1. On considère l'algorithme suivant :

Entrée : n un entier
Résultat : ? ? ? ?

début
  Donner à S la valeur 0;
  pour i **de** 1 **a** n **faire**
    Donner à S la valeur $S + i$
  fin
  retourner : S
fin

Algorithme 1:

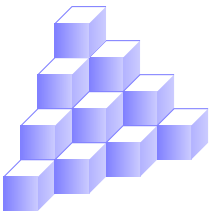
   (a) Quelle valeur rend-il pour $n = 0$?
   (b) De façon générale, quel résultat rend-il ?
   (c) Réécrire cet algorithme sous forme récursive.

2. (a) Que fait l'algorithme suivant :

```
Entrée : n un entier
Résultat : ????
début
    a reçoit 0;
    b reçoit 0;
    pour i de 1 a n faire
        a reçoit a + i;
        b reçoit a + b;
    fin
    retourner : b
fin
```

Algorithme 2:

☞ *aide :*



(b) Le réécrire sous forme récursive.

3. On considère l'algorithme suivant :

```
Entrée : n un entier
Résultat : ????
début
    si n=0 alors
        retourner : 1
    sinon
        retourner : f(n-1)*2
    fin
fin
```

Fonction f(n)

(a) Traduire cet algorithme sous `XCAS` ou `SCILAB`.

(b) Dérécursifier cet alogrithme.

4. On considère la suite u définie par

$$\begin{cases} u_0 = 5 \\ u_{n+1} = \sqrt{1+u_n} \end{cases}$$

(a) Ecrire un algorithme itératif, que l'on traduira sous `XCAS` ou `SCILAB`, qui, pour n donné, retourne u_n.

(b) Même question, avec un algorithme récursif.

(c) En utilisant les fonction précédentes, écrire un programme qui affiche tous les termes de la suite u vérifiant $|u_{n+1} - u_n| > 10^{-5}$.
Quel problème soulève cette façon de procéder ?

2 En vrac

1. On considère l'algorithme suivant :

```
Entrée : a un tableau;
n et k des entiers
Résultat : ? ? ? ?
début
    si k − 1 < n/2 alors
        retourner : a
    sinon
        tmp reçoit a[k];
        a[k] reçoit a[n − k + 1];
        a[n − k + 1] reçoit tmp;
        retourner : vrac(a,n,k−1)
    fin
fin
```

Fonction vrac(a,n,k)

   (a) Si $a = [1, 2, 3, 4, 5, 6]$, que va rendre l'exécution de $vrac(a, 6, 6)$?

   (b) Sa traduction peut poser un problème. De quelle nature ?
   ☞ *paramètres de la fonction et pile d'exécution*

   (c) Comment remédier simplement à ce problème ?

   (d) Ecrire la version itérative de cet algorithme.

2. L'algorithme suivant a été écrit pour réaliser la même tâche que précédemment :

```
Entrée : a un tableau;
n et k des entiers
Résultat : ? ? ? ?
début
    si k = n alors
        retourner : a
    sinon
        tmp reçoit a[n];
        a[n] reçoit a[k];
        a[k] reçoit tmp;
        retourner : vrac2(a,n,k + 1)
    fin
fin
```

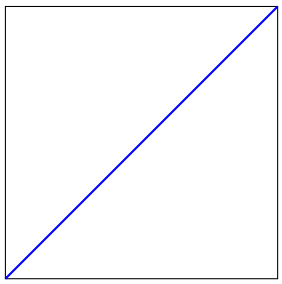
Fonction vrac2(a,n,k)

   Pourquoi n'est-il pas correct ?

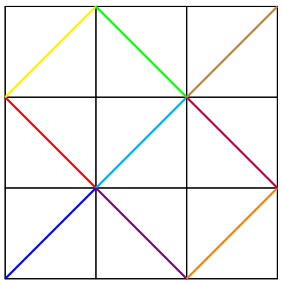
3 Des fractales

3.1 Courbe de Péano

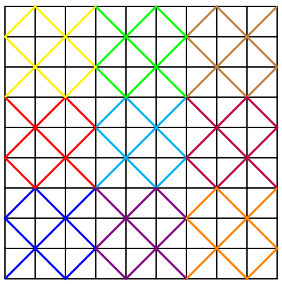
Rappelons le principe :



on part de la diagonale d'un carré



on découpe ce carré en 9 carrés, puis on trace les diagonales



on réitère ce processus sur chaque diagonale

Dans la suite, nous réaliserons des algorithmes destinés à être traduits sous `XCAS`.
Pour cela, nous avons utiliserons le mode `logo`, accessible via les touches `Alt` + `D`.
Ce mode met à notre disposition plusieurs commandes permettant de déplacer un curseur graphique, que l'on appellera tortue : à notre disposition plusieurs fonctions, bien pratiques pour ce que l'on veut faire :

- `avance(n)` : la tortue avance de n pas (par défaut n=10).
- `tourne_gauche(n)` : la tortue tourne à gauche de n degrés (par défaut n=90).
- `tourne_droite(n)` : la tortue tourne à droite de n degrés (par défaut n=90)
- `efface` : efface l'écran de la tortue ou recule de n pas en effaçant.
- `saute` : La tortue saute (avance sans laisser de traces) de n pas (par défaut n=10).

1. Compléter les séquences suivantes afin d'obtenir une fonction qui reproduit la seconde figure, ℓ désignant la longueur de la diagonale du grand carré :
   ☞ *on suppose que la tortue est déjà dans la bonne direction.*

Entrée : un réel ℓ
Résultat : Le déplacement sur une diagonale de longueur ℓ

début
```
avance(...);
tourne_gauche(...);
avance(...);
...
```
fin

Fonction Deplacement(ℓ)

2. Modifier l'algorithme précédent, pour obtenir un algorithme récursif du tracé de la courbe de Péano d'ordre n :

Entrée : n un entier, un réel ℓ
Résultat : La courbe de Péano d'ordre n sur une diagonale de longueur ℓ

début
```
...;
```
fin

Fonction Peano(n,ℓ)

☞ *On prendra soin de repérer les points de l'algorithme où doit intervenir la récursivité dans le premier algorithme.*
On utilisera la variable n pour le contrôle de la terminaison.

3. Traduire l'algorithme précédent sous `XCAS`, et réaliser le tracé de la courbe de Péano pour $n = 4$ et $\ell = 270$.
   Quelles sont les instructions dont on peut changer l'ordre dans l'algorithme sans modifier le résultat ?

3.2 D'autre exemples

1. En s'inspirant de la première partie, réaliser un algorithme, et le traduire sous `XCAS`, afin d'obtenir la courbe du dragon d'ordre n.
2. Réaliser un flocon de VonKoch

4 Problème du labyrinthe

Préléminaires :
Dans le dossier `Labyrinthe`, charger, sous `SCILAB`, le fichier `BoiteOutils.sce` et l'exécuter dans la console. Ce fichier fournit 3 programmes :

- *LabC(n) : permet de construire une matrice nxn associée à un labyrinthe ;*
- *LabA(M,n) : permet d'afficher une matrice M de taille nxn associée à un labyrinthe ;*
- *dedale(fi,fj,di,dj) : exploration d'un labyrinthe stocké dans une matrice L (variable globale), dont la sortie est aussi stockée dans une variable globale (si et sj).*
  di,dj désignent la postion de l'entrée et fi, fj la première position explorée.
  ☞ *par convention, la case inférieure gauche a la position 1,1.*

1. Faire un essai, en utilisant la matrice M sauvegardé dans le fichier $M.dat$:
   `load('M.dat');L=M;size(L);LabA(L,10);si=6;sj=1;dedale(4,9,4,10)`
2. Construire un labyrinthe et tester le programme dedale sur ce labyrinthe.
3. Comment modifier l'ordre d'exploration des bifurcations ?
4. La modification de cet ordre peut-il permettre de trouver le chemin le plus court ?
5. Construire un labyrinthe mettant en echec le programme dedale (présence d'un cycle).
   ☞ *Au cas où, un exemple de matrice est sauvegardé dans la variable `Mcycle` (fichier `Mcycle.dat`).*
6. Modifier l'algorithme de façon à traiter les cycles éventuels.
   ☞ *on pourra, par exemple, utiliser un marqueur sur les cases observées, compatible avec le test `deplacement valide`*
7. Réfléchir à un algorithme permettant d'obtenir toutes les solutions, puis la solution la plus courte.

5 Récursivité multiple

1. Ecrire l'algorithme d'une foncton récursive $c(n, p)$ qui, pour les entiers n et p, retourne le coefficient binomial $\begin{pmatrix} n \\ p \end{pmatrix}$.
2. Traduire cet algorithme sous `XCAS` et le tester sur différentes valeurs.
3. Compléter l'algorithme suivant, sachant qu'il utilise la construction du triangle de Pascal pour obtenir le même résultat.

```
Entrée : n et p des entiers
variables locales : a un tableau (n+1)x(p+1)
Résultat : p parmi n
début
    a := tableau nul de dimension (n+1)x(p+1);
    pour i de 0 a p faire
        a[i,i] := 1;
    fin
    pour i de 0 a n faire
        a[i,0] := 1;
    fin
    pour i de 1 a n faire
        pour i de 1 a p faire
            a[i,j] := ... ;
        fin
    fin
    retourner : ...
fin
```

Fonction c(n,p)

4. Comparer les temps d'exécution des deux fonctions.
5. Réaliser une version itérative, utilisant un tableau à une seule dimension.

6 Diviser pour régner

1. On se propose de reprendre le jeu du Plus-Moins, et d'en écrire un algorithme récursif. *Principe : le joueur choisit mentalement un nombre entier entre deux bornes, fixées préalablement (n et p par exemple), et l'algorithme procède alors par élimination dichotomique.* Dans les fichiers mis à disposition, on trouvera dans le dossier PlusMoins des fichiers `PMtrous` en version `SCILAB` ou `XCAS`. Compléter ces fichiers afin d'obtenir une réponse au problème posé.