ALGORITHMES RÉCURSIFS

L'adjectif « récursif » est associé à des objets mathématiques dont la définition fait intervenir l'objet lui-même : pour simplifier, l'objet est défini grâce à lui-même...

1 Ensembles bien fondés

1.1 Généralités

Définition. Soit E un ensemble muni d'une relation d'ordre notée comme d'habitude $\leqslant$.

On dit que l'ensemble ordonné $(E, \leqslant)$ est bien ordonné (ou que $\leqslant$ est un bon ordre) si et seulement si toute partie non vide de E admet un plus petit élément.

On dit que l'ensemble ordonné $(E, \leqslant)$ est bien fondé (ou que $\leqslant$ est un ordre bien fondé) si et seulement si il n'existe pas de suite infinie strictement décroissante d'éléments de E.

On déduit des définitions les quelques propriétés suivantes :

Proposition 1 *Un bon ordre est un ordre total et tout ensemble bien ordonné est bien fondé.*

Proposition 2 *Dans un ensemble bien fondé, toute suite infinie décroissante est stationnaire.*

Définition. Dans un ensemble ordonné E, si A est une partie de E et a un élément de A, on dit que a est un élément minimal dans A si et seulement si il n'existe pas dans A d'élément strictement inférieur à a.

$$a \text{ est minimal dans } A \text{ si et seulement si } \forall x \in A \quad x \leqslant a \Rightarrow x = a$$

Proposition 3 (caractérisation équivalente) *E est bien fondé si et seulement si toute partie non vide de E possède au moins un élément minimal.*

Remarque. Ne pas confondre « élément minimal » et « minimum » ! Il y a bien sûr un lien : si A a un minimum m, alors m est évidemment un élément minimal, mais la réciproque est fausse.

Exemples

a) $\mathbb{N}$, muni de l'ordre naturel, est bien ordonné ;

b) $\mathbb{N} - \{0, 1\}$, muni de l'ordre de la divisibilité ($a \leqslant b$ si et seulement si a divise b) est bien fondé, mais pas bien ordonné ;

c) $\mathbb{N}^2$, muni de l'ordre $(a, b) \leqslant (c, d) \iff (a \leqslant c \text{ et } b \leqslant d)$, est bien fondé, mais pas bien ordonné.

1.2 Démonstration par induction sur un ensemble bien fondé

Théorème 4 (Principe d'induction) *Soit E un ensemble bien fondé, on pose $\mathscr{M}(E)$ l'ensemble des éléments minimaux de E. Soit $\mathscr{P}(x)$ une proposition dépendant d'une variable x appartenant à E.*

On suppose que :

- *pour tout $x \in \mathscr{M}(E)$, $\mathscr{P}(x)$ est vraie*
- *pour tout $x \in E$, si $\mathscr{P}(y)$ est vraie pour tous les éléments y tels que $y < x$, alors $\mathscr{P}(x)$ est vraie*

Alors pour tout $x \in E$, $\mathscr{P}(x)$ est vraie.

Remarque. Dans le cas de l'ensemble bien fondé $\mathbb{N}$, on retrouve le principe de la démonstration par récurrence forte.

2 Fonctions récursives

2.1 Théorème fondamental

Théorème 5 *Soit F un ensemble bien fondé, X, Y deux autres ensembles.*

On suppose données :

- *v une application de X dans F*
- *B une partie de X et φ une application de B dans Y ;*
- *$d_1, ..., d_k$ des applications de $X - B$ dans X telles que pour tout $i \in \{1, ..., k\}$ et $x \in X - B$, $v(d_i(x)) < v(x)$;*
- *C une application de $X \times Y^k$ dans Y*

Alors il existe une unique application f de X dans Y telle que :

(*) pour tout $x \in B$, $f(x) = \varphi(x)$;

(**) pour tout $x \in X - B$, $f(x) = C\big(x, f(d_1(x)), ..., f(d_k(x))\big)$.

On dit que f a été définie récursivement (ou par induction) sur X grâce aux deux propriétés (*) et (**).

Dans ce théorème, les éléments de B sont appelés les cas de base, les applications d_i sont les applications de descente et C est la fonction de calcul.

2.2 Algorithmes récursifs et terminaison

Corollaire 6 *Toute fonction récursive est calculable par un algorithme dit récursif. Avec les mêmes notations que dans le théorème précédent, l'algorithme décrit ci-dessous termine et décrit le calcul de $f(x)$ pour tout x de X :*

```
F(x) =
  si x element de B alors retourner phi(x)
  sinon retourner C( x, F(d1(x)), ..., F(dk(x)) );
```

En pratique, l'ensemble bien fondé est souvent $\mathbb{N}$ muni de l'ordre usuel.

Pour résoudre récursivement un problème algorithmique :

- on choisit l'ensemble F et l'application v adaptés au problème ;
- on détermine les cas de base, pour lesquels le résultat est immédiat ;
- pour les autres valeurs de x, en supposant qu'on est capable de calculer $f(y)$ pour y tel que $v(y) < v(x)$, on explicite la façon de calculer $f(x)$ en fonction d'un certain nombre de telles valeurs $f(y)$.

Alors le théorème précédent et son corollaire assure que l'algorithme termine. Autrement dit, l'algorithme termine car **les valuations des paramètres des appels récursifs décroissent strictement dans un ensemble bien fondé**, donc forment nécessairement des suites finies.

2.3 Correction des algorithmes récursifs

Pour prouver la correction de l'algorithme, on utilise une proposition $\mathscr{P}(x)$, dite invariant d'appel récursif, du type « le calcul de `F(x)` fournit une solution au problème », qui doit vérifier les propriétés :

- pour tout $x \in B$, $\mathscr{P}(x)$ est vraie ;
- pour tout $x \in X - B$, si on suppose que $\mathscr{P}(y)$ est vraie pour tout y tel que $v(y) < v(x)$, alors $\mathscr{P}(x)$ est vraie.

Alors d'après le principe d'induction, la proposition est toujours vraie, donc l'algorithme calcule dans tous les cas ce qu'on attend de lui.

Enfin, la complexité se calcule elle aussi par induction.

Exemples d'algorithmes récursifs à présenter en cours :

a) calcul de la factorielle (complexité)

b) plus généralement, calcul de suites définies par des relations de récurrences : suite récurrente simple, suite de Fibonacci (complexité)

c) somme des termes d'un tableau : algorithme récursif direct, algorithme récursif encapsulé, comparaison des complexités

d) exponentiation classique (complexité)

e) multiplication rapide, exponentiation rapide (complexité)

f) pgcd binaire (complexité)