Introduction à l'Algorithmique II

Chapitre : Récursivité

A.U: 2020 – 2021

Fonctions Récursives

Fonctions Récursives

- Introduction: Il arrive, en mathématique, que des suites soient définies de la manière suivante :

  u_0 = constante

  $u_n = f(u_{n-1})$

- Exemple :

La suite factorielle : n! = n*(n-1)!, pour n≥1 avec 0!=1, peut s'ecrire :

f(0)=1

f(n)=n*f(n-1)

Ce que l'on peut traduire par : f(n) = (si n=0 alors 1 sinon n*f(n-1)).

Fonctions Récursives

- Cela peut se traduire en algorithmique par :

```
fonction factorielle_rec(n :entier) : entier
début
si (n=0) alors retourne(1)
sinon retourne(n*factorielle_rec(n-1))
finsi
Fin
```

- Dans la fonction factorielle_rec(), on constate que la fonction s'appelle elle-même.
- Ceci est possible, puisque la fonction factorielle_rec() est déclarée avant son utilisation (c'est l'entête d'une fonction qui la déclare).

Fonctions Récursives

- Que se passe-t-il lorsque on calcul factorielle_rec(3) ?

Factorielle_rec(3) = 3 * factorielle_rec(2)

= 3 * (2*factorielle_rec(1))

= 3 * (2 * (1*factorielle_rec(0)))

= 3 * (2 * (1 * (1)))

= 3 * (2 * (1))

= 3 * (2)

= 6

Fonctions Récursives

- La récursivité est un concept fondamental en mathématiques et en informatique.

- Un programme récursif est un programme qui s'appelle lui-même.

- Pour éviter la **boucle** infinie, il faut que le programme cesse de s'appeler lui-même.

  → Un programme récursif doit contenir une condition **d'arrêt** (ou de terminaison) qui autorise le programme à ne plus faire appel à lui-même.

- Fonction récursive:
  - Directe: Contient appel à elle même
  - Indirecte (croisée): contient un appel à une fonction qui emmène à l'appel de la fonction initiale.

Fonctions Récursives

- **Souvent** la récursivité dans les fonctions est ainsi:

```
fonction Fonc_Rec(paramètres) : Type_retoure
début
   Si (cond arret 1 vérifiée)
     retourner valeur_1
   FinSi
   appel(s) récursivité Fonc_Rec(paramètres')
Fin
```

- Attention, une mauvaise condition d'arrêt → Solution fausse.

- Une fonction récursive peut avoir **plusieurs conditions d'arrêt** . Exemple: Fonction de Fibonacci

Fonctions Récursives

- Suite de Fibonacci

$F_n = F_{n-1} + F_{n-2}$, pour n ≥2 avec F_0=0 et F_1=1

- On a un programme récursif simple associé a cette relation :

```
fonction Fibo1(n entier) : entier
début
  si (n = 0 ou n = 1) alors retourne(n)
  sinon retourne(Fibo1(n-1) + Fibo1(n-2))
  finsi
Fin
```

- Le nombre d'appels nécessaires au calcul de F_n est égal au nombre d'appels nécessaires au calcul de F_{n-1} plus celui relatif au calcul de F_{n-2},

- ceci correspond bien à la définition de la suite de Fibonacci .

Fonctions Récursives

- La récursivité
  - Est un concept proche de l'esprit humain, **MAIS**
  - n'est pas nécessairement la **meilleure** solution de résolution d'un Pb en terme de temps d'execution ou d'espace mémoire réservé (**complexité**)
- Exemple: 2ème algorithme de calcul de **Fn** (en utilisant un tableau)

```
constante : N = 26
fonction Fibo2(entier :n) : entier
variable i: entier ; tableau F[N] : entier
Début
   F[0]  ← 0
   F[1]  ← 1
   pour i allant de 2 à n faire
       F[i]  ← F[i-1] + F[i-2]
   Finpour
   retourne F[n]
Fin
```

Fonctions Récursives

Exemple de récursivité: Recherche Dichotomique Récursive RDR dans un **tableau trié**: retourne la position de elem s'il existe et -1 sinon

```
fonction RDR(T[]:entier,elem:entier,Deb:entier,Fin : entier):Entier
variable milieu : entier
Début
si Fin < Deb
retourne -1
milieu ← (Deb + Fin) / 2
si elem = T[milieu] retourne milieu
   sinon si elem <  T[milieu]
         retourne rech_dich_rec(T,elem,Deb,milieu-1)
         sinon retourne rech_dich_rec(T,elem,milieu +1,Fin)
         finsi
Finsi
Fin
```