

Analyse des Algorithmes Récursives

Un Algorithme (fonction ou procédure) est dit récursif si il fait appel à lui-même.
l'implémentation récursive suit toujours la définition récursive de la solution, et nécessite la condition d'arrêt de la récursivité

exemple

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

↳ Algorithme Fact (n: entier): entier

Début

si n=0 alors return 1

ssi;

fin;

sinon return n * Fact(n).

condition d'arrêt de la récursivité

Appel

récursif

Comment Analyser la complexité temporelle?

- 1- déterminer l'équation de récurrence correspondante à $T(n)$
- 2- Résoudre cette équation de récurrence

réu @

il existe trois méthodes de résolution.

- Pour déduction: par substitution de proche en proche jusqu'à trouver le cas général et terminer (sortir) à l'aide de la condition d'arrêt
- Deviner une solution, puis la démontrer par récurrence.
- Utiliser la solution de certaines équations de récurrences connues en mathématique.

Exemple du factoriel

1) équation de récurrence

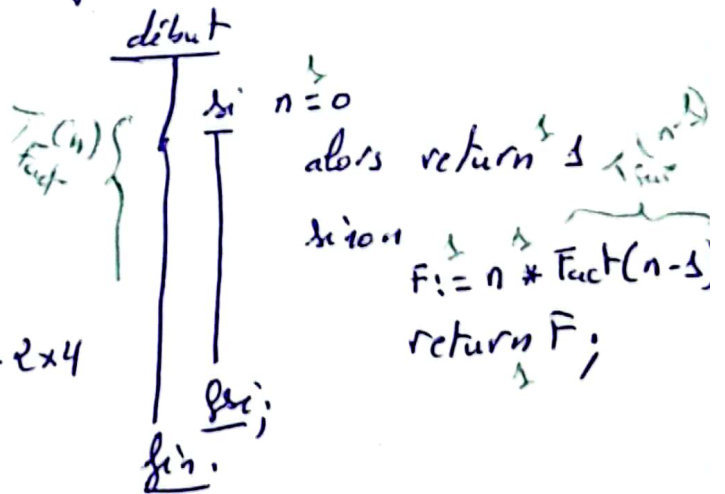
$$T(n)_{\text{fact}} = \begin{cases} T(n-1) + 1 & \text{si } n > 0 \\ 1 & \text{si } n = 0 \end{cases}$$

2) résolution par déduction

$$\begin{aligned} T(n) &= T(n-1) + 1 \\ &= (T(n-2) + 1) + 1 = T(n-2) + 2 \times 1 \\ &= (T(n-3) + 1) + 2 \times 1 \\ &= T(n-3) + 3 \times 1 \\ &\vdots \\ &= T(n-n) + n \times 1 \\ &= T(0) + n \times 1 = 1 + n \end{aligned}$$

$$T(n) = 1 + n = O(n)$$

fonction fact (n: entier): entier



récurrence