Résumé

▪ Définition :

Un algorithme ou un traitement est appelé récurrent s'il utilise une structure de traitement répétitive pour produire un résultat qui peut découler d'autres résultats de traitements précédents.

▪ Ordre de récurrence :

Ordre 1 :

La détermination du résultat d'un traitement dépendra du résultat d'un traitement précédent.

Exemple :

Soit la suite U définie par :

$U_0 = 5$

$U_n = 2 * U_{n-1} + 1.5$ pour tout $n \geq 1$

Le calcul de la valeur d'un terme de la suite U_n dépendra de la valeur du terme précédent U_{n-1}.

Ordre 2 :

La détermination du résultat d'un traitement dépendra du résultat des deux traitements précédents.

Exemple :

Soit la suite U définie par :

$U_0 = 1$

$U_1 = 3$

$U_n = 2 * U_{n-1} + 3 * U_{n-2}$ pour tout $n \geq 2$

Le calcul de la valeur d'un terme de la suite U_n dépendra de la valeur des deux termes précédents U_{n-1} et U_{n-2}.

Ordre n :

La détermination du résultat d'un traitement dépendra du résultat de n traitements précédents.

Les algorithmes récurrents

Exemple :

Soit M une matrice de taille n * m entiers et soit SL la somme des sommes S_i avec :

S_i est la somme des entiers d'une $i^{ème}$ ligne de M.

$SL = S_0 + S_1 + ... + S_{n-1}$

$S_i = M [i , 1] + M [i , 2] + ... + M [i , m-1]$

$S_i = S_i + M [i , j]$ avec j allant de 0 à m-1

Le calcul d'une somme S_i, à l'instant j, dépend que du contenu de la case M [i , j].

On dit que le calcul d'une S_i est récurent **d'ordre 1**.

Par contre pour déterminer SL, il faut calculer n S_i, donc le calcul de la somme SL est récurrent **d'ordre n**.