CM 4 : programmation dynamique

LECTURES :

Chapitre 8 IAAD : Dynamic Programming

Supports de cours

QUESTIONS, REFLEXIONS

1) Soit l'énoncé récursif de Fibonnacci

$F_0 = 0, F_1 = 1, \forall n \geq 2, \quad F_n = F_{n-1} + F_{n-2}$

Et sa traduction directe récursive.

On peut le transformer en une version itérative en « commençant » par 0 puis 1 etc et calculant de proche en proche jusqu'à n.

prev ←-1, res ←1

Pour i de 0 à n

Etc..

2) Considérons maintenant

$$\binom{n}{k} = C_n^k = \frac{n!}{k!(n-k)!}.$$

Qui peut s'écrire

$$\binom{n}{k} + \binom{n}{k+1} = \binom{n+1}{k+1}$$

Qui donne comme algorithme (récursif) :

- C(n,m) =1 si m = 0
- C(n,m) = 1 si n = m
- C(n,m) = C(n-1,m)+C(n-1,m-1)

On peut le transformer en une version itérative en « commençant » par 0,0 puis 1,0 et 1,1, puis 2,0 2,1 et 2,2 etc et calculant de proche en proche jusqu'à n,m

Ecrivez cet algorithme (cf triangle de pascal)

Quelle est sa complexité temporelle et spatiale ? Justifiez.

3) Mémoisation

On repart des algorithmes récursifs (par exemple fibonacci fib(n)).

L'idée est maintenant de ne pas refaire les calculs déjà faits, c'est-à-dire que quand on appelle la fonction pour un nombre k on va regarder si fib(k) a déjà été calculé (et stocké quelque part) auquel cas on renvoie la valeur sinon on fait le calcul récursif, on récupère la valeur et la stocke quelque part puis renvoie cette valeur

Modifiez l'algorithme récursif de sorte que :

- Les cas triviaux sont fib(0) et fib(1) mais aussi fib(k) où on a déjà fait le calcul
- Le cas récursif sinon : on fait le calcul récursif en sauvegardant le résultat avant de renvoyer la valeur (et du coup au prochain appel avec le paramètre ça deviendra un cas trivial).

Complexité ??