Activité : Programmation dynamique

Suite de Fibonacci

Chaque valeur de cette suite est égale à la somme des 2 valeurs qui la précèdent.

Au départ, les deux valeurs sont 1 et 1. Par somme, la troisième valeur est 2, puis la suivante vaut 3, puis 5 et ainsi de suite.

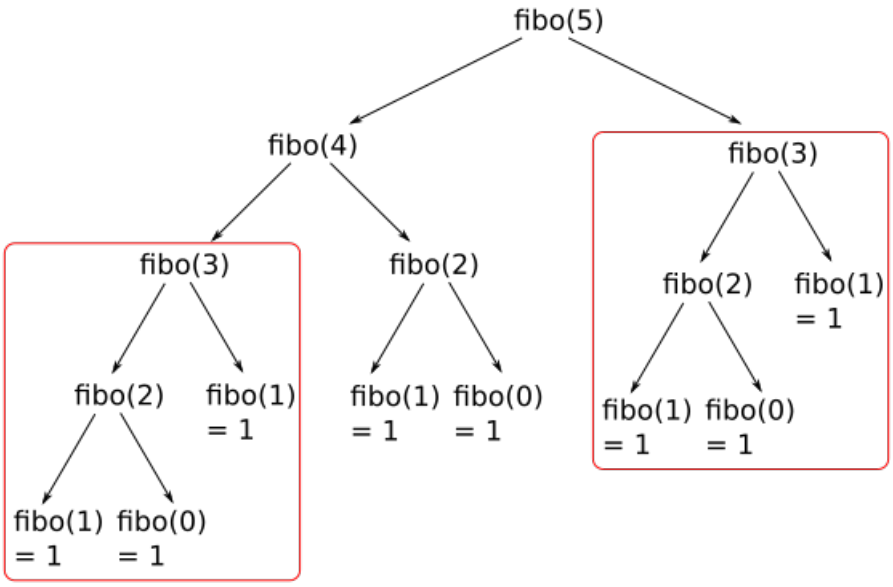
On note chaque valeur par référence à sa position :

- fib(0)=1 et fib(1)=1 pour les deux premières valeurs,
- fib(2)=fib(0)+fib(1)=1+1=2 pour la troisième valeur,
- fib(3)=fib(2)+fib(1)=2+1=3 pour la quatrième valeur,
- fib(4)=fib(3)+fib(2)=3+2=5 pour la cinquième valeur,
- et ainsi de suite.

On remarque que cette suite (définie par une double récurrence) se calcule en connaissant les deux termes précédents. La programmation récursive s'applique facilement pour calculer les différentes valeurs de cette suite.

1) Écrire le code en Python de la fonction récursive `fibo` qui a en paramètre le nombre entier `n` indiquant la position du nombre à calculer et renvoie la valeur de la suite de Fibonacci de position n.

2) On a représenté les appels récursifs sous forme d'arbre binaire pour le calcul de `fibo(5)`.



Représenter les appels récursifs de `fibo(6)`.

Combien de fois doit-on calculer `fibo(3)`, `fibo(4)` et `fibo(5)` ?

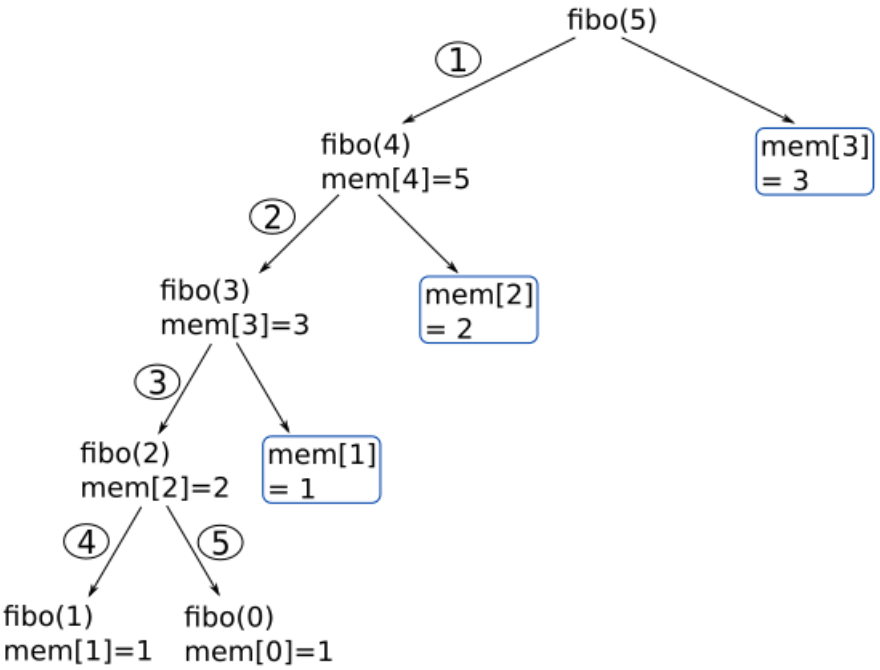
Programmation dynamique

La **programmation dynamique** consiste à éviter de recalculer une valeur déjà calculée en la conservant dans un tableau de valeurs. Le procédé est le suivant :

- pour calculer une nouvelle valeur, on regarde si les valeurs nécessaires ont déjà été calculées.
- si oui, on les utilise pour le nouveau calcul,
- si non, on effectue le calcul de la valeur nécessaire,
- si la nouvelle valeur calculée n'est pas dans le tableau, on l'ajoute au tableau des valeurs calculées.

Par exemple, le calcul de `fibo(5)` nécessite de calculer les valeurs de `fibo(0)` à `fibo(4)`. Lorsque ces valeurs sont calculées, elles sont mémorisées dans un tableau et utilisées avant de provoquer un appel récursif inutile.

On représente ci-dessous le calcul de fibo(5).



1) Écrire un code en Python qui permet de calculer les nombres de la suite de Fibonacci en utilisant la programmation dynamique, c'est à dire en mémorisant les valeurs calculées dans une liste et ainsi en évitant des appels récursifs.

2) **a)** Calculer avec la fonction récursive `fibo(35)`. Mesurer si possible le temps d'exécution.

b) Calculer la même valeur avec la fonction qui utilise la programmation dynamique. Mesurer, si possible, le temps d'exécution.

3) Combien y a t-il d'appels récursifs à chaque appel de la fonction `fibo(3)` ? `fibo(4)` ? `fibo(n)` ?

4) Le nombre maximal d'appels récursifs est fixé à 3000. Quel est la plus grande valeur de `n` que l'on peut saisir en argument de la fonction `fibo` ?

5) Combien d'appels récursifs sont réalisés en programmation dynamique ?