Programmation dynamique

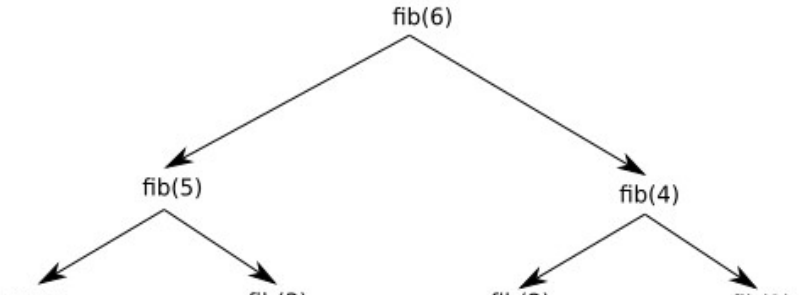
A) Une première approche avec la suite de Fibonnacci

Définition de la suite de Fibonnacci : 0 1 1 2 3 5 8 .. (un terme est égale à la somme de ses deux précédents)

- **Ecrire** le code en récursif et tester pour n=5,8,10,50

```
def fib(n):
```

- Pour n=6, il est possible d'illustrer le fonctionnement de ce programme avec le schéma ci-dessous à compléter :



fib(1) =1 fib(0) =0

En observant attentivement le schéma ci-dessus, remarquez que de nombreux calculs sont inutiles, car effectué plusieurs fois : par exemple on retrouve le calcul de fib(4) à …. endroits, fib(3) à ….. endroits, fib(2) à …..endroits .

On pourrait donc grandement simplifier le calcul en calculant une fois pour toutes fib(4), fib(3),fib(2) … en "mémorisant" le résultat et en le réutilisant quand nécessaire .

On va utiliser un tableau mem qui va **mémoriser** les valeurs de fib(i) et qui seront donc retourner directement par l'accès au tableau mem sans à refaire tous les calculs (c'est-à-dire les appels récursifs) :

Remarque : il faut quand même réaliser 1 fois le calcul pour le stocker dans le tableau mem

On utilise une fonction qui va créer le tableau mem et lancer la fonction récursive

```
def fib_mem(n):
    mem = [None]*(n+1)  #permet de créer un tableau contenant n+1 zéro
    return fib_mem_c(n,mem)

def fib_mem_c(n,m):
    if n==0 or n==1:
        m[n]=n
        return n
    elif m[n]!=None:
        return m[n]
    else:
        m[n]=fib_mem_c(n-1,m) + fib_mem_c(n-2,m)
```

Comparer le temps d'exécution de fib et fib_mem pour différentes valeurs de n