POLYNÉSIE – 2026 – sujet1

EXERCICE 1 (6 POINTS)

Cet exercice porte sur la notion de complexité.

On définit une suite d'entiers U_n de la façon suivante :

$$U_0 = 0,\ U_1 = 1,\ U_n = U_{n-1} + U_{n-2} \text{ pour tout } n \geq 2.$$

Les premières valeurs de cette suite sont donc les suivantes : $U_0 = 0$, $U_1 = 1$, $U_2 = 1$, $U_3 = 2$, $U_4 = 3$, $U_5 = 5$.

On donne la fonction suivante permettant de calculer les termes de cette suite :

1. Compléter la fonction récursive `suite` qui prend en paramètre un entier *n* et renvoie la valeur de U_n.

```
def suite(n):
    if n < 2:
        valeur = ...
    else:
        valeur = suite(n - 1) + suite(...)
    return ...
```

On appelle *p* la fonction donnant le nombre d'additions à effectuer pour calculer le terme d'indice *n* de la suite U_n avec la fonction proposée ci-dessus. On peut montrer que cette fonction est la suivante :

$$p(0) = 0\ ;\ p(1) = 0\ ;\ p(n) = 1 + p(n-1) + p(n-2) \text{ pour tout } n \geq 2.$$

2. Écrire une fonction `nb_additions_suite` prenant en paramètre un entier *n* et renvoyant la valeur de *p(n)* où *p* est la fonction définie ci-dessus.

3. Calculer les valeurs *p(2)*, *p(4)* et *p(7)*.

On propose une nouvelle fonction pour calculer les termes de la suite U_n :

```
def suite2(n, valeurs_calculees = [0, 1]):
    if n < len(valeurs_calculees):
        valeur = valeurs_calculees[n]
    else:
        valeurs_calculees.append(suite2(n - 1, valeurs_calculees)
                                 + suite2(n - 2, valeurs_calculees))
    return valeurs_calculees[n]
```

4. Recopier et compléter le tableau ci-dessous avec le nombre d'additions effectuées par la fonction `suite2` pour les calculs des premiers termes de la suite U_n.

complexité de suite2	
n	**nombre d'additions**
2	1
3	2
4	3
5	
6	
7	

On appelle *q* la fonction donnant le nombre d'additions à effectuer pour calculer le terme d'indice *n* de la suite U_n avec la fonction `suite2`.

5. Expliquer pourquoi `suite2` est plus efficace que `suite` et donner le nom du procédé algorithmique utilisé pour obtenir cette efficacité.

6. Proposer une version itérative, non récursive, `suite3` du calcul des termes de la suite U_n.

7. Expliquer pourquoi la version itérative est préférable à la version récursive pour obtenir un terme quelconque de la suite U_n.