TP14

FONCTIONS RÉCURSIVES (2)

Corrigé

Exercice 1 Exponentiation rapide

Lorsque l'on souhaite calculer a^n , on peut effectuer $n-1$ multiplication en effectuant le produit :

$$a^n = a \times a \times \cdots \times a$$

Cependant l'exemple suivant montre que l'on peut calculer a^{15} différemment :

$$a^{15} = a \times (a^7)^2 = a \times (a \times a^6)^2 = a \times (a \times (a^3)^2)^2 = a \times (a \times (a \times a^2)^2)^2$$

ou encore :

$$a^{15} = a \times a^{14} = a \times (a^2)^7 = a \times \left(a^2 \times (a^2)^6\right) = a \times \left(a^2 \times \left((a^2)^2\right)^3\right) = a \times \left(a^2 \times \left(a^4 \times (a^4)^2\right)\right).$$

Il y a ici 3 multiplications par a et trois mises au carré, soit 6 multiplications (contre 14 avec la version normale).

1. Décomposer a^{27}

$$a^{27} = a \times (a^{13})^2 = a \times (a \times (a^6)^2)^2 = a \times (a \times ((a^3)^2)^2)^2 = a \times (a \times ((a \times a^2)^2)^2)^2$$

2. Dans l'algorithme précédent, exprimer dans chacun des cas p en fonction de n.

   — si $n = 0$ alors $a^1 = 1$
   — si n est pair alors $a^n = (a^{n//2})^2$
   — si n est impair alors $a^n = a \times (a^{n//2})^2$

3. Compléter la fonction `expoR(a,n)` qui renvoie le calcul de a^n selon le principe d' exponentiation rapide.

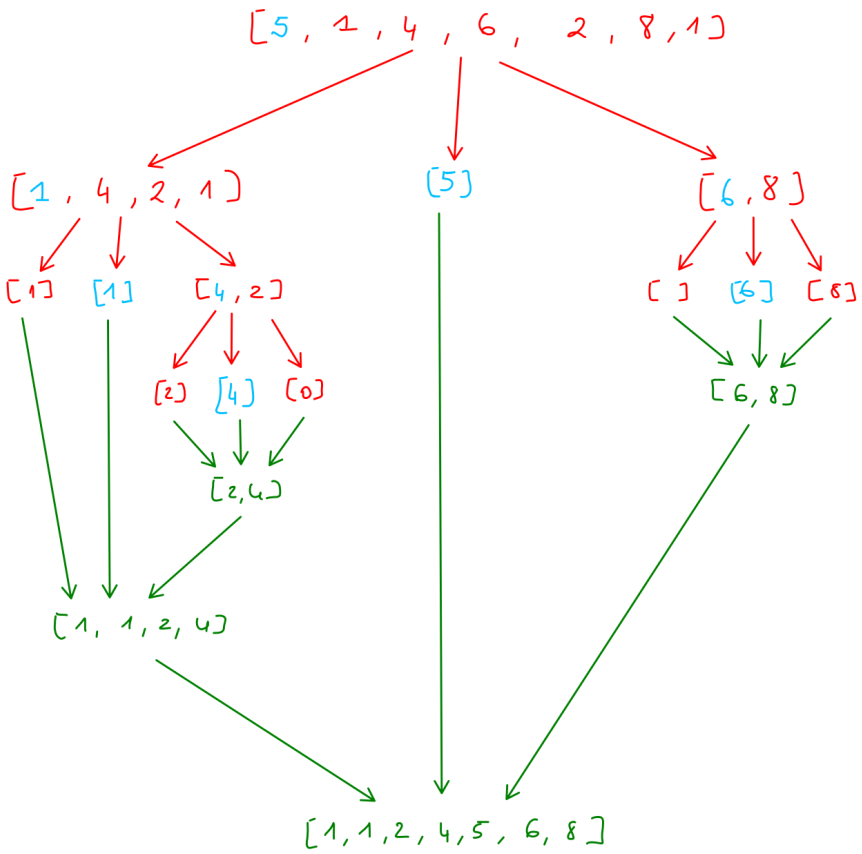
```
def expoR(a,n):
    if n == 0 :
        return 1
    elif n%2 == 0:
        return expoR(a,n//2)**2
    else :
        return a * expoR(a,n//2)**2

# Version 2, sans utiliser **
def expoR(a,n):
    if n == 0 :
        return 1
    elif n == 1 :
        return a
    elif n == 2 :
        return a*a
    elif n%2 == 0:
        return expoR(expoR(a,n//2),2)
    else :
        return a * expoR(expoR(a,n//2),2)
```

```python
# version 3 - correspond à la 2e décomposition de a**15
def expoR(a,n):
    if n == 0 :
        return 1
    elif n%2 == 0:
        return expoR(a**2,n//2)
    else :
        return a * expoR(a**2,n//2)
```

4. Tester la fonction `expoR(2,2)` et `expoR(2,5)` par exemple.

Exercice 2 Tri rapide

1. Trier à la main la liste L = [5, 1, 4, 6, 2, 8, 1]. Faire un schéma.



2. Quelle est la condition d'arrêt ?
   Si la liste est de longueur $\leqslant 1$ (vide ou avec un seul élément), alors on renvoie la liste elle-même.

3. Programmer le tri rapide.

```
    def tri_rapide(L):
        if len(L) <= 1 :
            return L
        else :
            pivot = L[0]
            partie1 = [] # élements inférieurs au pivot
            partie3 = [] # élements supérieurs au pivot
            for element in L[1:]:
                if element <= pivot:
                    partie1.append(element)
                else:
                    partie3.append(element)
            return tri_rapide(partie1) + [pivot] + tri_rapide(partie3)
```

Exercice 3 **Comptage de "a"**

On cherche à déterminer le nombre de a dans une chaîne de caractère à l'aide de la fonction `compte_a(chaine)` .

Par exemple :

```
>>> chaine='blabla'

>>> compte_a(chaine)
2
```

Compléter la fonction `compte_a(chaine)` qui prend en entrée une chaine de caractère et compte le nombre de a dans cette chaîne de façon récursive.

```
def compte_a(chaine):
    if len(chaine) == 0:
        return 0
    else:
        if chaine[0] == "a":

            return 1 + compte_a(chaine[1:])
        else:
            return  compte_a(chaine[1:])
```