TP14

FONCTIONS RÉCURSIVES (2)

À préparer : exercice 1

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} en suivant le premier exemple ci-dessus..

L'algorithme d'exponentiation rapide est la transposition au cas général de l'exemple suivant :

— si $n = 0$ alors $a^0 = 1$;

— si $n = 2p$ alors $a^n = (a^p)^2$;

— si $n = 2p+1$ alors $a^n = a \times (a^p)^2$.

Alors que l'algorithme naïf demande de l'ordre de n multiplications pour calculer a^n, l'algorithme exponentiation rapide se contente de l'ordre de $\ln(n)$ multiplications.

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

— si $n = 0$ alors $a^0 = 1$

— si n est pair alors $a^n = (a^{\cdots\cdots\cdots\cdots})^2$

— si n est impair alors $a^n = a \times (a^{\cdots\cdots\cdots\cdots})^2$

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 ..........:

        return ......................

    else:

        return ......................
```

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

Exercice 2 **Tri rapide**

On souhaite trier une liste de nombres dans l'ordre croissant. Il existe de nombreux algorithmes pour réaliser ceci, nous en avons déjà vu plusieurs (tri à bulles, tri par insertion, tri par sélection). Nous allons aujourd'hui travailler sur le tri rapide qui utilise un algorithme récursif.

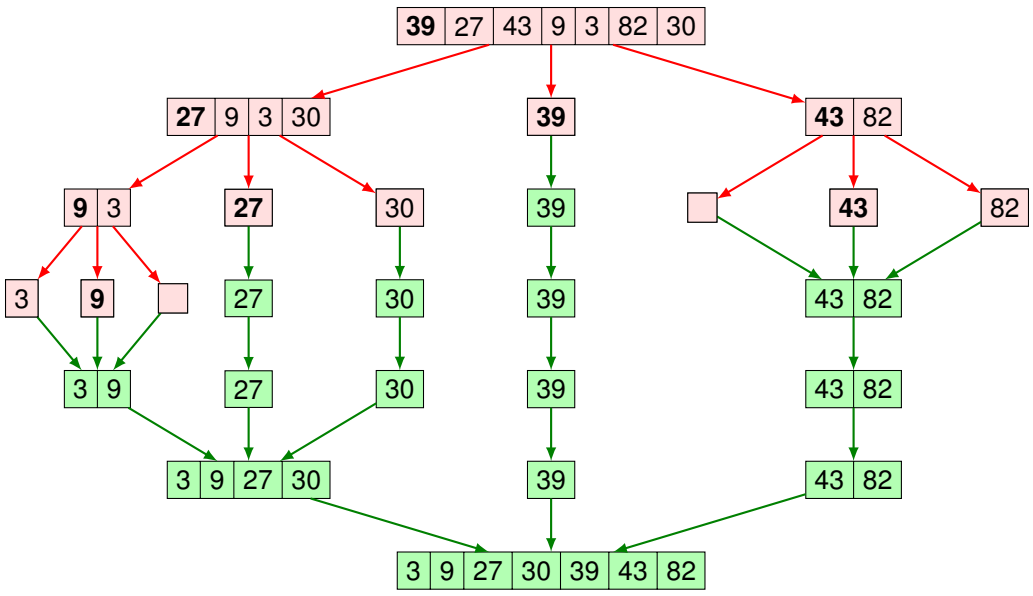
Principe du tri rapide :

— On choisit un pivot dans la liste. Nous choisirons le premier élément de la liste pour ce pivot, mais on pourrait faire autrement (dernier élément, pivot aléatoire, etc.).

— On sépare en trois : la liste des éléments inférieurs au pivot (partie 1), le pivot et la liste des éléments supérieurs au pivot (partie 3).

— On trie les parties 1 et 3, de manière récursive.

— On regroupe la liste triée : [partie 1 triée, pivot, partie 3 triée].



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 ?

3. Programmer le tri rapide.

```
def tri_rapide(L):
    if ..................... :
        return ..............
    else :
        pivot = ...........
        # Créer les parties 1 et 3









        return
```

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 .....

    else:

        if chaine[0] == .......:

            return ......................

        else:

            return  ...................
```