Tri rapide, Tri par paquets, et médiane

1. Partage, tri rapide (cours)

1. Écrire une fonction `partage(L,g,d)` prenant pour arguments une liste `L` de nombres (entiers ou flottants) et deux entiers `g` et `d`. On veut partager la partie de liste `L[g:d]`, en prenant comme pivot `L[g]`, et renvoyer l'indice final du pivot.
Par exemple, `partage1([4,6,3,2,1,40,5,7,15,9,10],2,8)` doit modifier la liste en `[4,6,2,1,3,5,40,7,15,9,10]` et renvoyer 4.
2. Écrire une fonction récursive `tri_rapide_rec(L,g,d)` qui prend en argument une liste `L` et deux indices `g, d` qui sont les limites du tableau et qui permet de trier la partie `L[g:d]` du tableau. On utilisera la fonction `partage(L,g,d)` définie précédemment.
3. À l'aide de la fonction précédente, écrire une fonction `tri_rapide(L)` renvoyant une copie triée de la liste `L`.
4. Modifier les fonctions précédentes de manière à retourner, outre la liste triée, le nombre total de comparaisons effectuées au cours du tri.
5. Tester votre programme sur une liste composée de 40 000 entiers aléatoirement choisis entre -999 et 999 (on pourra utiliser par exemple la fonction `randint` de la bibliothèque `random`).

2. Application : recherche de la médiane

On donne une version différente du partage :

```
def partage1(L,x):
    L1 = []
    L2 = []
    for elt in L:
        if elt<=x:
            L1.append(elt)
        else:
            L2.append(elt)
    return (L1,L2,len(L1)-1)
```

Une application courante des algorithmes de tri est la **recherche de la médiane** d'une liste. De façon naïve, on peut la trouver simplement en **triant** la liste et en **prenant le $(N//2)^{\text{ème}}$ élément** de la liste triée, où `N` est la longueur de la liste. Néanmoins, cela est rarement efficace car il faut trier la liste entière pour ne conserver au final qu'une seule valeur...

Une stratégie plus fructueuse est donc d'**adapter les algorithmes de tri**, du moins ceux qui sont récursifs, de sorte à minimiser la quantité de calculs. Pour cela, une idée possible est, à chaque appel récursif, de ne trier **que** la partie de la liste contenant la médiane cherchée. Le tri rapide s'y prête particulièrement bien.

6. **Après avoir compris ce que fait `partage1`, En adaptant le tri rapide, écrire une fonction `mediane_rec(L)` n'effectuant l'appel récursif que sur la partie contenant la médiane, et renvoyant celle-ci une fois la recherche terminée.**
7. **Tester cette fonction et évaluer sa complexité. La comparer à celle du tri rapide. Commenter.**