TD3 : Récursivité

Exercice 1.
Il existe deux définitions récursives de la fonction puissance :

$$a^b = \begin{cases} 1 & \text{si } b = 0 \\ a \times a^{b-1} & \text{sinon} \end{cases}$$

$$a^b = \begin{cases} 1 & \text{si } b = 0 \\ a \times a^{b-1} & \text{si } b \text{ est impair} \\ a^{\frac{b}{2}} a^{\frac{b}{2}} & \text{si } b \text{ est pair} \end{cases}$$

Pour chacune des définitions, donner l'algorithme récursif correspondant ainsi que sa complexité.

Exercice 2.
Utiliser le Master Theorem pour prouver que la complexité du tri fusion est $O(n \log n)$ sur une liste de taille n.

Exercice 3.
On considère une permutation P de n éléments, représentée dans un tableau. On cherche à évaluer le nombre d'inversions au sein de P, c'est à dire le nombre de paires $(i, j) \in [n]^2$ telles que $i < j$ mais j apparaît avant i dans P.

1. Compter le nombre d'inversions dans la permutation `[6,3,8,5,2,1,4,7]`.
2. Donner un algorithme quadratique pour résoudre ce problème.
3. On cherche maintenant à améliorer l'algorithme précédent en utilisant le paradigme Diviser pour Régner. Pour cela, on coupe la permutation P en deux sous-permutations P_0 et P_1. On compte alors les inversions (i, j) qui peuvent être de deux types : soit i et j apparaissent dans la même sous-permutation, soit i apparaît dans une sous-permutation et j apparaît dans l'autre (inversion mixte).
   (a) Supposons que P_0 et P_1 sont triées. Montrer que l'on peut compter les inversions mixtes en temps linéaire.
   (b) Donner un algorithme Diviser pour Régner qui compte le nombre d'inversions de n'importe quelle permutation P de n élements en temps $O(n \log n)$.