

Remarquons que les appels récursifs s'arrêteront forcément puisque les deux sous-listes sont de taille strictement inférieure à la liste initiale.

3.3 Complexité

- Déterminons la complexité $F(m)$ de la fonction fusion dans le cas où on fusionne deux listes de taille m

$$F(m) = 3 + 5 \times \underbrace{2m}_{\substack{\uparrow \\ \text{nombre maximal de tour de boucle}}} + 3 = \underline{6 + 10m}$$

Nous allons nous intéresser à la complexité de la fonction tri_fusion dans le cas d'une liste de départ de taille 2^p . Notons $C(2^p)$ cette complexité.

- Exprimons $C(2^p)$ en fonction de p et de $C(2^{p-1})$.

$$\begin{aligned} C(2^p) &= 2 + 2 + C(2^{p-1}) + C(2^{p-1}) + F(2^{p-1}) + 1 \\ &= 5 + 2C(2^{p-1}) + F(2^{p-1}) = 5 + 2C(2^{p-1}) + 6 + 10 \times 2^{p-1} \\ \boxed{C(2^p) &= 11 + 5 \times 2^p + 2C(2^{p-1})} \end{aligned}$$

- En notant $C(1) = \lambda$ on pourrait alors vérifier, en calculant de proche en proche les $C(2^p)$, que pour tout entier naturel p , $C(2^p) = (a + \lambda)2^p + bp2^p - a$
- En supposant que n est une puissance de 2, déterminons $C(n)$ en fonction de n :

$$\begin{aligned} n = 2^p \Leftrightarrow p = \log_2(n), \text{ donc } C(n) &= (a + \lambda)n + bn \log_2(n) - a \\ \boxed{C(n) &= O(n \log_2(n))} \end{aligned}$$

On a donc montré que la complexité, même dans le pire des cas, est en $O(n \log(n))$ (complexité **quasi-linéaire**).

4 Tri par insertion

4.1 Présentation

Le tri par insertion est sans doute le plus naturel. Il consiste à insérer successivement chaque élément dans l'ensemble des éléments déjà triés. C'est le procédé que l'on utilise, en général, pour classer un jeu de cartes (on pioche les cartes une par une et on range chaque carte piochée dans son jeu déjà trié)

Le tri par insertion s'effectue en place, son coût en mémoire est donc constant.

Son principe est d'insérer successivement chaque élément $a[i]$ dans la portion de la liste située avant l'indice i , qui est déjà triée.