# TD 3 : Analyse de Programmes et Récursivité (suite)

3 décembre 2023

**Exercice 5**

1) On note $C_1(n)$ le coût maximal du tri par insertion sur une liste de taille $n$.

Sur une entrée de taille 0, le coût est de 1 opération.

Sur une entrée de taille $n$, on aura un appel de la fonction de tri par insertion sur une liste de taille $n-1$ (coût $C(n-1)$), et l'insertion d'un élément dans la liste triée. Dans le pire cas, l'insertion d'un élément se fait en parcourant tous les éléments de la liste. Le coût de cette opération d'insertion est donc de $n-1$.

$C(n) = n-1+C(n-1) = n-1+n-2+C(n-2) = ... = (\sum_{i=1}^{n-1} i)+C(0) = \frac{n(n-1)}{2}+1 = \mathcal{O}(n^2)$ : la complexité dans le pire cas est quadratique.

2) On note $C_2(n)$ le coût maximal du tri fusion sur une liste de taille $n$.

Sur une entrée de taille 0 ou 1, le coût est de 1 opération.

Sur une entrée de taille $n$, un appel de la fonction tri fusion entraîne :

- Un appel de la fonction division, avec un coût en $\mathcal{O}(n)$ ($n$ insertions d'éléments dans une liste ou dans l'autre, en temps constant)
- Deux appels de la fonction tri fusion, sur des entrées de taille $\frac{n}{2}$ si $n$ est pair, et $\frac{n-1}{2}$ et $\frac{n+1}{2}$ si $n$ est impair ;
- Un appel de la fonction fusion avec un coût en $\mathcal{O}(n)$ ($n$ insertions d'éléments dans la liste, en temps constant)

Le coût cumulé de l'opération de division et de l'opération de fusion respectivement avant et après les appels récursifs sur la fonction tri fusion est donc borné par $Kn$ avec $K$ constant.

On a donc :

$C_2(n) \leq Kn + C(\frac{n-1}{2}) + C(\frac{n+1}{2})$ si $n$ est impair ;

$C_2(n) \leq Kn + 2 \times C(\frac{n}{2})$ si $n$ est pair.

Pour tous $n \leq n_0$, on a $C_2(n) \leq C_2(n_0)$. On pose $k = \lceil \log_2(n) \rceil$ et $n_0 = 2^k$ : on a $\log_2(n) \leq k = \log_2(n_0)$, d'où $n \leq n_0$.

$C_2(n) \leq C_2(n_0) \leq Kn_0 + 2C_2(\frac{n_0}{2}) \leq Kn_0 + 2 \times (K\frac{n_0}{2} + C_2(\frac{n_0}{4})$

$\leq \sum_{i=1}^{k} Kn_0 = k \times Kn_0 = K \times n_0 \log_2(n_0) \leq K(n+1)\log_2(n+1) = \mathcal{O}(n\log_2(n))$

La complexité dans le pire cas est donc en $\mathcal{O}(n\log(n))$.

3) On note $C_3(n)$ la complexité dans le pire cas du tri rapide sur une entrée de taille $n$.

Pour une entrée de taille 0, on a 1 opération.

Pour une entrée de taille $n$ : soit $k$ le nombre d'éléments inférieurs à l'élément pivot dans le pire cas. Un appel de l'algorithme sur une entrée de taille $n$ entraîne :

- Un déplacement des éléments à gauche et à droite du pivot, en $\mathcal{O}(n)$ ($n$ comparaisons au total)
- Deux appels du tri rapide respective sur l'ensemble des éléments à gauche et des éléments à droite du pivot, avec des coûts $C_3(k)$ et $C_3(n-k-1)$.

On a donc $C_3(n) \leq K \times n + C_3(k) + C_3(n-k-1)$. Le pire cas apparaît lorsque tous les éléments sont ensemble à gauche ou à droite du pivot, c'est-à-dire quand $k = 0$ ou $k = n-1$. Dans ce cas, $C_3(n) \leq K \times n + C_3(0) + C_3(n-1) \leq K \times n + 1 + K \times (n-1) + 1 + C_3(n-2) \leq$ $... \leq K\sum_{i=0}^{n}(i+1) = K\frac{(n+1)(n+2)}{2} = \mathcal{O}(n^2)$.

La complexité dans le pire cas du tri rapide est donc quadratique.

*Remarque : on peut se demander pourquoi le tri est appelé tri rapide si sa complexité dans le pire cas est d'un ordre de grandeur supérieur à celui du tri fusion. En réalité, le coût moyen sur l'ensemble des permutations possibles est en* $\Theta(n\log(n))$ *: on a en moyenne* $\log(n)$ *couches d'appels récursifs avec un coût total linéaire à chaque couche, comme dans le tri fusion. De plus, il s'agit dans le cas moyen d'un des algorithmes dont le coût réel est le plus rapide.*

*L'ordre de grandeur du pire cas apparaît sur des entrées où la quasi-totalité des éléments sont d'un côté du pivot, comme par exemple lorsque l'on choisit le premier élément comme pivot et que l'entrée est une liste presque triée. Pour éviter cela, on peut par exemple permuter au hasard les éléments de la liste, ou bien prendre à chaque appel un élément au hasard parmi les éléments de la liste comme pivot.*