Sur la Complexité de Quicksort (très optionnel)

(L1, cours Python n°6)

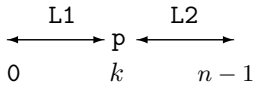
jpr, Nice, 12 Octobre 2016

1 Le Pire des Cas

L'exercice de TD 6.2 comportait une coquille. La complexité du *tri rapide* Quicksort n'est pas en $n\log(n)$ dans le pire des cas, mais en moyenne (voire dans le meilleur cas). Dans le pire des cas – par exemple où la liste `L` est déjà croissante et où le pivot est le premier élément `L[0]` – la phase de scission produit avec un coût $\mathcal{O}(n)$ pour `L1` la liste vide et pour `L2` la liste `L[1:]`. Le même phénomène se répète pour trier `L2` avec un coût de $n-1$, etc et l'on obtient un coût final en $1+2+3+...+n = n(n+1)/2 = \mathcal{O}(n^2)$. Le pire des cas a donc un coût quadratique. Mais il ne nous intéressait pas dans cet exercice.

2 L'Analyse en Moyenne

Nous sommes bien d'accord que cette lecture est optionnelle et hors-programme pour l'examen. Elle enchantera au moins les matheux. Les analyses d'algorithme en moyenne sont en général difficiles, car qui dit moyenne dit probabilité. J'essaye de faire au plus simple sans trop de dégâts.
Le pivot peut se retrouver au final en position k ($0 \leq k < n$) avec une probabilité $1/n$. S'il est en position k, la liste `L1` sera de longueur k et la liste `L2` sera de longueur $n-k-1$.



Dans ce cas, que vaut le coût c_n de trier une liste `L` de longueur n ? Il faudra payer un coût linéaire pour l'étape de scission de `L` en `L1` et `L2`, puis un coût c_k pour trier par récurrence `L1`, puis un coût c_{n-k-1} pour trier `L2`, puis enfin un coût linéaire pour concaténer `LT1`, le pivot et `LT2`. Simplifions le coût linéaire en simplement n (les fameuses *mathématiques de combat*[1]). Au final, pour $n \geq 2$:

$$c_n = c_k + c_{n-k-1} + n$$

Oui mais voilà, le pivot n'a qu'une chance sur n d'être en position k. On fait donc la moyenne pondérée de ces possibilités et il vient pour le cas *moyen* :

$$c_n = \frac{1}{n}\sum_{k=0}^{n-1}(c_k + c_{n-k-1} + n) = \frac{1}{n}\sum_{k=0}^{n-1}(c_k + c_{n-k-1}) + n$$

Travaillons un peu au chalumeau cette expression redoutable.

$$c_n = \frac{1}{n}(c_0 + c_{n-1} + c_1 + c_{n-2} + \cdots + c_{n-1} + c_0) + n$$

En regroupant les deux sommes inversées, on parvient à une forme un peu plus simple (E_n) :

$$c_n = \frac{2}{n}\sum_{k=0}^{n-1} c_k + n$$

1. Cherchez *street fighting math* sur Google...

soit encore :

$$nc_n = 2\sum_{k=0}^{n-1} c_k + n^2 \quad (E_n)$$

Ah-ah, on utilise alors l'astuce des *sommes télescopiques* en calculant $(E_n) - (E_{n-1})$. Les deux sommes vont en effet se détruire mutuellement et il ne restera que des miettes :

$$nc_n = 2\sum_{k=0}^{n-1} c_k + n^2 \quad (E_n)$$

$$(n-1)c_{n-1} = 2\sum_{k=0}^{n-2} c_k + (n-1)^2 \quad (E_{n-1})$$

et en faisant $(E_n) - (E_{n-1})$, le télescopage produit :

$$nc_n - (n-1)c_{n-1} = 2c_{n-1} + 2n$$

soit :

$$nc_n = (n+1)c_{n-1} + 2n$$

ou encore :

$$\frac{c_n}{n+1} = \frac{c_{n-1}}{n} + \frac{2}{n+1}$$

ce qui n'est qu'une définition par récurrence de la suite $(\frac{c_n}{n+1})$ dont il est facile de calculer le terme général en déroulant la récurrence :

$$\frac{c_n}{n+1} = \frac{c_{n-2}}{n-1} + \frac{2}{n} + \frac{2}{n+1} = \frac{c_{n-3}}{n-2} + \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1} = \cdots = \frac{c_1}{2} + (\frac{2}{3} + \frac{2}{4} + \cdots + \frac{2}{n} + \frac{2}{n+1})$$

d'où en introduisant la somme harmonique $\mathcal{H}_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$ et en tenant compte de $c_1 = 0$:

$$c_n = 2(n+1)(\mathcal{H}_{n+1} - \frac{3}{2}) \approx 2n\mathcal{H}_n$$

Or $\mathcal{H}_n \approx \ln(n) + \gamma$ où $\gamma \approx 0.577$ est la constante d'Euler [2] (la somme $1 + \frac{1}{2} + \frac{1}{3} + \cdots$ est donc infinie). Donc au final (ouf) :

$$c_n \approx 2n \ln n$$

Je ne sais pas vous, mais moi, ça m'a donné soif tout ça...

2. Voir par exemple sur Internet la Wikipédia à l'article *Série Harmonique*.