

ANNEXE C

*Diviser pour régner*

Hugo SALOU MPI\*

Dernière mise à jour le 1<sup>er</sup> mars 2023

# TABLE DES MATIÈRES

On se place dans un contexte dans l'idée de coder des algorithmes avec la stratégie « diviser pour régner. » Par exemple, on considère un algorithme de calcul de maximum divisant le tableau en deux à chaque itération.<sup>1</sup> Le calcul usuel de maximum est en  $\Theta(n)$ . On peut espérer que cet algorithme divisant le tableau ait une complexité logarithmique. Calculons cette complexité.

Soit  $(C_n)_{n \in \mathbb{N}}$  la suite donnant le nombre de comparaisons entre éléments du tableau<sup>2</sup> effectuées par l'algorithme `aux_max` sur un intervalle de taille  $|j - i + 1| = n$ . On a  $C_0 = 0, C_1 = 0$ , et pour tout  $n \geq 2, C_n = C_{\lfloor \frac{n}{2} \rfloor} + C_{\lceil \frac{n}{2} \rceil} + 1$ .

FIGURE 1

Soit  $(v_p)_{p \in \mathbb{N}}$  la suite définie par  $v_p = C_{2^p}$ . Ainsi,  $v_0 = 0$  et

$$v_{p+1} = C_{2^{p+1}} = C_{\lfloor \frac{2^{p+1}}{2} \rfloor} + C_{\lceil \frac{2^{p+1}}{2} \rceil} + 1 = 2v_p + 1.$$

Calculons  $v_p$  :

$$\begin{aligned} v_p &= 2v_{p-1} + 1 \\ &= 2(2v_{p-2} + 1) + 1 \\ &= 2^2 v_{p-2} + 2^1 + 2^0 \\ &\vdots \\ &= 2^p v_0 + \sum_{i=1}^{p-1} 2^i \\ &= 2^p - 1 \end{aligned}$$

Ce résultat ne s'applique que pour les puissances de 2, mais, par un argument de croissance, on peut en déduire un résultat pour tout entier  $n$ . Montrons que la suite  $(C_n)_{n \in \mathbb{N}}$  est croissante. En effet, par récurrence forte, soit  $n \geq 2$ .

— Si  $n$  est pair, alors  $C_n = 2 \times C_{\frac{n}{2}} + 1$  et  $C_{n-1} = C_{\frac{n-2}{2}} + C_{\frac{n-2}{2}+1} + 1 = C_{\frac{n}{2}-1} + C_{\frac{n}{2}} + 1$ .

Et,  $C_{\frac{n}{2}-1} \leq C_{\frac{n}{2}}$  par hypothèse de récurrence. D'où,  $C_n \geq C_{n-1}$ .

— De même si  $n$  est impair.

Ainsi, soit  $n \in \mathbb{N}^*$ . On pose alors  $p = \lfloor \log_2 n \rfloor$  donc  $p \leq \log_2 n$ . Par croissance de  $C$ , on a  $C_{2^p} \leq C_n \leq C_{2^{p+1}}$  donc  $v_p \leq c_n \leq v_{p+1}$ . Ainsi,

$$\begin{aligned} 2^p - 1 &\leq C_n \leq 2^{p+1} - 1 \\ 2^{\lfloor \log_2 n \rfloor} - 1 &\leq C_n \leq 2^{\lfloor \log_2 n \rfloor + 1} - 1 \\ 2^{\log_2(n)-1} - 1 &\leq C_n \leq 2^{\log_2(n)+1} \\ \frac{1}{2}n - 1 &\leq C_n \leq 2n - 1 \end{aligned}$$

Ceci étant vrai pour tout  $n \in \mathbb{N}^*$ , on a  $C_n = \Theta(n)$ . On peut remarquer que cet algorithme a une complexité équivalente à un algorithme itératif. Mais,<sup>3</sup> la complexité de cet algorithme s'améliore si les calculs se font en parallèles.

Autre exemple, le tri fusion a une complexité en  $C_n = C_{\lfloor \frac{n}{2} \rfloor} + C_{\lceil \frac{n}{2} \rceil} + n$ , ce qui donne une complexité en  $\Theta(n \log n)$ .

1. C'est le principe des tournois sportifs.  
2. *i.e.* le nombre d'appels à la fonction `max`.  
3. et c'est tout l'intérêt pour les tournois sportifs