Une Khôlle
Calcul de complexité

Amar AHMANE

31 janvier 2022

Énoncé On considère l'algorithme suivant implémenté en python

```
1                   def foo(lst):
2                           if len(lst) == 0:
3                                   return 1
4                           m = len(lst)//3
5                           return foo(lst[:m]) + foo(lst[m:])
```

En calculer la complexité au pire des cas.

Éléments de réponse Si on note $T(n)$ le nombre d'opérations élémentaires effectués par l'algorithme lorsque l'entrée est de taille n, on remarque que

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \mathcal{O}(1)$$

Ceci vient de l'appel récursif fait à la ligne 5 : le coût total est égal au coût des appels sur des entrées de tailles respectives $\frac{n}{3}$ et $\frac{2n}{3}$ plus celui de l'addition. On peut alors représenter dans un arbre binaire les appels récursifs et les additions qui sont faites

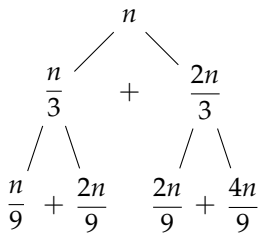

FIGURE 1 – Arbre représentant les appels récursifs à la fonction `foo`

La première chose que l'on remarque est que la valeur du noeud tout à gauche du dernier étage de l'arbre semble décroitre plus rapidement que celle du noeud tout à droite du dernier étage du fait de la présence d'une puissance de 2 en facteur. En fait, si l'on regarde de plus près, notamment en dessinant le troisième étage, on remarque un motif et on arrive à dénombrer les différentes tailles qui apparaîssent : faisons ça

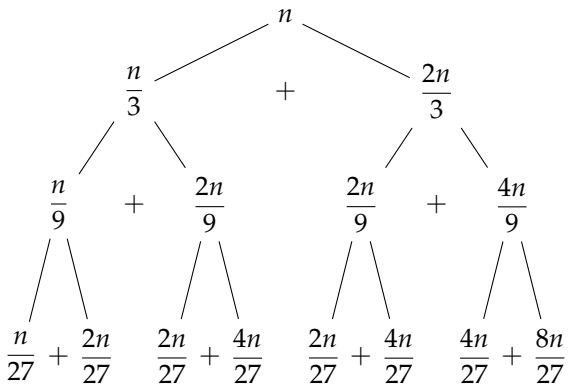

FIGURE 1 – Arbre représentant les appels récursifs à la fonction `foo`

On fait alors la conjecture que, pour un étage k donné, pour un entier $i \in [\![0, k]\!]$, le problème de taille $\frac{2^i n}{3^k}$ est traîté C_k^i fois.

Montrons cela par récurrence :

— Pour $k = 0$, tout est OK, il n'y a que la racine qui est traîtée C_0^0 fois i.e 1 fois.

— Pour un étage k donné, on suppose que notre conjecture est vérifiée. On montre qu'elle reste vraie à l'étage $k+1$. Soit $i \in [\![0, k+1]\!]$: pour arriver à un problème de taille $\frac{2^i n}{3^{k+1}}$, on a du soit partir d'un problème de taille $\frac{2^{i-1} n}{3^k}$ et descendre à droite, soit partir d'un problème de taille $\frac{2^i n}{3^k}$ et descendre à gauche : ceci n'est pas vrai si $i = 0$ ou $i = k+1$, en écartant ces cas particuliers, on a que le nombre de problèmes de taille $\frac{2^i n}{3^{k+1}}$ est égal à

$$C_k^i + C_k^{i-1} = C_{k+1}^i$$

Dans le cas où $i = 0$, il n'y a qu'une seule possibilité et pour $i = k+1$ aussi, ce qui achève de montrer que notre conjecture reste vraie à l'étage $k+1$.

— Conclusion : notre conjecture est vraie quel que soit l'étage auquel on se place.

On estime alors le nombre d'opérations élémentaires (c'est à dire le nombre d'additions) à la moitié du nombre de noeuds décrits par $(k, i) \in \mathbb{N}^2$ tels que $\frac{2^i n}{3^k} \geq 1$. C'est ainsi que, pour un étage k donné, et pour $i \in [\![0, k]\!]$, on notera

$$\beta_{ki} = \begin{cases} 0 & \text{si } \frac{2^i n}{3^k} < 1 \\ 1 & \text{sinon} \end{cases}$$

En se rappelant que le terme tout à droite du dernier étage est celui qui décroît le moins rapidement et qu'il représente un problème de taille $\frac{2^m n}{3^m}$ où m est l'étage, on en déduit qu'il détermine la hauteur de l'arbre puisqu'il est le dernier à être plus grand que 1, ainsi la hauteur est $m = \lfloor \log_{\frac{3}{2}}(n) \rfloor$. D'où la majoration suivante pour S, le nombre de sommes effectuées :

$$\begin{aligned} S &= \frac{1}{2} \sum_{k=0}^{m} \sum_{i=0}^{k} C_k^i \beta_{ki} \\ &\leq \sum_{k=0}^{m} \sum_{i=0}^{k} C_k^i \frac{2^i n}{3^k} \\ &\leq \sum_{k=0}^{m} \frac{n}{3^k} \sum_{i=0}^{k} C_k^i 2^i \\ &\leq \sum_{k=0}^{m} \frac{n}{3^k} 3^k \\ &\leq \sum_{k=0}^{m} n \\ &\leq (m+1)n = \mathcal{O}(n \log_{\frac{3}{2}}(n)) \end{aligned}$$

Ce qui achève de montrer que $T(n) = O(n \log_{\frac{3}{2}}(n))$.