TD8 : Master theorem

1 Rappels

Le master theorem est un théorème central pour l'étude de la complexité des algorithmes de type *diviser pour régner*. Lorsque la complexité $T(n)$ de l'algorithme en fonction de la taille n des données peut être amenée sous la forme

$$T(n) = aT\left(\frac{n}{b}\right) + f(n)$$

le théorème permet dans la plupart des cas de démêler la récurrence et de fournir la complexité directement en fonction de n.

1. si $f(n) = \mathrm{O}(n^c)$ avec $c < \log_b a$, alors la complexité de f est suffisamment faible devant la complexité liée à la récursion. Dans ce cas
$$T(n) = \Theta(n^{\log_b a}).$$
2. si $f(n) = \Theta(n^{\log_b a})$, alors les deux éléments interagissent et la complexité finale est donnée par
$$T(n) = \Theta(n^{log_b a} \log n).$$
3. si $f(n) = \Omega(n^c)$ avec $c > \log_b a$, alors la complexité de f est prépondérante devant la complexité liée à la récursion. Dans ce cas il n'est pas toujours possible de déterminer la complexité finale à cause des appels récursifs à f. Si f respecte le critère de régularité : $\exists k < 1, \exists n_0 \geq 0, \forall n \geq n_0, af(n/b) \leq kf(n)$, alors
$$T(n) = \Theta(f(n)).$$

2 Cas synthétiques

a) Exprimez la complexité de l'algorithme suivant sous la forme $T(n) = aT\left(\frac{n}{b}\right)+f(n)$, et résolvez sa complexité en utilisant le master theorem.

```
Fonction exo() → entier
    données : n : la taille de l'entrée
    Algorithme
        si n = 0 alors
            retourner 1
        sinon
            res ← 0
            pour i allant de 0 à n faire
                res = 2 × res
            pour i allant de 1 à 9 faire
                res ← res + exo(n/3)
            pour i allant de 0 à n faire
                pour j allant de 0 à n faire
                    pour k allant de 0 à j faire
                        res = res + 1
            retourner res
```

b) Pour chacune des formulations suivantes, donnez la complexité résultant de l'application du théorème :

$$T(n) = 16T\left(\tfrac{n}{4}\right) + \Theta(n) \qquad T(n) = 36T\left(\tfrac{n}{6}\right) + \Theta(n^2) \qquad T(n) = 5T\left(\tfrac{n}{3}\right) + \Theta(n^2)$$

3 Retour en primaire

Nous allons étudier ici un algorithme pour la multiplication de deux très grands nombres entiers. Nous ne considérerons donc plus que la multiplication ou l'addition de deux nombres sont des opérations réalisées en temps constant, mais plutôt que c'est l'addition et la multiplication de deux *chiffres* qui peut être réalisée en temps constant.

3.1 Souvenirs d'école

c) Posez l'addition de 54362 et 986473. Quelle est la complexité de cette opération en terme d'addition de chiffres, par rapport au nombre de chiffres n des nombres fournis ?

d) Posez la multiplication de 456 et 123. Quelle est la complexité de cette opération en terme d'additions et de multiplications de chiffres, par rapport au nombre de chiffres n des nombres fournis ?

3.2 Multiplication de Karatsuba

La multiplication de Karatsuba est un algorithme qui améliore la complexité de l'algorithme classique qui vous a été enseigné. Le principe consiste à couper les nombres en deux et calculer des multiplications de nombres deux fois plus petits pour obtenir le résultat final. Dans toute la suite vous supposerez pour simplifier que n est une puissance de 2.

Soit a un grand nombres à n chiffres. Soit a_1 le nombre constitué des $\frac{n}{2}$ chiffres les plus forts de a et a_0 le nombre constitué des $\frac{n}{2}$ chiffres les plus faibles. Mathématiquement, nous avons :

$$a = a_1 \times 10^{\frac{n}{2}} + a_0.$$

e) Quelle la complexité de la reconstruction de a à partir de a_1 et a_0 ?

Soient b un autre grand nombre à n chiffres, b_1 et b_0 les nombres définis de la même manière que a_1 et a_0 pour découper b.

f) D'après le master theorem, quelle est la complexité de la multiplication en la réalisant via la formule suivante :

$$a \times b = a_1 \times b_1 \times 10^n + (a_0 \times b_1 + a_1 \times b_0) \times 10^{\frac{n}{2}} + a_0 \times b_0$$

La multiplication de Karatsuba consiste à faire mieux en ne calculant que trois produits au prix de quelques sommes. Les produits $a_1 \times b_1$ et $a_0 \times b_0$ sont toujours calculés, et le troisième est $(a_0 - a_1) \times (b_1 - b_0)$.

g) Voyez vous comment obtenir le résultat à partir de ces trois produits ?

h) Quelle complexité obtenez vous via le master theorem ?