Algorithmique & Complexité

Tom Niget

NB : Je supposerai ici que les formules passées en entrée sont de la forme C=A×B.

1 Unaire, une bande

L'algorithme consiste à, B fois, soustraire A de C.

- Pour chaque chiffre (à droite) de B : n fois
  - Pour chaque chiffre de A, retirer le chiffre le plus à gauche de C : $O(n)$

Les deux boucles engendrent une complexité de $O(n^2)$.

Vu sous un autre angle, le nombre d'étapes sera approximativement de $C + (C - 1) + (C - 2) + \dots + A$, et avec $C \geq A$, cette valeur est de l'ordre de C^2, d'où une complexité totale de $O(n^2)$.

2 Unaire, trois bandes

On profite ici des trois bandes pour pouvoir faire un parcours linéaire (après copie initiale de valeurs sur les deux bandes). La troisième bande contient la valeur de A, la deuxième contient B. Chaque étape du parcours se déplace dans B en effaçant un chiffre de C, et en effaçant un chiffre de A dès qu'on arrive « au bout » de B, auquel cas on change de sens de déplacement de B et où on réitère. Lorsque C est vide, A doit également être vide.

- Atteindre le = ; recopier A ; recopier B : $O(n)$
- Tant que A non nul, soustraire B à C et décrémenter A après chaque soustraction : $O(n)$

L'algorithme est en $O(n)$.

3 Binaire, trois bandes

L'algorithme consiste à, pour chaque chiffre 1 de B, ajouter A à un accumulateur situé sur la deuxième bande, à la bonne position. Pour cela, la troisième bande stocke la position où le prochain ajout doit être effectué. Une fois cette multiplication longue effectuée, on compare l'accumulateur avec C.

- Atteindre la droite : $O(n)$
- Pour chaque chiffre de B : n fois
  - Si chiffre = 1
    - Ajouter A à l'accumulateur, puis revenir à droite : $O(n)$
  - Si chiffre = 0
    - Ne rien faire, simplement déplacer la tête de l'accumulateur : $O(1)$
  - Décrémenter B (retirer le chiffre le plus à droite) : $O(1)$
- Comparer C et l'accumulateur : $O(n)$

Dans le meilleur cas (B est une puissance de 2), l'algorithme s'exécutera en $O(n)$.

Dans le cas général, l'algorithme est en $O(n^2)$.

4 Binaire, une bande (efficace)

L'algorithme consiste à, B fois, soustraire A de C. On optimise un peu en détectant la parité ; si C est pair (finit par un 0), au moins un des deux facteurs doit l'être, et vice versa. Et si C est pair, on peut diviser toute l'équation par 2 (C ainsi qu'un facteur pair), et elle sera toujours vrai, mais on aura retiré un chiffre à traiter.

- Parcourir C : $O(n)$.
- Tant que C est non nul : n fois
  - Si le dernier chiffre de C est un 0, C est pair, un des deux facteurs est donc pair.
    - Diviser C par 2 : $O(1)$.
    - Aller jusqu'à la fin de A : $O(n)$
    - Si A est également pair, diviser A par 2 puis aller tout à droite : $O(n)$
    - Sinon, aller à la fin de B et diviser B (qui ne peut qu'être pair) par 2 : $O(n)$
  - Sinon, aller directement à la fin : $O(n)$
  - Si B est pair, le diviser par 2, puis retour à gauche et diviser C par 2 : $O(n)$
  - Sinon, décrémenter B et soustraire A à C : $O(n^2)$

Dans le meilleur cas (produit de puissances de 2), l'algorithme s'exécutera en $O(n^2)$ (car il n'effectuera qu'une longue suite de divisions par 2).

Dans le pire cas (produit de nombres de la forme $2^n - 1$), l'algorithme s'exécutera en $O(n^3)$ (car il devra alors effectuer une longue suite de soustractions avec seulement quelques divisions par 2).

Dans le cas général, il s'exécute en $\mathrm{O}(n^x)$, avec $2 \leq x \leq 3$. On peut donc dire qu'il est en $O(n^3)$.

5 Unaire, une bande (efficace)

Confer question 1.