Matthieu Boyer
Planche A

Question de Cours

Soient $f, g : \mathbb{N}^+ \to \mathbb{N}^+$. Démontrer ou réfuter les propriétés suivantes :

1. Si $f = \mathcal{O}(g)$, alors $g = \mathcal{O}(f)$.
2. $|\sin(n)| = \mathcal{O}(1)$.
3. $n^{100} = \mathcal{O}(2^n)$.
4. $1 + \frac{36!}{n} = \mathcal{O}(1)$.

Sumer

L'algorithme de Karatsuba est une méthode de multiplication de type diviser pour régner qui surpasse la méthode apprise à l'école. Pour multiplier deux nombres x et y de n chiffres en base B, on les divise en deux parties de taille $n/2$ telles que $x = a \cdot B^{n/2} + b$ et $y = c \cdot B^{n/2} + d$. Le produit complet s'écrit $ac \cdot B^n + (ad + bc) \cdot B^{n/2} + bd$. Alors que la méthode classique calcule les quatre produits ac, ad, bc et bd, Karatsuba remarque que le terme central peut être obtenu via l'identité $ad + bc = (a + b)(c + d) - ac - bd$. Cela ne nécessite que trois multiplications de nombres de taille $n/2$ (à savoir ac, bd et $(a + b)(c + d)$), réduisant ainsi la complexité à $O(n^{\log_2 3})$.

1. Démontrer la propriété de complexité ci-dessus.

En 1854, des archéologues ont découvert des tablettes d'argile sumériennes, gravées vers 2000 av. J.-C., listant les carrés d'entiers jusqu'à 59. Cette découverte a mené certains chercheurs à conjecturer que les anciens Sumériens effectuaient des multiplications par réduction au carré, en utilisant une identité comme $x \cdot y = (x^2 + y^2 - (x - y)^2)/2$. Malheureusement, ces mêmes chercheurs sont muets sur la manière dont les Sumériens auraient calculé les carrés de plus grands nombres.

2. Décrivez une variante de l'algorithme de Karatsuba qui calcule le carré de n'importe quel nombre à n chiffres en un temps $\mathcal{O}(n^{\log 3})$, en réduisant le problème au calcul du carré de trois nombres de $n/2$ chiffres.
3. Décrivez un algorithme récursif qui calcule le carré de n'importe quel nombre à n chiffres en un temps $\mathcal{O}(n^{\log_3 6})$, en réduisant le problème au calcul du carré de six nombres de $n/3$ chiffres.
4. Décrivez un algorithme récursif qui calcule le carré de n'importe quel nombre à n chiffres en un temps $\mathcal{O}(n^{\log_3 5})$, en réduisant le problème au calcul du carré de seulement cinq nombres de $(n/3 + \mathcal{O}(1))$ chiffres.

Matthieu Boyer
Planche B

Question de Cours

Soient $f, g : \mathbb{N}^+ \to \mathbb{N}^+$. Démontrer ou réfuter les propriétés suivantes :

1. $n = \mathcal{O}(\log n)$.
2. Si $\log^*(n) = k$ tel que $\log^k(n) = 1$ (log itéré), $\log^*(n) = \mathcal{O}(1)$.
3. $f(n) = \mathcal{O}(f^2(n))$.

Récursion

24. Considérez l'algorithme récursif classique suivant pour calculer la factorielle $n!$ d'un entier non négatif n :

```
FACTORIELLE(n) :
si n = 0
retourner 1
sinon
retourner n * FACTORIELLE(n - 1)
```

1. Combien de multiplications cet algorithme effectue-t-il ?
2. Combien de bits sont nécessaires pour écrire $n!$ en binaire ? On se souviendra que $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$.
3. Votre réponse en (b) devrait vous convaincre que le nombre de multiplications n'est pas une bonne estimation du temps d'exécution réel de FACTORIELLE. Quel est le temps d'exécution de FACTORIELLE si l'on considère que multiplier un nombre de k chiffres par un nombre de l chiffres prend un temps $O(k \cdot l)$?
4. L'algorithme récursif suivant calcule également la fonction factorielle, mais en utilisant un groupement différent des multiplications :

```
PRODUIT_DESCENDANT(n, m) : // Calcule n!/(n - m)!
si m = 0
retourner 1
sinon si m = 1
retourner n
sinon
retourner PRODUIT_DESCENDANT(n, sol(m/2)) * PRODUIT_DESCENDANT(n - sol(m/2),
plaf(m/2))
```

   Quel est le temps d'exécution de PRODUIT_DESCENDANT(n, n) si l'on utilise la multiplication apprise à l'école primaire ?
5. Décrivez et analysez une variante de l'algorithme de Karatsuba qui multiplie n'importe quel nombre de k chiffres et n'importe quel nombre de l chiffres, pour tout $k \geq l$, en temps $O(k \cdot l^{\log 3 - 1})$.

Matthieu Boyer
Planche C

Question de Cours

Soient $f, g : \mathbb{N}^+ \to \mathbb{N}^+$. Démontrer ou réfuter les propriétés suivantes :

1. Si $f = \mathcal{O}(g)$ alors $\log f = \mathcal{O}(\log g)$.
2. Si $f = \mathcal{O}(g)$ alors $2^{f(n)} = \mathcal{O}(2^{g(n)})$.
3. $f(n) = \mathcal{O}(f(n/2))$.

BLIIIIIIIIIIIIIIIIIIIIIT

La plupart du matériel graphique supporte une opération appelée blit, ou transfert de bloc, qui copie rapidement une zone rectangulaire d'une carte de pixels d'un endroit à un autre.

Supposons que nous voulions faire pivoter une carte de $n \times n$ pixels de 90 degrés dans le sens horaire. Une façon de faire, quand n est une puissance de 2, est de diviser la carte en quatre blocs de taille $n/2 \times n/2$, de déplacer chaque bloc vers sa position finale en utilisant une séquence de cinq blits, puis de faire pivoter récursivement chaque bloc.

1. Prouvez que les deux versions de l'algorithme (déplacement avant ou après la récursion) sont correctes quand n est une puissance de 2.
2. Exactement combien de blits l'algorithme effectue-t-il quand n est une puissance de 2 ?
3. Décrivez comment modifier l'algorithme pour qu'il fonctionne pour un n arbitraire. Combien de blits votre algorithme modifié effectue-t-il ?
4. Quel est le temps d'exécution de votre algorithme si un blit de taille $k \times k$ prend un temps $O(k^2)$?
5. Et si un blit de taille $k \times k$ ne prend qu'un temps $O(k)$?