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 $\exp \circ f = \mathcal{O}(\exp \circ g)$.
2. Si $f = \mathcal{O}(g)$ alors $\frac{1}{f} = \mathcal{O}(\frac{1}{g})$.

PGCD

Le plus grand commun diviseur de deux entiers positifs x et y, noté $pgcd(x, y)$, est le plus grand entier d tel que x/d et y/d soient des entiers.

```
EUCLIDE_PGCD(x, y) :
si x = y
retourner x
sinon si x > y
retourner EUCLIDE_PGCD(x - y, y)
sinon
retourner EUCLIDE_PGCD(x, y - x)
```

1. Prouvez que EUCLIDE_PGCD calcule correctement $pgcd(x, y)$. Plus précisément :
   (a) Prouvez que EUCLIDE_PGCD(x, y) divise à la fois x et y.
   (b) Prouvez que chaque diviseur de x et y est un diviseur de EUCLIDE_PGCD(x, y).
2. Quel est le temps d'exécution dans le pire des cas de EUCLIDE_PGCD(x, y), en tant que fonction de x et y ?
3. Prouvez que l'algorithme suivant calcule également le pgcd.

```
FAST_EUCLIDE_PGCD(x, y) :
si y = 0
retourner x
sinon si x > y
retourner FAST_EUCLIDE_PGCD(y, x mod y)
sinon
retourner FAST_EUCLIDE_PGCD(x, y mod x)
```

4. Quelle est sa complexité, en supposant que $x \mod y$ prend $\mathcal{O}(\log x \cdot \log y)$?
5. Prouvez que l'algorithme suivant (PGCD binaire) calcule aussi $pgcd(x, y)$.

```
BINAIRE_PGCD(x, y) :
si x = y retourner x
sinon si x et y sont pairs retourner 2 * BINAIRE_PGCD(x/2, y/2)
sinon si x est pair retourner BINAIRE_PGCD(x/2, y)
sinon si y est pair retourner BINAIRE_PGCD(x, y/2)
sinon si x > y retourner BINAIRE_PGCD((x - y)/2, y)
sinon retourner BINAIRE_PGCD(x, (y - x)/2)
```

6. Quelle est la complexité de cet algorithme, en supposant que $x - y$ se calcule en $\mathcal{O}(\log x + \log y)$ et $z/2$ en $\mathcal{O}(\log z)$?

Matthieu Boyer
Planche B

Question de Cours

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

1. Si $f = \mathcal{O}(g)$ alors $f^2 = \mathcal{O}(g^2)$.
2. Si $f(n) = \min\{k \mid k^2 \leq n\}$, alors $f(n) = \mathcal{O}(\sqrt{n})$.

Karatsuba

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 ci-dessus.
2. Décrivez et analysez une variante de l'algorithme de Karatsuba qui multiplie n'importe quel nombre de m chiffres avec n'importe quel nombre de n chiffres, pour tout $n \geq m$, en temps $\mathcal{O}(nm^{\log 3 - 1})$.
3. Décrivez un algorithme pour calculer la représentation décimale de 2^n en temps $\mathcal{O}(n^{\log 3})$, en utilisant l'algorithme de la partie précédent comme sous-programme.
4. Décrivez un algorithme de type diviser pour régner pour calculer la représentation décimale d'un nombre binaire arbitraire de n bits en temps $\mathcal{O}(n^{\log 3})$.
5. Supposons que nous puissions multiplier deux nombres de n chiffres en temps $\mathcal{O}(M(n))$. Décrivez un algorithme pour calculer la représentation décimale d'un nombre binaire arbitraire de n bits en temps $\mathcal{O}(M(n) \log n)$.

Matthieu Boyer
Planche C

Question de Cours

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

1. $n^2 = \mathcal{O}(n^3)$.
2. Si $f(n) = \mathcal{O}(g(n))$ et f et g sont des polynômes à coefficients entiers positifs, alors $\deg \frac{f}{g} \leq 0$.

Cibles et massacres récursifs

À la fin du second acte du blockbuster Fast and Impossible XIII : Les derniers gardiens de la justice renouvelée, le vilain Dr. Metaphor hypnotise la Hero League entière, les dispose en une longue ligne au bord d'une falaise, et ordonne à chaque héros de tirer sur les héros les plus proches qui sont plus grands qu'eux, à leur gauche et à leur droite, à un signal prédéfini.

Supposons que nous ayons les tailles de n héros, dans l'ordre de gauche à droite, dans un tableau $Ht[1..n]$. Pour éviter les disputes de salaire, les producteurs insistent pour que deux héros n'aient jamais la même taille. Nous pouvons alors calculer les cibles de gauche et de droite de chaque héros en temps $O(n^2)$ en utilisant l'algorithme de force brute suivant :

```
QUI_CIBLE_QUI(Ht[1..n]) :
pour j allant de 1 à n
// Trouver la cible de gauche L[j] pour le héros j
L[j] = AUCUN
pour i allant de 1 à j - 1
si Ht[i] > Ht[j] alors
L[j] = i
// Trouver la cible de droite R[j] pour le héros j
R[j] = AUCUN
pour k allant de n vers j + 1 par pas de -1
si Ht[k] > Ht[j] alors
R[j] = k
retourner L[1..n], R[1..n]
```

1. Décrivez un algorithme qui calcule la sortie de QUI_CIBLE_QUI en temps $O(n \log n)$.
2. Prouvez qu'au moins $n/2$ des n héros sont des cibles. C'est-à-dire, prouvez que les tableaux de sortie $L[1..n]$ et $R[1..n]$ contiennent au moins $n/2$ valeurs distinctes (différentes de AUCUN).
3. Hélas, le plan diabolique du Dr. Metaphor réussit. Au signal, tous les héros tirent simultanément sur leurs cibles, et toutes les cibles tombent de la falaise, apparemment mortes. Dr. Metaphor répète son expérience encore et encore : après chaque massacre, il force les héros restants à choisir de nouvelles cibles selon le même algorithme, puis à tirer sur leurs cibles au signal suivant. Finalement, seul le membre le plus petit de la Hero League reste en vie. Décrivez et analysez un algorithme pour calculer le nombre de rounds avant que le processus mortel du Dr. Metaphor ne se termine. Pour obtenir tous les points, votre algorithme doit s'exécuter en temps $O(n)$.