# DS INFO N°3

## Exercice 1 (Questions de cours - 11 minutes).

1. Quelle est la complexité **spatiale** de l'algorithme des tours de Hanoï ? Donner une justification en quelques lignes et/ou avec un petit dessin, sans y passer trop de temps. (3 minutes)
2. Écrire en OCaml l'algorithme d'exponentiation rapide dans sa version **itérative** (5 minutes)
3. Donner (sans justification) les **complexités temporelles asymptotiques** des algorithmes suivants, vus en cours ou en TD/TP. Vous préciserez de quoi dépend cette complexité et expliquerez les notations choisies. La réponse se fera en recopiant et remplissant le tableau suivant sur votre copie (prévoir suffisamment de place pour la deuxième colonne). (3 minutes)

| Algorithme | Complexité temporelle au pire |
|---|---|
| Recherche dichotomique | |
| Exponentiation naïve | |
| Tours de Hanoï | |
| Tri à bulles | |
| Exponentiation rapide | |
| Tri insertion | |
| Recherche séquentielle | |
| Multiplication russe | |
| Algorithme de recherche naïf de motif dans un texte | |
| Concaténation de deux listes | |

## Exercice 2 (Division euclidienne et algorithme d'Euclide).

**1.** On définit le *quotient* $q$ et le *reste* $r$ de la division euclidienne de $a \in \mathbb{Z}$ par $b \in \mathbb{N}^*$ (donc $b > 0$) comme l'unique couple d'entiers $q \in \mathbb{Z}$ et $r \in \mathbb{Z}$ vérifiant $a = bq + r$ avec $0 \leq |r| < b$. On dit que $a$ est divisible par $b$ si le reste de la division euclidienne de $a$ par $b$ vaut 0.

Pour cette question, on pourra se limiter au cas où $a \geq 0$.

**a.** **Pour cette question seulement, la multiplication et la division sont interdites.** Écrire une fonction **itérative** en langage C

```
void div_euclid(int a, int b, int *q, int *r)
```

qui reçoit deux paramètres entiers `a` et `b` et qui calcule le quotient et le reste de la division euclidienne de $a$ par $b$ . On respectera scrupuleusement le prototype imposé pour cette fonction. On pourra faire quelques exemples sur la copie pour se donner des idées et ensuite généraliser.

**b.** Prouver la terminaison de votre fonction.

**c.** Prouver la correction de votre fonction.

**d.** Quelle est la complexité temporelle de cette fonction ?

**2.** Le *plus grand diviseur commun* de deux entiers $a$ et $b \geq 0$, noté $\text{pgcd}(a, b)$, est le plus grand entier **positif** qui divise à la fois $a$ et $b$.

**a.** Pour un nombre $m \in \mathbb{Z}$, que vaut $\text{pgcd}(m, 0)$ ? Justifier de manière claire et concise.

**b.** Montrer que

$$\forall m \in \mathbb{Z}, \forall n \in \mathbb{N}, \text{pgcd}(m, n) = \text{pgcd}(n, r) \tag{1}$$

où $r$ est le reste de la division euclidienne de $m$ par $n$.

**c.** Utiliser cette propriété de manière répétée pour calculer le PGCD de $a = 63$ et $b = 11$ en décrivant toutes les étapes du calcul.

**d.** En vous aidant de cet exemple, écrire en **C** un algorithme **itératif**

```
void euclide(int a, int b, int *d)
```

permettant de calculer le PGCD de deux nombres $a$ et $b$. *Indication : prenez le temps de valider toutes les étapes de votre algorithme, sur plusieurs exemples, en faisant tourner l'algorithme à la main.*

**e.** L'algorithme fonctionne-t-il pour $a < b$ ? Que se passe-t-il précisément ?

**f.** Prouver rigoureusement la correction partielle de votre algorithme. *Indication :* on pourra s'appuyer sur le résultat (Equation 1) prouvé précédemment.

**g.** Montrer la *correction totale* de votre fonction.

**h.** Écrire en OCaml une version **récursive** de l'algorithme d'Euclide.

## Exercice 3 (Algorithme de calcul de racine).

Soit $a$ et $b$ deux nombres réels tels que $a < b$ et $f : [a, b] \to \mathbb{R}$ une fonction continue strictement monotone telle que $f(a)$ et $f(b)$ sont de signes opposés.

1. Montrer qu'une telle fonction admet une **unique racine** : $\exists! \, x_0 \in [a, b]$, $f(x_0) = 0$. *Indication : On pourra notamment faire appel à un théorème du cours de mathématiques.*
2. Soit $\alpha$ et $\beta$ deux réels non nuls. Quel test (un seul test) permet de répondre à la question : « $\alpha$ et $\beta$ sont-ils de signes opposés ? »
3. Soit $c$ la valeur centrale de l'intervalle $[a, b]$ et $f(c)$ son image par $f$. En quoi la connaissance de la valeur $f(c)$ permet d'avancer dans la recherche de la racine ?
4. A partir de la constatation de la question précédente, écrire en OCaml un algorithme `racine` permettant de calculer, à $\varepsilon$ près, la valeur de la racine d'une fonction $f$ respectant les critères ci-dessus. La fonction valeur absolue est incluse dans la bibliothèque standard OCaml et se nomme `abs_float`.
5. Expliquer pourquoi la fonction suivante vérifie bien les préconditions de votre algorithme :
$$\begin{array}{rccc} f : & [0.25, 0.8] & \to & \mathbb{R} \\ & x & \to & \cos(x\pi) \end{array}$$
Quelle est la réponse attendue si l'on donne cette fonction en entrée de l'algorithme ?
6. A l'aide de la calculatrice, tester pas à pas à la main votre algorithme sur la fonction précédente avec $\varepsilon = 10^{-2}$, en laissant une trace de votre test, par exemple en faisant un tableau de suivi de variables.
7. Prouver la terminaison de l'algorithme. *Indication : on sort légèrement du cadre mathématique vu dans le cours mais cela ne pose pas de problème particulier ici. Faites vous confiance, la logique mathématique reste la même.*
8. De quels paramètres dépend la complexité de l'algorithme ? Donnez une expression asymptotique de la complexité dans le pire des cas.