Algorithmique
TD n° 1

Exercice 1 : Ordre de grandeur

- Dans un calcul de complexité d'une fonction f, on note souvent une expression du type $f(n) \in O(g(n))$ ou $f(n) \in \Theta(g(n))$ Que représente n dans cette expression ? Que veut dire O ? Que veut dire Θ ?
- Ordonner les fonctions suivantes par ordre asymptotique de grandeur. (Si $f(n) \in O(g(n))$, on notera $f << g$).
  n^2, $lg(n)$, $n\sqrt{n}$, $3n$, $lglg(n)$, n, n^3, $2n$, $n!$, $nlg(n)$, $\sqrt{n}$
- Peut-on dire :
  - $n \in O(n)$, $n \in \Theta(n)$, $n^2 \in O(2^n)$
  - $n \in O(n^2)$, $n \in \Theta(n^2)$, $n^2 \in \Theta(2^n)$,
  - $n^2 \in O(n)$, $n^2 \in \Theta(n)$, $\sqrt{n} + n \in \Theta(n)$
  - $3 * lg(n) \in \Theta(lg(n))$, $nlg(n) \in O(n^2)$, $n + lg(n) \in O(n)$
  - $f(n) \in \Theta(g(n)) \implies g(n) \in \Theta(f(n))$
  - $f(n) \in O(g(n)) \implies g(n) \in O(f(n))$
  - $f(n) + g(n) \in O(h(n)) \implies f(n) \in O(h(n))$ et $g(n) \in O(h(n))$
  - $f(n) + g(n) \in \Theta(h(n)) \implies f(n) \in \Theta(h(n))$ et $g(n) \in \Theta(h(n))$
  - $f(n) * g(n) \in O(h(n)) \implies f(n) \in O(h(n))$ et $g(n) \in O(h(n))$
  - $f(n) * g(n) \in \Theta(h(n)) \implies f(n) \in \Theta(h(n))$ et $g(n) \in \Theta(h(n))$

Exercice 2 : Découpe de chocolat

Un tablette de chocolat est composée de $n * m$ carreaux. On souhaite séparer tous les carreaux, et pour cela la seule opération dont on dispose est de prendre un morceau de chocolat et de le couper en deux verticalement ou horizontalement (en suivant une ligne ou une colonne).

- Proposer un algorithme pour séparer tous les carreaux. Quelle est sa complexité ? Appliquer le sur une tablette $2 * 5$.
- Quel est le nombre minimal d'opérations permettant de séparer tous les carreaux (quel que soit l'algorithme) ? Votre algorithme est-il optimal ?

Exercice 3 : Retour en primaire, apprenons à multiplier

Imaginons que l'opérateur $*$ pour la multiplication n'existe pas. Nous allons ici redéfinir des algorithmes pour effectuer cette opération sur des entiers positifs et calculer leur complexité.

- L'algorithme suivant est-il juste ? Quelle est sa complexité ?

```
uint mult(uint a, uint b) {
    uint ret_val = 0;
    for (uint i=0 ; i<a ; i++) ret_val += b;
    return ret_val;
}
```

- Lorsque vous effectuez une multiplication à la main (comme appris en primaire), quel algorithme utilisez-vous ? Écrire cet algorithme et déduire sa complexité.
- Est-il meilleur que celui présenté avant ? Pouvez-vous le prouver ?
- À partir de l'algorithme précédent, écrire un algorithme puissance qui calcule la puissance b d'un entier a ($a, b \in \mathbb{N}$). Quelle est sa complexité ?
- Superbonus - Une semaine avant d'avoir eu le cours, pouvez-vous écrire un second algorithme puissance qui a une complexité $lg(nb\ mult)$?

Exercice 4 : Attrapez les tous

Il y a quelques années, le jeu Pokémon Go sur smartphone a connu un gros succès. L'un des principes de ce jeu est de se rendre physiquement à certains endroits pour aller y capturer des pokémons à l'aide de son smartphone. Imaginons que l'on puisse connaître à l'avance la position géographique (coordonnées x et y sur une carte plane) de chacun des pokémons qui nous intéressent (uniquement les 151 premiers bien évidemment !). Nous connaissons également notre position de départ x_0, y_0.

- pouvez-vous écrire un algorithme qui calcule le plus court chemin à parcourir pour tous les capturer (l'algorithme n'a pas besoin d'être efficace) ? Quelle est sa complexité ?
- Pouvez-vous créer un algorithme significativement mieux ?