DM – Miller-Rabin

Ci-suit les réponses aux questions posées.

Question 1

Nous avons choisi le langage de programmation Python qui nous permet de travailler plus efficacement sur les nombres de grande taille. Nous avons utilisé la bibliothèque secrets qui permet de gérer des nombres entiers de grande taille. Cette bibliothèque permet de générer de manière sécurisée des données aléatoires. Les opérations de base ne sont pas implémentées dans cette bibliothèque, cependant, Python utilise ses opérateurs intégrés pour effectuer des opérations sur les nombres de grande taille automatiquement (comme l'addition, la soustraction, la multiplication, la division, etc...).

Question 2

Un nombre aléatoire est un nombre dont chaque chiffre est obtenu par tirage au sort à égalité de chances. C'est-à-dire que ce nombre ne peut pas être prédit. En Python, la bibliothèque qui permet de générer des aléatoires est la bibliothèque secrets (celle-ci permettant de générer des nombres aléatoires de plusieurs milliers de bits).

Question 3

La fonction Decomp() est implémentée dans le fichier q3.py.

Question 4

La fonction ExpMod() est implémentée dans le fichier q4.py.

Question 5

La fonction MillerRabin() est implémentée dans le fichier q5.py.

Question 6

Les tests des nombres en hexadécimal sont effectués dans le fichier q6.py.

GEHIN Sandy
BELUCHE Quentin

Question 7

La fonction Eval() est implémentée dans le fichier q7.py.

Question 8

La fonction Eval() est testée dans le fichier q8.py.

Question 9

Nous constatons que plus la taille du nombre est grande, plus le nombre d'itérations nécessaires pour trouver un nombre probablement premier est grand. Cela est dû au fait que plus la taille du nombre est grande, plus il est difficile de trouver un nombre premier. Cela est dû à la complexité de l'algorithme de Miller-Rabin qui est en O(k * log(n)^3) avec k le nombre d'itérations et n la taille du nombre. Ainsi, plus la taille du nombre est grande, plus le nombre d'itérations nécessaires pour trouver un nombre probablement premier est grand.

Question 10

Le test qui permet de garantir si un nombre est premier ou non est le test de primalité AKS. La complexité de ce test est assez variante, si l'on raffine l'algorithme et qu'on utilise différentes variantes celui-ci peut atteindre une complexité de Ω(log(n)^7.5) Ce qui fait de ce test un test très lent pour les grands nombres. De plus, sa complexité temporelle est de Ω(log(n)^10.5) ce qui le rend encore plus lent. Sachant qu'un algorithme se dit réellement utilisable à au plus Ω(log(n)^3) de complexité temporelle, l'algorithme AKS est donc inutilisé en pratique.

GEHIN Sandy
BELUCHE Quentin