Devoir Maison – Test de Miller-Rabin

Master 1 - Informatique
UE Introduction à la cryptographie

Madison NOYER, Theo FAEDO

Session 2024

Table des Matières

1 Introduction

Ce projet, réalisé en duo par Madison NOYER et Théo FAEDO, se concentre sur le test de Miller-Rabin, un outil essentiel en théorie des nombres pour déterminer la primalité d'un nombre. Notre objectif principal est de comprendre et d'implémenter ce test de manière efficace.

La documentation est divisée en deux parties. Dans la première, un rapport PDF répond en détail aux questions posées, offrant des explications claires et des justifications solides. La seconde partie comprend notre programme, écrit en Python, un fichier makefile, un fichier readme et un fichier test.txt comprenant la sortie du terminal.

Ce projet démontre notre compréhension du test de Miller-Rabin, notre capacité à l'implémenter correctement, et à le valider rigoureusement par des tests spécifiques. Les tests sont automatisés et affichent un message seulement en cas d'échec. À l'exception du test sur n1, n2 et n3 qui affiche directement le résultat en sortie de terminal.

Explorez les sections suivantes pour une vue détaillée de notre approche et de nos résultats.

2 Avant d'implémenter le test de Miller-Rabin

Question 1: Nous avons opté pour le langage de programmation Python en raison de sa simplicité, de sa lisibilité du code et de la richesse de ses bibliothèques.

Python gère automatiquement les nombres entiers de grande taille grâce à la bibliothèque intégrée appelée "bigint" ou "integers longs". Donc python prend en charge cela nativement, ce qui simplifie grandement le processus de développement en éliminant le besoin d'importer des bibliothèques supplémentaires pour la manipulation de grands entiers.

La bibliothèque intégrée de Python pour la gestion des entiers de grande taille, appelée "bigint" ou "integers longs", implémente toutes les opérations arithmétiques de base que l'on peut effectuer sur des entiers. Elle permet également d'effectuer des opérations de comparaison entre les entiers de grande taille

Question 2: Définition : Un nombre aléatoire est un nombre dont chaque chiffre est obtenu par tirage au sort à égalité de chances.

En informatique, un nombre aléatoire provient d'un processus imprévisible. Dans le cadre de notre projet, où des nombres de plusieurs milliers de bits sont requis, nous privilégions la bibliothèque secrets de Python pour garantir une génération de nombres aléatoires de haute qualité et de grande taille.

La bibliothèque secrets offre une fonction secrets.randbits(n) qui génère un entier de n bits de manière cryptographiquement sécurisée. Cette méthode est particulièrement adaptée aux besoins de notre projet. On utilise également secrets.randbelow(n) pour générer un nombre aléatoire dans la plage allant de 0 à n-1, ce qui peut être utile dans les cas où vous avez besoin d'un nombre aléatoire dans une plage spécifique.

3 Implémentation et tests du test de Miller-Rabin

Question 6: Pour chaque nombre, nous allons effectuer un test pour déterminer s'ils sont pseudo-premiers ou composés avec cpt = 20.
Voici les résultats :
n1 (768 bits) = 1
n2 (768 bits) = 0
n3 (1024 bits) = 1

Nous concluons que n2 est composé tandis que n1 et n3 sont probablement premiers.

Question 9: Sur le graphique, avec la taille en bits du nombre en abscisse et la moyenne des 100 tests de la valeur de compteur en ordonnée, on observe que la moyenne augmente à mesure que la taille de bits du nombre augmente. Grâce à cette visualisation, il est possible de constater la création d'une droite, ce qui indique une relation proportionnelle entre la taille de bits et la moyenne des itérations.

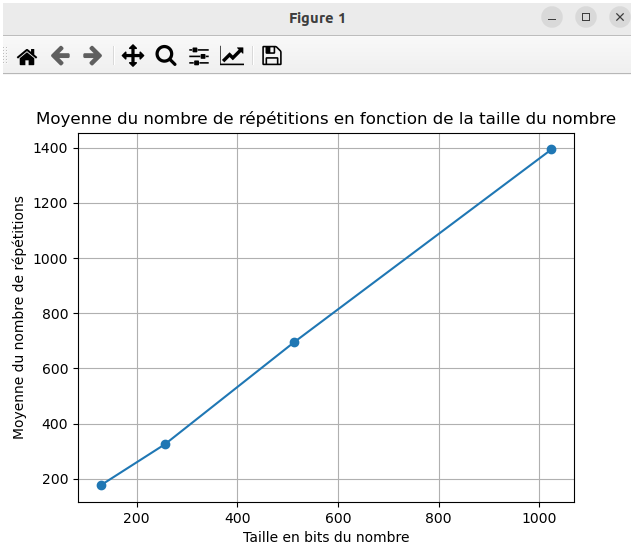

Figure 1: Taille de b utilisée : 128, 256, 512, 1024

Question 10: Le test qui permet de garantir si un nombre est premier ou non est appelé le "test de primalité de AKS".Ce test est basé sur des techniques algorithmiques différentes et plus efficaces que les tests probabilistes comme le test de Miller-Rabin.

La complexité du test de primalité de AKS est polynomiale. Plus précisément, sa complexité est de l'ordre de $O((\log n)^6)$, où n est le nombre testé.