Algorithme Rabin-Miller (1980)

L'algorithme de Miller-Rabin est une méthode de test de primalité probabiliste. Conçu par Michael Rabin en 1980, il est basé sur une extension des idées du test de primalité développé par Gary Miller. C'est un outil crucial en cryptographie, notamment pour la génération de grands nombres premiers utilisés dans les protocoles de chiffrement tels que RSA.

Principes de base :

Le test de Rabin-Miller repose sur des notions de théorie de nombres, en particulier sur le petit théorème de Fermat, qui dit que si ‘p’ est un nombre premier et ‘a’ est un entier non-divisible par ‘p’ alors :

A^(p-1) = 1 mod p

Pour un nombre impair n que le voudrai tester, on donne n-1 comme 2^s * d où ‘d’ est impair. L’algorithme vérifie ensuite si ‘n’ vérifie bien certaine règles de nombre premier.

Déroulement de l’algorithme :

1) **Réduction de n-1 :**
   - Trouver ‘d’et ‘s’ tels que n-1 = 2^s * d, avec ‘d’ impair
2) **Choix de ‘a’ :**
   - Prendre aléatoirement un ‘a’ entre [2, n-2]
3) **Exponentiation modulaire :**
   - Calculer : x = a^d mod n, si x = 1 ou x = n-1 alors ‘n’ est probablement premier.
4) **Carrés successifs :**
   - Calculer : x^2 mod n, s-1 fois ou jusqu’à ce que x = n-1, si x = n-1 alors ‘n’ est probablement premier.
5) **Observation :**
   - Si aucune condition n’est remplis, ‘n’ n’est probablement pas premier.
   - On peut répéter ces étape plusieurs fois pour augmenter la véridicité de ce test.

Choix de A :

Importance du choix de ‘a’ :

Le choix de la base ‘a’ dans le test de primalité de Miller-Rabin est une étape déterminante pour l'efficacité et la précision du test. Voici pourquoi le choix de ‘a’ est si crucial et comment il peut influencer les résultats du test :

- Rabin-Miller est probabiliste, ce qui peut dire que sa capacite a déterminer si un nombre est premier dépend de la base ‘a’, il est préférable que ‘a’ soit un « bon témoin » pour réduire la probabilité d’erreurs.

Comment choisir ‘a’ :

- La méthode la plus courante est de choisir ‘a’ aléatoirement dans l'intervalle [2, n-2]
- Pour les nombres inférieurs à certaines limites, il a été prouvé que tester avec des petites bases prédéfinies peut suffire pour confirmer la primalité. Par exemple, il est connu que pour

  n< 2, 152, 302, 898, 746, 322, 56, il suffit de tester,
  a= 2, 3, 5, 7, 11, 13, 17, 19, 23, 29,31, 37
  Pour déclarer ‘n’ premier avec certitude.

Exponentiation modulaire

L'exponentiation modulaire est une opération mathématique qui consiste à élever un nombre à une certaine puissance et à prendre le résultat modulo un autre nombre. Elle permet de réaliser des calculs efficaces sur de grands nombres tout en contrôlant leur taille pour éviter les débordements :

Principes de base

L’exponentiation modulaire calcule a^b mod m où :

- ‘a’ est la base
- ‘b’ est l’exposant
- ‘m’ est le modulo

Le but est de trouver le reste de la division de a^b par m. (a^b) pouvant être très grand on se sert de ‘m’. Cette méthode utilise des propriétés mathématiques pour simplifier les calcule et les garder « gérable ».

Carrés successifs

Après avoir calculé x= a^d mod n où 'a' est une base aléatoire et 'd' est tel que n-1 = 2^s * d

Avec 'd' impair, le test de Miller-Rabin ne s'arrête pas si 'x' n'est ni 1 ni n-1. A ce stade, le test entre dans une phase de vérification supplémentaire appelée "carrés successifs" :

1) **Initialisation :**
   - On commence avec 'x', qui vient de x = a^d mod n.
   - Si x = 1 ou x = n-1, le test s'arrête et 'n' est probablement premier (pour la base 'a' donnée).
2) **Boucle :**
   - On calcule x = x^2 mod n, est on répète ce calcule jusqu'à ce que (x = 1) ou (x = n-1) ou jusqu'à (s -1) fois.
   - Si x = n-1, le test s'arrête et 'n' est probablement premier pour la base 'a' testé.
   - Si x = 1, alors 'n' est probablement non-premier pour la base 'a' testé, le test s'arrête.
3) **Arrêt**
   - Si l'algorithme se termine sans que x = n-1, alors n est non-premier

Probabilité et complexité

Probabilité d'erreur :

- Chaque test avec une base 'a' différente a une probabilité d'au moins 1/4 de dire qu'un nombre non premier 'est premier', d'où l'utilité de répéter le test, avec k test effectue la probabilité d'erreur et de 4^(-k).

Complexité :

- La complexité de l'algorithme est de l'ordre $O((\log n)_3)$, ceci est du grâce au multiplication modulaire.

Utilisation dans le code :

Utilisation Mult_mod :

- **Fonctionnalité :**

  La fonction Mult_mod effectue la multiplication modulaire, une opération pour manipuler de grand nombre sans risquer de dépassement (overflow)

  Elle calcule : (factor1 * factor2) modulo (mod)

- **Application Rabin-Miller :**

  Dans les test Rabin-Miller, Mult_mod est utilisé pour réaliser des multiplications répétitives dans les calcule des carrés successifs. Le modulo assure que même de très grand nombre reste dans la limite gérable.

Utilisation Expo_mod :

- **Fonctionnalité :**

  La fonction Expo_mod effectue l'exponentiation modulaire qui est importante pour réduire la complexité de l'exponentiation de grands nombres.

  Elle calcule : base^(exp) modulo (mod)

- **Application Rabin-Miller :**

  Dans le test Rabin-Miller, Expo_mod est utilisé pour calculer :

  a^d mod n dans la première phase du test où 'a' est une base aléatoire et 'd' est dérivé de (n-1), cette partie aide grandement à déterminer s un nombre est probablement premier ou non.

Utilisation Est_premier :

- **Fonctionnalité :**

  La fonction Est_premier contient l'ensemble du « test de primalité » de Rabin-Miller, elle utilise « Mult_mod » et « Expo_mod » pour vérifier si un nombre donné est premier avec une probabilité qui dépend du nombre de tours effectué.

- **Application Rabin-Miller :**

  Dans le test Rabin-Miller, Est_premier est concu pour tester la primalité de grands nombres (nécessaire pour la sécurité des algorithmes de chiffrement a clé publique comme RSA). En augmentant le nombre de round on augmente la fiabilité du résultat donné.

Conclusion :

Les fonctions Mult_mod et Expo_mod dans est_premier permettent d’effectuer le test Rabin-Miller de manière optimisé. Ces fonctions travaillent ensemble pour tester de grand nombre en maintenant les opérations dans la limite gérable et en assurant la précision nécessaire pour ce test de primalité. Cette approche est essentielle pour les applications de sécurité ou la certitude de la primalité est crucial, et ou la performance ne peut pas être sacrifié.