Licence informatique - L3 Année 2009/2010

Conception d'algorithmes et applications (LI325)

COURS 10

Résumé. Nous consacrons cette séance aux algorithmes probabilistes de type Monte Carlo. Nous présentons deux algorithmes issus de la théorie des nombres que sont le test de primalité de Miller-Rabin et la détermination des racines carrées modulaires.

1. Algorithmes Probabilistes

1.1. **Les algorithmes de type Monte Carlo.** On dira qu'un algorithme est de type *Monte Carlo d'erreur* $\epsilon < 1$, si, pour toute entrée concordante avec la spécification, la probabilité que cet algorithme retourne un résultat erroné vaut au plus ϵ. On remarquera qu'à la différence des algorithmes de type Las Vegas, le temps d'exécution des algorithmes de type Monte Carlo est déterministe. L'aléa ici est dans la fiabilité de la réponse renvoyée.

Dans le cas des problèmes de décision, c'est à dire quand l'algorithme retourne oui ou non, on distinguera deux raffinements :

Un algorithme est de type Monte Carlo à *erreur unilatérale* si pour toute instance pour laquelle la réponse est non, l'algorithme ne se trompe jamais (il retourne non avec probabilité 1). Par contre, il peut répondre non avec probabilité non nulle sur des instances où la réponse devrait être oui. En conséquence, quand un algorithme de type Monte Carlo à *erreur unilatérale* retourne oui, on est sûr qu'il renvoie la bonne réponse.

Un algorithme est de type Monte Carlo à *erreur bilatérale* si l'algorithme peut se tromper avec probabilité non nulle sur au moins une instance dont la réponse doit être oui et une instance dont la réponse doit être non.

1.1.1. *Application du principe Monte Carlo à la théorie des nombres : le test de primalité de Miller-Rabin.* Savoir déterminer si un nombre est premier est un problème fondamental en cryptologie. De nombreux algorithmes de cryptage reposent sur des problèmes d'arithmétique modulo de très grands nombres premiers (RSA, log discret...). Or déterminer en temps polynomial en n si un nombre à n chiffres est premier ou non est un problème très difficile. Il a fallu attendre 2002, pour qu'un algorithme déterministe polynomial soit enfin trouvé. Avant, il existait des algorithmes déterministes mais la preuve que leur complexité était polynomiale utilisait l'hypothèse de Riemann généralisée (qui reste une conjecture !)

Néanmoins, il existe des algorithmes probabilistes de type Monte Carlo très efficaces et très simples à implémenter. Nous allons décrire ici l'un deux qui est le test de Miller-Rabin.

Le test de Miller-Rabin est, stricto sensus, un test de non primalité de type Monte Carlo à erreur unilatérale 1/4. En d'autres termes, s'il renvoie non premier, alors le nombre n'est pas premier et s'il renvoie premier alors avec 3 chances sur 4, le nombre est vraiment premier.

Le test de Miller-Rabin repose sur la propriété suivante :

Propriété 1. *Soit p un nombre premier impair et $a \in \{1, ..., p-1\}$, on note r l'unique entier impair tel que $p - 1 = r.2^s$ alors :*

- *soit $a^r = 1 \mod p$*
- *soit $a^{r2^j} = -1 \mod p$ pour un entier j, $0 \leq j \leq s-1$.*

Preuve. Par le petit théorème de Fermat, $a^{p-1} = a^{r2^s} = 1 \mod p$. Donc, ou bien $\forall j$, $0 \leq j \leq s$, $a^{r2^j} = 1 \mod p$ et en particulier $a^r = 1 \mod p$. Ou bien il existe un j tel que $a^{r2^j} \neq 1 \mod p$, on prend alors le plus grand j possible. Ce choix de j implique que, $(a^{r2^j})^2 = a^{r2^{j+1}} = 1 \mod p$. Donc a^{r2^j} est une racine carrée modulaire de 1, c'est à dire 1 ou -1. Pour ne pas contredire les hypothèses il s'ensuit que $a^{r2^j} = -1 \mod p$. □

Définition 1. *Soit n un entier impair et $a \in \{1, ..., n-1\}$, soient l'entier r impair et l'entier s tels que $n-1 = r.2^s$. On dit que a est un* témoin fort de non-primalité *pour n, si $a^r \neq 1 \mod n$ et $\forall j, 0 \leq j \leq s-1$, $a^{r.2^j} \neq -1 \mod n$.*

Un nombre premier n'a donc pas de témoin fort de non-primalité (heureusement sinon la terminologie aurait été sérieusement mal appropriée !). Le lemme que nous ne démontrons pas ici (voir Cours d'algèbre de Demazure, pour la preuve) explique l'emploi de témoin *fort* :

Lemme 1. *Tout entier n, impair et non premier, a au moins $3(n-1)/4$ témoins forts de non-primalité.*

Le test de Miller-Rabin cherche des témoins de non-primalité forts. En voici le pseudo-code :

Algorithme 1 : Test de Miller-Rabin

Entrées : Un entier n impair.
Sorties : Composé ou probablement premier

```
1  TEST(n)
2  r := n − 1 ;
3  s := 0 ;
4  tant que r est pair faire
5  |  r := r/2 ; s := s + 1 ;
6  Tirer un entier a uniformément dans {1, ..., n − 1};
7  x := a^r mod n;
8  si x = 1 ou x = n − 1 alors
9  |  retourner probablement premier
10 j := 1;
11 tant que j < s faire
12 |  x := x^2 mod n;
13 |  si x = n − 1 alors
14 |  |  retourner probablement premier
15 |  si x = 1 alors
16 |  |  retourner Composé
17 |  j := j + 1;
18 retourner probablement premier
```

Analyse de l'algorithme Test de Miller-Rabin :

Preuve de Terminaison :
Le premier Tant que ligne 4 se termine (car par exemple r diminue à chaque itération et reste positif). Le deuxième Tant que ligne 11 se termine après au plus s itérations.

Preuve de Validité :
Nous cherchons ici à vérifier que le pseudo-code renvoie composé ssi le nombre a tiré aléatoirement est un témoin fort de non-primalité de n. Tout d'abord le Tant que ligne 4 permet de calculer l'unique entier impair r et l'entier s tels que $n-1 = r.2^s$. Ligne 8 si $x = a^r = \pm 1 \mod n$ alors a n'est pas un témoin fort de non-primalité de n et l'on renvoie probablement premier. La boucle Tant que ligne 11 permet de calculer les valeurs $a^{r.2^j} \mod n$ pour j variant de 1 à $s-1$, de plus, si à une étape de la boucle on trouve $x = -1$ alors a n'est pas un témoin fort de non-primalité de n et l'on peut renvoyer probablement premier. De même, si $x = 1$ et que x n'a jamais valu -1 avant, alors il est inutile de continuer le calcul des puissances (toutes égales à 1 après), on peut immédiatement conclure que a est un témoin fort de non-primalité de n et l'on peut renvoyer Composé.

Analyse de la Complexité en nombre de multiplications modulaires.
Ligne 7, il faut calculer a^r, en utilisant une approche diviser pour régner, on peut effectuer ce calcul en $O(\ln(r))$ multiplications. Comme $r \leq n$ ceci compte pour $O(\ln(n))$ multiplications. La boucle Tant que ligne 11 est effectuée au plus s fois et $s < \log_2(n)$. Comme à chaque itération on doit

faire une multiplication, cette boucle compte pour $O(\ln(n))$ multiplications. L'algorithme effectue donc $O(\ln(n))$ multiplications modulaires. (Un produit modulo n peut être réalisé en $O(\ln(n)^2)$.

Le test de Miller-Rabin décrit ci-dessus renvoie, si n est un entier impair non premier, Composé avec une probabilité supérieure à 3/4. Ceci est clairement trop faible pour un usage sûr tel que la cryptologie l'exige. Néanmoins, et c'est là la grande force des algorithmes de type Monte Carlo, si l'on relance k fois le test de Miller-Rabin sur le même entier impair non premier n la probabilité que le test se trompe, c'est à dire qu'il renvoie les k fois Probablement premier, est de $(1/4)^k$ ce qui décroit exponentiellement vite vers 0 quand k tend vers l'infini.

En pratique, 6 appels (au lieu de 44) au test de Miller-Rabin suffisent pour avoir un résultat sûr à 2^{-80} près. Ceci est du au fait que le lemme 1 donnant une borne du nombre de témoins forts est très lache. Le test de Miller-Rabin est donc aujoud'hui encore, l'un des tests de primalités les plus efficaces connus.

1.1.2. *Le calcul de racines carrées modulaires.* En cryptologie, un certain nombre de cryptages repose sur des applications mathématiques computationnellement asymétriques, on dit qu'une application f est *computationnellement asymétrique* si f est une bijection de A dans B, telle que le calcul de $f(a)$ peut se faire en temps polynomial pour tout $a \in A$, alors que le calcul de $f^{-1}(b)$ est difficile (au moins NP-Complet). De telles applications existent, par exemple l'exponentiation modulaire et sa réciproque le logarithme discret (voir cours de crypto). Nous allons montrer ici que le carré modulaire n'est pas (d'un point de vue probabiliste) une application computationnellement asymétrique.

Un entier x est un *résidu quadratique modulo* p, s'il existe un entier y tel que $0 < y < p$ et $y^2 = q \mod p$. L'entier y est alors une *racine carrée de* x *modulo* p. Il est bon de remarquer que 0 n'est pas considéré comme un résidu quadratique bien que $0^2 = 0$.

Commençons par un petit rappel d'algèbre qui nous sera bien utile par la suite :

Propriété 2. *$\mathbb{Z}/p\mathbb{Z}$ est un corps ssi p est premier. Dans un corps, un polynôme de degré n a au plus n racines distinctes.*

Donc, un résidu quadratique a au plus 2 racines carrées distinctes. Si y est une racine carrée de x modulo p, alors $-y$ est aussi et y et $-y$ sont différents mod p (p est impair). Par conséquent, les $p-1$ entiers $\{1, \ldots, p-1\}$ sont les racines carrées d'exactement $\frac{p-1}{2}$ résidus quadratiques différents : *exactement la moitié des entiers non nuls modulo p sont des résidus quadratiques.*

Nous allons dans un premier temps donner une caractérisation des résidus quadratiques :

Propriété 3. *Soit p un nombre premier impair et $x \in \{1, ..., p-1\}$. On a pour tout x, $x^{(p-1)/2} = \pm 1$, de plus $x^{(p-1)/2} = 1$ si et seulement si x est un résidu quadratique modulo p.*

Preuve. Posons $z = x^{(p-1)/2} \mod p$, on a $z^2 = 1 \mod p$ car par le petit théorème de Fermat $x^{(p-1)} = 1 \mod p$. Donc z est une racine modulo p de 1 et vaut 1 ou -1. Ceci conclut la première partie de la proposition.

Si x est un résidu quadratique, il existe un entier y tel que $0 < y < p$ et $y^2 = x \mod p$. D'après le petit théorème de Fermat : $x^{(p-1)/2} = y^{(p-1)} = 1 \mod p$.

Dans $\mathbb{Z}/p\mathbb{Z}$, une équation polynômiale ne peut avoir plus de solutions que son degré. Nous connaissons déjà $(p-1)/2$ racines à l'équation $x^{(p-1)/2} = 1 \mod p$ (les résidus quadratiques), donc aucun non-résidu ne peut en être solution. □

Il est donc assez rapide de tester si un entier x est un résidu quadratique, en faisant une exponentiation modulaire. Notre problème est donc de trouver l'une des deux racines modulaires d'un résidu quadratique. Remarquez que si l'on a trouvé une racine x alors l'autre est $-x$. Il y a un fait assez surprenant dans ce problème, suivant que p est congru 1 ou 3 modulo 4 la solution est simple ou difficile à trouver. Voici la partie simple :

Propriété 4. *Soit p un nombre premier tel que $p = 4k + 3$ alors si x est un résidu quadratique, une racine modulaire de x est x^{k+1}.*

Preuve. Comme x est un résidu quadratique, $x^{(p-1)/2} = 1$ avec $(p-1)/2 = 2k+1$, on a donc $x^{2k+2} = x$. Il s'ensuit que x^{k+1} est une racine carrée modulo p. □

Il reste donc à traiter le cas où p est de la forme $p = 4k+1$, il n'est connu à l'heure actuelle, aucun algorithme déterministe permettant de trouver une racine carrée modulaire en temps polynomial pour ce cas. Néanmoins, il existe un algorithme de type Monte-Carlo assez simple à temps moyen polynomial que l'on va décrire.

Soit x un résidu quadratique et $\sqrt{x}$ sa racine carrée modulaire qui se trouve dans $\{1, ..., (p-1)/2\}$. Si a, b, c et d sont des entiers, en prenant $e = ac + xbd \mod p$ et $f = ad + bc \mod p$ on a, sans connaître la valeur $\sqrt{x}$, la formule :

$$(a + b\sqrt{x})(c + d\sqrt{x}) = e + f\sqrt{x} \mod p$$

Il s'ensuit qu'il est possible en temps polynomial (en faisant une exponentiation modulaire) de calculer u et v tels que $(\alpha + \beta\sqrt{x})^{(p-1)/2} = (u + v\sqrt{x})$ sans connaître explicitement la valeur $\sqrt{x}$.

Voici le pseudo-code de l'algorithme :

Algorithme 2 : Calcul de racine carrée modulo p

Entrées : Un entier p premier impair et un résidu quadratique x modulo p.
Sorties : Une racine carrée modulo p de x ou ECHEC

1 **si** $p = 3 \mod 4$ **alors**
2 **retourner** $x^{(p+1)/4} \mod p$
3 Tirer uniformément un entier a dans $\{1, ..., p-1\}$;
4 **si** $a^2 = x \mod p$ **alors**
5 **retourner** a
6 Calculer c et d tels que $(a + \sqrt{x})^{(p-1)/2} = (c + d\sqrt{x}) \mod p$;
7 **si** $c \neq 0 \mod p$ **alors**
8 **retourner** *ECHEC*
9 **retourner** $d^{-1} \mod p$

Preuve de Terminaison :
Immédiat.
Preuve de Validité :
Il faut prouver que d^{-1} est bien une racine carrée de x. Tout d'abord $(a+\sqrt{x})^{(p-1)/2} = \pm 1$ car son carré vaut 1 par le petit théorème de Fermat. Deuxièmement, on remarque que $(a - \sqrt{x})^{(p-1)/2} = c - d\sqrt{x}$. Il suffit pour cela de développer $(a + \sqrt{x})^{(p-1)/2}$. Donc $(a - \sqrt{x})^{(p-1)/2}$ vaut aussi ± 1. Il y a alors deux cas, soit $(a + \sqrt{x})^{(p-1)/2}$ et $(a - \sqrt{x})^{(p-1)/2}$ sont de même signe (c'est à dire tous les deux égaux à 1 ou tous les deux égaux à -1). Dans ce cas, on a $c - d\sqrt{x} = c + d\sqrt{x}$ et donc $d = 0$ et $c = \pm 1$; soit $(a + \sqrt{x})^{(p-1)/2}$ et $(a - \sqrt{x})^{(p-1)/2}$ sont de signes contraires, on a alors $-c + d\sqrt{x} = c + d\sqrt{x}$ et donc $c = 0$ et $d\sqrt{x} = \pm 1$. On a donc que si $c = 0$ alors $d\sqrt{x} = \pm 1$ et donc $d^{-1} = \pm\sqrt{x}$ est une racine carrée de x.
Analyse de la Complexité en nombre de multiplications modulaires.
Si l'on passe Ligne 3, il faut calculer $x^{(p+1)/4} \mod p$, en utilisant une approche diviser pour régner, on peut effectuer ce calcul en $O(\ln(p))$ multiplications. Le calcul de c et d ligne 7 nécessite lui aussi de l'ordre de $O(\ln(n))$ multiplications. L'algorithme effectue donc $O(\ln(n))$ multiplications modulaires. (Un produit modulo n peut être réalisé en $O(\ln(n)^2)$).
Il reste à déterminer avec quelle probabilité l'algorithme retourne ECHEC. On se place dans le cas où p est de la forme $4k+1$ (c'est à dire le cas intéressant). On remarque que $c \neq 0$ ssi $(a+\sqrt{x})^{(p-1)/2}$ et $(a - \sqrt{x})^{(p-1)/2}$ sont de même signe et donc dans ce cas $(a + \sqrt{x})^{(p-1)/2}.(a - \sqrt{x})^{(p-1)/2} = (a^2 - x)^{(p-1)/2} = 1$. Donc $c \neq 0$ ssi $a^2 - x$ est un résidu quadratique. Il faut donc compter le nombre de résidus quadratiques de la forme $a^2 - x$ avec $a \in \{1, ..., p-1\}$. Pour cela, prenons l'application $f(y) = \frac{y-\sqrt{x}}{y+\sqrt{x}}$. Cette application est une bijection de $\{1, ..., p-1\} \setminus \{\sqrt{x}, p - \sqrt{x}\}$ dans $\{2, ..., p-2\}$ (Il suffit de montrer qu'elle est injective). Or, $a^2 - x$ est un résidu quadratique ssi $f(a)$ est un résidu quadratique. Le nombre de résidus quadratiques dans $\{2, ..., p-2\}$ est $(p-1)/2 - 2$ car 1 et -1 ne sont pas dans l'ensemble. En conséquence, lorsque p est de la forme $4k + 1$, la probabilité que l'algorithme retourne ECHEC est $\frac{p-5}{2(p-1)} < 1/2$.