COMPLEX
Algorithmes Probabilistes

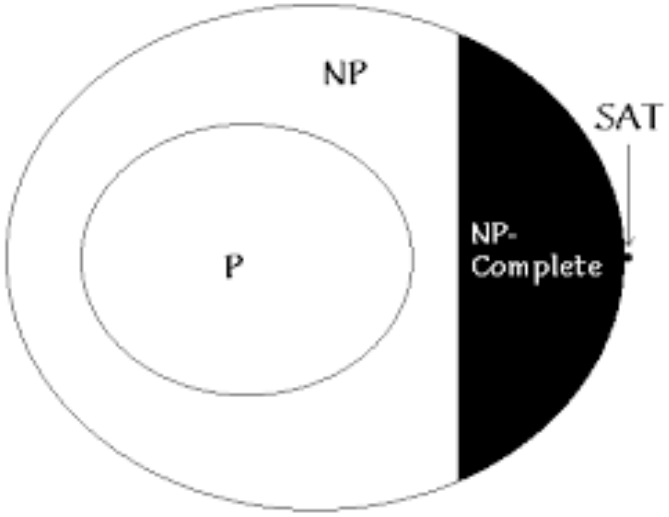
Ludovic Perret

UPMC Univ. Paris 06 & INRIA
LIP6

Outline

Algorithme Déterministe

- Il retourne toujours une solution **correcte**, et
- pour une **même entrée**, il retourne toujours la **même réponse** (et toujours avec la **même complexité**)



https://openclassrooms.com/

Algorithme Probabiliste

- Pour une **même entrée**, l'algorithme ne retourne **pas toujours la même réponse** (et pas toujours avec la même complexité)
- la sortie n'est pas forcément correcte

https://www.authenticfx.com/

Las-Vegas

Définition

- Un algorithme de type **Las-Vegas** est un algorithme probabiliste qui retourne toujours un résultat correct
- Le temps d'exécution **peut varier** ; la complexité d'un algorithme de Las-Vegas est une **variable aléatoire**

- `QuickSort` avec pivot aléatoire
  - $\Theta(n^2)$ dans le pire cas,
  - $\Theta(\log(n) \cdot n)$ en moyenne.

`http://instants-captures.fr/`

Monte-Carlo

Définition

- Un algorithme de type **Monte-Carlo** est un algorithme qui termine toujours en temps polynomial
- Le résultat **n'est pas toujours correct**, mais on contrôle la probabilité d'erreur

http://www.casinosenfrance.net/

Outline

Motivation

Contexte

Cryptographie à clef publique $\equiv$ fonction à sens unique avec trappe.

- Problème *difficile* (dans NP).

FACT

- **Entrée :** un entier non nul $N \in \mathbb{N}$
- **Question :** trouver un diviseur *non-trivial* p de N

PREMIER

- **Entrée :** un entier non nul $N \in \mathbb{N}$
- **Question :** N est-il premier ?

http://viterbi.usc.edu/

Primalité

Génération de Grands Premiers

Soit $\pi(N) = \#\{n \in [1, \ldots, N] \mid n \text{ premier}\}$. Alors :

$$\pi(N) \approx_{N \to \infty} \frac{N}{\log(N)}.$$

Primalité

Génération de Grands Premiers

Soit $\pi(N) = \#\{n \in [1, \ldots, N] \mid n \text{ premier}\}$. Alors :

$$\pi(N) \approx_{N \to \infty} \frac{N}{\log(N)}.$$

Historique

- PREMIER $\in$ NP.

Primalité

Génération de Grands Premiers

Soit $\pi(N) = \#\{n \in [1, \ldots, N] \mid n \text{ premier}\}$. Alors :

$$\pi(N) \approx_{N \to \infty} \frac{N}{\log(N)}.$$

Historique

- PREMIER $\in$ NP.
- Tests de primalité probabilistes.
  - Si TestPREMIER(N) retourne composé, alors N est toujours composé
  - Si TestPREMIER(N) retourne premier, alors N est peut être premier

Primalité

Génération de Grands Premiers

Soit $\pi(N) = \#\{n \in [1, \ldots, N] \mid n \text{ premier}\}$. Alors :

$$\pi(N) \approx_{N \to \infty} \frac{N}{\log(N)}.$$

Historique

- PREMIER $\in$ NP.
- Tests de primalité probabilistes.
  - Si TestPREMIER(N) retourne composé, alors N est toujours composé
  - Si TestPREMIER(N) retourne premier, alors N est peut être premier
- Algorithme déterministe en $O(\log(N)^{15/2+\varepsilon})$, pour $\varepsilon > 0$.

M. Agrawal, N. Kayal, et N. Saxena.
"PRIMES is in P".
Annals of Mathematics, 2004.

Outline

L'ensemble des entiers modulaires

Deux entiers a et b sont dits congruents modulo N, noté $a \equiv b \bmod N$ si, et seulement si :

$$N \mid a - b.$$

Définition

$$\mathbb{Z}_N = \{\overline{0}, \ldots, \overline{N-1}\}.$$

http://fr.123rf.com/

Addition/Soustraction

Pour tout $a, b \in \mathbb{Z}_N$:

$$\overline{a} \oplus \overline{b} = \overline{s} \text{ avec } s \equiv (a+b) \mod N.$$
$$\overline{a} \ominus \overline{b} = \overline{s} \text{ avec } s \equiv (a-b) \mod N.$$

Complexité : $O(\log(N))$.

$\oplus$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{0}$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{1}$	$\overline{1}$	$\overline{2}$	$\overline{0}$
$\overline{2}$	$\overline{2}$	$\overline{0}$	$\overline{1}$

$$\mathbb{Z}_3 = \{\overline{0}, \overline{1}, \overline{2}\}$$

Multiplication dans $\mathbb{Z}_N$

Multiplication

Pour a, b dans $\mathbb{Z}_N$:

$$\overline{a} \otimes \overline{b} = \overline{s} \text{ avec } s \equiv (a \times b) \mod N.$$

Complexité : $O(\log^2(N))$.

$\otimes$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$
$\overline{2}$	$\overline{0}$	$\overline{2}$	$\overline{1}$

$$\mathbb{Z}_3 = \{\overline{0}, \overline{1}, \overline{2}\}$$

$\otimes$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$	$\overline{0}$
$\overline{1}$	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
$\overline{2}$	$\overline{0}$	$\overline{2}$	$\overline{0}$	$\overline{2}$
$\overline{3}$	$\overline{0}$	$\overline{3}$	$\overline{2}$	$\overline{1}$

$$\mathbb{Z}_4 = \{\overline{0}, \overline{1}, \overline{2}, \overline{3}\}$$

Éléments Inversibles

Théorème

Soit $x \in \mathbb{Z}_N$, alors x est inversible modulo N si et seulement si $\mathrm{pgcd}(x, N) = 1$.

- La complexité du pgcd est en $O(\log^2(N))$.

Définition

L'ensemble des éléments inversibles modulo N est noté $\mathbb{Z}_N^\times$.
Pour un entier $N \in \mathbb{N}$, on note :

$$\varphi(N) = \#\{x \in [1, \ldots, N-1] \mid \mathrm{pgcd}(x, N) = 1\}.$$

- Si p est premier, $\varphi(p) = p - 1$.

Calcul de l'inverse modulaire

Proposition

Calculer l'inverse modulo N de $x \in \mathbb{Z}_N^\times$ est en $O(\log^2(N))$.

Principe

Il existe (u, v) t.q. $u \cdot x + v \cdot N = \text{pgcd}(x, N) = 1$, c'est à dire :

$$u \cdot x \equiv 1 \mod N.$$

Calculer l'inverse de x, c'est donc calculer u (et v).

- On utilise l'algorithme d'Euclide étendu.

Proposition

Soient $a \in \mathbb{Z}_N$ et $k < N$, on calcule $a^k \mod N$ en $O(\log^3(N))$.

Euler/Fermat

Théorème (Fermat)

Soient N premier et un entier $a \in \mathbb{Z}$. Nous avons :

$$a^N \equiv a \bmod N.$$

Si $\mathrm{pgcd}(a, N) = 1$, alors :

$$a^{N-1} \equiv 1 \bmod N.$$

Euler/Fermat

Théorème (Fermat)

Soient N premier et un entier $a \in \mathbb{Z}$. Nous avons :

$$a^N \equiv a \bmod N.$$

Si $\mathrm{pgcd}(a, N) = 1$, alors :

$$a^{N-1} \equiv 1 \bmod N.$$

Théorème (Euler)

Pour tout entiers N et a tels que $\mathrm{pgcd}(a, N) = 1$, alors :

$$a^{\varphi(N)} \equiv 1 \mod N.$$

Outline

Test de Fermat – (I)

Principe Général des Tests

Condition calculatoirement facile à vérifier qui caractérise les premiers.

- Test probabiliste $\equiv$ condition nécessaire.
- Test déterministe AKS $\equiv$ condition nécessaire et suffisante.

Test de Fermat – (I)

Principe Général des Tests

Condition calculatoirement facile à vérifier qui caractérise les premiers.

- Test probabiliste $\equiv$ condition nécessaire.
- Test déterministe AKS $\equiv$ condition nécessaire et suffisante.

Théorème (Fermat)

Soit N premier. Alors, pour tout entier a tel que $\mathrm{pgcd}(a, N) = 1$:

$$a^{N-1} \equiv 1 \bmod N.$$

Test de Fermat – (I)

Principe Général des Tests

Condition calculatoirement facile à vérifier qui caractérise les premiers.

- Test probabiliste $\equiv$ condition nécessaire.
- Test déterministe AKS $\equiv$ condition nécessaire et suffisante.

Théorème (Fermat)

Soit N premier. Alors, pour tout entier a tel que $\mathrm{pgcd}(a, N) = 1$:

$$a^{N-1} \equiv 1 \bmod N.$$

- Soit $a, 1 < a < N$ tel que $\mathrm{pgcd}(a, N) = 1$ et $a^{N-1} \not\equiv 1 \bmod N$ alors N est composé et a est un **témoin de Fermat**
- Si N est un entier composé impair et $a, 1 < a < N$. Alors, N est **pseudo-premier** en base a si

$$a^{N-1} \equiv 1 \bmod N.$$

Test de Fermat

`TestFermat`(N)

- Choisir aléatoirement $a \in [2, \ldots, N-1]$
- Si $\text{pgcd}(a, N) > 1$ retourner `composé`.
- Calculer `Reste` $\equiv a^{N-1}$ mod N
- Si `Reste` $\not\equiv 1$ mod N alors retourner `composé`.
- Retourner `premier`

Complexité

$$O\left(\log^3(N)\right).$$

Test de Fermat

`TestFermat`(N)

- Choisir aléatoirement $a \in [2, \ldots, N-1]$
- Si $\text{pgcd}(a, N) > 1$ retourner `composé`.
- Calculer `Reste` $\equiv a^{N-1} \bmod N$
- Si `Reste` $\not\equiv 1 \bmod N$ alors retourner `composé`.
- Retourner `premier`

Caractéristiques

- Si `TestFermat`(N) retourne `composé` alors N n'est pas premier.
- Si `TestFermat`(N) retourne `premier`, alors N est **peut être premier**.
  - L'algorithme échoue pour les a tels que $\text{pgcd}(a, N) = 1$, $a^{N-1} \equiv 1 \bmod N$ mais N composé. Ce sont les a tels que N est pseudo-premier en base a (par exemple, $341 = 11 \cdot 31$ pour $a = 2$).

Nombres de Carmichaël

Définition

Un **nombre de Carmichaël** est un entier N composé qui $\forall a \in [2, \ldots, N-1]$ avec $\mathrm{pgcd}(a, N) = 1$ est pseudo-premier en base a.

Nombres de Carmichaël

Définition

Un **nombre de Carmichaël** est un entier N composé qui $\forall a \in [2, \ldots, N-1]$ avec $\mathrm{pgcd}(a, N) = 1$ est pseudo-premier en base a.

- `TestFermat`(N) échoue donc systématiquement si N est un nombre de Carmichaël.
- $+$ petit nombre de Carmichaël est $561 = 3 \cdot 11 \cdot 17$.

Densité des Nombres de Carmichaël [Harman, 2002]

Soit $C(N)$ le nombre d'entiers de Carmichaël $\leq N$. Alors $\exists \beta > 0.33$:

$$C(N) > N^{\beta},$$

pour N "suffisamment" grand.

- $C(1014) = 44706$, et $\lceil 1014^{0.33} \rceil = 41687$

Nombres de Carmichaël

Définition

Un **nombre de Carmichaël** est un entier N composé qui $\forall a \in [2, \ldots, N-1]$ avec $\mathrm{pgcd}(a, N) = 1$ est pseudo-premier en base a.

$$\Pr_{n \in [1,\ldots,N]}(n \text{ est de Carmichaël}) \leq \frac{C(N)}{N} \leq \frac{1}{N^{0.66}}.$$

Outline

Rabin-Miller

Lemme

Soient N un premier et X un entier positif. Si $X^2 \equiv 1 \bmod N$ alors $X \equiv \pm 1 \bmod N$.

Rabin-Miller

Lemme

Soient N un premier et X un entier positif. Si $X^2 \equiv 1 \mod N$ alors $X \equiv \pm 1 \mod N$.

Théorème

Soit N un premier. Nous écrivons $N - 1 = r \cdot 2^s$, avec r impair et $s \geq 0$. Soit $a \in \mathbb{Z}$ tel que $\mathrm{pgcd}(a, N) = 1$, alors :

$$a^r \equiv 1 \mod N \text{ ou } \exists j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \equiv -1 \mod N.$$

Rabin-Miller

Théorème

Soit N un premier. Nous écrivons $N-1 = r \cdot 2^s$, avec r impair et $s \geq 0$. Soit $a \in \mathbb{Z}$ tel que $\mathrm{pgcd}(a, N) = 1$, alors :

$$a^r \equiv 1 \bmod N \text{ ou } \exists j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \equiv -1 \bmod N.$$

- N composé si $\exists a, 1 < a < N$ tel que $\mathrm{pgcd}(a, N) = 1$ et

$$a^r \not\equiv 1 \bmod N \text{ et } \forall j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \not\equiv -1 \bmod N.$$

Rabin-Miller

Théorème

Soit N un premier. Nous écrivons $N-1 = r \cdot 2^s$, avec r impair et $s \geq 0$. Soit $a \in \mathbb{Z}$ tel que $\mathrm{pgcd}(a, N) = 1$, alors :

$$a^r \equiv 1 \bmod N \text{ ou } \exists j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \equiv -1 \bmod N.$$

- N composé si $\exists a, 1 < a < N$ tel que $\mathrm{pgcd}(a, N) = 1$ et

$$a^r \not\equiv 1 \bmod N \text{ et } \forall j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \not\equiv -1 \bmod N.$$

Définition

Soient N un entier composé impair et $a, 1 < a < N$ tel que $\mathrm{pgcd}(a, N) = 1$. On écrit $N-1 = r \cdot 2^s$, avec r impair. Nous dirons que N est **pseudo-premier fort** en base a si :

$$a^r \equiv 1 \bmod N \text{ ou } \exists j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \equiv -1 \bmod N.$$

Rabin-Miller

Théorème

Soit N un premier. Nous écrivons $N-1 = r \cdot 2^s$, avec r impair et $s \geq 0$. Soit $a \in \mathbb{Z}$ tel que $\mathrm{pgcd}(a, N) = 1$, alors :

$$a^r \equiv 1 \bmod N \text{ ou } \exists j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \equiv -1 \bmod N.$$

Définition

Soient N un entier composé impair et $a, 1 < a < N$ tel que $\mathrm{pgcd}(a, N) = 1$. On écrit $N-1 = r \cdot 2^s$, avec r impair. Nous dirons que N est **pseudo-premier fort** en base a si :

$$a^r \equiv 1 \bmod N \text{ ou } \exists j, 0 \leq j \leq s-1 \text{ tel que } a^{2^j r} \equiv -1 \bmod N.$$

Théorème

Soit N un entier impair composé. Il existe au plus $N/4$ entiers $a, 1 < a < N$ tel que N est pseudo-premier fort en base a.

Test de Rabin-Miller

`RabinMiller`(N)

- Choisir aléatoirement $a \in [2, \ldots, N-1]$
- Si $\text{pgcd}(a, N) > 1$ retourner `composé`
- Trouver (r, s) tel que $N - 1 = r \cdot 2^s$ avec r impair
- Calculer `Reste` $\equiv a^r \bmod N$
- Si `Reste` $\equiv \pm 1 \bmod N$ alors retourner `premier`
- Pour $j, 1 \leq j \leq (s-1)$ faire

  `Reste` $\leftarrow$ `Reste`$^2 \bmod N$

  Si `Reste` $\equiv -1 \bmod N$ alors retourner `premier`
- Retourner `composé`

Complexité

$$O\left(\log^3(N)\right).$$

Test de Rabin-Miller

`RabinMiller`(N)

- Choisir aléatoirement $a \in [2, \ldots, N-1]$
- Si $\text{pgcd}(a, N) > 1$ retourner `composé`
- Trouver (r, s) tel que $N - 1 = r \cdot 2^s$ avec r impair
- Calculer `Reste` $\equiv a^r \bmod N$
- Si `Reste` $\equiv \pm 1 \bmod N$ alors retourner `premier`
- Pour $j, 1 \leq j \leq (s-1)$ faire

  `Reste` $\leftarrow$ `Reste`$^2 \bmod N$

  Si `Reste` $\equiv -1 \bmod N$ alors retourner `premier`
- Retourner `composé`

Caractéristiques

- Si `RabinMiller`(N) retourne `composé` alors N n'est pas premier.
- Si `RabinMiller`(N) retourne `premier`, alors N est **peut être premier**.

Test de Rabin-Miller

`RabinMiller`(N)

- Choisir aléatoirement $a \in [2, \ldots, N-1]$
- Si $\text{pgcd}(a, N) > 1$ retourner `composé`
- Trouver (r, s) tel que $N - 1 = r \cdot 2^s$ avec r impair
- Calculer `Reste` $\equiv a^r \bmod N$
- Si `Reste` $\equiv \pm 1 \bmod N$ alors retourner `premier`
- Pour $j, 1 \leq j \leq (s-1)$ faire

  `Reste` $\leftarrow$ `Reste`$^2 \bmod N$

  Si `Reste` $\equiv -1 \bmod N$ alors retourner `premier`
- Retourner `composé`

Si `RabinMiller`(N) retourne premier alors :

$$\Pr_N(\texttt{RabinMiller}(N) \text{ retourne } \texttt{premier} \mid N \text{ est composé}) \leq \frac{1}{4}$$

Outline

Motivation

Soit $n > 0$ et une matrice de *Vandermonde* :

$$\mathrm{Vd}(x_1, \ldots, x_n) = \begin{pmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ 1 & x_2 & \ldots & x_2^{n-1} \\ 1 & \vdots & \ldots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{pmatrix}$$

Motivation

Soit $n > 0$ et une matrice de *Vandermonde* :

$$\mathrm{Vd}(x_1, \ldots, x_n) = \begin{pmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ 1 & x_2 & \ldots & x_2^{n-1} \\ 1 & \vdots & \ldots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{pmatrix}$$

On sait **montrer mathématiquement** :

$$\mathrm{Det}\big(\mathrm{Vd}(x_1, \ldots, x_n)\big) = \prod_{1 \leq i < j \leq n} (x_j - x_i).$$

Motivation

Soit $n > 0$ et une matrice de *Vandermonde* :

$$\mathrm{Vd}(x_1, \ldots, x_n) = \begin{pmatrix} 1 & x_1 & \ldots & x_1^{n-1} \\ 1 & x_2 & \ldots & x_2^{n-1} \\ 1 & \vdots & \ldots & \vdots \\ 1 & x_n & \ldots & x_n^{n-1} \end{pmatrix}$$

On sait **montrer mathématiquement** :

$$\mathrm{Det}\big(\mathrm{Vd}(x_1, \ldots, x_n)\big) = \prod_{1 \leq i < j \leq n} (x_j - x_i).$$

On souhaite **vérifier** l'identité :

$$\mathrm{Det}\big(\mathrm{Vd}(x_1, \ldots, x_n)\big) - \prod_{1 \leq i < j \leq n} (x_j - x_i) = 0.$$

Motivation

On souhaite **vérifier** l'identité :

$$\mathrm{Det}\big(\mathrm{Vd}(x_1,\ldots,x_n)\big) - \prod_{1\leq i<j\leq n}(x_j - x_i) = 0.$$

- Pour $n = 2$, $\mathrm{Det}\big(\mathrm{Vd}(x_1,x_2)\big) = x_2 - x_1$. est un poly. de degré ≈ 5000 avec 2^{5000} termes.
- Pour $n = 100, \mathrm{Det}\big(\mathrm{Vd}(x_1,\ldots,x_{100})\big)$ est un poly. de degré ≈ 5000 avec 2^{5000} termes.

Motivation

On souhaite **vérifier** l'identité :

$$\mathrm{Det}\big(\mathrm{Vd}(x_1,\ldots,x_n)\big) - \prod_{1\le i<j\le n}(x_j - x_i) = 0.$$

Définition

- $\mathbb{K}[x_1,\ldots,x_n]$ l'ensemble des polynômes en les variables $x_1,\ldots,x_n$ dont les coefficients sont dans $\mathbb{K}$
- Un **monôme** est un produit de puissances des variables, i.e. $x_1^{\alpha_1}\cdots x_n^{\alpha_n}$.
- Le **degré** d'un monôme $x_1^{\alpha_1}\cdots x_n^{\alpha_n}$ est $\sum_{i=1}^{n}\alpha_i$.
- Le nombre de monômes de degré $\le d$ en n variables $\binom{n+d}{d} \in O(n^d)$.

Polynomial Idendity Testing

Définition

Soit $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > 0$. Nous dirons que f est donné par une **boîte noire** s'il est possible d'évaluer f efficacement, i.e. en temps polynomial (sans forcement accéder à ses coefficients).

https://en.wikipedia.org/wiki/Black_box

- Ex : $\mathrm{Det}\big(\mathrm{Vd}(x_1, \ldots, x_n)\big)$.

PIT

- **Entrée :** un polynôme $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > 0$ donné par une boîte-noire.
- **Question :** Est-ce que le polynôme f est nul ?

PIT – Cas Univarié

UnivPIT

- **Entrée :** un polynôme $f \in \mathbb{K}[X]$ de degré $d > 0$ donné par une boîte-noire.
- **Question :** Est-ce que le polynôme f est nul ?

PIT – Cas Univarié

UnivPIT

- **Entrée :** un polynôme $f \in \mathbb{K}[X]$ de degré $d > 0$ donné par une boîte-noire.
- **Question :** Est-ce que le polynôme f est nul ?

Lemme

Il existe une algorithme polynomial déterministe pour résoudre UnivPIT.

PIT – Cas Univarié

UnivPIT

- **Entrée :** un polynôme $f \in \mathbb{K}[X]$ de degré $d > 0$ donné par une boîte-noire.
- **Question :** Est-ce que le polynôme f est nul ?

Lemme

Il existe une algorithme polynomial déterministe pour résoudre UnivPIT.

Démonstration.

Si f est non nul, alors d s'annule sur au plus d points distincts. L'algorithme consiste à évaluer le polynôme f sur $d+1$ poins distincts $r_1, \ldots, r_{n+1} \in \mathbb{K}$. Si $f(r_i) = 0$, pour tout $i, 1 \leq i \leq n+1$, alors nous pouvons dire que f est nul. Sinon, f est nul.

□

PIT – Cas Général

Lemme de Schwartz-Zippel-DeMillo-Lipton

Soit $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > 0$, et $S \subseteq \mathbb{K}$. Si f est **non-nul**, alors :

$$\Pr_{(r_1,\ldots,r_n)\in S^n}\big(f(r_1,\ldots,r_n) = 0\big) \leq \frac{d}{|S|},$$

avec $|S|$ qui dénote la taille d'un sous-ensemble fini $S \subseteq \mathbb{K}$.

The Curious History of the Schwartz-Zippel Lemma

The Curious History of the Schwartz-Zippel Lemma

NOVEMBER 30, 2009

by rjlipton

tags: Algorithms, identity testing, polynomial, Problems, random, randomness, Schwartz-Zippel

On the discovery that randomness helps exponentially for identity testing

Jack Schwartz was one of most famous mathematicians who worked in the theory of linear operators, his three volume series with Nelson Dunford is a classic. Yet he found the time, the creative energy, and the problem solving ability to make seminal contributions to many diverse areas of science—not just mathematics or theory—but science in general. He worked on, for example: parallel computers, the design of new programming languages, fundamental questions of computational theory, and many more wonderful problems from a wide range of areas. He is greatly **missed** by our community as well as all of science.

Today I want to talk about the **Schwartz-Zippel** Lemma. Or the Schwartz-Zippel-DeMillo-Lipton Lemma. Or the Schwartz Lemma. The whole point is to discuss the curious history of this simple, but very useful result.

First, I thought that I would give one short story that shows somethings about the character of Jack. When he was working on a problem, that problem was the most important thing in the world. He had tremendous ability to focus. He also was often ahead of his time with the problems he worked on; his important work on parallel computers was way before it was the critical problem it is today.

Once at a **POPL** meeting Jack Schwartz was scheduled to talk right before lunch. His talk was on his favorite topic at the time: the programming language **SETL**. He loved to talk about SETL, I think it was one of his favorite topics of all times—but that is just my opinion.

In those days there were no parallel sessions, and the speaker right before Jack gave a very short talk. That left almost a double period before lunch. Right after being introduced, Schwartz said with a smile,

https://rjlipton.wordpress.com/2009/11/30/the-curious-history-of-the-schwartz-zippel-lemma/

The Curious History of the Schwartz-Zippel Lemma


Richard A. DeMillo, Richard J. Lipton.
“A Probabilistic Remark on Algebraic Program Testing".
Inf. Process. Lett. 1978.


Jacob T. Schwartz.
“Probabilistic Algorithms for Verification of Polynomial Identities (invited)".
International Symposium on Symbolic and Algebraic Computation, 1979.


Richard Zippel.
“Probabilistic Algorithms for Sparse Polynomials".
International Symposium on Symbolic and Algebraic Computation, 1979.

PIT – Cas Général

Lemme de Schwartz-Zippel-DeMillo-Lipton

Soit $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > 0$, et $S \subseteq \mathbb{K}$. Si f est non-nul, alors :

$$\Pr_{(r_1,\ldots,r_n)\in S^n}\big(f(r_1,\ldots,r_n) = 0\big) \leq \frac{d}{|S|},$$

avec $|S|$ qui dénote la taille d'un sous-ensemble fini $S \subseteq \mathbb{K}$.

PIT – Cas Général

Lemme de Schwartz-Zippel-DeMillo-Lipton

Soit $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > 0$, et $S \subseteq \mathbb{K}$. Si f est non-nul, alors :

$$\Pr_{(r_1,\ldots,r_n)\in S^n}\big(f(r_1,\ldots,r_n) = 0\big) \leq \frac{d}{|S|},$$

avec $|S|$ qui dénote la taille d'un sous-ensemble fini $S \subseteq \mathbb{K}$.

Démonstration.

Cas de base. Pour $n = 1$, un polynôme univarié possède au plus d racines distinctes. □

PIT – Cas Général

Démonstration.

Induction. Soit $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > 0$ non nul. On dénote par k la plus grande puissance de x_1 qui divise les monômes de f.

$$f = \sum_{i=0}^{k} x_1^i \cdot f_i(x_2, \ldots, x_n).$$

Par définition, f_k est non-nul et de degré $\leq d - k$. Par hypothèse d'induction :

$$\Pr_{(r_2,\ldots,r_n)\in S^{n-1}}\left(f_k(r_2, \ldots, r_n) = 0\right) \leq \frac{d-k}{|S|}.$$

□

PIT – Cas Général

Démonstration.

Soient A l'événement "$f(r_1, \ldots, r_n) = 0$" et B l'événement "$f_k(r_2, \ldots, r_n) = 0$". Clairement, $A = (A \cap B) \cup (A \cap \bar{B})$ et

$$\begin{aligned} \Pr(A) &= \Pr(A \cap B) + \Pr(A \cap \bar{B}) \\ \Pr(A) &= \Pr(A|B)\Pr(B) + \Pr(A|\bar{B})\Pr(\bar{B}) \\ \Pr(A) &\leq \Pr(B) + \Pr(A|\bar{B}). \end{aligned}$$

Nous avons :

$$\Pr(B) = \Pr_{(r_2,\ldots,r_n) \in S^{n-1}} \left(f_k(r_2, \ldots, r_n) = 0 \right) \leq \frac{d-k}{|S|}.$$

□

PIT – Cas Général

Démonstration.

Soit $(r_2, \ldots, r_n) \in S^{n-1}$. On pose :

$$p(x_1) = f(x_1, r_2, \ldots, r_n) = \sum_{i=0}^{k} x_1^i \cdot f_i(r_2, \ldots, r_n).$$

Si "$f_k(r_2, \ldots, r_n) \neq 0$", alors p est un polynôme univarié non-nul de degré k. Finalement :

$$\Pr(A) \leq \Pr(B) + \Pr(A|\bar{B}) \leq \frac{d-k}{|S|} + \frac{k}{|S|} \leq \frac{d}{|S|}.$$

□

Résolution du PIT

Proposition

Soit $\mathbb{K}$ fini, et $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > |\mathbb{K}|$. Il existe une algorithme polynomial probabiliste pour résoudre PIT. Si l'algorithme retourne non-zero, alors f est bien non nul. Si l'algorithme retourne zero, alors f est nul avec probabilité $\leq \frac{d}{|\mathbb{K}|}$.

- Si $\mathbb{K}$ est infini, avec probabilité $\leq \varepsilon$, pour tout $0 < \varepsilon < 1$.

Résolution du PIT

Proposition

Soit $\mathbb{K}$ fini, et $f \in \mathbb{K}[x_1, \ldots, x_n]$ de degré $d > |\mathbb{K}|$. Il existe une algorithme polynomial probabiliste pour résoudre PIT. Si l'algorithme retourne non-zero, alors f est bien non nul. Si l'algorithme retourne zero, alors f est nul avec probabilité $\leq \frac{d}{|\mathbb{K}|}$.

- Si $\mathbb{K}$ est infini, avec probabilité $\leq \varepsilon$, pour tout $0 < \varepsilon < 1$.

Démonstration.

On évalue le polynôme f sur un point $(r_1, \ldots, r_n) \in \mathbb{K}^n$ tiré aléatoirement.

- Si $f(r_1, \ldots, r_n) \neq 0$, alors l'algorithme retourne non-zero.
- Si $f(r_1, \ldots, r_n) = 0$, alors l'algorithme retourne zero.

Finalement, si f est non-nul, alors :

$$\Pr_{(r_1,\ldots,r_n)\in\mathbb{K}^n}\big(f(r_1, \ldots, r_n) = 0\big) \leq \frac{d}{|\mathbb{K}|}.$$