SORBONNE UNIVERSITÉ

4I900

Complexité, Algorithmes Randomisés et Approchés

Rapport de Projet

Étudiants: Numéros:

Yannick LI 3305496

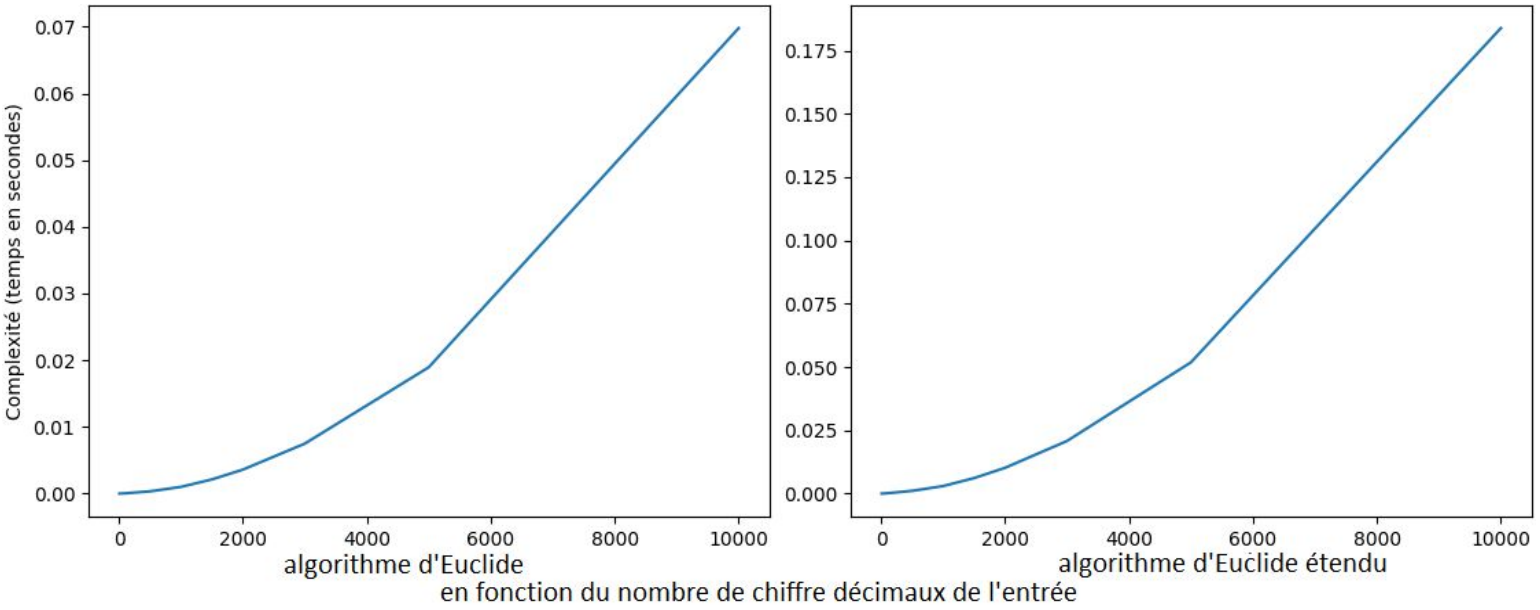
Keyvan Beroukhim 3506789

Décembre 2018

Exercice 1 : Arithmétique dans $\mathbb{Z}_n$

1.a. 1.b. Les calculs du pgcd (de deux nombres) et de l'inverse (de l'un modulo l'autre) peuvent se faire grâce aux algorithmes d'Euclide ou d'Euclide étendu et partagent donc la même complexité polynomiale théorique (en fonction de la taille des entrées).

1.c. Expérimentalement, pour une taille maximale des entrées fixée, nous mesurons le temps nécessaire aux algorithmes pour s'exécuter une centaine de fois (en donnant à chaque fois comme paramètres deux valeurs tirées indépendamment et uniformément dans l'ensemble {0, …, max}).



La faible complexité des algorithmes leur permet de traiter rapidement des entrées de l'ordre de 10^{10000}.
Expérimentalement, les courbes ont la même allure, cela témoigne du fait que les deux algorithmes partagent la même complexité asymptotique polynomiale.
L'algorithme d'Euclide étendu à une complexité environ 2,5 fois supérieure à celle de l'algorithme simple. Ceci est en partie expliqué par le fait que le calcul de l'inverse peut lever une exception si les deux paramètres ne sont pas inversibles.

1.d. Pour le calcul de exponentiation discrète modulaire binaire, on effectue autant d'itérations que l'exposant a de chiffres en base 2. Le coût d'une itération est bornée car la taille des opérandes est bornée.

Exercice 2 : Test naïf et recherche des nombres de Carmichael

2.a. Le test naïf de primalité consiste à vérifier que pour tout entier k inférieur à la racine carrée de N, on a k qui ne divise pas N.

2.b. L'entrée est le nombre N représenté en n bits.
$N = O(2^n)$ donc le nombre d'itération est :

$$\begin{aligned} sqrt(N) &= O(\, sqrt\,(2^n)\,) \\ &= O(\,(2^n)^{1/2}\,) \\ &= O(\,2^{n/2}\,) \\ &= O(\,(2^{1/2})^n\,) \\ &= O(\,1.41^n\,) \end{aligned}$$

Donc le nombre d'itération est exponentiel en la taille de l'entrée.

Si on considère que les opérations arithmétiques s'effectuent en O(1),
alors la complexité du test est en $O(\,1.41^n\,)$.
Si elles s'effectuent en O(bitsize(a) * bitsize(b)),
alors la complexité du test est $O(\,1.41^n * n * bitsize(sqrt(N))) = O(\,n^2 * 1.41^n\,)$

2.c. On teste pour tout k inférieur à 10^5 si k est premier. On obtient un total de 9592 nombres premiers inférieurs à 10^5.

2.d. Pour tester si un entier n est un nombre de **Carmichael**, on calcul sa **décomposition en produit de facteurs premiers** puis on applique le **critère de Korselt**.
On teste pour tout k inférieur à 10^5 si k est un nombre de Carmichael. Il n'y a que 16 nombres de Carmichael inférieurs à 10^5 :
[561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, 41041, 46657, 52633, 62745, 63973, 75361]

2.e. Pour générer des nombres de Carmichael de la forme pqr, on maintient une liste de nombres premiers initialisée à [2,3,5]. On teste toutes les combinaisons de 3 nombres parmi la liste puis on ajoute le nombre premier suivant et réitère l'opération autant de fois que souhaité.

2.f. En 5 minutes, le plus grand nombre de Carmichael trouvé est :
588909469501 = 1871 * 16831 * 18701

2.g.1 Soit n un nombre de Carmichael de la forme p*q*r avec p<q<r p,q,r premiers.

$$n = pqr = 1\ [r-1] \quad (\textit{par application du critère de Korselt})$$

$$\Rightarrow \quad pqr - r + 1 = 1\ [r-1]$$

$$\Rightarrow \quad r(pq-1) = 0\ [r-1]$$

$$\Rightarrow \quad r-1 \mid r\,(pq-1)$$

$$\Rightarrow \quad r-1 \mid pq-1 \quad (\textit{car } r \textit{ et } r-1 \textit{ sont premiers entre eux, par application du théorème de Gauss})$$

donc il existe h tel que :

$$pq - 1 = h(r-1) \qquad \text{(i)}$$

Montrons que 2<=h<=p-1 :

- Si h = 0 :

$$pq - 1 = 0$$

$$\Rightarrow \quad pq = 1$$

La seule solution dans N est p=q=1, ce qui est impossible car p et q sont premiers.

- Si h = 1 :

$$pq - 1 = r - 1$$

$$\Rightarrow \quad pq = r$$

$$\Rightarrow \quad p \mid r$$

Ce qui contredit le fait que r soit premier.

- Si h>=p :

$$p/h \leq 1$$

$$\Rightarrow \quad pq - 1 \geq h\,(r-1)\,p/h$$

$$\Rightarrow \quad pq - 1 \geq p\,(r-1)$$

$$\Rightarrow \quad p(q-r+1) \geq 1$$

Or q et r sont des nombres premiers distincts et r >= 5 (le troisième nombre premier) donc |q-r| >= 2. De plus, q<r donc q < r - 1

$$\Rightarrow \quad q - r + 1 < 0$$

$$\Rightarrow p\,(q-r+1) < 0$$ C'est une contradiction.

Au final, on a : **2<=h<=p-1**

De même en partant de n = 1 mod q-1, on obtient qu'il existe k tel que :

$$pr - 1 = k(q-1) \qquad \text{(ii)}$$

Nous n'avons pas que 2<=k<=p-1

2.g.2

$rh = pq - 1 + h$ *d'après* (i)

$prh - h = hk(q-1)$ *d'après* (ii)

En substituant rh dans cette dernière équation, on a :

$$p(pq - 1 + h) - h = hk(q-1)$$

$$\Rightarrow hk(q-1) = p(pq - 1 + h) - h$$

$$= p^2q - p + hp - h$$

$$= (p^2q - p + hp - h) + (p^2 - p^2) + (kp - kp) + (k - k)$$

$$= (p^2 - p + kp - k) + (hp - h - p^2 - kp + k + p^2q)$$

$$= (p + k)(p - 1) + (hp - h - p^2 - kp + k + p^2q)$$

$$= (p + k + h)(p - 1) + k(1 - p) + p^2(q - 1)$$

$$= (p + h)(p - 1) + k(p - 1) + k(1 - p) + p^2(q - 1)$$

$$= (p + h)(p - 1) + p^2(q - 1)$$

$$\Rightarrow (hk - p^2)(q - 1) = (p + h)(p - 1)$$

Bornons k :

d'après (hk - p²)(q - 1) = (p + h)(p - 1)

vu que (q - 1), (p + h) et (p - 1) sont strictement positifs, on obtient :

hk - p² > 0

donc k > p²/h

Par ailleurs, en multipliant les deux équation (i) et (ii), on obtient :

$(pq-1)(pr-1) = h(r-1)\,k(q-1)$

$donc\ k = 1/h * (pq-1)/(q-1) * (pr-1)/(r-1)$

$= 1/h * fp(q) * fp(r)$

$avec\ fp(x) = (px - 1) / (x - 1)$

f_p est décroissante donc on majore $f_p(q)$ par $f_p(p')$ et $f_p(r)$ par $f_p(p'')$

p' et p'' étant les deux nombres premiers consécutifs à p.

donc **$p^2/h + 1 <= k <= 1/h * f_p(p') * f_p(p'')$**

2.g.3. Si n est un nombre de Carmichael de la forme pqr, pour p fixé, les valeurs prises par h et k sont bornées.

Pour tout h dans [2 ... p - 1]

Pour tout k dans [p²/h + 1 ... 1/h * f_p(p') * f_p(p'')]

On calcule q = (p + h)(p - 1)/(hk - p²) + 1

r = (pq-1) / h + 1

On regarde ensuite si le nombre pqr formé est un nombre de Carmichael selon qu'il vérifie le critère de Korselt ou non. Si c'est le cas, on ajoute ce nombre à la liste des solutions.

2.h. On obtient :
Nombres de Carmichael de la forme pqr avec p = 3
561 = 3 * 11 * 17

Nombres de Carmichael de la forme pqr avec p = 5
1105 = 5 * 13 * 17
2465 = 5 * 17 * 29
10585 = 5 * 29 * 73

En fixant la valeur de p à celle d'un grand nombre premier, on trouve très rapidement des nombres de Carmichael plus grands qu'avec notre précédente méthode. Par exemple avec p = 4999, on trouve :
617461815971987281 = 4999 * 3418633 * 36130543

Exercice 3 : Test de Fermat

3.a. Si p est premier d'après le petit théorème de Fermat, quel que soit a :
$a^p = a\ [p]$.
Le test de primalité de Fermat consiste à dire que p est premier si $a^p = a\ [p]$ pour un entier a donné en paramètre.

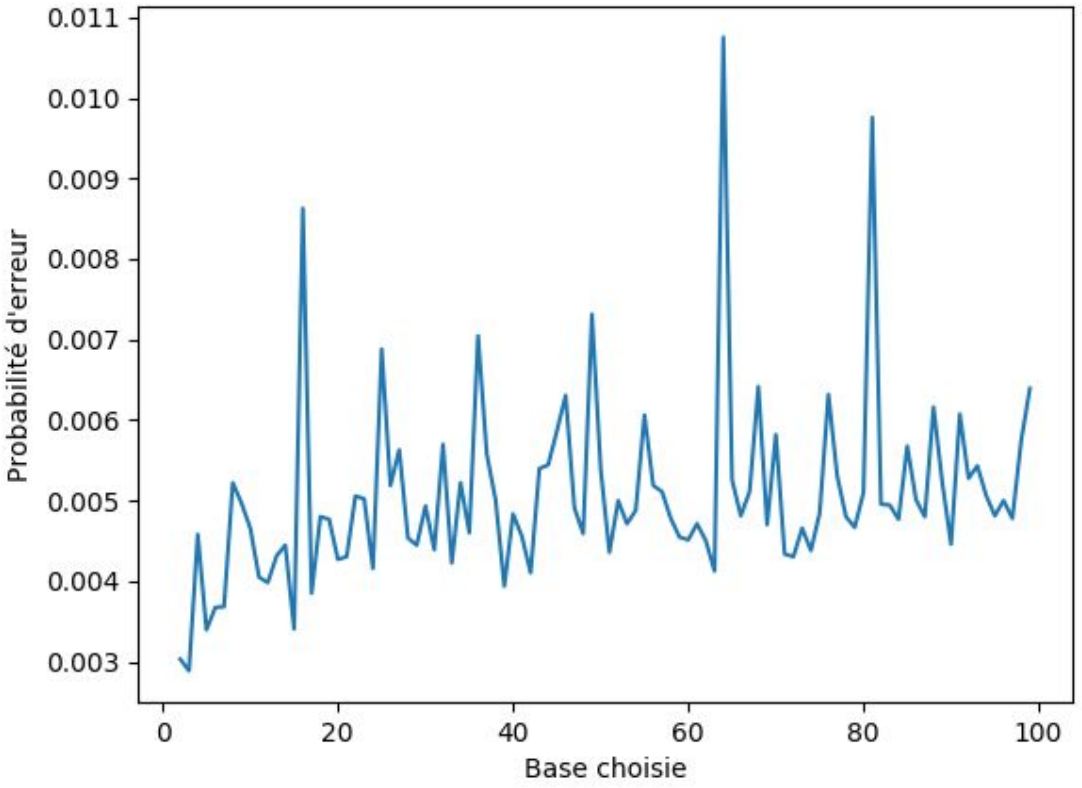
3.b. Le test de primalité de Fermat est un test probabiliste. Pour une entrée n, si l'algorithme renvoie "vrai" alors n est possiblement premier. Si il renvoie "faux" alors on est sûr que n est composé.

- Si n est un nombre premier, l'algorithme ne peut renvoyer que "vrai" quelle que soit la base utilisée. En effet, si il existait une base a telle que le test de Fermat en base a retourne "Composé" alors n serait alors composé de façon certaine, ce qui est contradictoire.

- Les nombres composés peuvent passer le test dans certaines bases - c'est à dire que l'algorithme retourne "possiblement premier" alors que le nombre est composé - on les appelle dans ce cas pseudo premier en cette base.
  Par exemple, 221 = 13*17 et 221 passe le test de Fermat dans les bases 21, 34 et 38 mais pas dans la base 37.

- Il existe des nombres composés particuliers appelés nombres de Carmichael tels que quelle que soit la base utilisée, le test de Fermat retourne "Possiblement premier".
  Par exemple, 561 = 3 * 11 * 17 et tous les autres nombres de Carmichael générés inférieurs à 10^5 passent les tests de Fermat pour toutes les bases testées.

3.c.



Probabilité d'erreur du test de Fermat sur les nombres composés inférieurs à 10^5 selon la base utilisée (variant de 2 à 100)

Quelle que soit la base utilisée, la probabilité d'erreur du test de Fermat est non nulle. Elle est en moyenne égale à 0,5% sur cette instance. Bien que relativement basse, la fiabilité de cet algorithme le rend **surclassé par d'autres** et **non utilisé**.

Exercice 4 : Test de Rabin et Miller

4.a. Le test de Miller-Rabin consiste à tirer plusieurs fois une base aléatoirement, et d'exécuter le test de primalité fort de Fermat dans cette base. Le nombre d'itérations à effectuer est donné en paramètre.

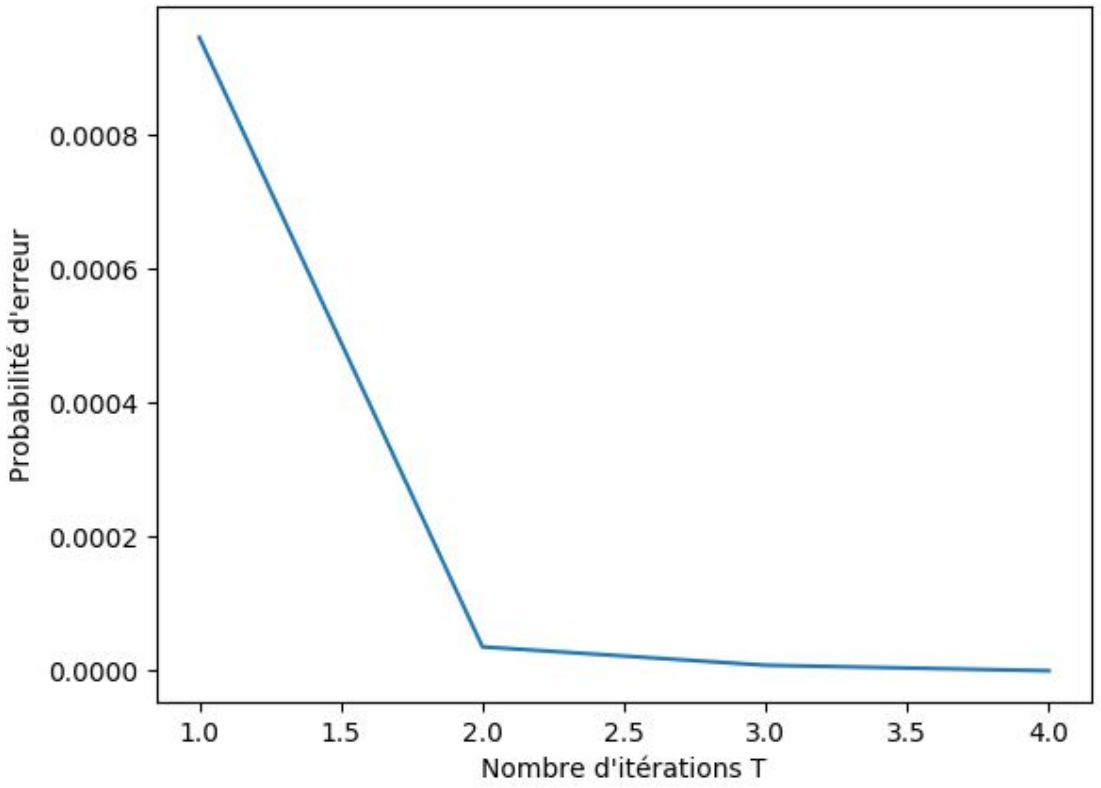
4.b. De la même manière que le test de Fermat, le test de Miller-Rabin est un test probabiliste. Pour une entrée n, si l'algorithme renvoie "vrai" alors n est possiblement premier. Si il renvoie "faux" alors on est sûr que n est composé.

- Si n est un nombre premier, le test de Miller-Rabin ne peut renvoyer que "vrai".

- Les nombres composés peuvent passer le test de Fermat fort dans certaines bases. On les appelle dans ce cas pseudo premiers forts en cette base.
  Par exemple, 121 = 11*11 passe le test de Fermat fort dans la base 3.
  4033 = 37*109 passe le test dans les bases 2, 17, 19 et 23.

- Les nombres de Carmichael n'ont pas la propriété de résister au test de fermat fort. Par exemple 561 = 3 * 11 * 17 ne passe le test de Fermat fort dans aucune des bases 2, 3, 5, 7, 11, 13, 17 et 19.

4.c



La probabilité d'erreur est nettement plus faible que celle du test de Fermat. Même avec une seule itération, la probabilité d'erreur de Miller-Rabin est déjà inférieure. A partir de T=4 on mesure à chaque fois un nombre d'erreur nul.
La probabilité d'erreur de cet algorithme est **exponentielle décroissante** : elle est en $O(4^{-T})$.
Quelle que soit la précision souhaitée on peut l'atteindre en augmentant la valeur du paramètre T, et la complexité de calcul l'algorithme est linéaire en la valeur de T.

4.d Pour générer deux grands nombres premiers, on tire aléatoirement des grandes valeurs et on les garde si ces nombres passent le test de Miller-Rabin. Grâce à cet algorithme, la **génération** de deux nombres premiers devient **bien plus aisée** que la **décomposition** de leur produit.