Université de Paris 7 | **TP08 Méthode de pollard** | **agreg 2013**

(F.Han [1])

Nous allons étudier la seconde partie du texte : exemple-Jury4 (méthode de Pollard). Pour la première partie vous pourrez regarder l'exemple dans l'aide xcas : `Exemples>crypto>rsa`

Thèmes reliés :

- Fonction random ou variable aléatoire. Pertinence du modèle choisi.
- Suites récurrentes et applications.
- Statistiques.
- Factorisation, Complexité.

L'exercice suivant peut être traité sans avoir lu le texte, il permet de découvrir l'algorithme et de le tester. Vous pourrez lire ensuite le texte pour avoir plus de détails.

Exercice I: Exemples de développements

1) a) Créer une procédure `bete`, qui trouve le plus petit diviseur de n différent de 1 et 2, et qui retourne n s'il n'y en a pas, (en testant tous les nombres impairs de 3 à $\sqrt{n}$) [2]. Commentez cet algorithme par rapport au crible d'Eratostene lorsque n est grand.

b) Comparez et commentez le temps d'execution de `p:=nextprime(10**55)` avec celui de `isprime(p)`.

c) Pour différentes valeurs de a et b, tester `bete` sur des nombres N du type `nextprime(a)*nextprime(b)`. Donner un ordre de grandeur sur le nombre de chiffres de N pour que `bete` ne réponde plus. Que fait nextprime ? Attention, que fait `rand(30)` par rapport à `rand(2^43)` dans certaines versions d'xcas ?

2) Si $n = pq$, où p et q sont premiers, comment trouver p si on a trouvé a et b tels que $a = b[p]$ et $a \neq b[n]$?

3) Créer une procédure `pollard` qui à partir de N trouve un diviseur strict de N en testant $(u_{2i} - u_i) \wedge N \notin \{1, N\}$ où $u_{i+1} = (u_i)^2 + 1[N]$, $u_0 = 1$. On programmera de manière efficace le calcul de u_{2i}. On remarquera que cette procédure peut ne pas aboutir.)

4) a) tester `pollard(N);` avec : `N:=nextprime(10^6)*nextprime(2*10^6);`

b) Testez `pollard` pour plusieurs valeurs de N où `bete` échoue.

5) Remarquer que si $u_i = u_j[p]$, alors $\forall s \geq 0, u_{i+s} = u_{j+s}$, et donc que si $u_i = u_j[p]$ alors il existe k tel que $u_k = u_{2k}[p]$

6) a) En modifiant `pollard`, faire une procédure `comptep` qui retourne une valeur flottante de $\frac{j}{\sqrt{p}}$ où j est le nombre de tours effectués par pollard, et p le diviseur trouvé par pollard.

b) Etudier le comportement de `comptep(N)` lorsque N grandit. Conclure que Pollard est en $O(\sqrt{p})$. On pourra créer une liste de valeurs, et dessiner cette liste.

7) a) Quelle est la probabilité pour qu'un couple (a, b) d'entiers inférieurs à $n = pq$ soit tel que $a = b \bmod n$? Celle pour que $a = b \bmod p$? Expliquer pourquoi pollard boucle parfois pour des n petits, mais que l'on n'observe pas ce phénomène pour des nombres difficiles à factoriser.

b) On fait i tirages entre 0 et p, quelle est la probabilité pour que 2 d'entre eux soient égaux ? (Cf anniversaires)

c) Créer la fonction :
`P:=i0->product(1-j/p,j=1..i0-1);`. En utilisant `limit` en `+infinity`, montrer que $\log(P(2))$ est équivalent à $-1/p$, $\log(P(3)) \simeq -3/p$, $\log(P(4)) \simeq -6/p$. Deviner une formule pour $\log(P(i))$ et testez la.

d) En utilisant cette formule, montrer que $P(i) > 1/2$ dès que i est de l'ordre [3] de $1.18\sqrt{p}$

8) Que se passe t'il si l'on prend comme fonction "random" une suite du type $u_{n+1} = a.u_n + c[n]$ au lieu de $(u_n)^2 + 1[n]$?

Exercice II:

L'algorithme de pollard repose sur l'hypothèse que les données sont piochées de manière aléatoire uniforme. Nous allons donc étudier ici si la suite récurrente choisie a des statistiques cohérentes avec cette hypothèse.

1. http ://www.math.jussieu.fr/∼han/agreg
2. Attention tout de même à ne pas calculer la racine carrée à chaque tour
3. C'est la borne du texte Jury4

1) Etudier/réviser l'utilisation des fonctions statistiques de votre logiciel. Par exemple pour xcas : `histogram`, `classes`, `diagramme_batons`.

2) Etudier un peu la répartition des données fournies par la fonction `rand(2)`. Par exemple donner un ordre de grandeur sur le nombre de tirages pour que la différence du nombre de 1 et de 0 ait l'air sur votre histogramme d'être inférieure à 5% du nombre de données.

3) a) Prenons un nombre N du type `N:=nextprime(10^6)*nextprime(2*10^6);`. On considère la suite $u_{i+1} = (u_i)^2 + 1$ $[N]$.

b) Choisissez un u_0, et programmez u par récurrence. Calculez $u(1)$, $u(2)$, $u(10000)$.

c) Etudier la répartition des (u_i) modulo N. (Créer une liste de 2000 termes). Attention, vu la taille de N, on regroupera les données en 40 classes, et l'on remarquera que le diagramme en batons est plus approprié que l'histogramme. Comment ne pas afficher les noms ? Peut on considérer qu'ils sont répartis de manière uniforme ?

d) Etudier graphiquement les couples $(u_i\ [N], u_{2i}\ [N])$. (Créez une fonction qui donne une liste de M couples). Conclusion ?