UNIVERSITÉ DU QUÉBEC À CHICOUTIMI

DEVOIR 4

PAR

JÉRÉMY BOUCHARD (BOUJ08019605)

JEAN-PHILIPPE SAVARD (SAVJ04079609)

ALEXIS VALOTAIRE (VALA09129509)

DEVOIR PRÉSENTÉ À

M. FRANÇOIS LEMIEUX

DANS LE CADRE DU COURS D'ALGORITHMIQUE (8INF433)

Note : Le code source LaTeX est disponible à l'URL suivant :
`https://github.com/AlexisCode101/algo-devoir-4`

Question 1

Donnez toutes les étapes de l'algorithme de Miller-Rabin pour tester si 97 est premier. Vous devez supposer que le générateur pseudo aléatoire retourne le nombre 5.

Ce que nous savons du problème:

$$n = 97$$
$$n - 1 = 96 = 2^5 * 3$$
$$b = 5$$

On peut alors exécuter l'algorithme de Miller-Robin pour déterminer si 97 est possiblement un nombre premier:

$$b^t = 5^3 = 125 \equiv 28 (mod\, 97)$$
$$b^{2t} = 28^2 = 784 \equiv 8 (mod\, 97)$$
$$b^{2^2 t} = 8^2 = 64 \equiv 64 (mod\, 97)$$
$$b^{2^3 t} = 64^2 = 4096 \equiv 22 (mod\, 97)$$
$$b^{2^4 t} = 22^2 = 484 \equiv 96 (mod\, 97) \equiv -1 (mod\, 97)$$

Puisque nous avons une valeur de -1 à la dernière itération montrée, nous pouvons donc dire que 97 est probablement premier sur une base 5.

Question 2

On peut montrer que si n est un nombre premier, alors pour tout entier positif b on a :

$$b^{(n-1)/2} mod\, n = J(b, n) \tag{1}$$

où $J(b, n)$ est une fonction appelée symbole de Jacobi (Il n'est pas nécessaire de connaître la définition exacte de $J(b, n)$ pour répondre à cette question).

D'autre part, si n est composé, alors la relation (1) est fausse pour au moins 50% de tous les entiers b pour lesquels $PGCD(b, n) = 1$.

a) En utilisant ces deux observations, trouvez un algorithme de Monte Carlo 50%-correct qui détermine si un entier positif n est premier. Vous pouvez supposer qu'il est possible de tester efficacement si la relation (1) est vraie.

Algorithme 1 : VerifSiPremierMonteCarlo50Correct

Données : *entier* n, *entier* k

Résultat : Vrai si n est premier ou Faux si n est composé

```
1 pour i ← 1 à k faire
2 |   b ← uniforme(2 ... n − 2)
3 |   si PGCD(b, n) = 1 alors
4 |   |   si b^((n−1)/2) mod n ≠ J(b, n) alors
5 |   |   |   retourner Faux
6 |   |   fin
7 |   fin
8 fin
9 retourner Vrai
```

b) Votre algorithme est-il biaisé? Si oui, est-il vrai-biaisé ou faux-biaisé? Expliquez.

Notre algorithme est biaisé puisqu'il possède une probabilité de 50% ce qui est la limite inférieure de la condition pour être considéré comme biaisé. De plus, il est faux-biaisé car nous sommes sûr d'avoir un n composé lorsque la fonction retourne $Faux$, contrairement à avoir un n premier où nous avons 50% de chance d'avoir un faux-vrai.

c) Montrez comment vous pouvez modifier votre algorithme pour en obtenir un qui soit 99.999%-correct.

Algorithme 2 : VerifSiPremierMonteCarlo99Correct

Données : *entier* n, *entier* k

Résultat : Vrai si n est premier ou Faux si n est composé

```
1  temoin ← 0
2  pour i ← 1 à k faire
3  |  b ← uniforme(2...n − 2)
4  |  si PGCD(b, n) = 1 alors
5  |  |  si b^((n−1)/2) mod n ≠ J(b, n) alors
6  |  |  |  temoin + +
7  |  |  fin
8  |  fin
9  fin
10 si temoin/k · 100 ≥ 50 alors
11 |  retourner Faux
12 fin
13 sinon
14 |  retourner Vrai
15 fin
```

Question 3

Considérez deux algorithmes de Monte-Carlo A et B pour résoudre un même problème P. A est p-correct et vrai-biaisé tandis que B est q-correct et faux-biaisé. Le temps d'exécution de A est $TA(n)$ et celui de B est $TB(n)$.

a) Utilisez ces deux algoritmes pour construire un algorithme de Las Vegas pour résoudre le problème

Il est possible de résoudre ce problème en utilisant un premier algorithme pour calculer une solution et si celle-ci ne produit pas un résultat exactement vrai, d'utiliser le second algorithme pour tester si la même entrée offre un résultat exactement faux. Cela est possible puisque les algorithmes sont biaisés différemment.

Algorithme 3 : Las Vegas en utilisant 2 Monte Carlo

Données : *entier* n
Résultat : Vrai ou Faux

```
1 pour i ← 1 à ∞ faire
2 |   si A(i) = Vrai alors
3 |   |   retourner Vrai
4 |   fin
5 |   si B(i) = Faux alors
6 |   |   retourner Faux
7 |   fin
8 fin
```

Puisque l'algorithm tourne sans arrêt (boucle sans fin) jusqu'à ce qu'une solution correcte ressorte, cela constitue un algorithme de Las Vegas.

b) Analysez le temps d'exécution de votre algorithme

Il est facile de calculer le temps d'exécution de cet algorithme. En effet, le premier algorithme à une probabilité de donner la réponse vrai au moins qvec une probabilité $0 < q < 0.5$ (car sinon il ne serais pas vrai-biaisé). De manière analogue, on trouve que la borne supérieure est d'une probabilité de 0,5.

On peux donc calculer le temps d'exécution de la manière suivante :

$$\sum_{i=1}^{\infty} i0,5^i * Boucle$$

$$\sum_{i=1}^{\infty} i0,5^i * (TA(n) + TB(n))$$

$$(TA(n) + TB(n)) \sum_{i=1}^{\infty} i0,5^i$$

$$2 * (TA(n) + TB(n))$$

On voit donc que le temps d'exécution est en temps :

$$2 * (TA(n) + TB(n)) = \mathcal{O}(2 * (TA(n) + TB(n))) \approx \mathcal{O}((TA(n) + TB(n))$$

Question 4

Trouvez les clefs publique et privée RSA avec les nombres premiers $p = 43$ et $q = 37$. En utilisant la valeur $e = 13$, montrez toutes les étapes de l'algorithme d'Euclide étendu permettant de trouver d (j'utilise ici les même noms de variables qu'en classe). Donnez toutes les étapes de vos calculs. Vous devez me donner la séquence de tous les appels récursifs, les paramètres utilisés et les valeurs de retour.

Avec les données du problème, nous avons les informations suivantes:

$$
\begin{aligned}
p &= 43 \\
q &= 37 \\
e &= 13 \\
n &= p * q = 43 * 37 = 1591 \\
\phi(n) &= (p-1)(q-1) = (43-1)(37-1) = 42 * 36 = 1512
\end{aligned}
$$

Puisque la fonction d'Euclide étendu est récursive, nous devons déterminer les valeurs de chaque appel. Pour la première itération, nous savons que:

$$
\begin{aligned}
a &= 13 \\
b &= 1512
\end{aligned}
$$

Pour les prochaines itérations, on obtient a et b de la manière suivante:

$$
\begin{aligned}
a &= b \\
b &= a (mod\, b)
\end{aligned}
$$

Nous répétons cette étape jusqu'à ce que b = 0, ce qui donne le tableau suivant:

itération	a	b	a (mod b)
1	13	1512	13
2	1512	13	4
3	13	4	1
4	4	1	0
5	1	0	-

Ensuite, nous devons trouver la valeur qui est retournée à chaque itération. Nous savons que la valeur retournée par l'itération où b = 0 est:

$$(t, x, y) = (a, 1, 0)$$

Pour les autres itérations, la valeur retournée est:

$$(t, x, y) = (t', y', x' - \lfloor a/b \rfloor * y')$$

Où (t',x',y') sont déterminés par la récursion de l'algorithme

Ce qui donne le tableau suivant:

itération	t	x	y	$(x' - \lfloor a/b \rfloor * y') = y$
5	1	1	0	-
4	1	0	1	$(1 - \lfloor 4/1 \rfloor * 0) = 1$
3	1	1	-3	$0 - \lfloor 13/4 \rfloor * 1) = -3$
2	1	-3	349	$(1 - \lfloor 1512/13 \rfloor * -3) = 349$
1	1	349	-3	$(-3 - \lfloor 13/1512 \rfloor * 349) = -3$

De ce qu'on a vu en classe (d = x), on peut alors dire que d = 349. On peut vérifier cette affirmation en testant la formule suivante:

$$\begin{aligned} e * x \ (mod\ \phi(n)) &= 1 \\ 13 * 349 \ (mod\ 1512) &= 1 \\ 4537 \ (mod\ 1512) &= 1 \end{aligned}$$

Donc,

$$d = 349$$