L’algorithme d’exponentiation rapide (Square-and-multiply)

L'algorithme de "Square and Multiply" (également connu sous le nom d'exponentiation rapide) est très apprécié en informatique, notamment dans le domaine de la cryptographie, pour calculer des puissances de manière efficace. Le principe est d’effectuer le calcul suivant: b^n mod m (où b (base), n(exposant) et m (modulo)) représentent potentiellement de très grands nombres.
L’exponentiation modulaire est une opération souvent utilisée lors des tests de primarité tels que le test de Fermat ou Solvay Strassen mais aussi dans des cryptosystèmes à clé publique tel que RSA. L’avantage d’utiliser SQM au détriment de l’exponentiation modulaire implémentée de manière naïve est de réduire le temps de calcul.

L'algorithme de Square and Multiply réduit considérablement le nombre d'opérations nécessaires pour calculer une puissance. Au lieu de multiplier la base par elle-même n−1 fois, l'algorithme utilise une approche logarithmique en termes de nombre d'opérations. Il ne nécessite que O(log(n)) multiplications, ce qui est significativement moins que les O(n) multiplications nécessaires dans l'approche naïve. L'efficacité de Square and Multiply est donc sans appel lorsqu’il s’agit de manipuler de grands nombres et devient encore plus prononcée dans les calculs modulo m, car chaque opération de multiplication peut être suivie d'une opération modulo pour garder les nombres gérables, sans augmenter la complexité de l'algorithme.

Algorithme 1 : Square-and-Multiply

Données : Un entier x, le module n et un exposant $H = \sum_{i=0}^{t} h_i 2^i$, avec $h_i \in \{0,1\}$ et $h_t = 1$
Résultat : $x^H \mod n$
$r \leftarrow x$;
pour $i = t-1, \ldots, 0$ **faire**
  $r \leftarrow r^2 \mod n$;
  si $h_i = 1$ **alors**
    $r \leftarrow r \cdot x \mod n$;
Renvoyer r ;

Étapes de l'algorithme :

1. Conversion en binaire : Convertir l'exposant n en sa représentation binaire.

2. Initialisation : Commencer avec une variable résultat initialisée à 1.

3. **Itération sur chaque bit** : - Pour chaque bit de n, de gauche à droite (du bit le plus significatif au moins significatif) effectuer un carré de la valeur actuelle du résultat (même si le bit est 0). Si le bit actuel est 1, multiplier le résultat par la base b. Appliquer le modulo m après chaque carré et chaque multiplication pour éviter que le nombre ne devienne trop grand.

4. **Résultat final** : Après avoir traité tous les bits de n, le résultat contient b^n mod n.

Cas pratique :

Supposons que nous voulons calculer 3^13mod7.

On doit d'abord exprimer l'exposant en numération binaire. (13 en binaire est 1101).

Initialisation :

Nous commençons avec le résultat initial à 1.

Itération à travers les bits de l'exposant :

- Premier bit (1) :

Nous effectuons une opération de carré sur le résultat actuel (1), puis nous multiplions par la base (3), ce qui donne 3. Ensuite, nous prenons le modulo 7, ce qui nous donne 3 mod 7 = 3.

- Deuxième bit (1) :

Nous effectuons une opération de carré sur le résultat actuel (3), puis nous multiplions par la base (3) à nouveau, ce qui donne 9. Ensuite, nous prenons le modulo 7, ce qui nous donne 9 mod 7 = 2.

- Troisième bit (0) :

Comme le bit est 0, on effectue aucune multiplication, et le résultat reste 2.

- Dernier bit (1) :

Nous effectuons une opération de carré sur le résultat actuel (2), puis nous multiplions par la base (3) à nouveau, ce qui donne 6. Ensuite, nous prenons le modulo 7, ce qui nous donne 6 mod 7 = 6.

Résultat :

Après avoir parcouru tous les bits de l'exposant, le résultat final est 6, ce qui est le résultat de 3^13mod 7 =3.

Pour n = 13, ce calcul requiert 7 opérations modulo (4 carrés et 3 multiplications), ce qui est très efficace comparé à l'approche naïve qui aurait nécessité 12 multiplications.

Sources :

https://fr.wikipedia.org/wiki/Exponentiation_rapide

http://mynath.free.fr/rsa/rsa-expo.php3

https://theses.hal.science/tel-01665008/document