MATH EN INFO

FICHE EXPLICATION N° 8

Multiplication

1 Descriptif

L'objectif de ce défi est d'implémenter une méthode calculant le produit du nombre binaire courant avec celui donné en paramètre. Malgré les apparences, l'algorithme de multiplication est très simple à coder si l'on prend le problème du bon côté.

Supposons que nous souhaitions faire la multiplication de a=11101 par b=1011. Nous allons utiliser que $b = 1000 + 10 + 1$ et donc que $ab = 1000 * a + 10 * a + 1 * a$. Or $1000 * a$ est obtenu en décalant a de 3, $10 * a$ est obtenu en décalant a de 1 et $1 * a$ en décalant a de 0.
En conclusion, ab s'obtient en calculant la somme de :

- a décalé de 3,
- a décalé de 1,
- a décalé de 0.

Vous noterez qu'il s'agit de l'algorithme que l'on réalise naturellement quand on pose une multiplication à la main.

2 Protocole

1. Une fois la connexion établie, le serveur commence par envoyer un premier message annonçant le début du défi :

   -- Début du défi : Multiplication --

   Ce message n'attend pas de réponse.
2. Le serveur envoie ensuite une série de nombres binaires (de taille aléatoire) deux par deux.
3. Pour chaque paire de nombres binaires, le serveur doit recevoir en retour un nombre binaire (sous forme binaire) égal au produit des deux nombres binaires envoyés.
4. Après chaque réponse, le serveur enverra un message commençant par "OK" ou "NOK" suivant si la réponse est correcte ou non.
5. A la fin du défi, le serveur enverra un message indiquant "Défi validé" ou "Défi échoué !". Aucune réponse n'est attendue.
6. Le serveur terminera la communication par le message "FIN", votre client devra alors fermer la socket. Aucune réponse n'est attendue.

3 Exemple de communication

Voici un exemple (incomplet) d'une communication pour ce défi. Dans cet exemple les "<" et ">" indiquent le sens de transfert de chaque message et ne doivent pas être présents dans la communication.

< -- Début du défi : Multiplication --
< 1000100010000111101101
< 111011111001001001
> 11111111100010010110000100100101001010101
< OK
< 101110010
< 100101
> 11010101111010
< OK
< 110010100110
< 101101
> 100011100100101110
< OK