DEVOIR SURVEILLÉ N° 2

Matière : **Informatique** **Filière** : **MPSI1 –G2**
Professeur : **A. ZBAKH** **Durée** : **1h30**

Remarque :

Si au cours du DS, un candidat repère ce qui peut lui sembler être une erreur d'énoncé, il le signale sur sa copie et poursuit sa composition en expliquant les raisons des initiatives qu'il est amené à prendre.

Exercice : l'algorithme RLE pour la compression de chaînes de caractères

La compression de données est le traitement informatique utilisant un algorithme particulier, qui permet de transformer une suite de bits A en une suite de bits B plus courte. Les suites A et B contiennent les mêmes informations, mais codées d'une manière différente

L'algorithme **RLE** (**Run-Length Encoding**), appelé en français le codage par plages, est un algorithme de compression informatique qui consiste à repérer et à éliminer la redondance des données. Toute suite de bits ou de caractères identiques est remplacée par un couple (caractère répété suivi de son nombre d'occurrences (son nombre de répétitions)).

Exemples:

- ✓ s1='CCCCBBBRCC' donne: s2='C4B3R1C2'
- ✓ s1 = 'AAARTTAAVVTTT' ,sa chaîne compressée s2='A3R1T2A2V2T3'

Ecrire la fonction: compresser(ch) qui prend en paramètre une chaine de caractères ch, et qui retourne la compression de ch selon l'algorithme **RLE.**

Problème : Codage cyclique

On se propose d'implanter plusieurs techniques de détection d'erreurs dans la transmission de données sur des moyens de communication non fiables. Les données transitant sur le réseau sont des séquences de bits que l'on notera 0 et 1. Ces séquences de bits sont découpées en mots $[b_0, b_1 \ldots , b_{n-1}]$ de longueur n ($n > 0$) .
Ces mots sont représentés par des tableaux d'entiers b de longueur n dont l'élément b_i a l'indice **i** vaut 0 ou 1 ($0 \leq i < n$). Le nombre d'erreurs de transmission du mot b est le nombre de bits ayant changé de valeur après la transmission.

Exemple : le mot b **[1,1,0,1,1,0,0] de taille n=7** sera représenté en Python par la liste :

i	***0***	***1***	***2***	***3***	***4***	***5***	***len(b)-1***
b	**1**	**1**	**0**	**1**	**1**	**0**	**0**

Partie I :Bit de parité

Le ou-exclusif $x \oplus y$ de deux bits x et y est défini par la table des valeurs suivante :

$x \backslash y$	0	1
0	0	1
1	1	0

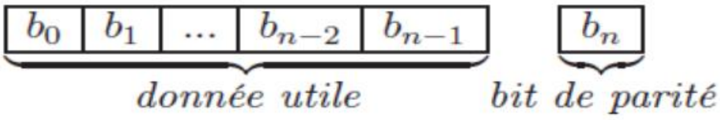
On remarque qu'on a : $0 \oplus x = x \oplus 0 = x$ et $x \oplus x = 0$ pour toute valeur de x. Donc x est son propre opposé pour $\oplus$.

Question 1 : ✍ Ecrire la fonction **ou_exclusif(x,y)** qui calcule et retourne le ou-exclusif x ⊕y des deux bits x et y pris comme arguments.

Exemple : l'appel de la fonction **ou_exclusif(1,0)** retourne 1

La technique du bit de parité consiste à rajouter un bit $\mathbf{b_n}$ (le bit de parité) aux données utiles $[\mathbf{b_0, b_1 \ldots, b_{n-1}}]$ de façon à ce que $[\mathbf{b_0, b_1 \ldots, b_{n-1}, b_n}]$ ait un nombre pair de bits à 1.

Ainsi, pour le tableau [1, 1, 0, 1, 1, 1, 0] on a 7 bits et 5 uns, donc le bit de parité sera 1. Le mot transmis sur le réseau est [1, 1, 0, 1, 1, 1, 0, **1**]. Schématiquement, on aura :



Question 2 : ✍ Ecrire la fonction **nombre1(b)** qui prend en paramètre un mot b et qui calcule et retourne le nombre de bits égaux à 1 dans b.

Exemple : l'appel de la fonction **nombre1 (**[1, 1, 0, 1, 1, 1, 0]**)** retourne 5

Question 3 : ✍ Ecrire la fonction **bit_parite(b)** qui prend en paramètre un mot b et qui retourne un mot (avec bit de parité) qui sera transmis sur le réseau

Exemple : l'appel de la fonction **bit_parite (**[1, 1, 0, 1, 1, 1, 0]**)** retourne le mot :

[1, 1, 0, 1, 1, 1, 0, **1**]

Question 4 : ✍ Ecrire la fonction **erreurs(b1,b2)** qui prend en paramètre le mot b1 avant la transmission et le mot b2 après la transmission et qui retourne le nombre d'erreurs de transmission

Exemple :

l'appel de la fonction **erreurs (**[1, 1, 0, 1, 1, 1, 0],[1, **0**, **1**, 1, **0**, 1, 0]**)** retourne 3

Partie II : Codage CRC

On peut considérer le tableau *b* de *n* bits $[\mathbf{b_0, b_1 \ldots, b_{n-1}}]$ comme les coefficients du polynôme :

$$P(X) = b_0X^{n-1} + b_1X^{n-2} + \cdots + b_{n-2}X + b_{n-1}$$

Le degré du polynôme $P(X) = b_0X^{n-1} + b_1X^{n-2} + \cdots + b_{n-2}X + b_{n-1}$ est la valeur maximale de *k* tel que b_{n-1-k} ne soit pas nul.

Question 5 : ✍ Ecrire la fonction **degre(b)** prenant en argument un tableau *b* de coefficients représentant le polynôme *P*(*X*) , et retournant le degré de *P* (Par convention, le degré du polynôme nul vaudra −1).

Exemple : Soit le mot b suivant :

i	*0*	*1*	*2*	*3*	*n-3*	*n-2*	*n-1*
b	**0**	**0**	**1**	**0**	**1**	**0**	**1**

l'appel de la fonction **degre(b)** retourne 4

La somme exclusive des deux polynômes *P*(*X*) et *Q*(*X*) dont les coefficients sont les tableaux $[\mathbf{b_0, b_1 \ldots, b_{n-1}}]$ et $[\mathbf{C_0, C_1 \ldots, C_{n-1}}]$ est définie par :

$$P(X)\oplus Q(X) = (b_0\oplus C_0)X^{n-1} + (b_1\oplus C_1)X^{n-2} + \cdots + (b_{n-2}\oplus C_{n-2})X + (b_{n-1}\oplus C_{n-1})$$

Bonne chance

Remarque :
P(*X*) est son propre opposé, puisque $P(X) \oplus P(X) = 0$.

Question 6 :	Ecrire la fonction **plus(b , C)** prenant comme arguments deux tableaux *b* et *c* représentant deux polynômes *P*(*X*) et *Q*(*X*) et qui retourne un tableau **d** représentant le polynôme somme **Exemple :** l'appel de la fonction **plus(**[1,0,1,0,0,0,0],[1,0,1,0,1,1,1]**)** retourne le polynôme de coefficients **[0, 0, 0, 0, 1, 1, 1]**

La technique, dite CRC (Cyclic Redundancy Check), ajoute plus de bits de contrôle à chaque mot transmis que la méthode du bit de parité. Elle considère un polynôme générateur G(X), bien choisi, de degré k ($0 < k \le n$), donné une fois pour toutes ; et elle construit pour tout mot b, correspondant au polynôme P(X), le reste R(X) de la division euclidienne de : $P(X) * X^k$ par G(X).

$R(X) = P(X) \cdot X^k$ mod $G(X)$ où **mod** est l'opération modulo.

Si **[$r_0, r_1, . . . r_{k-1}$]** sont les coefficients de R(X), la donnée transmise **[$b_0, b_1, . . . b_{n-1}, r_0, r_1, . . . r_{k-1}$]** correspondra à un polynôme multiple de G(X). Graphiquement, on a :

$$\underbrace{\boxed{b_0}\boxed{b_1}\boxed{\ldots}\boxed{b_{n-2}}\boxed{b_{n-1}}}_{donnée\ utile} \qquad \underbrace{\boxed{r_0}\boxed{\ldots}\boxed{r_{k-1}}}_{contrôle\ CRC}$$

Pour calculer le reste R(X) de la division de $P(X) * X^k$ par G(X), on utilise l'algorithme classique de la division :

- On aligne les bits valant 1 les plus à gauche du dividende et du diviseur, puis on retranche le diviseur au dividende (grâce à l'opération $\oplus$). Le degré du résultat est strictement inférieur à celui du dividende.
- Et on recommence la division en prenant le résultat comme nouveau dividende, jusqu'à ce que son degré soit strictement inférieur à k.

Ainsi, par exemple, pour les tableaux b = [0, 1, 1, 1, 0, 1] et g = [0, 1, 0, 1] (et donc k = 2), les étapes successives de la division donnent :

- On calcul $P(X) * X^2$ qui donne : [0, 1, 1, 1, 0, 1, 0, 0]
- On retranche le diviseur au dividende : qui donne [0, 0, 1, 0, 0, 1, 0, 0]

$$\begin{array}{rl} \oplus & [0, 1, 1, 1, 0, 1, 0, 0] \\ & [0, 1, 0, 1] \\ \hline & [0, 0, 1, 0, 0, 1, 0, 0] \end{array}$$

- Et on recommence la division en prenant le résultat comme nouveau dividende, jusqu'à ce que son degré soit strictement inférieur à k

[0, 0, 0, 0, 1, 1, 0, 0]
[0, 0, 0, 0, 0, 1, 1, 0]
[0, 0, 0, 0, 0, 0, 1, 1]

D'où la valeur [1, 1] pour le CRC.

Question 7 :	Ecrire la fonction **CRC(b,g)** prenant en argument deux tableaux b et g et qui retourne le tableau de bits correspondant aux coefficients du CRC du mot représenté par b par rapport au polynôme générateur représenté par le tableau g. (La valeur de l'argument b ne doit pas être modifiée).

Bonne chance