M3 L3 : Bande-passante, stockage et réseaux [Solution]

1. Bande-passante d'un ordinateur

a) Sachant qu'une plus grande bande passante permet une lecture plus rapide, on fait la comparaison entre les bandes passantes des ordinateurs. Pour l'ordinateur A c'est donné : 400 Mo/s. Pour l'ordinateur B on fait le calcul : 12.5×10^6 bloc/s ×4 mots/bloc ×4 octets/mot = 200 Mo/s. Alors on choisit l'ordinateur A parce que 400 Mo/s >200 Mo/s.

b) Même si le processeur de B est plus rapide, avec moitié moins de cache on ne risque pas de profiter de son gain en vitesse (une instruction est exécutée en environ 0,33ns alors qu'il faut 0,5 ns pour le processeur A, soit un gain d'environ 0,17ns par instruction). En effet, avec moitié moins de cache on a (en gros) deux fois plus de risques de faire des défauts de cache. Or chaque défaut de cache nous «coûte» de l'ordre de 100ns alors que 1 GHz de gain sur la vitesse du processeur ne fait gagner «que» 0,17ns par opération. Il faudrait donc plus de 588 (= 100/0,17) opérations réellement utiles du processeur B entre 2 défauts de cache (de A) pour pouvoir profiter du gain de vitesse de B en dépit de son cache plus petit, ce qui est peu réaliste.

2. Bande-passante bis

Le programme vérifie si x est premier.

a) Les accès de mémoire sont dans la table suivante :

Pas	Instruction	Accès Mémoire
1	$y \leftarrow 1$	lire @1 (défaut de cache)
2	$y \leftarrow 1$	lire bloc @0 en mémoire
3	$y \leftarrow 1$	placer bloc @0 en cache
4	$y \leftarrow 1$	Écrire 1 à @1 (en cache)
5	$x = 0$?	lire @0 (en cache)
6	$x = 1$?	utilise registre (x déjà lu)
7	$s \leftarrow 0$	lire @13 (défaut de cache)
8	$s \leftarrow 0$	lire bloc @12 en mémoire
9	$s \leftarrow 0$	placer bloc @12 en cache
10	$s \leftarrow 0$	Ecrire 0 à @13 (en cache)
11	$i \leftarrow 2$	lire @12 (en cache)
12	$s \leq x$?	utilise registre (s déjà lu)
13	$s \leq x$?	utilise registre (x déjà lu)
14	$y = 1$?	utilise registre (y déjà lu)
...	...	n'utilise plus que des registres

b) On a 4 accès en cache en total et 2 défauts de cache.

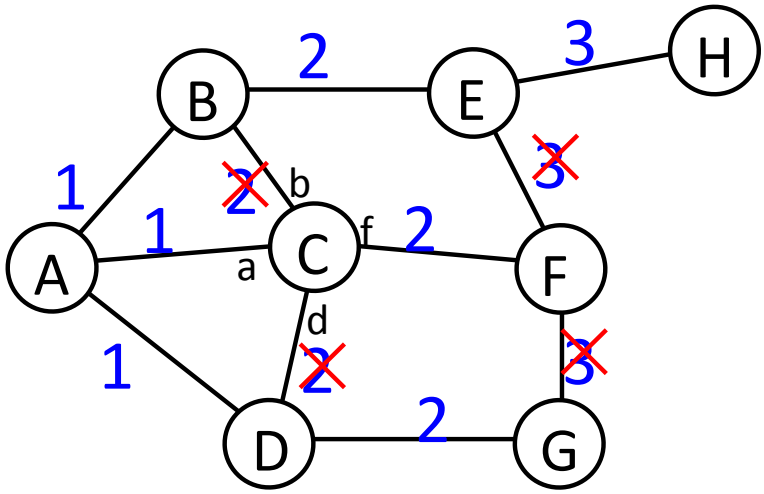
c) Pour exécuter le programme, le processeur accède la mémoire cache chaque fois qu'il veut lire ou écrire une variable. On compte 7 accès en tout, 4 en lecture comme nous l'avons vu et 3 en écriture à la fin (pour i, y et s; x, lui, n'a pas été modifié). On calcule donc le temps nécessaire pour chaque ordinateur de finir ce programme. Pour l'ordinateur A, on a besoin de 1 ns $\times 7 + 120$ ns $\times 2 = 247$ ns.

Pour l'ordinateur B, on a besoin de 1.2 ns $\times 7 + 100$ ns $\times 2 = 208$ ns.

d) Si on utilise cet exemple pour décider, on choisit l'ordinateur B; ce qui est normal ici, le programme utilisé ne nécessitant pas beaucoup de variables/mémoire, il y a peu d'accès à celle-ci mais «beaucoup» de défauts de cache (relativement au ratio d'environ 100 du temps d'accès mémoire par rapport à l'accès cache).

3. Routage IP

Propagation par ouï-dire de l'information sur le nœud A pour mettre à jour les tables de routage des autres nœuds :



Par exemple (solution non unique) :

table de C

A	1	a
B	1	b
D	1	d
E	2	b
F	1	f
G	2	d
H	3	f

4. Routage encore

C			D			E		
dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.
A	A	1	A	C	2	A	C	2
K	E	4	K	E	4	K	G	3

a)

F			G			H			I		
dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.
A	E	3	A	E	3	A	G	4	A	G	4
K	G	3	K	I	2	K	I	2	K	K	1
C			D			E					
dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.			
K	E	5	K	F	4	K	F	4			

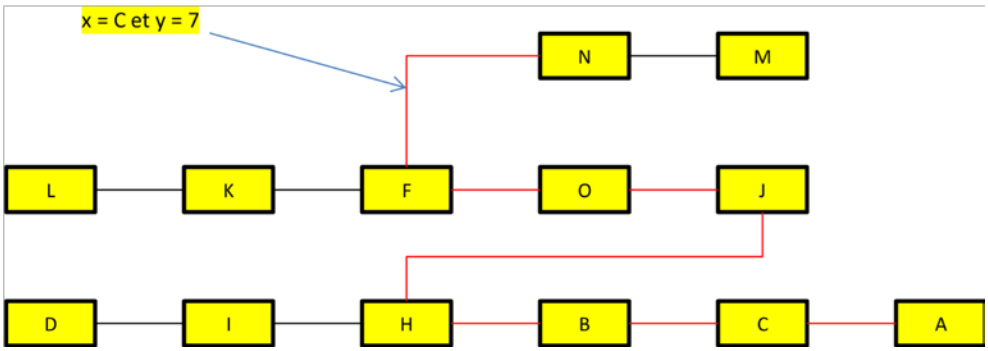
b)

F			G			H			I		
dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.
			A	F	4	A	F	4	A	G	5
C			D			E					
dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.			
K	D	5	K	F	4						

c)

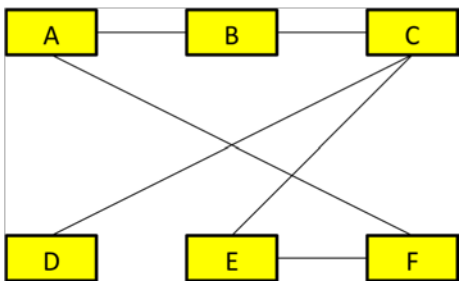
F			G			H			I		
dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.	dest.	dir.	dist.
A	D	3	A	F	4	A	F	4	A	G	5

5. Routage toujours



6 Encore un peu de routage?

oui et la longueur est 4 :



7 Synthèse musicale

7.1 Besoin de structure

Retrouvez l'amplitude du signal sonore au temps 0.01 s dans l'information suivante stockée sur le disque :
C'est évidemment impossible!...
Il faut la structure de ces informations pour les décoder. C'est justement le but du reste de l'exercice.

7.2 Echantillonnage du signal

Ce signal peut-il être reconstitué exactement à partir des échantillons du CD?
Oui sans souci : la bande passante est de $5\times440 = 2200$ Hz qui est bien inférieur à 44100/2=22'050 Hz.

Sinon, que devrait-on faire?
Sinon on devrait filtrer les hautes fréquences, celles supérieures à 22'050 Hz.

7.3 Codage du signal

L'amplitude du signal à chaque échantillon est stocké sur 16 bits. On peut stocker 65'536 valeurs, MAIS il faut considérer qu'on peut avoir des amplitudes négatives puisqu'il s'agit d'une somme de sinusoïdes: il faut donc un bit de signe. Avec les 15 bits restant on a donc 32768 niveaux distincts.

Est-ce possible de comprimer à 50%?
On parle *bien sûr* ici de compression sans perte et sans préfixe (pour redécoder), donc le th. de Shannon s'applique : on ne peut pas comprimer en dessous de 10.5 bit par valeur, soit une compression de 1-10.5/16 = 34%.

Quel taux de compression obtient-on avec Huffman (ordre de grandeur)?
Toujours par le th. de Shannon : $H \leq L \leq H+1$ donc on a entre 10.5 et 11.5 bit par valeur en moyenne, soit entre 28% et 34% de compression. Disons 30%.

7.4 Stockage sur disque : contenu du fichier

Combien de blocs (environ) utilise notre fichier?
1 seconde de musique donne 44'100 échantillons soit 485'100 bits soit 60'637 octets soit environ 60 Kio, c'est-à-dire 15 blocs.
Pour 2 s, il faut donc 30 blocs.

Nous avons un disque de 512 Gio : combien de blocs contient-il?

$$2^9 \cdot 2^{10} \cdot 2^{10} = 2^{29}\text{Koctets} = 2^{27}\text{blocs}$$

Combiens de bits sont nécessaire pour l'adresse des blocs?
Il faut donc (au moins) 27 bits.

Taille de la table représentée sur la première ligne de cette table ?
L'entier non-signé représenté sur cette ligne est bien 30, comme trouvé pour le nombre de blocs des 2 secondes de musique.

Quelle est l'adresse du bloc correspondant à l'amplitude du signal sonore au temps 0.01 s?
0.01 s correspond au 441e échantillons, c'est-à-dire à 4851 bits, soient 606 octets : c'est donc dans le 1er bloc. On doit donc décoder la 1re adresse, c'est-à-dire la 2e ligne, soit 00000000000000000000000000101011, qui représente **43** en binaire.

7.5 Stockage sur disque : nom absolu du fichier

Combien de blocs utilise alors le répertoire Musique?
La taille de la table de Musique est de $(1+30) \cdot 32$ bits (1 pour la taille et 30 adresses), soient 992 bits. 1 bloc suffit donc.

Quel est le début du fichier représentant le répertoire muche?
00000000000000000000000000000001 <<- taille = 1
00000000000000000000000000101010 <<- 42

7.6 Décodage du « message » de départ

Quelle est la valeur de l'amplitude de notre signal au temps 2/44100 s?
On décode : en 42 on trouve la table de notre fichier de musique. Le temps 2/44100 s correspond au 3e échantillon, qui est donc dans le 1erbloc, dont l'adresse est 43 (décodée plus haut dans la question 7.4).

On doit donc décoder le bloc 43, ce qui est fait en y cherchant des codes de Huffman. On trouve les 3 code successifs: 11001100110, 10101010, 0001110001 ce qui correspond donc aux amplitudes successives suivantes: 0 puis 5929 et enfin 11056.

Celle qui nous intéresse est la troisième = 11056, qui nous donne une amplitude (physique) de $4 \cdot 11056/32768$ = $\mathbf{1.35}$.