École de technologie supérieure	**Trimestre**	**Hiver 2016**
Département de génie logiciel et des TI	**Enseignant**	**Stéphane Coulombe**
	Préalable	**LOG120**
	Crédits	**4**

GTI310 Structure de données multimédias

Quatrième Travail Pratique

Sujet : Codeur d'image quasi-JPEG

Description

La compagnie Squeeze-media conçoit des bases de données d'images. Ils voudraient améliorer la performance de leur système en permettant de stocker plus d'images dans leur base de données. Puisque vous avez réussi avec brio le cours GTI310, vous leur suggérez d'utiliser une méthode de compression avec perte du genre que JPEG utilise.

Le patron est très enthousiaste à l'idée, mais l'ingénieur principal a plusieurs inquiétudes. Ce dernier ne veut pas utiliser de logiciels inconnus venant de l'extérieur de la compagnie. En particulier, il ne veut pas que vous utilisiez du code JPEG accessible sur le Web. Le code serait trop complexe et il ne veut pas le gérer. Il veut un codec simple fait de code simple même si le facteur de compression n'est pas optimal. Surtout, il veut pouvoir contrôler le facteur de qualité pour chaque image à coder. Concrètement, il veut pouvoir spécifier un facteur de qualité entier entre 1 (pire qualité) et 100 (meilleure qualité).

Vous vous assoyez donc avec l'ingénieur principal et lui expliquez le diagramme de flot du codec quasi-JPEG que vous planifiez implémenter. Les figures suivantes montrent les schémas de flot de l'encodeur et du décodeur (le décodeur effectue les opérations inverses de l'encodeur). Les images à l'entrée de l'encodeur seront en format PPM (voir les notes de cours pour le format de ce fichier) binaire. L'image d'entrée sera en couleur RVB. On la convertira en YUV pour la compression. Au décodeur, l'image sera reconvertie en format RVB dans le format PPM.

Le département de marketing décide de commercialiser votre format de compression sous le nom de **squeeze-light**.

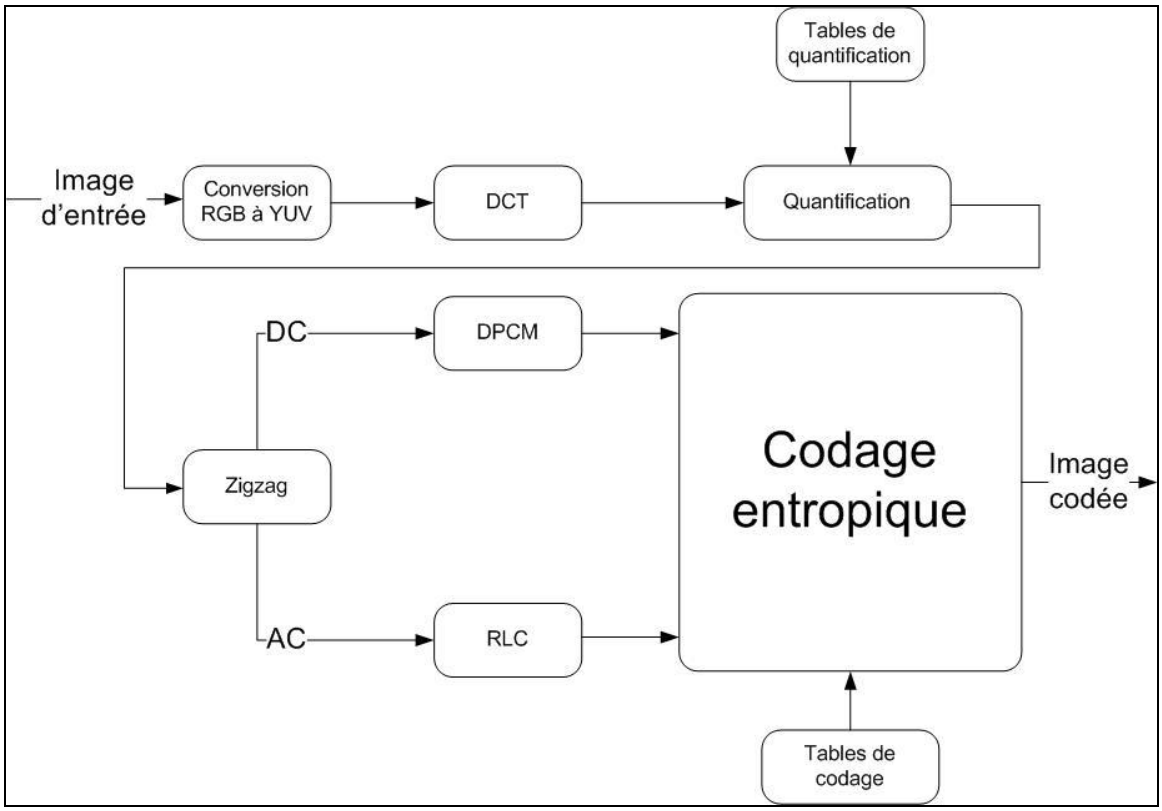


Figure 1 : Séquence d'encodage.

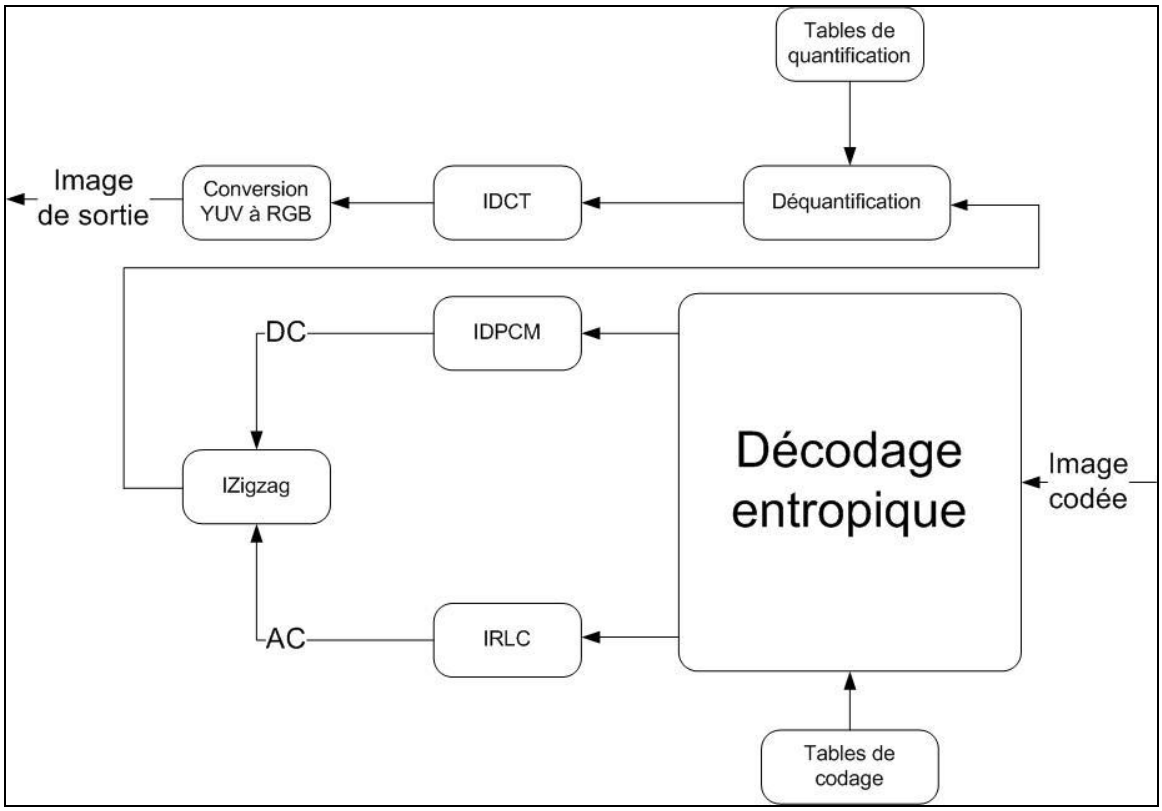


Figure 2 : Séquence de décodage.

Vous devez donc concevoir un logiciel interactif permettant à l'usager d'encoder ou de décoder des images en format squeeze-light (format privé à Squeeze-media). L'usager choisira le fichier à encoder, spécifiera le nom du fichier compressé ainsi que le facteur de qualité. Similairement pour le décodage, il choisira le fichier compressé et spécifiera le nom du fichier à la sortie. **Les détails et la syntaxe que le codec doit respecter sont décrits à l'annexe I.**

Analyse de complexité

Vous devez analyser (notation O) la complexité de chacune des opérations associées à la résolution du problème et mettre les résultats en commentaire dans chacune des procédures analysées.

Questions supplémentaires

1. Pour les images de référence lena.ppm et peppers.ppm, faites un graphique (en utilisant des facteurs de qualité de 1, 10, 20, 30, 50, 70, 90 et 100) :
    a. Du facteur de compression en fonction du facteur de qualité.
    b. Du niveau de qualité subjective en fonction du facteur de qualité.
2. Quelles modifications devriez-vous apporter à votre programme afin qu'il puisse supporter la compression en tons de gris en plus de couleur (c.-à-d. l'usager peut décider de conserver seulement l'information de tons de gris d'une image couleur lors de la compression)? Donnez les changements majeurs sans entrer dans les détails inutiles.

À remettre

- Un rapport de laboratoire
    - Voir le gabarit qui est disponible sur le site du cours.
    - Mettre le code source en annexe, à part le code qui vous a été fourni.
- Le projet contenant votre code source
    - NE PAS REMETTRE D'IMAGES!
- Les fichiers « .project » et « .classpath » (projet *Eclipse*) de façon à ce que le projet soit chargé rapidement.

Voir le document intitulé « Procédure de remise des laboratoires » sur le site du cours pour les détails supplémentaires concernant la remise.

Critères d'évaluation

1. Rapport de laboratoire : 40 points
    a. Introduction : 3 points
    b. Choix de design : 8 points
    c. Implémentation : 8 points
    d. Vérification : 8 points
    e. Analyse du projet : 8 points
    f. Conclusion : 2 points
    g. Réponse aux questions : 3 points
2. Programme : 60 points
    a. Qualité du code[1] : 10 points
    b. Fonctionnement : 45 points
        i. Robustesse : il ne faut pas que le programme plante
        ii. Logique : est-ce que le fonctionnement semble logique pour l'utilisateur.
        iii. Résultats attendus.
    c. Analyse de complexité : 5 points
3. Déductions
    a. Qualité du Français : jusqu'à 10 points
    b. Absence des fichiers « .project » et « .classpath » : 20 points
    c. Absence de librairies nécessaires à la compilation : 10 points
    d. Présentation du rapport de laboratoire : jusqu'à 5 points

Date de remise

Au plus tard le 11 avril 2016 à 23h59. Vous devez effectuer une remise électronique sur Moodle de votre travail (code et rapport en format .doc ou .docx).

Pénalité de retard

Aucun retard ne sera accepté. La note de zéro sera attribuée si le travail n'est pas remis dans les délais.

Plagiat et fraude

Les clauses du « Chapitre 10 : Plagiat et fraude » du « Règlement des études de 1er cycle » s'appliquent dans ce cours ainsi que dans tous les cours du département de génie logiciel et des TI.

ANNEXE I : Détails du codec quasi-JPEG

i. Conversion RGB à YUV

Voir notes de cours

ii. DCT

Le codec utilisera des blocs 8 pixels par 8 pixels. La DCT est défini pour un bloc de pixels f(i,j) de dimensions 8x8 par :

$$F(u,v) = \frac{C(u)C(v)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7}\left(\cos\left(\frac{(2i+1)u\pi}{16}\right)\cdot\cos\left(\frac{(2j+1)v\pi}{16}\right)\right)f(i,j)$$

$$\text{avec u,v=0,1,...,7 et } C(w) = \begin{cases}\frac{1}{\sqrt{2}} & \text{si } w = 0\\ 1 & \text{sinon}\end{cases}$$

avec i,j = 0, 1, …, 7

La DCT inverse (IDCT) est définie par :

$$\tilde{f}(i,j) = \sum_{u=0}^{7}\sum_{v=0}^{7}\frac{C(u)C(v)}{4}\left(\cos\left(\frac{(2i+1)u\pi}{16}\right)\cdot\cos\left(\frac{(2j+1)v\pi}{16}\right)\right)F(u,v)$$

$$\text{avec i,j=0,1,...,7}$$

iii. Quantification

Les tables de quantification Q pour la luminance (Y) et la chrominance (U et V) sont définies par :

$$Q_Y = \begin{bmatrix} 16 & 40 & 40 & 40 & 40 & 40 & 51 & 61 \\ 40 & 40 & 40 & 40 & 40 & 58 & 60 & 55 \\ 40 & 40 & 40 & 40 & 40 & 57 & 69 & 56 \\ 40 & 40 & 40 & 40 & 51 & 87 & 80 & 62 \\ 40 & 40 & 40 & 56 & 68 & 109 & 103 & 77 \\ 40 & 40 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 95 \end{bmatrix}$$

$$Q_U = Q_V = \begin{bmatrix} 17 & 40 & 40 & 95 & 95 & 95 & 95 & 95 \\ 40 & 40 & 40 & 95 & 95 & 95 & 95 & 95 \\ 40 & 40 & 40 & 95 & 95 & 95 & 95 & 95 \\ 40 & 40 & 95 & 95 & 95 & 95 & 95 & 95 \\ 95 & 95 & 95 & 95 & 95 & 95 & 95 & 95 \\ 95 & 95 & 95 & 95 & 95 & 95 & 95 & 95 \\ 95 & 95 & 95 & 95 & 95 & 95 & 95 & 95 \\ 95 & 95 & 95 & 95 & 95 & 95 & 95 & 95 \end{bmatrix}$$

Figure 3 Tables de quantification pour la luminance et la chrominance

L'opération de quantification pour chaque bloc 8x8 de chaque composante (Y, U et V) est définie par :

$$\hat{F}(u,v) = \begin{cases} F(u,v) \text{ si } f_Q = 100 \\ arrondir\left(\dfrac{F(u,v)}{\alpha Q(u,v)}\right) \text{ sinon} \end{cases}$$

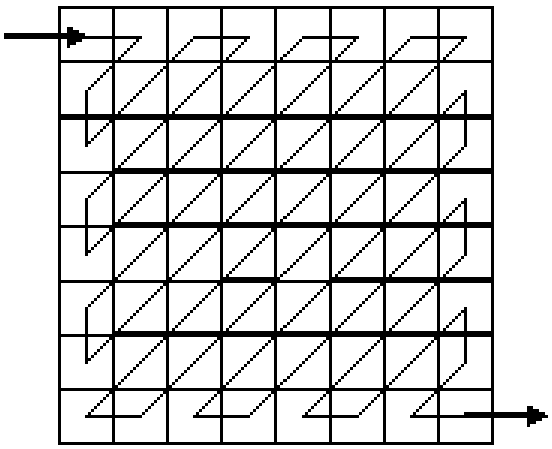
avec u,v=0,1,...,7

La valeur de α est calculée en fonction du facteur de qualité désiré. Si le facteur qualité, noté f_Q, est 100, alors on ne quantifie pas les coefficients. C'est équivalent à utiliser une matrice de quantification avec des valeurs de 1 pour chacun des éléments. Si $f_Q < 100$, alors on utilise la formule suivante :

$$\alpha = \begin{cases} \dfrac{50}{f_Q} & \text{si } 1 \leq f_Q \leq 50 \\ \dfrac{200\text{-}2f_Q}{100} & \text{si } 50 \leq f_Q \leq 99 \end{cases}$$

iv. Zigzag

La fonction de zigzag change l'ordre des coefficients pour rendre le codage plus efficace (pour avoir de plus longues séquences de zéros). Vous devez utiliser le parcours en zigzag illustré ci-dessous.



v. Traitement des coefficients DC

Tous les coefficients DC de l'image (ceux à la position F(0,0) de chaque bloc 8x8) devront être codés ensemble à l'aide de la méthode DPCM (méthode différentielle). Par exemple, si les DC sont 150, 155, 149, 152 et 144, alors après l'étape DPCM, on aura : 150, 5, -6, 3, -8.

Chaque coefficient DC encodé DPCM va se faire attribuer une paire de symboles (SIZE, AMPLITUDE), avec SIZE indiquant le nombre de bits requis pour représenter le coefficient d'AMPLITUDE tel que spécifié à la table suivante. SIZE est codé Huffman mais pas AMPLITUDE. Pour AMPLITUDE, on utilise le complément à un (c.-à-d. inverse tous les bits de sa version positive) pour les valeurs négatives.

SIZE	AMPLITUDE
1	-1, 1
2	-3, -2, 2, 3
3	-7..-4, 4..7
4	-15..-8, 8..15
.	.
.	.
.	.
10	-1023..-512, 512..1023

Par exemple, 150, 5, -6, 3, -8 serait codé comme suit : (8, 10010110), (3, 101), (3, 001), (2, 11), (4, 0111).

La table de Huffman utilisée pour coder SIZE est la suivante :

Category	Base Code	Length	Category	Base Code	Length
0	010	3	6	1110	10
1	011	4	7	11110	12
2	100	5	8	111110	14
3	00	5	9	1111110	16
4	101	7	A	11111110	18
5	110	8	B	111111110	20

Ainsi, (8, 10010110), (3, 101), (3, 001), (2, 11), (4, 0111) serait codé : **111110**10010110**00**101**00**001**100**111**101**0111

vi. Traitement des coefficients AC

Pour les coefficients AC, on utilise le RLC (ou RLE). La paire (RUNLENGTH, VALUE) est étendue à (RUNLENGTH, SIZE, AMPLITUDE), c.-à-d. VALUE est divisée en SIZE /AMPLITUDE comme pour les coefficients DC. On attribue 4 bits pour RUNLENGTH et 4 bits pour SIZE. Cela donne le code tel qu'illustré dans la table suivante :

		SSSS										
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	0	EOB	01	02	03	04	05	06	07	08	09	0A
	1	N/A	11	12	13	14	15	16	17	18	19	1A
	2	N/A	21	22	23	24	25	26	27	28	29	2A
	3	N/A	31	32	33	34	35	36	37	38	39	3A
	4	N/A	41	42	43	44	45	46	47	48	49	4A
R	**5**	N/A	51	52	53	54	55	56	57	58	59	5A
R	**6**	N/A	61	62	63	64	65	66	67	68	69	6A
R	**7**	N/A	71	72	73	74	75	76	77	78	79	7A
R	**8**	N/A	81	82	83	84	85	86	87	88	89	8A
	9	N/A	91	92	93	94	95	96	97	98	99	9A
	10	N/A	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA
	11	N/A	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA
	12	N/A	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA
	13	N/A	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA
	14	N/A	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA
	15	ZRL	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA

Notez que pour des séquences de plus de 15 zéros consécutifs, on utilise un code spécial : ZRL. Si on n'a aucune séquence de zéros, ou si un bloc se termine par une séquence de zéro, on utilise EOB. Les 8 bits composés de RUNLENGTH et SIZE sont ensuite codés Huffman, mais pas AMPLITUDE. Par exemple, (0,6) en format (RUNLENGTH, VALUE) sera représenté par (0, 3, 110) en format (RUNLENGTH, SIZE, AMPLITUDE). (0,3) sera représenté par (100). Donc (0,6) sera représenté par 100110. Les huit bits formés par RUNLENGTH et SIZE sont codés Huffman[2] tel que décrit à la figure de la page suivante.

[2] Voir l'ANNEXE III pour la table de Huffman complète.

vii. Syntaxe du fichier compressé

La syntaxe du fichier est comme suit :

Entête
- Code magique de Squeeze-light : 0xEA (8bits)
- Largeur de l'image : 16 bits signés
- Hauteur de l'image : 16 bits signés
- Nombre de composantes de couleur: 8 bits non signés
- Facteur de Qualité : 8 bits non signés

Composante DC Y Bloc 1
Composante DC Y Bloc 2
…
Composante DC U Bloc 1
Composante DC U Bloc 2
…
Composante DC V Bloc 1
Composante DC V Bloc 2
…
Composante AC Y Bloc 1
Composante AC Y Bloc 2
…
Composante AC U Bloc 1
Composante AC U Bloc 2
…
Composante AC V Bloc 1
Composante AC V Bloc 2
…

ANNEXE III : Tables de Huffman pour AC

Run/size	Code length	Code word
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	0111000
0/7	8	01111000
0/8	10	1111110110
0/9	16	1111111110000010
0/A	16	1111111110000011
1/1	4	1100
1/2	5	11011
1/3	7	1111001
1/4	9	111110110
1/5	11	11111110110
1/6	16	1111111110000100
1/7	16	1111111110000101
1/8	16	1111111110000110
1/9	16	1111111110000111
1/A	16	1111111110001000
2/1	5	11100
2/2	8	11111001
2/3	10	1111110111
2/4	12	111111110100
2/5	16	1111111110001001
2/6	16	1111111110001010
2/7	16	1111111110001011
2/8	16	1111111110001100
2/9	16	1111111110001101
2/A	16	1111111110001110
3/1	6	111010
3/2	9	111110111
3/3	12	111111110101
3/4	16	1111111110001111
3/5	16	1111111110010000
3/6	16	1111111110010001
3/7	16	1111111110010010
3/8	16	1111111110010011
3/9	16	1111111110010100
3/A	16	1111111110010101
4/1	6	111011
4/2	10	1111111000
4/3	16	1111111110010110
4/4	16	1111111110010111
4/5	16	1111111110011000
4/6	16	1111111110011001
4/7	16	1111111110011010
4/8	16	1111111110011011
4/9	16	1111111110011100
4/A	16	1111111110011101
5/1	7	1111010
5/2	11	11111110111
5/3	16	1111111110011110
5/4	16	1111111110011111
5/5	16	1111111110100000
5/6	16	1111111110100001

5/7	16	1111111110100010
5/8	16	1111111110100011
5/9	16	1111111110100100
5/A	16	1111111110100101
6/1	7	1111011
6/2	12	111111110110
6/3	16	1111111110100110
6/4	16	1111111110100111
6/5	16	1111111110101000
6/6	16	1111111110101001
6/7	16	1111111110101010
6/8	16	1111111110101011
6/9	16	1111111110101100
6/A	16	1111111110101101
7/1	8	11111010
7/2	12	111111110111
7/3	16	1111111110101110
7/4	16	1111111110101111
7/5	16	1111111110110000
7/6	16	1111111110110001
7/7	16	1111111110110010
7/8	16	1111111110110011
7/9	16	1111111110110100
7/A	16	1111111110110101
8/1	9	111111000
8/2	15	111111111000000
8/3	16	1111111110110110
8/4	16	1111111110110111
8/5	16	1111111110111000
8/6	16	1111111110111001
8/7	16	1111111110111010
8/8	16	1111111110111011
8/9	16	1111111110111100
8/A	16	1111111110111101
9/1	9	111111001
9/2	16	1111111110111110
9/3	16	1111111110111111
9/4	16	1111111111000000
9/5	16	1111111111000001
9/6	16	1111111111000010
9/7	16	1111111111000011
9/8	16	1111111111000100
9/9	16	1111111111000101
9/A	16	1111111111000110
A/1	9	111111010
A/2	16	1111111111000111
A/3	16	1111111111001000
A/4	16	1111111111001001
A/5	16	1111111111001010
A/6	16	1111111111001011
A/7	16	1111111111001100
A/8	16	1111111111001101
A/9	16	1111111111001110
A/A	16	1111111111001111
B/1	10	1111111001
B/2	16	1111111111010000
B/3	16	1111111111010001
B/4	16	1111111111010010
B/5	16	1111111111010011
B/6	16	1111111111010100
B/7	16	1111111111010101
B/8	16	1111111111010110

B/9	16	1111111111010111
B/A	16	1111111111011000
C/1	10	1111111010
C/2	16	1111111111011001
C/3	16	1111111111011010
C/4	16	1111111111011011
C/5	16	1111111111011100
C/6	16	1111111111011101
C/7	16	1111111111011110
C/8	16	1111111111011111
C/9	16	1111111111100000
C/A	16	1111111111100001
D/1	11	11111111000
D/2	16	1111111111100010
D/3	16	1111111111100011
D/4	16	1111111111100100
D/5	16	1111111111100101
D/6	16	1111111111100110
D/7	16	1111111111100111
D/8	16	1111111111101000
D/9	16	1111111111101001
D/A	16	1111111111101010
E/1	16	1111111111101011
E/2	16	1111111111101100
E/3	16	1111111111101101
E/4	16	1111111111101110
E/5	16	1111111111101111
E/6	16	1111111111110000
E/7	16	1111111111110001
E/8	16	1111111111110010
E/9	16	1111111111110011
E/A	16	1111111111110100
F/0	11	11111111001
F/1	16	1111111111110101
F/2	16	1111111111110110
F/3	16	1111111111110111
F/4	16	1111111111111000
F/5	16	1111111111111001
F/6	16	1111111111111010
F/7	16	1111111111111011
F/8	16	1111111111111100
F/9	16	1111111111111101
F/A	16	1111111111111110

Voir le site suivant pour les opérateurs binaires en Java :
http://leepoint.net/notes-java/data/expressions/bitops.html

Le code nécessaire pour faire le codage entropique vous est fourni. Vous devriez y jeter un coup d'œil pour voir comment est faite la manipulation de bits en Java.