Département d'informatique et de génie logiciel

Compression de données
IFT-4003/IFT-7023

Notes de cours
Codage arithmétique: génération du code binaire

Édition Hiver 2012

Mohamed Haj Taieb

Local: PLT 2113

Courriel: mohamed.haj-taieb.1@ulaval.ca

Faculté des sciences et de génie
Département de génie électrique et de génie informatique

UNIVERSITÉ LAVAL

Plan de la présentation

- Codage arithmétique:
  - Génération d'un code binaire
  - Unicité et efficacité du code
  - Algorithme d'implémentation
  - Implémentation entière
  - Comparaison avec le codage de Huffman

Génération du tag

- Génération du tag
  - Pour une séquence x on a obtenu un tag T_X.
  - Mais on cherche a représenter la séquence x avec un code binaire unique et efficace.
  - Comme le tag est unique → la représentation binaire du tag est unique → représentation binaire unique de la séquence.
  - Cependant le tag peut prendre une infinité de valeur de l'intervalle qui lui est associé → code peut être long.
- Troncation du code
  - Même si le code est unique il peut être non performant.
  - Il faut tronquer le code tout en préservant son unicité.

Unicité et efficacité du code arithmétique

- Représentation binaire tronqué du tag
  - Le tag est un nombre réel tq $T_X \in [0\ 1)$.
  - On considère une représentation binaire tronquée du tag de longueur:

$$l\ x = \left\lceil \log_2 \frac{1}{P\ x} \right\rceil + 1 \text{ bits}$$

- Exemple: code binaire généré à partir du tag
  - Soit un alphabet formé de trois lettre A= $\{a_1, a_2, a_3, a_4\}$ avec $P(a_1)=1/2$, $P(a_2)=1/4$, $P(a_3)=1/4$ et $P(a_4)=1/8$.
  - On considère un code binaire tronqué du tag T_X.
  - Un tag est alloué à chaque symbole: séquence formée d'un symbole unique.

Exemple de code binaire généré à partir du tag

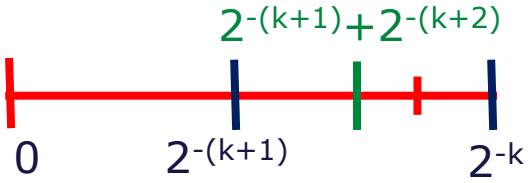
$$\bar{T}_X(a_i) = F_X(i-1) + \frac{1}{2}P(x=i)$$

- Tag d'un tag pour un symbole unique:
  - $P(a_1)=1/2$, $P(a_2)=1/4$, $P(a_3)=1/4$ et $P(a_4)=1/8$
  - $0.25=0x2^{-1} + 1x2^{-2} + 0x\ 2^{-3}+ 0 \ldots \rightarrow 0.25=010_2$.
  - $0.8125=1x2^{-1} + 1x2^{-2} + 1x\ 2^{-3} + 1x\ 2^{-4} + 0 \ldots \rightarrow 1111_2$.

Symbole	F_X	T_X	Binaire $T_{X\|base\ 2}$	$l(x) = \left\lceil \log_2 \frac{1}{P(x)} \right\rceil + 1$	Code $\lfloor \bar{T}_X(x)_{base\ 2} \rfloor_{l(x)}$
1	0.5	0.25	.010	2	01
2	0.75	0.625	.101	3	101
3	0.875	0.8125	.1101	4	1101
4	1.0	0.9375	.1111	4	1111

Décodabilité unique de ce code

- Montrons d'abord l'unicité du code: $\left\lfloor \bar{T}_X(x)_{\text{base 2}} \right\rfloor_{l(x)}$ $l(x) = \left\lceil \log_2 \frac{1}{P(x)} \right\rceil + 1$
  - On a $T_X(x) \in [F_X(x-1), F_X(x))$ et toute valeur de cet intervalle peut être un identificateur unique.
  - Donc pour montrer l'unicité de $\left\lfloor \bar{T}_X(x)_{\text{base 2}} \right\rfloor_{l(x)}$ il faut prouver qu'il se trouve dans $[F_X(x-1), F_X(x))$.
  - $T_X = b_1 x 2^{-1} + b_2 x 2^{-2} + \ldots + b_{l(x)} x 2^{-l(x)} + b_{l(x)+1} x 2^{-(l(x)+1)} \ldots$
  - $T_X = b_1 b_2 \ldots b_{l(x)} b_{l(x)+1} \ldots$ $b_i \in \{0,1\}$
  - $\left\lfloor \bar{T}_X(x)_{\text{base 2}} \right\rfloor_{l(x)} = b_1 b_2 \ldots b_{l(x)}$
  - $\left\lfloor \bar{T}_X(x)_{\text{base 2}} \right\rfloor_{l(x)} = b_1 x 2^{-1} + b_2 x 2^{-2} + \ldots + b_{l(x)} x 2^{-l(x)}$
  - $\bar{T}_X(x) - \left\lfloor \bar{T}_X(x)_{\text{base 2}} \right\rfloor_{l(x)} = b_{l(x)+1} x 2^{-(l(x)+1)} + b_{l(x)+2} x 2^{-(l(x)+2)} \ldots$
  - Et comme: $\sum_{i=k+1}^{N} 2^{-i} < 2^{-k} \Rightarrow 0 \le \bar{T}_X(x) - \left\lfloor \bar{T}_X(x) \right\rfloor_{l(x)} < 2^{-l(x)}$



Unicité du code $\lfloor \bar{T}_X(x) \rfloor_{l(x)}$

- Unicité du code:

$$0 \le \bar{T}_X(x) - \lfloor \bar{T}_X(x) \rfloor_{l(x)} < 2^{-l(x)}$$

  - Ainsi $\boxed{\lfloor \bar{T}_X(x) \rfloor_{l(x)} \le \bar{T}_X(x) < F_X(x)}$
  - Par ailleurs on a:

$$l(x) = \left\lceil \log_2 \frac{1}{P\ x} \right\rceil + 1 \Rightarrow 2^{-l(x)} = 2^{-\left\lceil \log_2 \frac{1}{P\ x} \right\rceil - 1} \le 2^{-\log_2 \frac{1}{P\ x} - 1} = \frac{P\ x}{2}$$

$$\bar{T}_X(x) = F_X(x-1) + \frac{1}{2}P(x) \Rightarrow \frac{1}{2}P(x) = \bar{T}_X(x) - F_X(x-1) > 2^{-l(x)} > \bar{T}_X(x) - \lfloor \bar{T}_X(x) \rfloor_{l(x)}$$

$$\bar{T}_X(x) - F_X(x-1) > \bar{T}_X(x) - \lfloor \bar{T}_X(x) \rfloor_{l(x)}$$

$$\boxed{\lfloor \bar{T}_X(x) \rfloor_{l(x)} > F_X(x-1)}$$

  - Finalement on obtient $\lfloor \bar{T}_X(x) \rfloor_{l(x)} \in [F_X(x-1), F_X(x))$
  - Le tag tronqué à une longueur l(x) est représentation unique du tag lui-même.

Décodabilité unique (1)

- Montrons ensuite décodabilité unique:
  - Pour montrer la décodabilité unique de ce code il suffit de vérifier que ce code est un code préfixe.
- Si b est un préfixe de a alors b ∈ [a, a+2^{-n}):
  - Soit a ∈ [0,1) tq a=$b_1x2^{-1}+b_2x2^{-2}+\ldots+b_nx2^{-n}$ =$b_1b_2\ldots b_n$
  - Soit b ∈ [0,1) tq b=$b_1x2^{-1}+b_2x2^{-2}+\ldots+b_nx2^{-n}+b_{n+1}x2^{-(n+1)}$ +…+ b_mx2^{-m} =$b_1b_2\ldots b_nb_{n+1}\ldots b_m$
  - b-a= $b_{n+1}x2^{-(n+1)}$ +…+ b_mx2^{-m} < $2^{-(n+1)}$ +…+2^{-m}< 2^{-n}
- Soient x et y 2 séquences distinctes:
  - Pour montrer que y n'est un pas préfixe de x il faut:

$$\text{Montrer que } \left\lfloor \bar{T}_X(y) \right\rfloor_{l(y)} \notin [\left\lfloor \bar{T}_X(x) \right\rfloor_{l(x)}, \left\lfloor \bar{T}_X(x) \right\rfloor_{l(x)} + 2^{-l(x)})$$

Décodabilité unique (2)

- Montrons que $\lfloor \bar{T}_X(y) \rfloor_{l(y)} \notin [\lfloor \bar{T}_X(x) \rfloor_{l(x)}, \lfloor \bar{T}_X(x) \rfloor_{l(x)} + 2^{-l(x)})$
- On a $\lfloor \bar{T}_X(x) \rfloor_{l(x)} \in [F_X \quad x-1 \ , F_X(x))$ et $\lfloor \bar{T}_X(y) \rfloor_{l(y)} \in [F_X(y-1), F_X \quad y \ )$

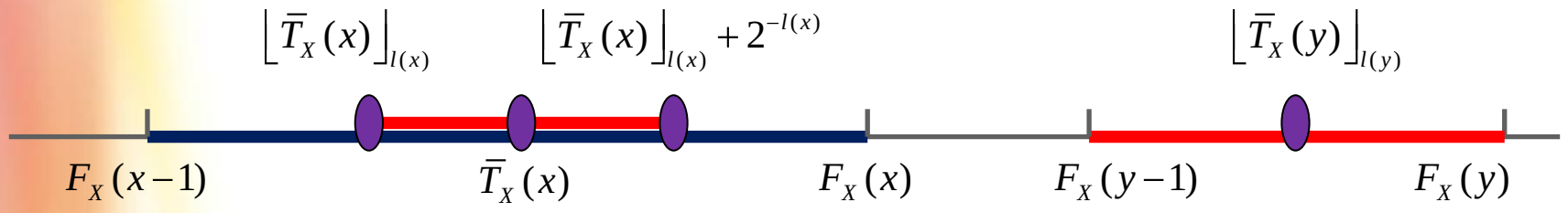
Il reste à prouver que : $\lfloor \bar{T}_X(x) \rfloor_{l(x)} + 2^{-l(x)} < F_X(x)$

On a: $F_X(x) - \lfloor \bar{T}_X(x) \rfloor_{l(x)} > F_X(x) - \bar{T}_X(x) = \frac{P(x)}{2} > 2^{-l(x)}$

$\Rightarrow \lfloor \bar{T}_X(x) \rfloor_{l(x)} + 2^{-l(x)} < F_X(x)$

- Donc la représentation binaire tronqué du tag $\lfloor \bar{T}_X(x) \rfloor_{l(x)}$ de longueur

$l(x) = \left\lceil \log_2 \frac{1}{P(x)} \right\rceil + 1$ est un code préfixe donc uniquement décodable.



Efficacité de ce code (1)

$$l(x) = \left\lceil \log_2 \frac{1}{P(x)} \right\rceil + 1$$

- Longueur moyenne du code:
  - Même si le code est uniquement décodable il faut aussi vérifier son efficacité. Soit une séquence x de longueur m:

$$l_{Arith}^{m} = \sum_{\forall x: l(x)=m} P(x)l(x)$$

$$= \sum P(x)\left(\left\lceil \log_2 \frac{1}{P(x)} \right\rceil + 1 \right)$$

$$< \sum P(x)\left(\log_2 \frac{1}{P(x)} + 1 + 1\right)$$

$$= \sum P(x) \log_2 \frac{1}{P(x)} + 2\sum P(x)$$

$$= H(X^{(m)}) + 2$$

Mohamed Haj Taieb, Département de génie électrique et de génie informatique

Efficacité de ce code (2)

- Longueur moyenne du code:
  - Comme on ne peut pas compresser une séquence au-delà de son entropie les bornes de la longueur moyenne de notre code sont données par:

$$H(X^{(m)}) \leq l^{m}_{Arith} < H(X^{(m)}) + 2$$

$$\frac{H(X^{(m)})}{m} \leq l_{Arith} < \frac{H(X^{(m)}) + 2}{m}$$

avec l_{Arith} la longueur moyenne d'un symbole

$$\boxed{H(X) \leq l_{Arith} < H(X) + \frac{2}{m}} \text{ car } H(X^{(m)}) = mH(X)$$

  - En augmentant m on s'approche de l'entropie.

Algorithme d'implémentation (1)

- Problème de précision finie du système
  - Nous avons développer dans le cours précédent un algorithme récursif pour la détermination des bornes de l'intervalle du tag:

$$x = (x_1 x_2 ... x_n)$$
$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_X(x_n)$$
$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)}) F_X(x_n - 1)$$

  - Lorsque n augmente l'intervalle du tag devient de plus en plus petit. Avec un système de précision finie à un certain moment on va perdre de vue cet intervalle:

[0.7788995, 0.7788996) → [0.778899, 0.778899)

Algorithme d'implémentation (2)

- Solution:
  - Redimensionnement de l'intervalle du tag: On augmente la taille de l'intervalle mais il faut préserver l'information.
  - Codage incrémental: transmission du code portion par portion au fur et à mesure que l'encodage progresse.
  - On part de l'observation de 3 possibilités concernant l'intervalle lors de l'encodage:
    1. L'intervalle du tag est inclus dans la moitié inférieure: [0,0.5).
    2. L'intervalle du tag est inclus dans la moitié supérieure: [0, 1).
    3. L'intervalle contient le point central 0.5.

Algorithme d'implémentation (3)

- L'intervalle du tag est inclus dans l'une des moitiés :
  - Lorsque l'intervalle se trouve dans l'une des moitiés, il restera dans cette moitié le long de l'encodage de la séquence.
  - Le bit le plus significatif est 0 si l'intervalle se trouve dans [0, 0.5).
  - Le bit le plus significatif est 1 si l'intervalle se trouve dans [0, 1).
  - Ainsi lorsque l'intervalle se trouve dans l'une des moitiés le bit le plus significatif devient connu.
  - L'encodeur peut alors informer le décodeur en envoyant 0 ou 1 de quel moitié il s'agit.

Algorithme d'implémentation (4)

- Redimensionnement:
  - Une fois l'encodeur et le décodeur connaissent la moitié contenant le tag, on peut ignorer l'autre moitié.
  - Pour implémenter l'algorithme sur un système de précision finie on peut redimensionner la moitié en question pour regagner l'intervalle [0, 1) au complet:

$$E_1:[0, 0.5)\rightarrow[0,1); \quad E_1(x)=2x$$

$$E_2:[0.5, 1)\rightarrow[0,1); \quad E_2(x)=2(x-0.5).$$

- Codage incrémental:
  - Mapping → la perte de l'information sur le MSB.
  - Mais ce n'est pas un grave car on a déjà envoyé le MSB. On continue le codage et à chaque fois on se trouve dans une moitié on envoie un bit → codage incrémental.

Génération de tag avec redimensionnement (1)

- Exemple: génération du tag avec redimensionnement de la séquence: 1 3 2 1

$$[Rappel]$$
$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$
$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

  - Considérons la source suivante:

Lettres	a	b	c
Probabilité	0.8	0.02	0.18
$F_X(x)$: $F_X(0)=0$	$F_X(1)=0.8$	$F_X(2)=0.82$	$F_X(3)=1$

  - Observation 1: **1**

$$u^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)})F_X(x_1) = 0 + 1 \times F_X(1) = 0.8$$
$$l^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)})F_X(x_1 - 1) = 0 + 1 \times F_X(0) = 0$$

  - L'intervalle [0, 0.8) ne se trouve ni dans [0,0.5) ni dans [0.5,1) donc on continue sans aucun redimensionnement ni encodage incrémental pour le moment.

Génération de tag avec redimensionnement (2)

$$[Rappel]$$
$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$
$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- Observation 2: 1 **3**

$$u^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)})F_X(x_3) = 0 + 0.8 \times F_X(3) = 0.8 \times 1 = 0.8$$
$$l^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)})F_X(x_3 - 1) = 0 + 0.8 \times F_X(3-1) = 0.8 \times 0.82 = 0.656$$

- L'intervalle [0.656, 0.8) se trouve entièrement dans [0.5,1) donc on envoie le code binaire **1** et on redimensionne l'intervalle.

$$u^{(2)} = 2 \times (0.8 - 0.5) = 0.6$$
$$l^{(2)} = 2 \times (0.656 - 0.5) = 0.312$$

code binaire envoyé: **1**

Génération de tag avec redimensionnement (3)

$$[Rappel]$$
$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$
$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- Observation 1: 1 3 **2**
- Sans redimensionnement on avait:

$$u^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)})F_X(x_2) = 0.656 + 0.144 \times 0.82 = 0.77408$$
$$l^{(3)} = l^{(2)} + (u^{(2)} - l^{(2)})F_X(x_2 - 1) = 0.656 + 0.144 \times 0.8 = 0.7712$$

- Avec redimensionnement.

$$u^{(3)} = 0.312 + (0.6 - 0.312)F_X(x_2) = 0.312 + 0.288 \times 0.82 = 0.54816$$
$$l^{(3)} = 0.312 + (0.6 - 0.312)F_X(x_2 - 1) = 0.312 + 0.288 \times 0.8 = 0.5424$$

- L'intervalle [0.5424, 0.54816) se trouve entièrement dans [0.5,1) donc on envoie le code binaire **1** et on redimensionne l'intervalle.

$$u^{(3)} = 2 \times (0.54816 - 0.5) = 0.09632$$
$$l^{(3)} = 2 \times (0.5424 - 0.5) = 0.0848$$

code binaire envoyé: 1 **1**

Génération de tag avec redimensionnement (4)

- Observation 1: 1 3 **2**
- L'intervalle [0.0848, 0.09632) ∈ [0, 0.5) donc on envoie le code binaire **0** et on redimensionne:

$$u^{(3)} = 2 \times 0.09632 = 0.19264$$
$$l^{(3)} = 2 \times 0.0848 = 0.1696$$

- L'intervalle [0.1696, 0.19264) ∈ [0, 0.5) donc on envoie le code binaire **0** et on redimensionne:

$$u^{(3)} = 2 \times 0.19264 = 0.38528$$
$$l^{(3)} = 2 \times 0.1696 = 0.3392$$

- L'intervalle [0.3392, 0.38528) ∈ [0, 0.5) donc on envoie le code binaire **0** et on redimensionne:

$$u^{(3)} = 2 \times 0.38528 = 0.77056$$
$$l^{(3)} = 2 \times 0.3392 = 0.6784$$

code binaire envoyé: 1 1 **0 0 0**

Génération de tag avec redimensionnement (5)

- Observation 1: 1 3 **2**
- L'intervalle [0.6784, 0.77056) ∈ [0.5, 1) donc on envoie le code binaire **1** et on redimensionne:

$$u^{(3)} = 2\times(0.77056 - 0.5) = 0.54112$$
$$l^{(3)} = 2\times(0.6784 - 0.5) = 0.3568$$

- L'intervalle [0.3568, 0.54112) ne se trouve ni dans [0,0.5) ni dans [0.5,1) donc on continue
- Le bits envoyé suite au redimensionnement et lors de l'encodage incrémental est le même que le bit le plus significatifs (MSB) des bornes de l'intervalle et par conséquent le tag i.e. 0.6784=.1xx.. et 0.77056=.1xx..
- Le redimensionnement est un décalage vers la gauche de la représentation binaire du tag.

code binaire envoyé: 1 1 **0 0 0 1**

Génération de tag avec redimensionnement (6)

- Observation 1: 1 3 2 **1**

$$u^{(4)} = 0.3568 + (0.54112 - 0.3568)F_X(x_1) = 0.3568 + 0.18432 \times 0.8 = 0.504256$$
$$l^{(4)} = 0.3568 + (0.54112 - 0.3568)F_X(x_1 - 1) = 0.3568 + 0.18432 \times 0 = 0.3568$$

- A ce point on est arrivé à la fin de la séquence.
- Donc on envoie le code binaire généré suite au redimensionnement et une valeur du tag se trouvant dans l'intervalle [0.3568, 0.504256).
- Dans ce cas il est commode d'envoyer 0.5=.1 0 0 .. = 1

code binaire envoyé: 1 1 0 0 0 1

Génération de tag avec redimensionnement (7)

- Remarques:
- Avec les opération de redimensionnement et d'encodage incrémental l'intervalle du tag est [0.3568, 0.504256).

$$u^{(4)} = 0.3568 + (0.54112 - 0.3568)F_X(x_1) = 0.3568 + 0.18432 \times 0.8 = 0.504256$$
$$l^{(4)} = 0.3568 + (0.54112 - 0.3568)F_X(x_1 - 1) = 0.3568 + 0.18432 \times 0 = 0.3568$$

- Sans redimensionnement on avait

$$u^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})F_X(x_1) = 0.7712 + 0.00288 \times 0.8 = 0.773504$$
$$l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})F_X(x_1 - 1) = 0.7712 + 0.00288 \times 0 = 0.7712$$

- La largeur de l'intervalle passe de 0.002304 à 0.147456 soit exactement 64 plus large.

Génération de tag avec redimensionnement (8)

- Remarques:
- Avec les opération de redimensionnement et d'encodage incrémental le code envoyé est 1 1 0 0 0 1 pour le codage incrémental et 1 pour le tag.
- $1100011 = 1x2^{-1}+1x2^{-1}+ 1x2^{-6}+1x2^{-7} = 0.7734375$
- Sans redimensionnement on avait

$$u^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})F_X(x_1) = 0.7712 + 0.00288 \times 0.8 = 0.773504$$
$$l^{(4)} = l^{(3)} + (u^{(3)} - l^{(3)})F_X(x_1 - 1) = 0.7712 + 0.00288 \times 0 = 0.7712$$

- On remarque bien que 0.7734375 se trouve bien dans l'intervalle du tag.
- Pour le décodage on peut utiliser cette valeur de tag et suivre la même procédure vu le chapitre précédent.

Décodage incrémental

- Décodage habituel:
  - Utilisation du tag 0.7734375 et imiter l'encodeur dans sa procédure sans redimensionnement ni encodage incrémental.
  - Peut on effectuer un décodage incrémental.
- Décodage incrémental:
  1. Comment commencer le décodage?
  2. Comment continuer le décodage?
  3. Comment arrêter le décodage?
  - On va répondre à ces trois questions à travers les deux exemples suivants.

Exemple: décodage incrémental (1)

[Rappel]

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- Exemple: décodage de 1100011
  - On utilise des mots code de longueur 6.
  - Tout comme l'encodeur, on initialise $u^{(0)}$ à 1 et $l^{(0)}$ à 0.
  - On prend les 6 premiers bits 110001=0.765625.
- T_X=0.767625, intervalle=[0,1): $F_X(1)$=0.8, $F_X(2)$=0.82, $F_X(3)$=1

$$u^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)})F_X(x_n) = 0 + (1-0)F_X(x_n)$$

$$l^{(1)} = l^{(0)} + (u^{(0)} - l^{(0)})F_X(x_n - 1) = 0 + (1-0)F_X(x_n - 1)$$

$$\left.\begin{aligned} u^{(1)} &= F_X(x_n) \\ l^{(1)} &= F_X(x_n - 1) \end{aligned}\right\} \Rightarrow 0.765625 \in [F_X(x_n - 1), F_X(x_n))$$

Et comme pour $n = 1$, on a $[F_X(x_n - 1), F_X(x_n)) = [0, 0.8)$

Le premier élément de la séquence est 1.

Décodage: **1**

Exemple: décodage incrémental (2)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X=0.767625, intervalle=[0,0.8), 1100011

$F_X(1)$=0.8, $F_X(2)$=0.82, $F_X(3)$=1

  - L'intervalle [0,0.8) n'est pas inclus dans l'une des moitié donc on n'effectue pas de redimensionnement.

$$u^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)})F_X(x_n) = 0 + (0.8 - 0)F_X(x_n)$$

$$l^{(2)} = l^{(1)} + (u^{(1)} - l^{(1)})F_X(x_n - 1) = 0 + (0.8 - 0)F_X(x_n - 1)$$

$$0.765625 \in [0.8F_X(x_n - 1), 0.8F_X(x_n))$$

$$0.765625/0.8 = 0.95953125 \in [F_X(x_n - 1), F_X(x_n))$$

$$\Rightarrow n = 3$$

Méthode 2 : Remise en échelle

$$\begin{bmatrix} F_X(1)\times(0.8-0) = 0.64 \\ F_X(2)\times(0.8-0) = 0.656 \Rightarrow T_X = 0.767625 \in [0.656, 0.8) \Rightarrow n = 3 \\ F_X(3)\times(0.8-0) = 0.8 \end{bmatrix}$$

Décodage: 1 **3**

Exemple: décodage incrémental (3)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X=0.767625, intervalle=[0.656,0.8): 1100011

$F_X(1)$=0.8, $F_X(2)$=0.82, $F_X(3)$=1

  - L'intervalle [0.656,0.8) est inclus dans [0.5, 1) donc on effectue le redimensionnement.

$$u^{(2)} = 2 \times (0.8 - 0.5) = 0.6$$

$$l^{(2)} = 2 \times (0.656 - 0.5) = 0.312$$

  - L'encodage incrémental: envoie d'un bit 1.
  - Le décodage incrémental: décalage vers la droite tout en maintenant 6 bits pour le tag
  - **1100011**→1**100011**→**100011**=1x2^{-1}+1x2^{-5}+1x2^{-6}
  - T_X=0.546875

Décodage: 1 **3**

Exemple: décodage incrémental (4)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X=0.546875, intervalle=[0.312,0.6): 1100011

$F_X(1)=0.8$, $F_X(2)=0.82$, $F_X(3)=1$

  - L'intervalle [0.312,0.6) n'est pas inclus dans aucune moitié.

$$u^{(3)} = 0.312 + (0.6 - 0.312)F_X(x_n)$$

$$l^{(3)} = 0.312 + (0.6 - 0.312)F_X(x_n - 1)$$

$$0.546875 \in [0.312 + 0.288F_X(x_n - 1), 0.312 + 0.288F_X(x_n))$$

$$(0.546875 - 0.312)/0.288 = 0.815538194 \in [F_X(x_n - 1), F_X(x_n))$$

$$\Rightarrow n = 2$$

$$u^{(3)} = 0.312 + (0.6 - 0.312) \times 0.82 = 0.54816$$

$$l^{(3)} = 0.312 + (0.6 - 0.312) \times 0.8 = 0.5424$$

Décodage: 1 3 **2**

Exemple: décodage incrémental (5)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X=0.546875, intervalle=[0.5424,0.54816): 1100011

$F_X(1)=0.8$, $F_X(2)=0.82$, $F_X(3)=1$

- L'intervalle [0.5424,0.54816) est inclus dans [0.5, 1) donc on effectue le redimensionnement.

$$u^{(3)} = 2 \times (0.54816 - 0.5) = 0.09632$$

$$l^{(3)} = 2 \times (0.5424 - 0.5) = 0.0848$$

- L'encodage incrémental: envoie d'un bit 1.
- Le décodage incrémental: décalage vers la droite tout en maintenant 6 bits pour le tag
- 1**1000011**→11**000110**→**000110**=1x2^{-4}+1x2^{-5}
- T_X=0.09375

Décodage: 1 3 **2**

Exemple: décodage incrémental (6)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X=0.09375, intervalle=[0.0848,0.09632): 1100011

$F_X(1)=0.8$, $F_X(2)=0.82$, $F_X(3)=1$

- L'intervalle [0.0848,0.09632) est inclus dans [0, 0.5) donc on effectue le redimensionnement.

$$u^{(3)} = 2 \times 0.09632 = 0.19264$$

$$l^{(3)} = 2 \times 0.0848 = 0.1696$$

- L'encodage incrémental: envoie d'un bit 0.
- Le décodage incrémental: décalage vers la droite tout en maintenant 6 bits pour le tag
- 11**000110**→110**001100**→**001100**=1x2^{-3}+1x2^{-4}
- T_X= 0.1875

Décodage: 1 3 **2**

Exemple: décodage incrémental (7)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X= 0.1875, intervalle=[0.1696, 0.19264): 1100011

$F_X(1)=0.8$, $F_X(2)=0.82$, $F_X(3)=1$

  - L'intervalle [0.1696, 0.19264) est inclus dans [0, 0.5) donc on effectue le redimensionnement.

$$u^{(3)} = 2 \times 0.19264 = 0.38528$$

$$l^{(3)} = 2 \times 0.1696 = 0.3392$$

  - L'encodage incrémental: envoie d'un bit 0.
  - Le décodage incrémental: décalage vers la droite tout en maintenant 6 bits pour le tag
  - 110**001100**→1100**011000**→**011000**=1x2^{-2}+1x2^{-3}
  - T_X= 0.375

Décodage: 1 3 **2**

Exemple: décodage incrémental (8)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X= 0.1875, intervalle=[0.3392, 0.38528): 1100011

$$u^{(3)} = 2 \times 0.38528 = 0.77056$$

$$l^{(3)} = 2 \times 0.3392 = 0.6784$$

- L'encodage incrémental: envoie d'un bit 0.
- Le décodage incrémental: décalage vers la droite:
- 1100**011000**→11000**110000**→**110000**=1x2^{-1}+1x2^{-2}

$$u^{(3)} = 2 \times (0.77056 - 0.5) = 0.54112$$

$$l^{(3)} = 2 \times (0.6784 - 0.5) = 0.3568$$

- L'encodage incrémental: envoie d'un bit 1.
- Le décodage incrémental: décalage vers la droite:
- 11000**110000**→110001**100000**→**100000**=1x2^{-1}
- T_X=0.5

Décodage: 1 3 **2**

Exemple: décodage incrémental (9)

$[Rappel]$

$$u^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n)$$

$$l^{(n)} = l^{(n-1)} + (u^{(n-1)} - l^{(n-1)})F_X(x_n - 1)$$

- T_X= 0.5, intervalle=[0.3568,0.54112): 1100011

$F_X(1)=0.8$, $F_X(2)=0.82$, $F_X(3)=1$

$$u^{(3)} = 2 \times (0.77056 - 0.5) = 0.54112$$

$$l^{(3)} = 2 \times (0.6784 - 0.5) = 0.3568$$

$$u^{(3)} = 0.3568 + (0.54112 - 0.3568)F_X(x_n)$$

$$l^{(3)} = 0.3568 + (0.54112 - 0.3568)F_X(x_n - 1)$$

$$0.5 \in [0.3568 + 0.18432F_X(x_n - 1), 0.3568 + 0.18432F_X(x_n))$$

$$(0.5 - 0.3568)/0.18432 = 0.77690972 \in [F_X(x_n - 1), F_X(x_n))$$

$$\Rightarrow n = 1$$

Décodage: 1 3 2 **1**