Algorithmique Avancée

Master Big Data & Internet des Objets
(BDIO)

Chaînes de Caractères : compression de données :: Codage de Huffman

Animé par : Dr. ibrahim GUELZIM
Email : ib.guelzim@gmail.com

Sommaire

- Rappels
  - Introduction et notions générales
  - Analyse et conception d'algorithmes
  - Complexité d'algorithmes classiques : 3 Tris de tableaux, 2 recherches dans un tableau, Schéma de Hörner
  - Preuves d'algorithmes
- Autres algorithmes de tri :
  - Tri par fusion
  - Tri par Tas
- Complexité moyenne :
  - Application au Tri rapide
  - Structures de Données Probabilistes :
    - Notions sur les Tables de Hachage et Fonctions de Hachage,
    - Bloom Filter,
    - Count Min Sketch
- Programmation dynamique
- Traitements de chaines de Caractères :
  - Recherche de chaine de caractères
  - **Compression de données**

Codage de l'information

- Codage de texte :
  - Code ASCII : { A-Z ou a-z } sur 8 bits

  Exemple 1:
    - TXT = "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"
    - 35 caractères
    - Codage : 35 * 8 = 280 bits

  - Amélioration 1:
    - Alphabet restreint Σ : { A, C, D, E, T }
      - $|\Sigma| = 5$
      - $2^2 < 5 \leq 2^3$

      → 3 bits pour coder
      - Longueur FIXE pour chaque code

    Exemple 1: Codage :: 35 * 3 = 105 bits

    - Codage à Longueur Fixe

Codage de l'information

- Amélioration 2:

- Astuce : " **Coder le symbole le plus fréquent sur un nombre minimal de bits** "

Exemple 1:

- TXT = "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"
- Fréquences d'apparitions des lettres :
  - A : 15
  - E : 9
  - C : 6
  - D : 4
  - T : 1
- Comment coder ?
- Exemple : !
  - A sur 1 bit Ex : "0"
  - E sur 2 bits Ex : "01"
  - C sur 3 bits Ex : "001"
  - D sur 4 bits Ex : "0101"
  - T sur 5 bits Ex : "00100"
- TxtCode = "00100010100001010100001010100000100100100..."

Codage de l'information

– Amélioration 2

Définition Préfixe :

- Soit ∑ un alphabet fini = ensemble fini de symboles
  - Exemples d'alphabet : ∑1 = {0, 1}, ∑2 = {a, b, c,…, z}

- On appelle Σ^* ("sigma-étoile") l'ensemble de toutes les chaînes (**mots**) de longueur finie utilisant les symboles de l'alphabet Σ. (Ex: "01001", abca")

- La chaîne (mot) **vide** de longueur **zéro**, notée **ε** appartient aussi à Σ^*

- La longueur de la chaîne x (notée |x|) est le nombre de symboles dans x.

- La concaténation de deux mots x et y est notée xy (ou x.y ou x + y) et |xy| = |x| + |y|

- Une chaîne w est un **préfixe** d'une chaîne x, si $\exists\ y \in \Sigma^*$ / x = wy

→ Une chaine x est préfixe d'elle même (x = xε)

Codage de l'information

- Amélioration 2:
  - Idée :

  " **Coder le symbole le plus fréquent sur un nombre minimal de bits** "
    - Rappel code : {A : "0", E : "01", C : "001", D : "0101", T : "00100"}
  - Décoder:
    (TxtCode="00100010100001010100001010100000100100100…")
    - 001 : "0"."01" → "AE"

      OU : "001" → "C"
    - 0101 : "01"."01" → "EE"

      OU : "0101" → "D"

    →PK Ambiguité ? (plusieurs façons d'interpréter la même donnée)

    →un symbole est un préfixe d'un autre

    → E est préfixe de D : "D" = "EE"

    → C est préfixe de T : "T" = "CAA"

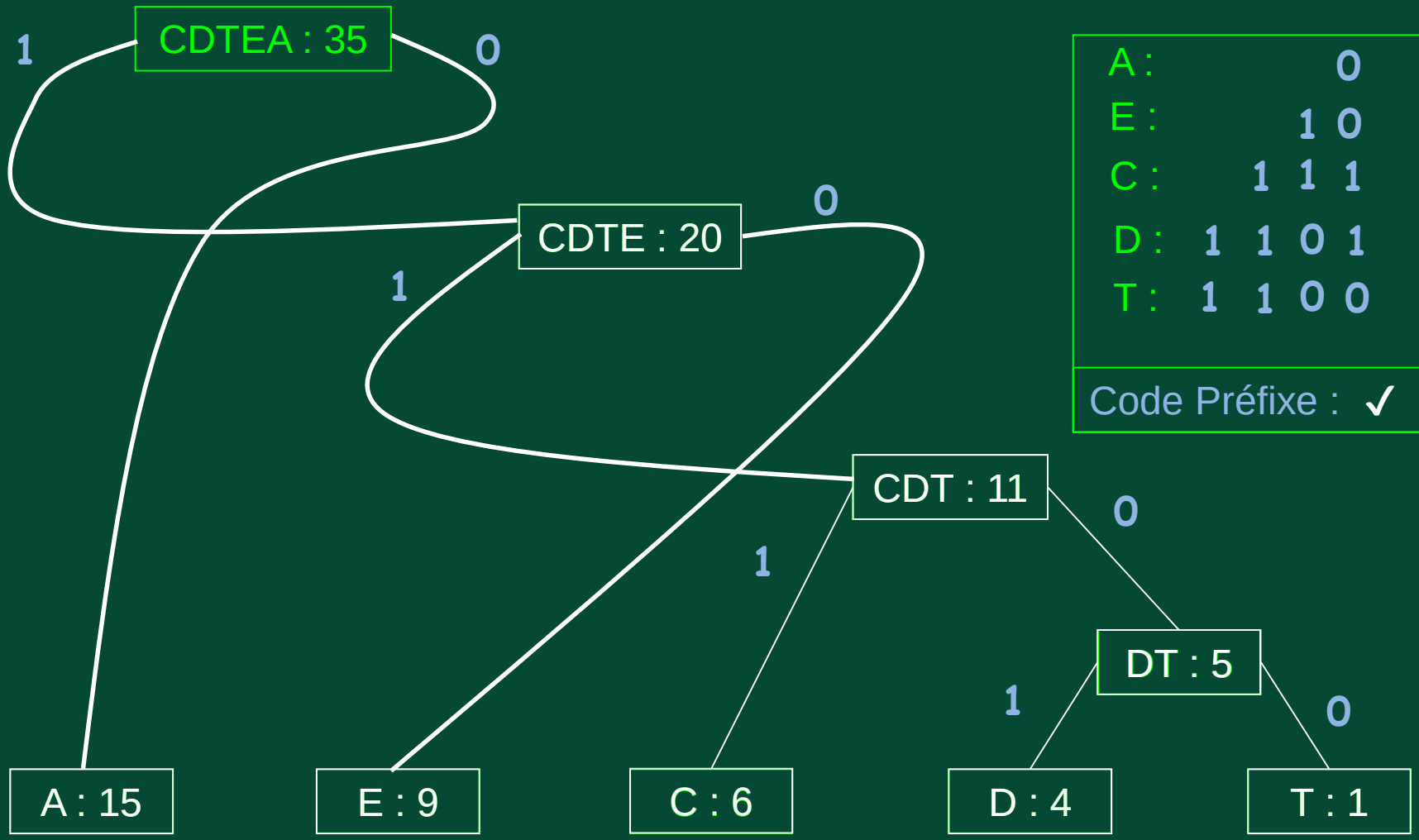
Solution pour éviter l'ambiguité :

- Proposer un codage où aucun symbole n'en soit un préfixe d'un autre.
- Codage Préfixe → Codage de Huffman

Codage de Huffman

- Application : Exemple 1

TXT = "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)



A :	0
E :	1 0
C :	1 1 1
D :	1 1 0 1
T :	1 1 0 0

Code Préfixe : ✓

Codage de Huffman

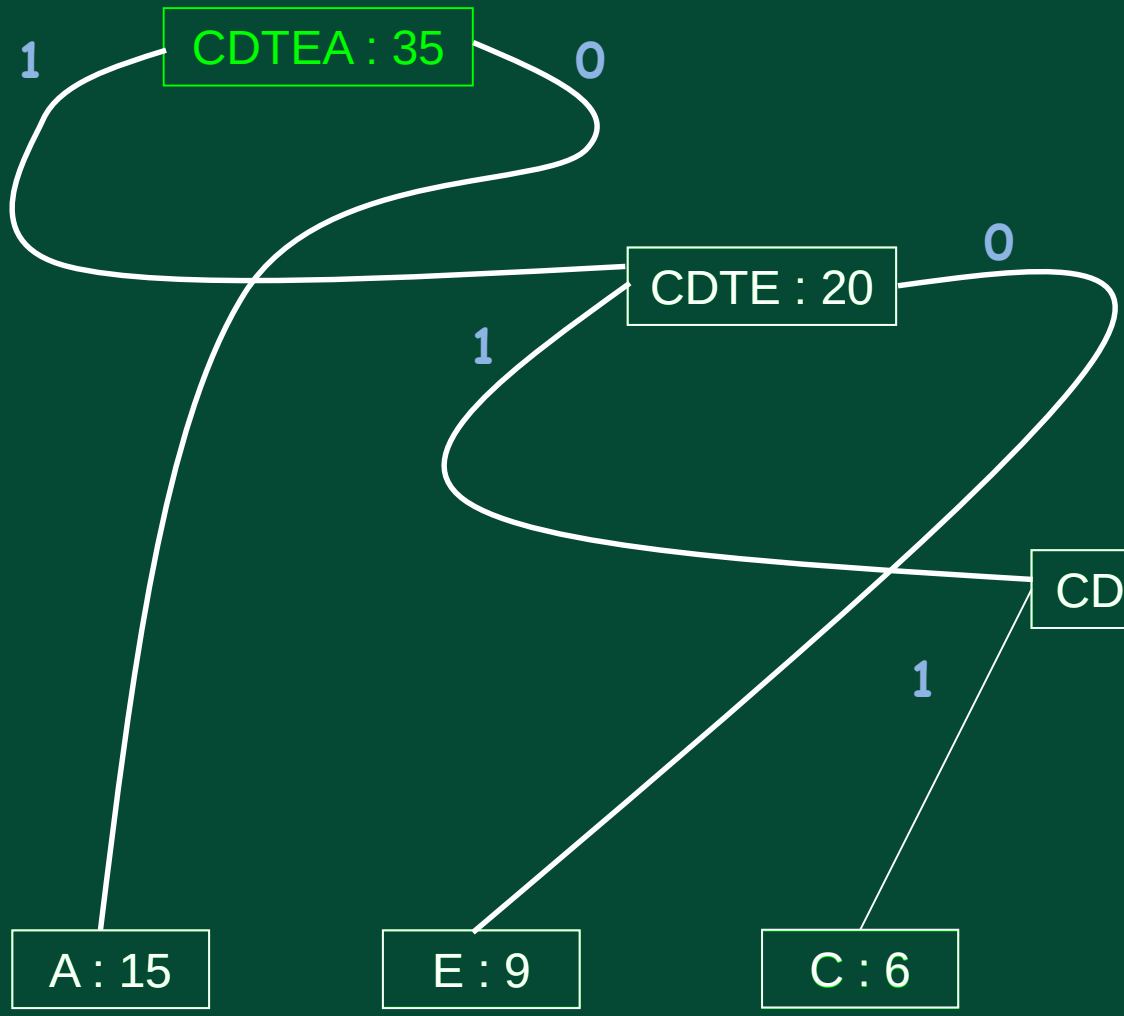
- Comment coder texte de longueur **NC** ?
  - Créer un tableau de symboles TS[] contenant les symboles alphanumériques ainsi que leurs fréquences d'apparition.
  - Trier TS[] par ordre décroissant des fréquences d'apparition
  - Un nœud est orphelin s'il ne possède pas de père.
  - Construire l'arbre binaire de Huffman de la façon suivante:
    1. Les symboles du TS trié dans un ordre décroissant constituent les feuilles de l'arbre (de gauche à droite)
    2. Créer un père des deux nœuds orphelins ayant les plus petites fréquences :
       i. Donner au nœud père une fréquence égale à la somme des fréquences de ses deux fils.
       ii. La nouvelle valeur du symbole est la concaténation du fils droit au fils gauche { Remarque : Fq(FG) ≥ Fq(FD) }
       iii. Affecter à la branche reliant le père au fils gauche le symbole binaire "1" et au fils droit "0"
    3. Si la fréquence du père est égale à NC → **Arrêt**

       Sinon, aller à l'étape 2
  - Le symbole binaire de chaque caractère est construit en écrivant de droite à gauche la valeur binaire du chemin allant de la feuille vers la racine.

Codage de Huffman

- Application : Exemple 1

TXT = "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)



A :	0
E :	1 0
C :	1 1 1
D :	1 1 0 1
T :	1 1 0 0

Code Préfixe : ✓

Codage de Huffman :: Pseudo Code 1 / 11

- Comment ?
  I. Lire le texte à coder et construire l'alphabet relative
  II. Calculer les fréquences d'apparitions des lettres (symboles)
  III. Affecter un code binaire (succession de 0,1) à chaque symbole
  IV. Vérifier la justesse (En : correctness) de l'algorithme
    - Coder un texte alphanumérique en binaire
    - Décoder le texte binaire en texte alphanumérique

- Outils :
  Ranger les symboles dans un tableau de sorte que chaque élément contienne:
    - Symbole alphanumérique
    - Fréquence du symbole
    - Symbole binaire

  → Sol : Structure

Codage de Huffman :: Pseudo Code 2 / 11

- Construire la structure utilisée :
  - Structure Symb

    Début

      AS : Chaine de caracteres # Alphanumeric String

      Fq : Entier # Frequence d appartition dans le texte

      BS : Chaine de caracteres # Binary String

    Fin
  - Pour utiliser un tableau de structure :

    Variable TS[] : Structure Symb

AS:	Fq:
BS :	

AS:	Fq:	AS:	Fq:	AS:	Fq:	AS:	Fq:	AS:	Fq:
BS :		BS :		BS :		BS :		BS :	

- Procédé (coder le contenu d'un fichier) :
  - Mettre le fichier dans une variable considérée comme chaine
  - Parcourir le fichier lettre par lettre pour construire l'alphabet
    - Si la lettre existe dans l'alphabet, incrémenter sa fréquence par 1
    - Sinon, l'ajouter à l'alphabet en lui affectant la fréquence 1
    - Réalisation : Ecrire une fct qui réalise les points ci-dessus

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)




FinPour
```

Récapitulatif du corps du Pseudo code

Codage de Huffman :: Pseudo Code 3 / 11

- écrire une fct qui
  - renvoie -1 si le caractère à chercher n'appartient pas au tableau de symboles de l'alphabet
  - sinon renvoie la position du caractère dans le tableau,

```
Fonction SymbInTS(car : caractere, TS[] : structure Symb)
  Début
    Pour i de 0 à Len(TS)-1 Faire # Len(TS) : longueur du tableau TS (analogie / chaine)
                  Si car = TS[i].AS alors
                              retourner i
                  FinSi
    FinPour
    retourner -1          # cas où si car ne fait pas partie de TS[]
  Fin
```

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])


        FinSi
FinPour
```

Récapitulatif du corps du Pseudo code

Codage de Huffman :: Pseudo Code 4 / 11

- Écrire une fonction qui **construit** (crée et initialise) un symbole

```
Fonction ConstrSymb (C : caractère) : Structure Symb
   variable Sb : Structure Symb
   Début
     Sb.AS ← C
     Sb.Fq  ← 1
   Fin
```

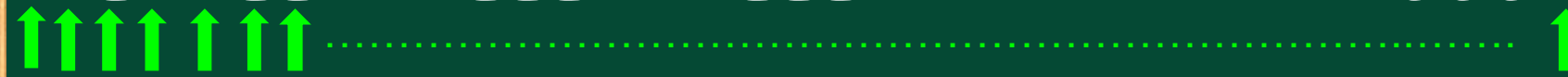
```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors           # élément NON TROUVÉ
           TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon                       # élément existant dans l'alphabet
           TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
```

Récapitulatif du corps du Pseudo code

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"



- TS [ ] (Tableau vide au départ : de Longueur 0)

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 3	AS: T	Fq: 1
BS :		BS :		BS :		BS :		BS :	

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon
            TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
CopTS ← TriDec(TS), NbLetAlph ← Len(TS) , NbSymbPart ← NbLetAlph - 1   # initialisation
```

Récapitulatif du corps du Pseudo code

Codage de Huffman :: Pseudo Code 5 / 11

- Remarques:
  - Trier TS[] par ordre décroissant / Fréquence des symboles

```
Variable NbLetAlph : Entier
NbLetAlph ← Len(TS)
```

  - Créer une copie de TS[] pour construire l'arbre de Huffman :

```
Variable CopTS [] : structure Symb
CopTS ← TS          # on suppose que cette affectation est permise
```

Codage de Huffman : Construction Tableau Symboles

- Exemple 1

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS :		BS :	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS :		BS :	

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon
            TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
CopTS ← TriDec(TS), NbLetAlph ← Len(TS) , NbSymbPart ← NbLetAlph – 1   # initialisation
Pour i de 0 à NbLetAlph-2 Faire
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart ].AS , "0")          # Fils droit
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart – 1 ].AS , "1")      # Fils gauche









FinPour
```

Récapitulatif du corps du Pseudo code

Codage de Huffman :: Pseudo Code 6 / 11

```
Fonction AjSymbBinChSymb( TS[] : Structure Symb, ch : chaine de
                                  caractère, SB : caractère)
# Ajoute Le Symbole Binaire SB aux éléments de la chaine ch appartenant au tableau TS.
    Début
        Lg ← Len( ch )
        Pour i de 0 à  Lg-1 Faire
            pos ← SymbInTS( ch[i], TS )      # On suppose que ts les car de ch ∈ l'alphabet
            TS[ pos ].BS = SB + TS[ pos ].BS
        FinPour
    Fin
```

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS : 1		BS : 0	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS :		BS :	

```
AjSymbBinChSymb(TS, CopTS[ NbSymbPart ].AS , "0")        # Fils droit
AjSymbBinChSymb(TS, CopTS[ NbSymbPart - 1 ].AS , "1")    # Fils gauche
```

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon
            TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
CopTS ← TriDec(TS), NbLetAlph ← Len(TS) , NbSymbPart ← NbLetAlph – 1   # initialisation
Pour i de 0 à NbLetAlph-2 Faire
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart ].AS , "0")          # Fils droit
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart – 1 ].AS , "1")      # Fils gauche

   CopTS[ NbSymbPart – 1 ].AS ← CopTS[ NbSymbPart – 1 ].AS + CopTS[ NbSymbPart ].AS
   CopTS[ NbSymbPart – 1 ].Fq  ← CopTS[ NbSymbPart - 1 ].Fq  + CopTS[ NbSymbPart ].Fq



FinPour
```

Récapitulatif du corps du Pseudo code

Codage de Huffman :: Pseudo Code 7 / 11

- **Remarques et commentaires:**

 # Concaténation du champ AS en utilisant l'operateur +

CopTS[NbSymbPart – 1].AS ← CopTS[NbSymbPart – 1].AS + CopTS[NbSymbPart].AS

 # Incrémentation du champ Fq en utilisant l'operateur +

CopTS[NbSymbPart – 1].Fq ← CopTS[NbSymbPart - 1].Fq + CopTS[NbSymbPart].Fq

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS : 1		BS : 0	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS :		BS :	

CopTS[NbSymbPart - 1].AS ← CopTS[NbSymbPart - 1].AS + CopTS[NbSymbPart].AS
CopTS[NbSymbPart - 1].Fq ← CopTS[NbSymbPart - 1].Fq + CopTS[NbSymbPart].Fq

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS : 1		BS : 0	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: DT	Fq: 5	AS: T	Fq: 1
BS :		BS :		BS :		BS :		BS :	

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon
            TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
CopTS ← TriDec(TS), NbLetAlph ← Len(TS) , NbSymbPart ← NbLetAlph – 1   # initialisation
Pour i de 0 à NbLetAlph-2 Faire
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart ].AS , "0")          # Fils droit
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart – 1 ].AS , "1")      # Fils gauche

   CopTS[ NbSymbPart – 1 ].AS ← CopTS[ NbSymbPart – 1 ].AS + CopTS[ NbSymbPart ].AS
   CopTS[ NbSymbPart – 1 ].Fq  ← CopTS[ NbSymbPart - 1 ].Fq  + CopTS[ NbSymbPart ].Fq

   NbSymbPart ← NbSymbPart - 1
   InsererElem( CopTS , NbSymbPart )  # Repositionnement de l'élément à la position NbSymbPart
FinPour
```

Récapitulatif du corps du Pseudo code

Codage de Huffman : Construction Tableau Symboles

- Exemple 1

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS : 1		BS : 0	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: DT	Fq: 5
BS :		BS :		BS :		BS :	

Ajouter "0" sur les symboles de la chaine représentant le dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple 1

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS :		BS : 01		BS : 00	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: DT	Fq: 5
BS :		BS :		BS :		BS :	

Ajouter "0" sur les symboles de la chaine représentant le dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS : 1		BS : 01		BS : 00	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: DT	Fq: 5
BS :		BS :		BS :		BS :	

Ajouter "1" sur les symboles de la chaine représentant l'avant dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS : 1		BS : 01		BS : 00	

- CopTS[] # copie Non triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: E	Fq: 9	AS: CDT	Fq: 11
BS :		BS :		BS :	

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon
            TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
CopTS ← TriDec(TS), NbLetAlph ← Len(TS) , NbSymbPart ← NbLetAlph – 1   # initialisation
Pour i de 0 à NbLetAlph-2 Faire
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart ].AS , "0")              # Fils droit
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart – 1 ].AS , "1")          # Fils gauche

   CopTS[ NbSymbPart – 1 ].AS ← CopTS[ NbSymbPart – 1 ].AS + CopTS[ NbSymbPart ].AS
   CopTS[ NbSymbPart – 1 ].Fq  ← CopTS[ NbSymbPart - 1 ].Fq  + CopTS[ NbSymbPart ].Fq

   NbSymbPart ← NbSymbPart – 1
   InsererElem( CopTS , NbSymbPart )  # Repositionnement de l'élément à la position NbSymbPart
FinPour
```

Récapitulatif du corps du Pseudo code

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS : 1		BS : 01		BS : 00	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: CDT	Fq: 11	AS: E	Fq: 9
BS :		BS :		BS :	

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS :		BS : 1		BS : 01		BS : 00	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: CDT	Fq: 11	AS: E	Fq: 9
BS :		BS :		BS :	

Ajouter "0" sur les symboles de la chaine représentant le dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS : 0		BS : 1		BS : 01		BS : 00	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: CDT	Fq: 11	AS: E	Fq: 9
BS :		BS :		BS :	

Ajouter "0" sur les symboles de la chaine représentant le dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS : 0		BS : 1		BS : 01		BS : 00	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: CDT	Fq: 11	AS: E	Fq: 9
BS :		BS :		BS :	

Ajouter "1" sur les symboles de la chaine représentant l'avant dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS : 0		BS : 11		BS : 101		BS : 100	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: CDT	Fq: 11	AS: E	Fq: 9
BS :		BS :		BS :	

Ajouter "1" sur les symboles de la chaine représentant l'avant dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS : 0		BS : 11		BS : 101		BS : 100	

- CopTS[] # copie Non triée dans ordre décroissant de TS

AS: A	Fq: 15	AS: CDTE	Fq: 20
BS :		BS :	

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS :		BS : 0		BS : 11		BS : 101		BS : 100	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: CDTE	Fq: 20	AS: A	Fq: 15
BS :		BS :	

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS : 0		BS : 0		BS : 11		BS : 101		BS : 100	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: CDTE	Fq: 20	AS: A	Fq: 15
BS :		BS :	

Ajouter "0" sur les symboles de la chaine représentant le dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS : 0		BS : 10		BS : 111		BS : 1101		BS : 1100	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: CDTE	Fq: 20	AS: A	Fq: 15
BS :		BS :	

Ajouter "1" sur les symboles de la chaine représentant l'avant dernier élément de CopTS

Codage de Huffman : Construction Tableau Symboles

- Exemple

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD"

AS: A	Fq: 15	AS: E	Fq: 9	AS: C	Fq: 6	AS: D	Fq: 4	AS: T	Fq: 1
BS : 0		BS : 10		BS : 111		BS : 1101		BS : 1100	

- CopTS[] # copie triée dans ordre décroissant de TS

AS: CDTEA	Fq: 35
BS :	

Fin Partie 2

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon
            TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
CopTS ← TriDec(TS), NbLetAlph ← Len(TS) , NbSymbPart ← NbLetAlph – 1   # initialisation
Pour i de 0 à NbLetAlph-2 Faire
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart ].AS , "0")          # Fils droit
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart – 1 ].AS , "1")      # Fils gauche

   CopTS[ NbSymbPart – 1 ].AS ← CopTS[ NbSymbPart – 1 ].AS + CopTS[ NbSymbPart ].AS
   CopTS[ NbSymbPart – 1 ].Fq  ← CopTS[ NbSymbPart - 1 ].Fq  + CopTS[ NbSymbPart ].Fq

   NbSymbPart ← NbSymbPart - 1
   InsererElem( CopTS , NbSymbPart )  # Repositionnement de l'élément à la position NbSymbPart
FinPour
TxtCode ← CodeTxt(TXT , TS)
Ecrire ("Text Codé par Huffman : " , TxtCode)
```

Récapitulatif du corps du Pseudo code

Codage de Huffman :: Pseudo Code 8 / 11

- Écrire une fonction pour coder le texte alphanumérique en un code binaire
- À cette étape, l'alphabet est construit ainsi que les symboles binaires.

```
Fonction CodeTxt(Txt [] : caractere, TS[] : structure Symb) : Chaine de caractères
    Variables  codeHuff : chaine de caractères
               i , pos : Entier
   Début
        codeHuff ← ""
        Pour i de 0 à  Len(Txt)-1 Faire
          pos ← SymbInTS(Txt[ i ], TS)
          codeHuff ← codeHuff + TS[ pos ].BS
        FinPour
        return codeHuff
    Fin
```

Codage de Huffman :: Pseudo Code 9 / 11

```
# Illustration:
Ecrire ("Text Codé par Huffman : ")
TxtCode ← CodeTxt(Text, TS)
Ecrire( "TxtCodé = ", TxtCode )

TxtDecode ← DecodeBin(TxtCode , TS)
Ecrire ("Text Décodé : ")
Ecrire (TxtDecode)
```

- Expl 1:
  TXT = "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

TxtCodé =
"01000101000010101000010101000011111111100011111111111011101110111001101"

```
TXT ← LireToutFichier(NomFich)  # re-écrire la function si besoin
Pour i de  0 à Len(TXT) Faire  # construction du Tableau des Symboles TS
        pos ← SymbInTS(TXT[ i ], TS)
        Si pos = -1 alors
            TS[ Len(TS) ] ← ConstrSymb(TXT[ i ])
        Sinon
            TS[ pos ].Fq ←  TS[ pos ].Fq + 1
        FinSi
FinPour
CopTS ← TriDec(TS), NbLetAlph ← Len(TS) , NbSymbPart ← NbLetAlph – 1   # initialisation
Pour i de 0 à NbLetAlph-2 Faire
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart ].AS , "0")          # Fils droit
    AjSymbBinChSymb(TS, CopTS[ NbSymbPart – 1 ].AS , "1")      # Fils gauche

   CopTS[ NbSymbPart – 1 ].AS ← CopTS[ NbSymbPart – 1 ].AS + CopTS[ NbSymbPart ].AS
   CopTS[ NbSymbPart – 1 ].Fq  ← CopTS[ NbSymbPart - 1 ].Fq  + CopTS[ NbSymbPart ].Fq

   NbSymbPart ← NbSymbPart - 1
   InsererElem( CopTS , NbSymbPart )  # Repositionnement de l'élément à la position NbSymbPart
FinPour
TxtCode ← CodeTxt(TXT , TS)
Ecrire ("Text Codé par Huffman : " , TxtCode)
TxtDecode ← DecodeBin(TxtCode , TS)
Ecrire ("Text Décodé : " , TxtDecode)
```

Récapitulatif du corps du Pseudo code

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

✓

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"A"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"A"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé = ✓

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"A"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AE"

. . . . . .

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAA"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAA"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé = ✓

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAA"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAAC"

. . . . . .

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAACCCAAACCC"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé = ✓

"01000101000010101000010101010000111111111000111111111101101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAACCCAAACCC"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"01000101000010101000010101010000111111111000111111111101101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCD"

. . . . . .

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé = ✓

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDD"

Codage de Huffman :: Décoder

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"

- TxtCodé =

"0100010100001010100001010100001111111110001111111111011101110111001101"

- TxtDecodé =

"AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" ✓

Codage de Huffman :: Pseudo Code 10 / 11

- Écrire une fonction qui décode le texte binaire en alphanumérique
- À cette étape, l'alphabet est construit ainsi que les symboles binaires.

- Pour ce, écrire une fct qui vérifie si une chaine binaire BinCH représente un symbole de l'alphabet (rangé dans le tableau TS)

```
Fonction BSinTS (BinCH [ ] : caractère, TS[ ] : structure Symb): Entier
  Variable i : Entier
  Début
   Pour i de  0 à Len(TS)  Faire
        Si BinCH = TS[i].BS
          retourner ( i )
        FinSi
   FinPour
   retourner ( -1 )
  Fin
```

Codage de Huffman :: Pseudo Code 11 / 11

```
Fonction DecodeBin(MotBin[] : caractere, TS[] : structure Symb) : chaine de
                                                                  caracteres
Variables TxtDecode[ ]    : caractere
          i, j, d         : Entier
Début
  TxtDecode = ""
  i ← 0, d ← 1
  TantQue  (i < Len(MotBin))  Faire
       j ← i + d
       pos ← BSinTS(MotBin[i : j], TS) # MotBin[i : j] est la chaine de la position i à j-1
       Si pos = -1          # Symbole MotBin[i : j] NON trouvé
              d ← d + 1
       Sinon                # Symbole MotBin[i : j] trouvé
             TxtDecode ← TxtDecode + TS[pos].AS
              i ← i + d
              d ← 1
  FinTantQue
  retourner TxtDecode
Fin
```

Codage de Huffman :: Gain

- Expl 1: TXT= "AEAAEEAAAEEEAAAEEEAAACCCAAACCCDDDTD" (35 car)

Lettre	A	E	C	D	T
Fq	15	9	6	4	1
Binaire	"0"	"10"	"111"	"1101"	"1100"
Long Bin	1	2	3	4	4

- Longueur du message codé :
  15x1 + 9x2 + 6x3 + 4x4 + 1x4 = 71
- TxtCodé =

"0100010100001010100001010100001111111110001111111111101110111011001101"

- Longueur msg par codage ASCII (8 bit) : 35x8 = 280
- Evaluation de la compression :
  - *Quotient de compression : Q = (Taille Initiale)/(Taille Finale) = TI / TF*
    Application Exemple 1 : Q1 = 280 / 71
  - Taux de compression : T = 1 / Q
    App : T1 = 1 / Q1 ~ 0,25
  - Gain de compression : G = 1 – T = (TI - TF) / TI
    *App : G1 = 1 - T1 ~ 0,75 (**75 %**)*

Codage de Huffman :: Complexité

- Construction d'un tableau de K symbole (alphabet, muni des fréquences)
  - Après lecture du texte ou fichier de N caractères
  - Pour chaque lettre du fichier (N fois):
    - Insertion dans un tas (Log K) ou
    - Parcours du tas pour màj (Log K)
  - D'où : Complexité de la construction de l'alphabet est N Log(K)

- Pour la construction des Symboles Binaires (SB) :
  - Parcours de l'alphabet de longueur K
  - Pour K-2 itérations :
    - Suppression d'un élément du Tas (Log K)
    - Réinsertion du nœud de fusion (père) (Lok K)
  - D'où : Complexité de la construction des SB est K Log(K)
- Par conséquent la complexité du codage de Huffman est :
  - N Log(K) + K Log(K) (souvent N >> K) d'où

→ T(Huffman) = N Log (K)

FIN