La compression consiste à éviter les répétitions, pour économiser de la place.

Un dictionnaire contenant toutes les répétitions est créé lors des deux opérations. Il doit être construit de la même manière, à la compression et à la décompression, et contenir les mêmes informations.

Tous les ensembles de lettres qui sont lus sont placés dans le dictionnaire, et son numérotés. A chaque fois qu'un ensemble est lu, on regarde s'il en existe déjà un qui est identique. Si c'est le cas, on émet son numéro vers le fichier compressé. Sinon, on le rajoute à la fin du dictionnaire, et on écrit chacune des lettres dans le fichier compressé.

consiste à éviter les répétitions, de caractères ou de chaînes de caractères, pour économiser de la place.

L’algorithme de compression LZW construit une table de correspondance entre un code de taille fixée et un caractère lu, à partir du texte à compresser. La table est initialisée avec tous les caractères du code ASCII (pour notre projet).

Le déroulement de la compression est le suivant : Tous les ensembles de lettres qui sont lus sont placés dans le dictionnaire, et sont numérotés. A chaque fois qu'un nouvel ensemble est lu, on regarde dans le dictionnaire s'il existe déjà: si c'est le cas, on émet son numéro vers le fichier compressé, sinon on le rajoute à la fin du dictionnaire, et on écrit chacune des lettres dans le fichier compressé.

l'algorithme commence par lire le premier caractère de la chaîne (qui est dans la table ASCII car elle censée contenir tous les caractères) et renvoie le code correspondant.

En parallèle, le caractère va être '' retenu en mémoire '' pour former le premier caractère d'une nouvelle chaîne, qui servira à compléter le dictionnaire. Elle aura, alors, le code 256 (car la table ASCII contient 255 caractères).

L'algorithme lit ensuite le deuxième caractère (lui aussi se trouve dans la table ASCII) et renvoie son code. Cette fois-ci, le caractère servira à deux choses ; il complètera la chaîne de caractères qui commence par le caractère précédent et sera le premier caractère de la chaîne suivante (celle qui a le code 257).

En procédant de cette manière pour tout le texte à compresser, nous obtenons alors un code à la place d'une chaîne de caractères.

Au début de la compression, le dictionnaire est initialisé, et contient les 256 codes ASCII. Ainsi, l’algorithme ajoutera ses propres séquences à partir de l’adresse de la « case » 256.

Pour déceler les redondances de la séquence, le programme lit séquentiellement chaque octet un par un, et le place dans une mémoire tampon. Cet algorithme de compression fonctionne de manière itérative. Au départ, le programme met un caractère dans le buffer puis, si cette valeur est déjà contenue dans le dictionnaire, le programme met deux caractères dans le buffer. Ainsi, à chaque fois que le buffer contient une valeur connue, l’algorithme rajoute un caractère (exemple : étapes 19, 22, 24).. Dès que la valeur n’est pas déjà compactée dans le dictionnaire, le programme exécute plusieurs tâches : il ajoute premièrement cette valeur à la fin du tableau, il recopie dans le fichier l’adresse de la dernière séquence connue, et enfin, il recharge la mémoire tampon avec deux caractères. Il est important de noter qu’entre chaque itération, le programme n’écrit rien, d’où le gain de place.

Pour économiser de la place, la compression consiste à éviter les répétitions:  un *dictionnaire*  contenant toutes les répétitions est créé lors des deux opérations: compression et décompression. Il doit être construit de la même manière et contenir les mêmes informations.

Tous les ensembles de lettres qui sont lus sont placés dans le dictionnaire, et sont numérotés. A chaque fois qu'un nouvel ensemble est lu, on regarde dans le dictionnaire s'il existe déjà: si c'est le cas, on émet son numéro vers le fichier compressé, sinon on le rajoute à la fin du dictionnaire, et on écrit chacune des lettres dans le fichier compressé.

Les numéros des chaînes du dictionnaire doivent dépasser 255, sinon il y aurait un risque de confusion avec les caractères normaux dont le code *Ascii*  est inférieur ou égal à 255. Ici, nous choisirons des numéros supérieurs ou égaux à 256 (variable *numero* ) . Pour stocker ces nombres, les 8 bits qui constituent un octet ne suffiront sans doute pas. Il faudra alors écrire des nombres stockés sur 9, 10, 11,... bits. De façon générale, si les nombres utilisés pour le dictionnaire sont codés sur *n*  bits, et que ces *n*  bits ne suffisent plus, il faudra passer à un codage sur *n* +1 bits (variable *nb\_bits* ).   
Pour que le décompresseur soit prévenu de ce changement, on émet un code de contrôle ( ici on a choisi le code 0 ) vers le fichier compressé.