TP 36

Algorithme de Lempel-Ziv-Welsh
Décompression

Exercice 36.1 Écrire une fonction `input_code` telle que l'appel `input_code f d` va lire d bits dans le fichier f pour construire un code de longueur d.

```
input_code : in_channel_bits -> int -> int
```

I Décompression pour les petits

Il est très facile de décompresser un texte si la table construite lors de la phase de compression a été transmise en même temps que le texte compressé…

Exercice 36.2 Avec la table ci-dessous :

…	…	256	257	258
…	…	AB	BA	ABA

décompresser le message suivant :

65 66 256 258.

Exercice 36.3 Écrirer une fonction `lzw_decomp_naif` telle que l'appel `lzw_decomp_naif table d entree sortie` permet de décompresser le fichier `entree` vers le fichier `sortie` connaissant la table des suffixes et la longueur du code d.

```
lzw_decomp_naif : string array -> int -> string -> string -> unit
```

II Décompression pour les grands

Il est en réalité inutile de transmettre la table des suffixes avec le fichier compressé pour être en mesure de le décompresser. On peut le reconstruire au fur et à mesure de la décompression en déterminant les opérations de compression qui ont nécessairement conduit à l'émission des codes lus. On initialise la table avec les valeurs des 256 octets, puis l'on traite le premier code lu (qui est nécessairement inférieur à 256) en émettant cet octet. À partir du second code lu :

- on lit un code n avec $t[n] = xm'$ (avec x un octet et m' une séquence) ;
- on émet xm' ;
- on ajoute à la table mx, où c est le précédent code lu et $t[c] = m$.

Exemple. En entrée, le fichier contient les codes suivants : 65 65 66 257 256 257.

- On initialise le tableau pour les 256 premières cases, on lit le premier code (65) et on emet le caractère $t[65]$, *i.e.* `A`. La variable m prend la valeur `"A"` ;
- on lit ensuite le code suivant (65). Avec les notations introduites ci-dessus, on a alors : $x = \texttt{'A'}$ et $m' = \texttt{""}$. On emet `A` et $t[256]$ prend la valeur `"AA"` et m la valeur `A` ;
- on lit le code suivant (66). On a alors : $x = \texttt{'B'}$ et $m' = \texttt{""}$. On emet `B` et $t[257]$ prend la valeur `"AB"` et m la valeur `B` ;
- on lit le code suivant (257). On a alors : $x = \texttt{'A'}$ et $m' = \texttt{"B"}$. On emet `AB` et $t[258]$ prend la valeur `"AB"` et m la valeur `AB` ;
- on lit le code suivant (256). On a alors : $x = \texttt{'A'}$ et $m' = \texttt{"A"}$. On emet `AA` et $t[259]$ prend la valeur `"ABA"` et m la valeur `AA` ;
- on lit le code suivant (257). On a alors : $x = \texttt{'A'}$ et $m' = \texttt{"B"}$. On emet `AB` et $t[260]$ prend la valeur `"AAA"` et m la valeur `AB`.

On a alors épuisé tous les codes d'entrée ; c'est fini. Si on récapitule, on a récupéré en sortie la chaîne : `AABABAAAB` (c'est l'exemple introductif pour la compression – on retrouve bien la même chaîne).

Ce premier exemple masque en réalité une difficulté qui peut survenir lors de la phase de décompression. On note ici $|t|$ la taille effective de la table des suffixes (*i.e.* le nombre de cases effectivement utilisées au moment où on accède au tableau) :

- si lors de la décompression, on récupère un code $n < |t|$, il n'y a aucun problème, on associe aisément la chaîne correspondante ;
- mais il peut arriver que $n = |t|$; on n'a donc pas encore accès à la case du tableau qui nous intéresse ! Comment la reconstruire ?

Exercice 36.4 Tenter de décompresser le message suivant : 65 66 256 258.

Réfléchissons à comment cela peut se produire (du côté de la compression) :

- on sait que c' vient d'être ajouté après émission du code pour u, donc c' code $u' = ux$ pour un certain octet x ;
- d'un autre côté, x suit immédiatement u dans l'entrée, donc u' commence par x ;
- ainsi, $u' = ux$ avec x le premier octet de u qui est connu.

On en déduit la procédure complète suivante :

- on initialise la table avec une entrée pour chaque octet ;
- on maintient une variable c contenant le dernier code lu ;
- pour chaque code n que l'on lit :
  - si $n < |t|$ et $t[n] = xm$ (avec x un octet), on écrit xm en sortie ;
  - si $n = |t|$, on écrit $t[c]x$ en sortie, où x est le premier octet de $t[c]$;
  - dans les deux cas, on ajoute $t[c]x$ à la table, et l'on fait $c \leftarrow n$.

Exercice 36.5 Compresser puis décompresser la séquence suivante :

```
ABABCABCDABCDABCDA
```

Exercice 36.6 Écrirer une fonction `lzw_decomp` telle que l'appel `lzw_decomp d entree sortie` permet de décompresser le fichier `entree` vers le fichier `sortie` connaissant la longueur du code d.

```
lzw_decomp : int -> string -> string -> unit
```

Exercice 36.7 Quel est le livre compressé (avec une longueur de code égale à 15) dans le fichier `livre_mystere.txt` ?