# PROJET INFORMATIQUE M1
# *Burrows-Wheeler*

## 1 La transformée de Burrows-Wheeler

### 1.1 Définition

La transformée de Burrows-Wheeler, couramment appelée $BWT$ (pour *Burrows-Wheeler Transform*) est une technique de compression de données. Elle fut inventée par Michael Burrows et David Wheeler[1]. Cette technique fut rendue publique en 1994, suite à de précédents travaux de Wheeler en 1983. Il ne s'agit pas à proprement parler d'un algorithme de compression, car aucune réduction de taille n'est effectuée, au contraire (voir ci-dessous), mais bien d'une méthode de réorganisation des données: les probabilités pour que des caractères identiques initialement éloignés les uns des autres se retrouvent côte à côte sont alors augmentées. Cette technique est présente dans le format *bzip2* qui est actuellement l'un des formats offrant le plus grand quotient de compression.

### 1.2 Fonctionnement

Comme nous l'avons dit, la transformée de Burrows-Wheeler ne compresse pas les données, elle se contente de les réorganiser de manière à obtenir ultérieurement un meilleur taux de compression.

Tout d'abord, la chaîne de caractères à coder doit être copiée dans un tableau carré en décalant la chaîne d'un caractère vers la droite à chaque nouvelle ligne. Ces lignes sont ensuite classées par ordre alphabétique. Nous savons que, grâce au décalage, chaque dernière lettre de chaque ligne précède la première lettre de la même ligne, sauf pour la ligne originale dont on notera la position. De plus, comme les lignes sont rangées par ordre alphabétique, on peut retrouver la première colonne du tableau grâce à la dernière colonne.

**Prenons un premier exemple**. Supposons que la chaîne à coder soit *TEXTE*. On réalise tout d'abord le tableau.

| position | chaîne | | | | |
|---|---|---|---|---|---|
| 1 | T | E | X | T | E |
| 2 | E | T | E | X | T |
| 3 | T | E | T | E | X |
| 4 | X | T | E | T | E |
| 5 | E | X | T | E | T |

[1] Michael Burrows, D. J. Wheeler. "A block-sorting lossless data compression algorithm", 10th May 1994, Digital SRC Research Report 124.

Puis l'on classe ces chaînes par ordre alphabétique :

| position | chaîne | | | | |
|---|---|---|---|---|---|
| 1 (2) | E | T | E | X | T |
| 2 (5) | E | X | T | E | T |
| 3 (3) | T | E | T | E | X |
| 4 (1) | T | E | X | T | E |
| 5 (4) | X | T | E | T | E |

Pour la décompression, il est nécessaire de garder en mêmoire la position de la chaîne originale, ici 4. Le texte codé est donc la dernière colonne, soit : *4TTXEE*. Cette transformation n'apporte aucun gain de compression immédiat, au contraire, car il est nécessaire de transmettre des informations supplémentaires pour le décodage. Cependant, Burrows et Wheeler recommandent ensuite d'utiliser un algorithme de type *MTF*. Ainsi, la chaîne possédant de nombreuses répétitions de caractères contiendra beaucoup de 0. Ceci assure avec un algorithme de type codage de Huffman un quotient de compression élevé.

Lors de la décompression, la chaîne codée est rangée par ordre alphabétique (on reprend l'exemple précédent, cette fois-ci dans le sens de la décompression) :

| position | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| codé | T | T | X | E | E |
| classé | E | E | T | T | X |

C'est ici que l'on se sert du chiffre transmis (4). Nous savons que les deux caractères correspondant à cet indice ne se suivent pas et que le caractère de la ligne classée est le premier de la chaîne originale.

On part donc ici du $T$ en position 4. Ce $T$ est le deuxième de la ligne classée. On recherche donc le deuxième $T$ de la ligne codée, ce qui correspond à la position 2. Ce $T$ est donc suivi d'un $E$. Ce $E$ est le deuxième de la ligne classée. On retourne donc chercher le deuxième $E$ de la ligne codée. On arrive en position 5. Ce $E$ est suivi d'un $X \ldots$. On continue ainsi jusqu'à tomber sur le $E$ en position 4 de la ligne codée. La décompression est alors terminée. On retrouve bien nos données initiales, à savoir la chaîne *TEXTE*.

**Deuxième exemple**[2]. Supposons que la chaîne à coder soit *ABRACA*. On réalise tout d'abord le tableau.

| position | chaîne | | | | | |
|---|---|---|---|---|---|---|
| 1 | A | B | R | A | C | A |
| 2 | B | R | A | C | A | A |
| 3 | R | A | C | A | A | B |
| 4 | A | C | A | A | B | R |
| 5 | C | A | A | B | R | A |
| 6 | A | A | B | R | A | C |

[2]S. Mantaci, A. Restivo, M. Sciortino,"The Burrows-Wheeler transform: from data compression to combinatorics on words", AFL 2005.

Puis l'on classe ces chaînes par ordre alphabétique :

| position | chaîne | | | | | |
|---|---|---|---|---|---|---|
| 1 (6) | A | A | B | R | A | C |
| 2 (1) | A | B | R | A | C | A |
| 3 (4) | A | C | A | A | B | R |
| 4 (2) | B | R | A | C | A | A |
| 5 (5) | C | A | A | B | R | A |
| 6 (3) | R | A | C | A | A | B |

La position de la chaîne originale est 2. Le texte codé est donc la position du texte 2 suivie par dernière colonne, soit : *2CARAAB*.

Lors de la décompression, on a :

| position | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| codé | C | A | R | A | A | B |
| classé | A | A | A | B | C | R |

On part donc avec le caractère en position 2: $A$. Ce $A$ est le deuxième de la ligne classée. On recherche donc le deuxième $A$ de la ligne codée, ce qui correspond à la position 4. Ce $A$ est donc suivi d'un $B$. Ce $B$ est le premier (unique) de la ligne classée. On retourne donc chercher le premier $B$ de la ligne codée. On arrive en position 6. Ce $B$ est suivi d'un $R$ .... On continue ainsi jusqu'à ce que l'on retrouve la chaîne $ABRACA$.

**Réaliser un programme informatique (dans un langage à votre choix) qui implante le systeme de compression de Burrows-Wheeler : la transformé de BW, suivie par un algorithme de compression, comme par exemple MTF+un codage statistique.**

Des critères comme le fonctionnement/fiabilité, la commodité de manipulation et visuelle, clarté, originalité, seront pris en compte.