Συμπίεση

Η συμπίεση δεδομένων ελαττώνει το μέγεθος ενός αρχείου :

- Εξοικονόμηση αποθηκευτικού χώρου
- Εξοικονόμηση χρόνου μετάδοσης
- Τα περισσότερα αρχεία έχουν πλεονασμό στα δεδομένα τους

Είναι σημαντική η συμπίεση των δεδομένων σήμερα;

- Νόμος του Moore: «το πλήθος των τρανζίστορ που μπορούν να τοποθετηθούν σε ένα ολοκληρωμένο κύκλωμα διπλασιάζεται κάθε 18-24 μήνες»
- IBM report on Big Data 2011: «Καθημερινά παράγονται $2{,}5 \times 10^{15}$ bytes πληροφοριών»

Συμπίεση

Generic file compression.

- Files: GZIP, BZIP, 7z.
- Archivers: PKZIP.
- File systems: NTFS, HFS+, ZFS.

Multimedia.

- Images: GIF, JPEG.
- Sound: MP3.
- Video: MPEG, DivX™, HDTV.

Communication.

- ITU-T T4 Group 3 Fax.
- V.42bis modem.
- Skype.

Databases. Google, Facebook, ...

Google™

facebook

Κατηγορίες Συμπίεσης

Συμπίεση με απώλειες δεδομένων (lossy compression)

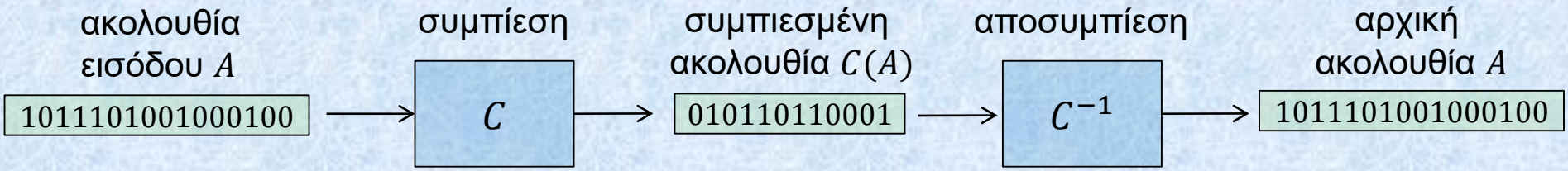
- π.χ. συμπίεση εικόνας και ήχου

Συμπίεση χωρίς απώλειες δεδομένων (lossless compression)

- π.χ. συμπίεση κειμένου, συμπίεση αρχείων

Κατηγορίες Συμπίεσης

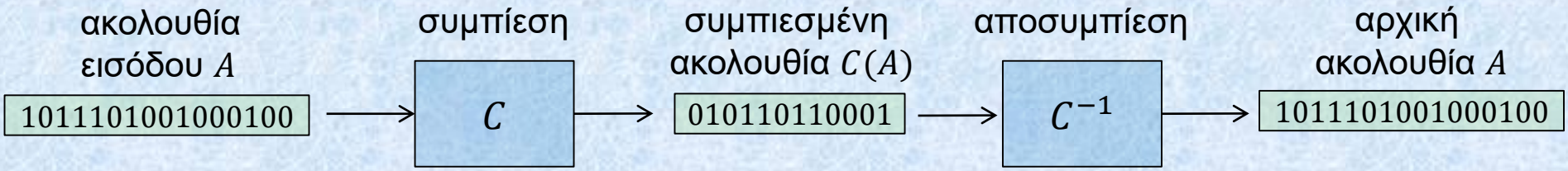
Συμπίεση χωρίς απώλειες (lossless compression)



$$\text{Λόγος συμπίεσης} = \frac{|C(A)|}{|A|}$$

Κατηγορίες Συμπίεσης

Συμπίεση χωρίς απώλειες (lossless compression)



$$\text{Λόγος συμπίεσης} = \frac{|C(A)|}{|A|}$$

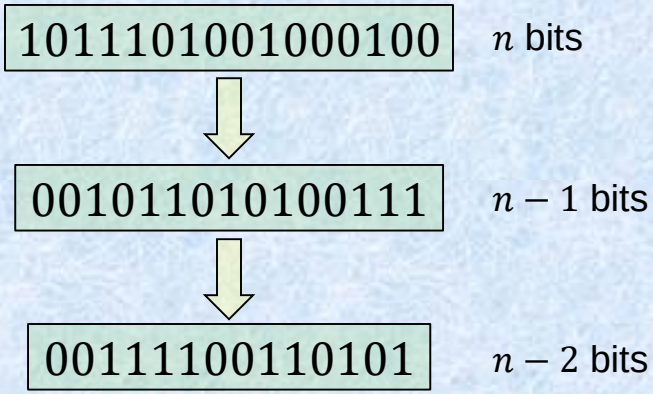
Θα δούμε δύο βασικές τεχνικές:

- Κωδικοποίηση χαρακτήρων σταθερού μήκους με κώδικα μεταβλητού μήκους (Huffman).
- Κωδικοποίηση χαρακτήρων μεταβλητού μήκους με κώδικα σταθερού μήκους (LZW).

Συμπίεση

Ερώτημα

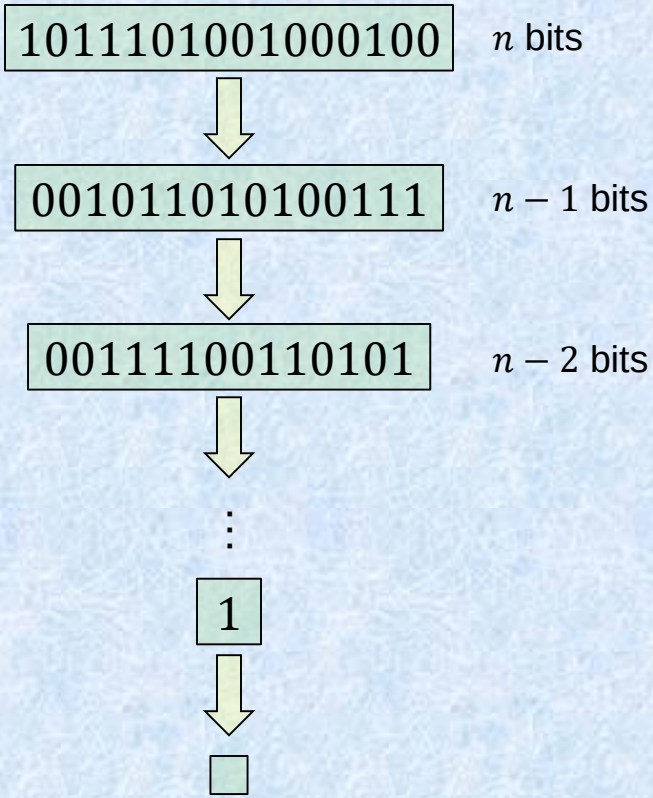
Μπορούμε να συμπιέσουμε χωρίς απώλειες οποιαδήποτε ακολουθία δυαδικών ψηφίων;



Συμπίεση

Ερώτημα

Μπορούμε να συμπιέσουμε χωρίς απώλειες οποιαδήποτε ακολουθία δυαδικών ψηφίων;



Συμπίεση

Ερώτημα

Μπορούμε να συμπιέσουμε χωρίς απώλειες οποιαδήποτε ακολουθία δυαδικών ψηφίων;

Ας υποθέσουμε ότι οποιαδήποτε ακολουθία n bit μπορεί να συμπιεστεί σε μια ακολουθία $< n$ bit.

Υπάρχουν 2^n διαφορετικές ακολουθίες των n bit. Άρα θα πρέπει να υπάρχουν τουλάχιστον τόσες ακολουθίες με $< n$ bit.

Όμως όλες οι ακολουθίες με $< n$ bit είναι

$$\sum_{k=0}^{n-1} 2^k = \frac{2^n - 1}{2 - 1} = 2^n - 1 < 2^n$$

Συμπίεση

Παράδειγμα

Ποιο ποσοστό όλων των ακολουθιών με 1000 δυαδικά ψηφία μπορεί να συμπιεστεί χωρίς απώλειες κατά 50% ή παραπάνω;

Υπάρχουν 2^{1000} διαφορετικές ακολουθίες των 1000 bit.

Το πλήθος των ακολουθιών με το πολύ 500 bit είναι

$$\sum_{k=0}^{500} 2^k = \frac{2^{501}-1}{2-1} = 2^{501}-1 < 2^{501}$$

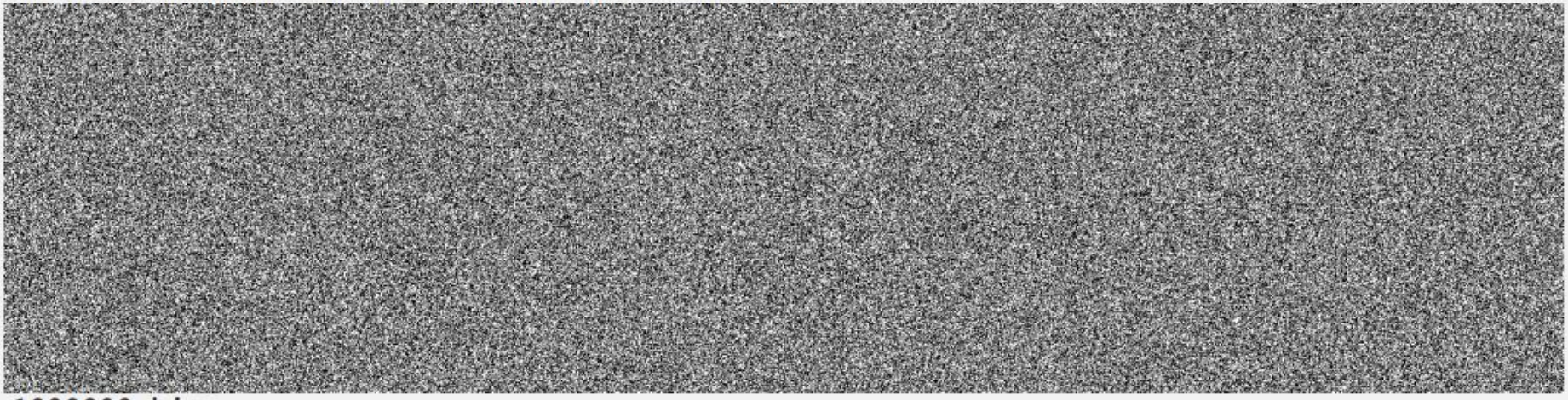
Έχουμε

$$\frac{2^{501}}{2^{1000}} = \frac{1}{2^{499}}$$

δηλαδή μπορούν να συμπιεστούν το πολύ 1 στις 2^{499} ακολουθίες!

Συμπίεση: Θεωρία υπολογισμού

```
% java RandomBits | java PictureDump 2000 500
```

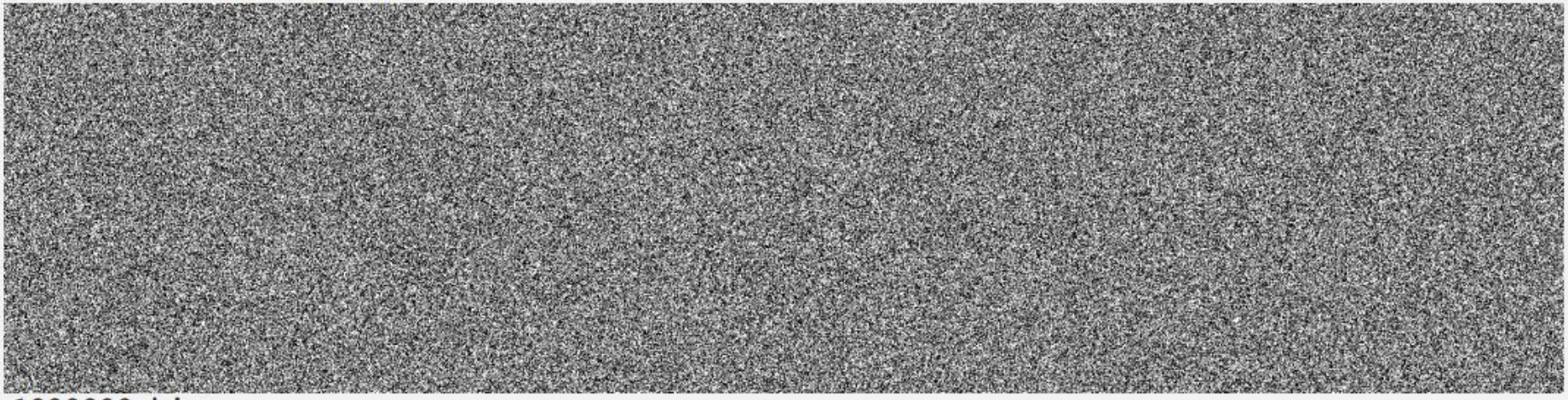


```
1000000 bits
```

A difficult file to compress: one million (pseudo-) random bits

Συμπίεση: Θεωρία υπολογισμού

```
% java RandomBits | java PictureDump 2000 500
```



```
1000000 bits
```

A difficult file to compress: one million (pseudo-) random bits

Παράγεται από το Java πρόγραμμα:

```
public class RandomBits {
        public static void main(String[] args) {
                int x = 11111;
                for (int i = 0; i < 1000000; i++) {
                        x = x * 314159 + 218281;
                        BinaryStdOut.write(x > 0);
                }
                BinaryStdOut.close();
        }
}
```

Κώδικες Huffman

Μέθοδος συμπίεσης δεδομένων:

Μας δίνεται αρχείο με n διαφορετικούς χαρακτήρες. Θέλουμε να αντιστοιχίσουμε τον κάθε χαρακτήρα σε ένα δυαδικό κωδικό, έτσι ώστε να ελαχιστοποιηθεί ο συνολικός αριθμός των bits όλου του αρχείου.

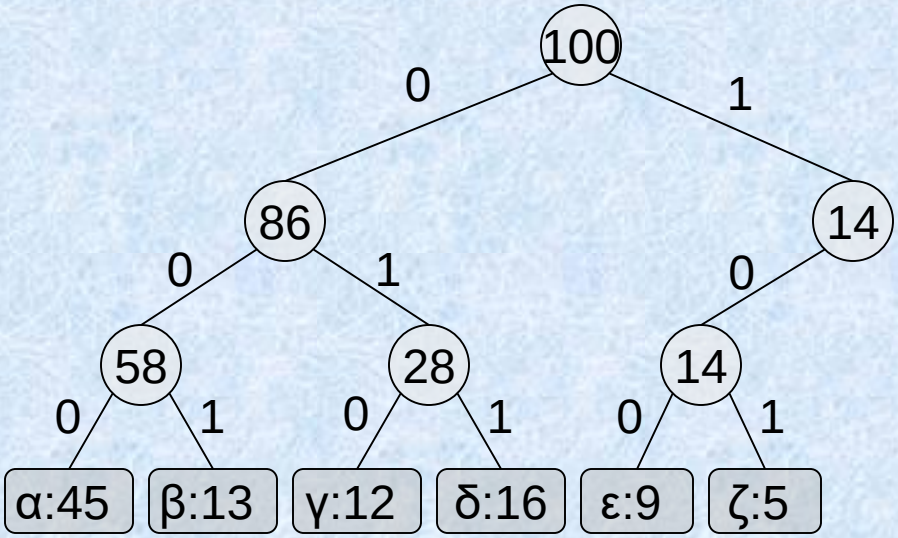
Π.χ. Αρχείο που αποτελείται από τους χαρακτήρες α,β,γ,δ,ε και ζ

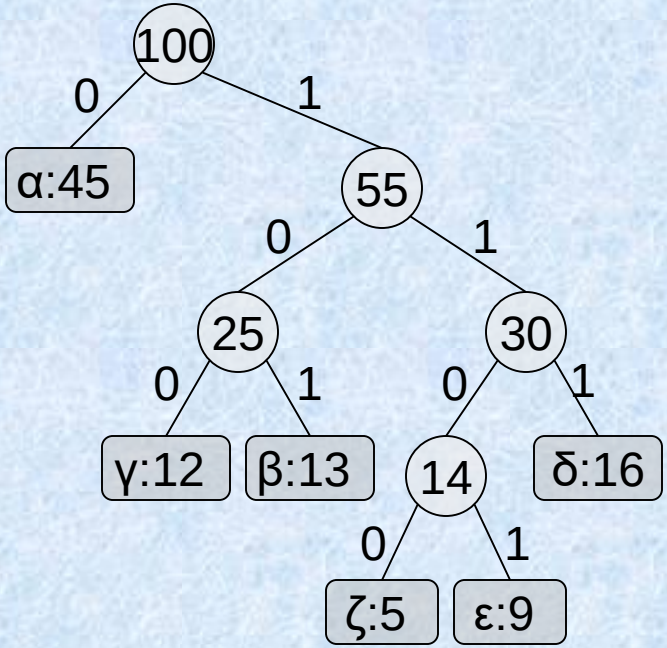
	α	β	γ	δ	ε	ζ
συχνότητα ($\times 10^3$)	45	13	12	16	9	5
κώδικας σταθερού μήκους	000	001	010	011	100	101
κώδικας μεταβλητού μήκους	0	101	100	111	1101	1100

Απροθηματικός κώδικας: κανένας κωδικός δεν αποτελεί πρόθημα άλλου κωδικού

Κώδικες Huffman

	α	β	γ	δ	ε	ζ
συχνότητα ($\times 10^3$)	45	13	12	16	9	5
κώδικας σταθερού μήκους	000	001	010	011	100	101
κώδικας μεταβλητού μήκους	0	101	100	111	1101	1100





$3 \cdot 100 = 300$

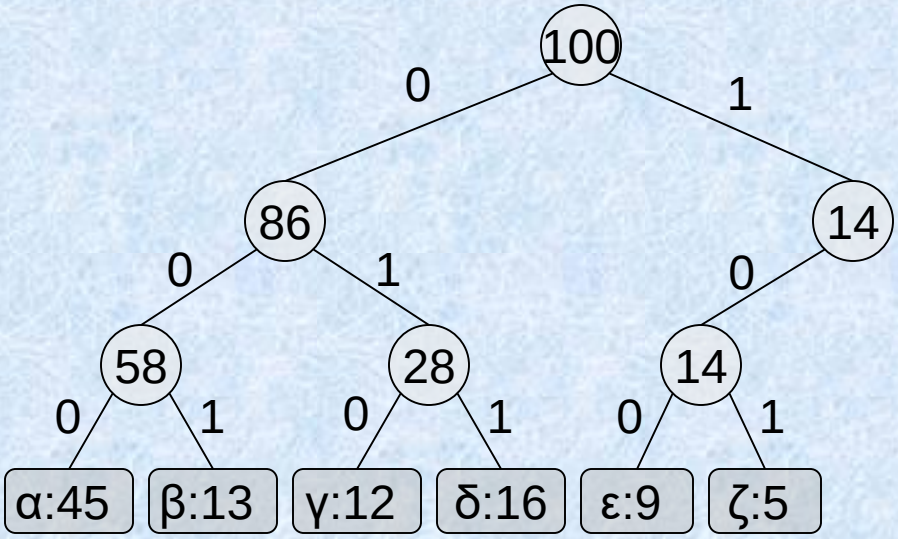
$1 \cdot 45 + 3 \cdot 13 + 3 \cdot 12 + 3 \cdot 16 + 4 \cdot 9 + 4 \cdot 5 = 224$

Κώδικες Huffman

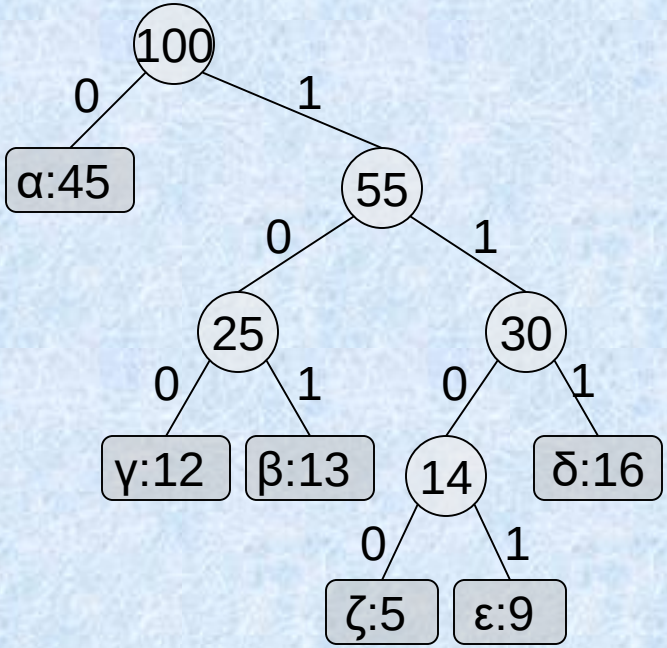
$f(c)$ συχνότητα του χαρακτήρα c

$d_T(c)$ βάθος του c στο δένδρο T

$B(T) = \sum_c f(c) d_T(c)$ συνολικός αριθμός bits κωδικοποίησης



3·100 = 300



1·45 + 3·13 + 3·12 + 3·16 + 4·9 + 4·5 = 224

Κώδικες Huffman

Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$

α:45 β:13 γ:12 δ:16 ε:9 ζ:5

Κώδικες Huffman

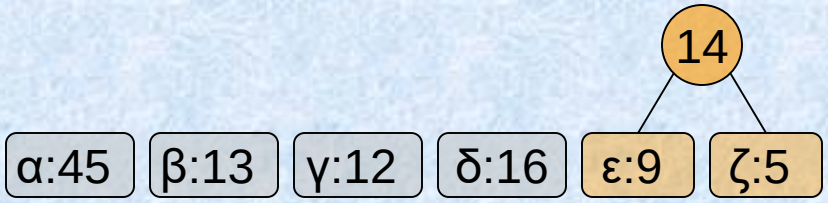
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

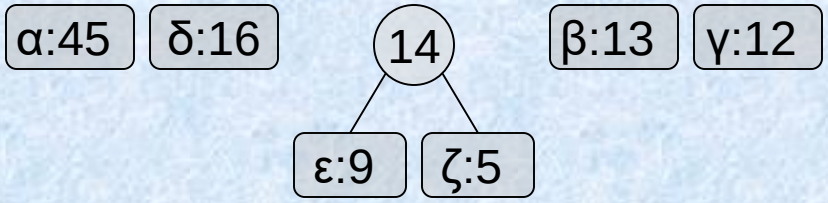
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

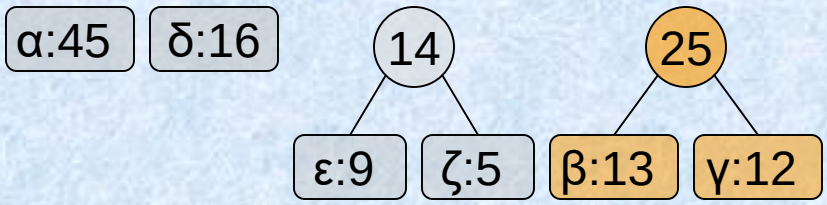
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

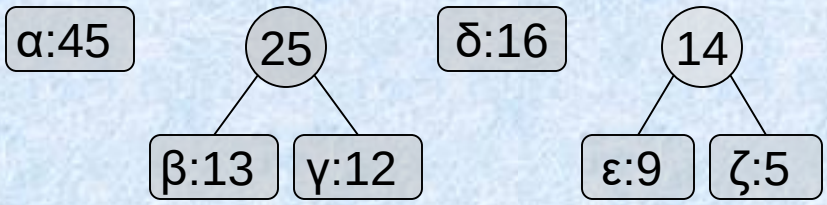
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

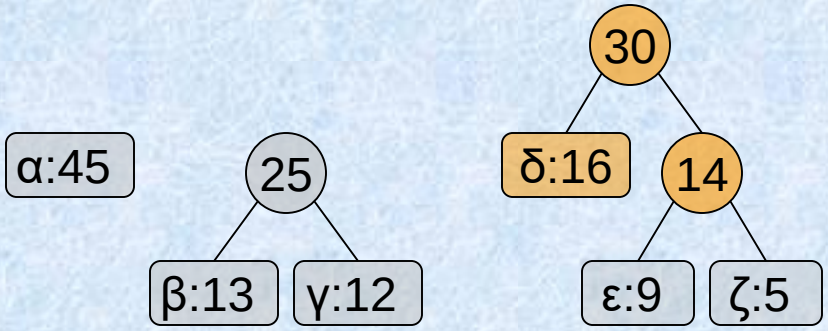
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

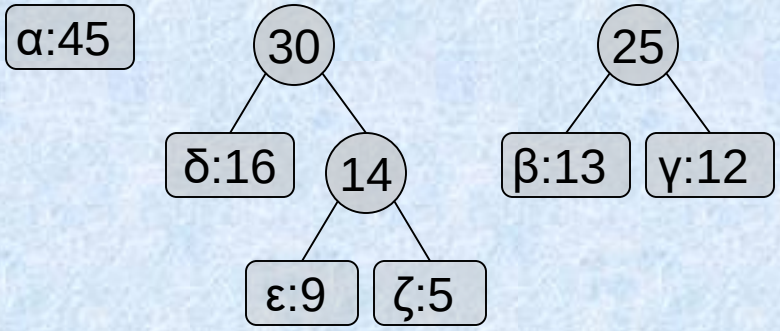
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

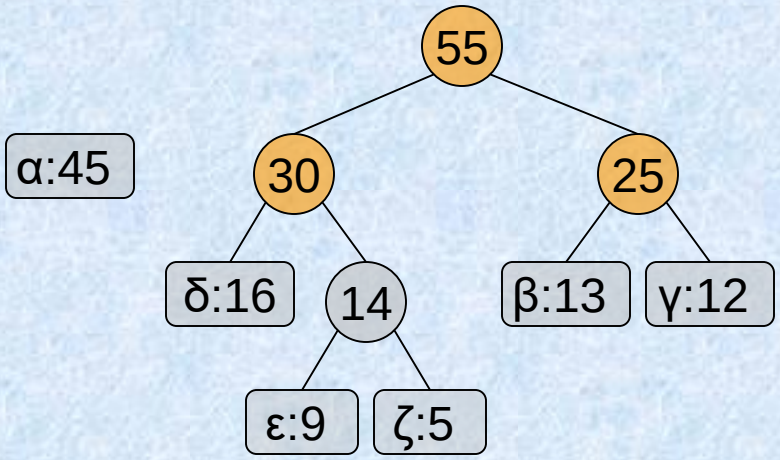
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

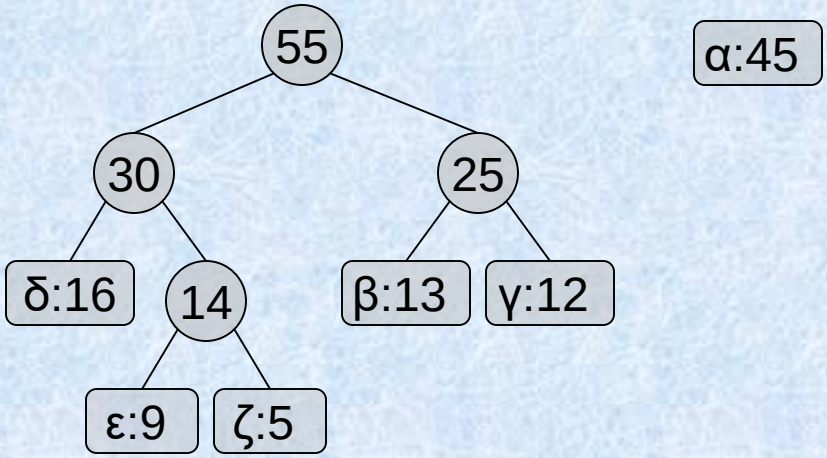
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

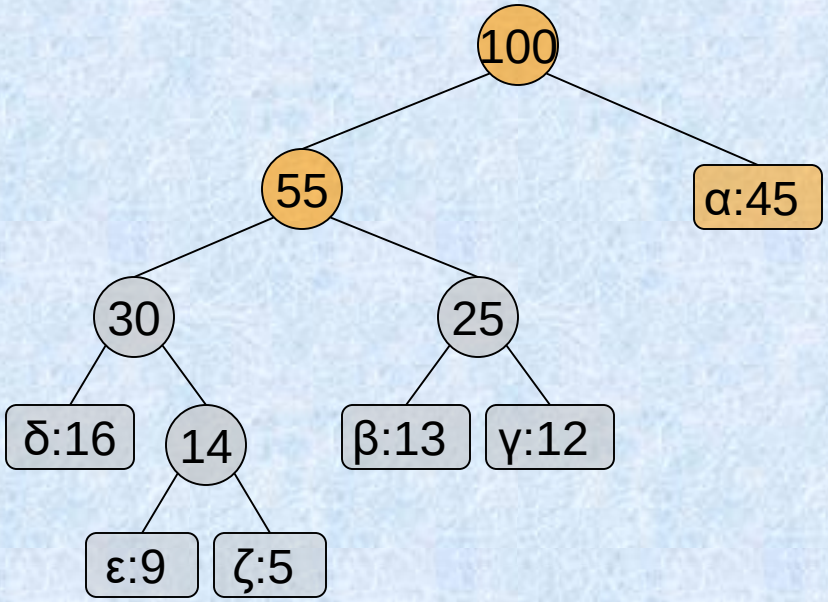
Κατασκευή του κώδικα

Σε κάθε βήμα συνδυάζουμε 2 χαρακτήρες με την **ελάχιστη συχνότητα**.

Έστω x και y δύο τέτοιοι χαρακτήρες.

Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα

$$f(z) \leftarrow f(x) + f(y)$$



Κώδικες Huffman

Υλοποίηση με ουρά προτεραιότητας

Σε κάθε βήμα :

- Συνδυάζουμε 2 χαρακτήρες, έστω x και y, με την ελάχιστη συχνότητα
  $\Rightarrow$ 2 εξαγωγές ελάχιστου

- Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα $f(z) \leftarrow f(x) + f(y)$
  $\Rightarrow$ εισαγωγή στοιχείου

Πλήθος των εξαγωγών ελάχιστου = πλήθος εισαγωγών

Κώδικες Huffman

Υλοποίηση με ουρά προτεραιότητας

Σε κάθε βήμα :

- Συνδυάζουμε 2 χαρακτήρες, έστω x και y, με την ελάχιστη συχνότητα
  $\Rightarrow$ 2 εξαγωγές ελάχιστου

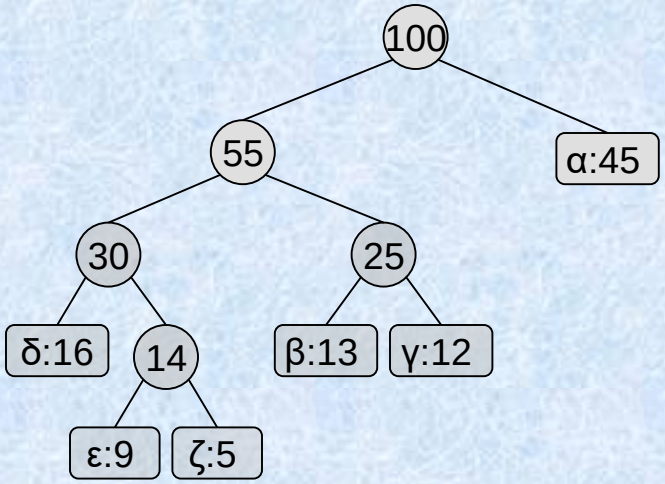
- Αντικαθιστούμε τους x και y από νέο χαρακτήρα z με συχνότητα $f(z) \leftarrow f(x) + f(y)$
  $\Rightarrow$ εισαγωγή στοιχείου

Πλήθος των εξαγωγών ελάχιστου = πλήθος εισαγωγών

= πλήθος κόμβων του δένδρου = $2n - 1$

όπου n ο αριθμός των διαφορετικών χαρακτήρων

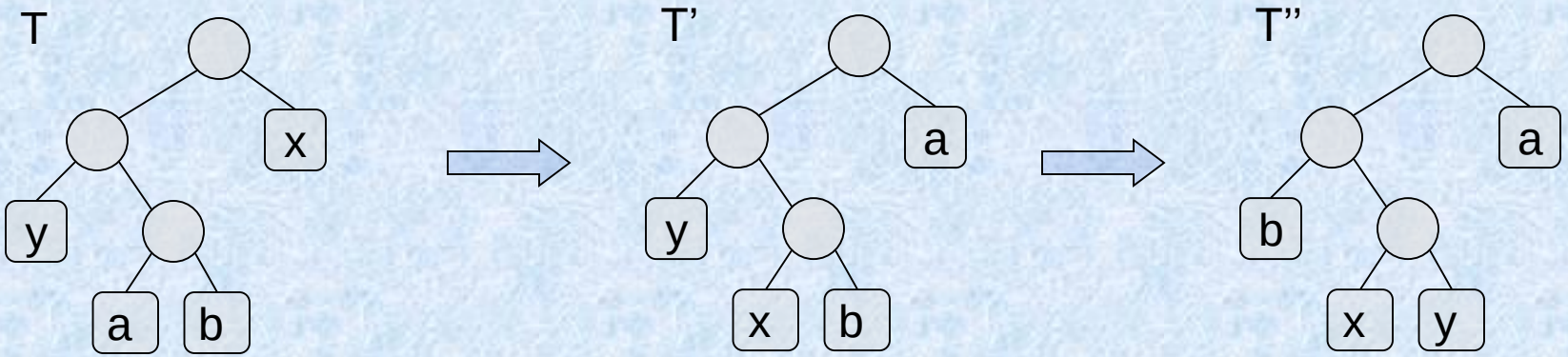
Συνολικός χρόνος κατασκευής = $O(n \log n)$



Κώδικες Huffman

Λήμμα Έστω x και y δύο χαρακτήρες με την ελάχιστη συχνότητα. Υπάρχει βέλτιστος κώδικας στον οποίο οι κωδικοί των x και y έχουν το ίδιο μήκος και διαφέρουν μόνο στο τελευταίο ψηφίο.

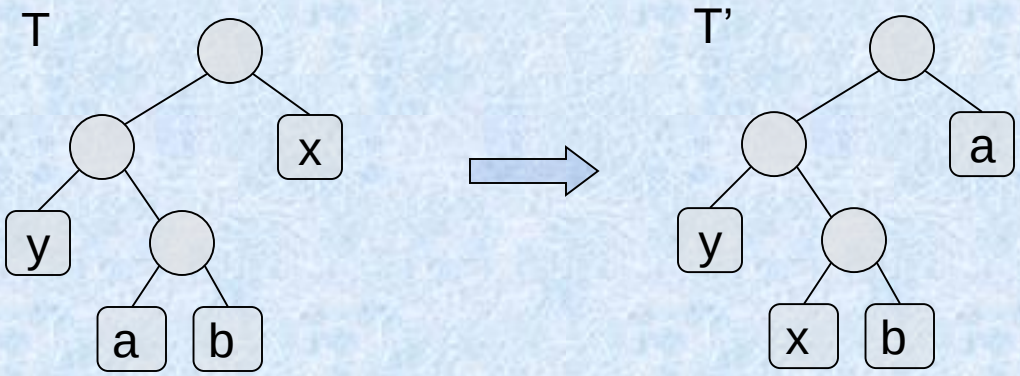
Έστω T ένα βέλτιστο δένδρο, με τους χαρακτήρες a και b να έχουν μέγιστο βάθος.



Κώδικες Huffman

Λήμμα Έστω x και y δύο χαρακτήρες με την ελάχιστη συχνότητα. Υπάρχει βέλτιστος κώδικας στον οποίο οι κωδικοί των x και y έχουν το ίδιο μήκος και διαφέρουν μόνο στο τελευταίο ψηφίο.

Έστω T ένα βέλτιστο δένδρο, με τους χαρακτήρες a και b να έχουν μέγιστο βάθος.



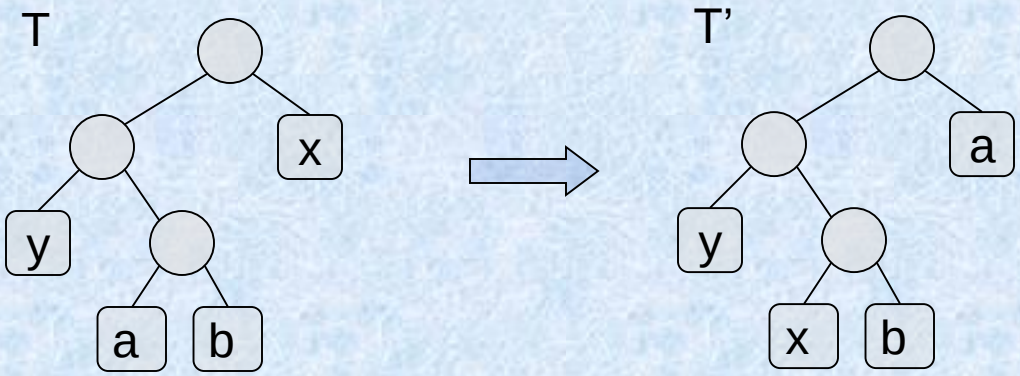
ανταλλαγή x και a

$$B(T) - B(T') = \sum_c f(c)d_T(c) - \sum_c f(c)d_{T'}(c)$$
$$= f(a)d_T(a) + f(x)d_T(x) - (f(a)d_{T'}(a) + f(x)d_{T'}(x))$$

Κώδικες Huffman

Λήμμα Έστω x και y δύο χαρακτήρες με την ελάχιστη συχνότητα. Υπάρχει βέλτιστος κώδικας στον οποίο οι κωδικοί των x και y έχουν το ίδιο μήκος και διαφέρουν μόνο στο τελευταίο ψηφίο.

Έστω T ένα βέλτιστο δένδρο, με τους χαρακτήρες a και b να έχουν μέγιστο βάθος.



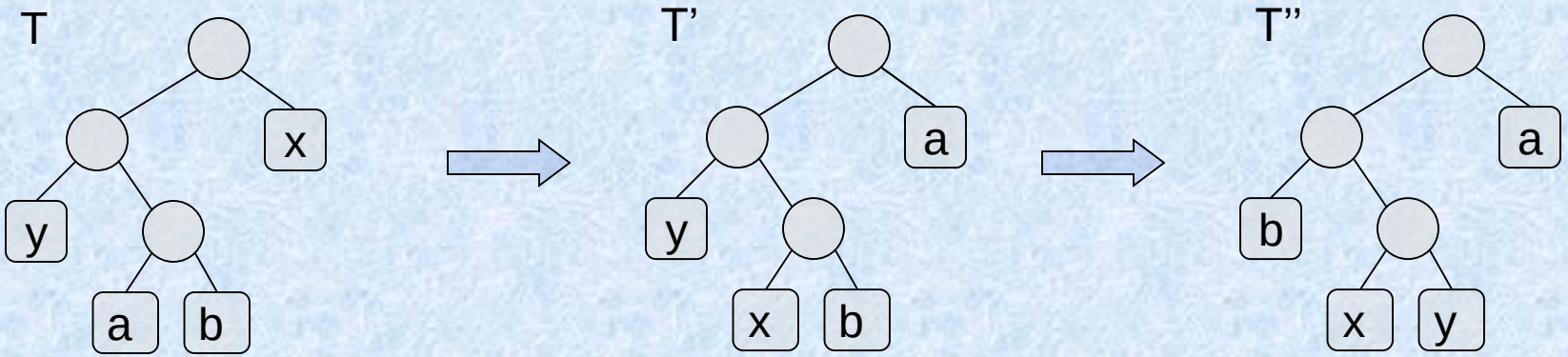
ανταλλαγή x και a

$$
\begin{aligned}
B(T) - B(T') &= \textstyle\sum_c f(c) d_T(c) - \sum_c f(c) d_{T'}(c) \\
&= f(a) d_T(a) + f(x) d_T(x) - (f(a) d_{T'}(a) + f(x) d_{T'}(x)) \\
&= f(a) d_T(a) + f(x) d_T(x) - f(a) d_T(x) - f(x) d_T(a) \\
&= (f(a) - f(x))(d_T(a) - d_T(x)) \geq 0
\end{aligned}
$$

Κώδικες Huffman

Λήμμα Έστω x και y δύο χαρακτήρες με την ελάχιστη συχνότητα. Υπάρχει βέλτιστος κώδικας στον οποίο οι κωδικοί των x και y έχουν το ίδιο μήκος και διαφέρουν μόνο στο τελευταίο ψηφίο.

Έστω T ένα βέλτιστο δένδρο, με τους χαρακτήρες a και b να έχουν μέγιστο βάθος.



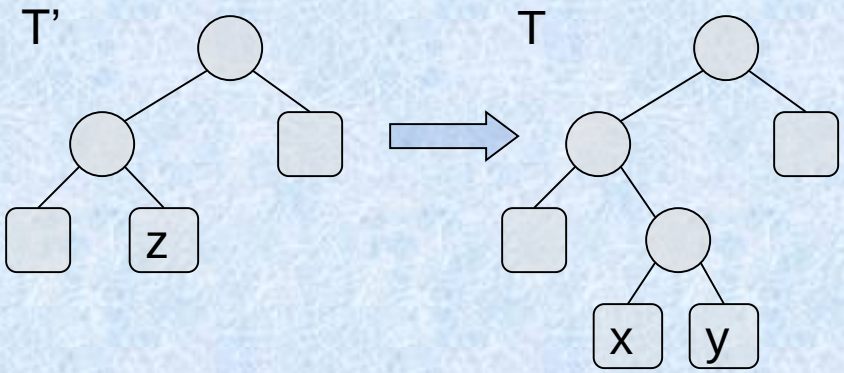
ανταλλαγή x και a ανταλλαγή y και b

$$B(T) - B(T') \geq 0 \qquad B(T') - B(T'') \geq 0$$

Συνεπώς, επειδή το T είναι βέλτιστο έχουμε $B(T) = B(T'')$

Κώδικες Huffman

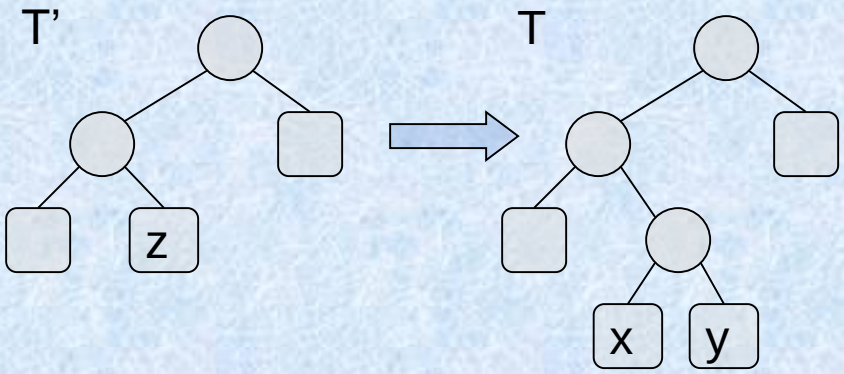
Λήμμα Έστω x και y δύο χαρακτήρες με την ελάχιστη συχνότητα και z ο χαρακτήρας που αντιστοιχεί στο συνδυασμό των x και y. Έστω T' ένα βέλτιστο δένδρο για το νέο αλφάβητο (με το z στη θέση των a και b). Τότε, το δένδρο T που προκύπτει από το T' με την αντικατάσταση του z είναι βέλτιστο για τον αρχικό αλφάβητο.



$$B(T) = B(T') + f(x) + f(y)$$

Κώδικες Huffman

Λήμμα Έστω x και y δύο χαρακτήρες με την ελάχιστη συχνότητα και z ο χαρακτήρας που αντιστοιχεί στο συνδυασμό των x και y. Έστω T' ένα βέλτιστο δένδρο για το νέο αλφάβητο (με το z στη θέση των a και b). Τότε, το δένδρο T που προκύπτει από το T' με την αντικατάσταση του z είναι βέλτιστο για τον αρχικό αλφάβητο.



$$B(T) = B(T') + f(x) + f(y)$$

Ας υποθέσουμε ότι το T δεν είναι βέλτιστο.

Έστω S ένα βέλτιστο δένδρο για το αρχικό αλφάβητο, με τα x και y αδέλφια στο μέγιστο βάθος.

Θεωρούμε το δένδρο S' που προκύπτει από την αντικατάσταση των x και y με το z.

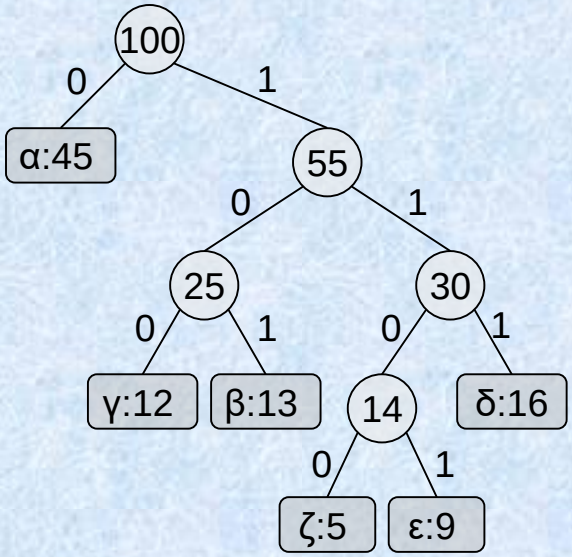
$$\begin{aligned} B(S') &= B(S) - f(x) - f(y) \\ &< B(T) - f(x) - f(y) \\ &< B(T') \end{aligned}$$

Άτοπο!

Κώδικες Huffman

Για να είναι δυνατή η αποκωδικοποίηση θα να στείλουμε και τον κώδικα μαζί με το κωδικοποιημένο κείμενο.

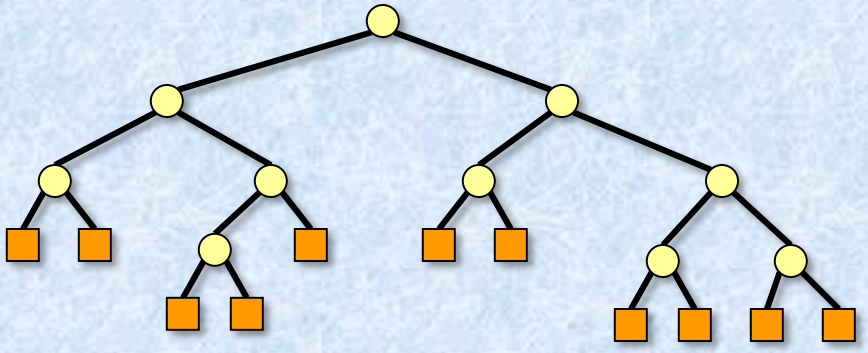
	α	β	γ	δ	ε	ζ
συχνότητα ($\times 10^3$)	45	13	12	16	9	5
κώδικας σταθερού μήκους	000	001	010	011	100	101
κώδικας μεταβλητού μήκους	0	101	100	111	1101	1100



Αρκεί να στείλουμε το δυαδικό δένδρο που αντιστοιχεί στον κώδικα:

Μπορεί να αναπαρασταθεί με n bits!

Διάσχιση Δυαδικού Δένδρου



$N = 10$

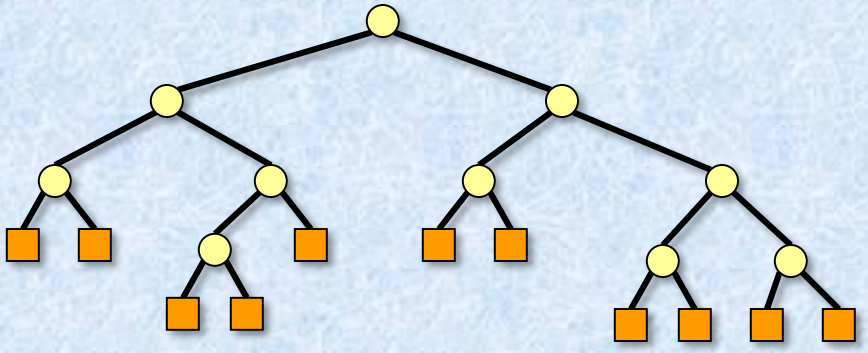
$$B = \left[\begin{array}{ccccccccccccccccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{ccccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Ένα δυαδικό δένδρο με N εσωτερικού κόμβους μπορεί να αναπαρασταθεί από μια ακολουθία B από $2N + 1$ δυαδικά ψηφία, που ικανοποιεί τις ακόλουθες συνθήκες:

- $N + 1$ ψηφιά είναι 0 και N ψηφία είναι 1
- Για κάθε θέση $i = 1, 2, \ldots, 2N + 1$, ο αριθμός των 1 που βρίσκονται πριν το b_i είναι μεγαλύτερος ή ίσος του αριθμού των 0 που βρίσκονται πριν το b_i

Διάσχιση Δυαδικού Δένδρου



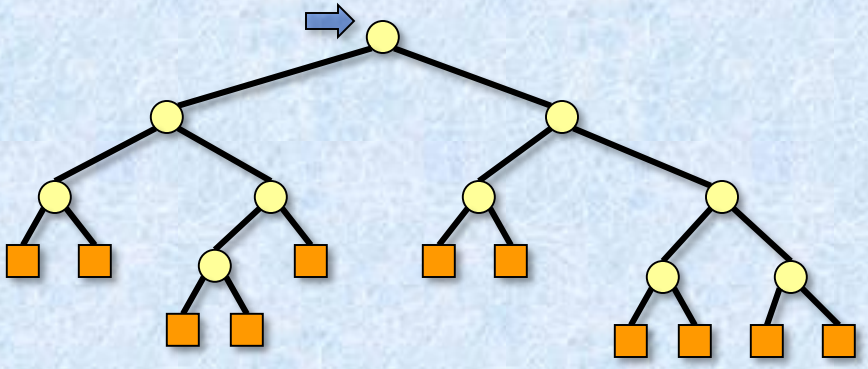
$N = 10$

$$B = \left[\begin{array}{cccccccccccccccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{cccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



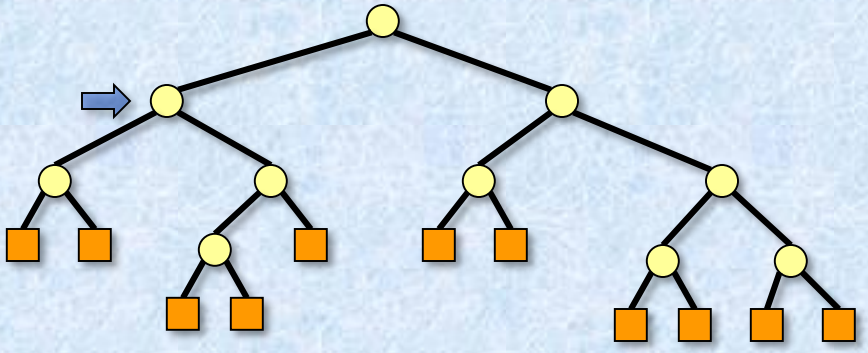
$N = 10$

$$B = \left[\begin{array}{cccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{cccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



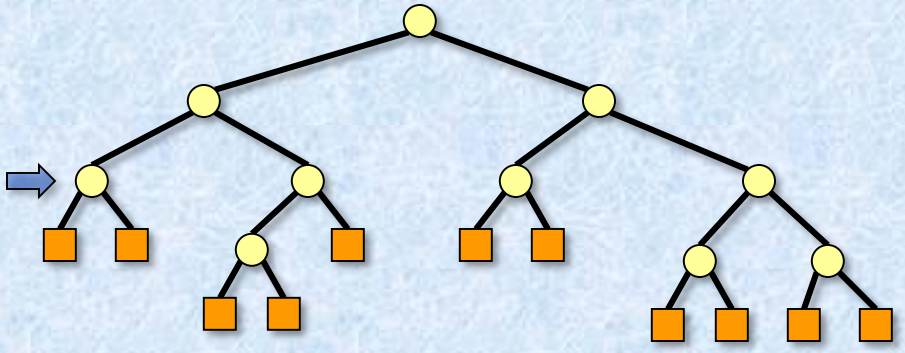
$N = 10$

$$B = \left[\begin{array}{cccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{cccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



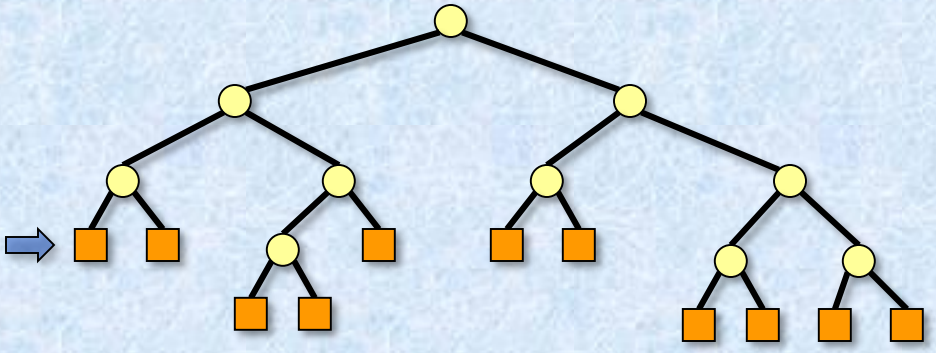
$N = 10$

$$B = \left[\begin{array}{cccccccccccccccccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{cccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου



$N = 10$

$$B = \left[\begin{array}{cccccc} b_0 & b_1 & b_2 & \ldots & b_{19} & b_{20} \end{array} \right]$$

$$= \left[\begin{array}{cccccccccccccccccccc} 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array} \right]$$

Εκτελούμε προδιατεταγμένη διάσχιση του δένδρου. Αν ο επόμενος κόμβος που συναντάμε είναι εσωτερικός τότε το επόμενο ψηφίο είναι 1, διαφορετικά, αν είναι εξωτερικός κόμβος, τότε το επόμενο ψηφίο είναι 0.

Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

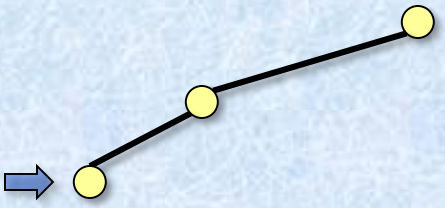
$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

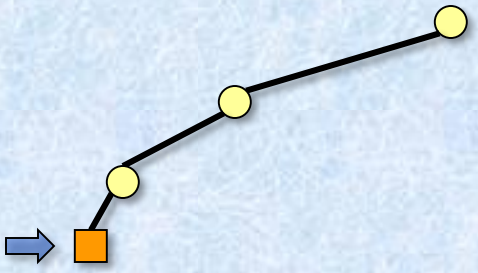


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

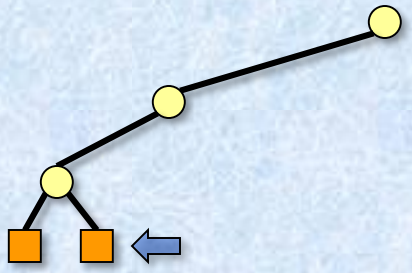


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

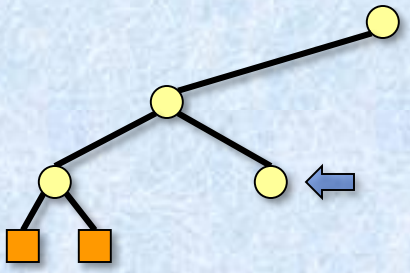


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

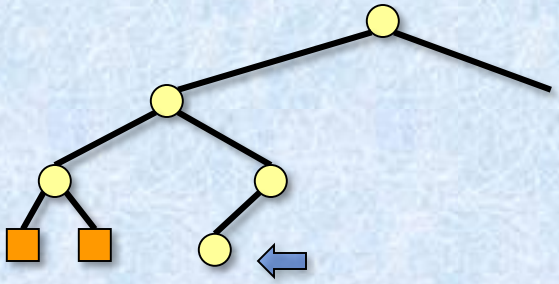


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$

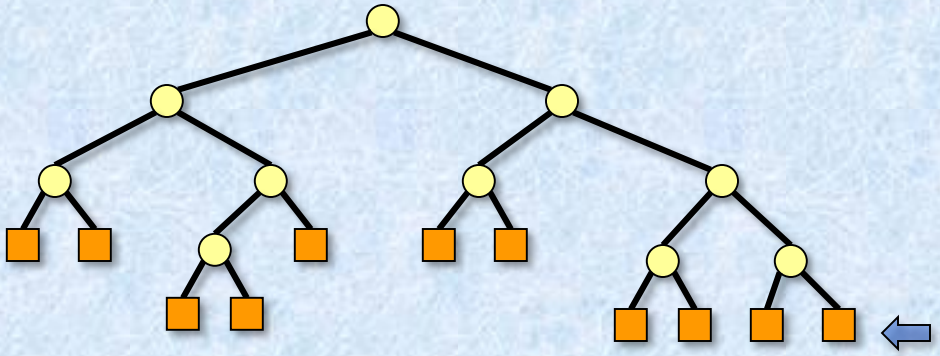


Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

$$B = [\ 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0\ ]$$



Διάσχιση Δυαδικού Δένδρου

Κατασκευή δυαδικού δένδρου από ακολουθία δυαδικών ψηφίων

Εξετάζουμε ένα-ένα τα ψηφία της ακολουθίας. Αν το επόμενο ψηφίο είναι 1 τότε δημιουργούμε νέο εσωτερικό κόμβο και κατασκευάζουμε αναδρομικά πρώτα το αριστερό και μετά το δεξί υποδένδρο. Διαφορετικά, αν το επόμενο ψηφίο είναι 0 τότε δημιουργούμε νέο εξωτερικό κόμβο.

Κατασκευή με αναδρομή

```
class Node {
        Item item; Node l; Node r;
        Node(Item v, Node l, Node r) {
                this.item = v;
                this.l = l;
                this.r = r;
        }
}
```

```
int k; // τρέχουσα θέση
Node createBT(char[] B)
{
    if (k == B.length) return null;
    if (B[k++] == '0') return null;

    Item v = …
    Node x = new Node(v,null,null);
    x.l = createBT(B);
    x.r = createBT(B);
    return x;
}
```

Αλγόριθμοι Lempel-Ziv

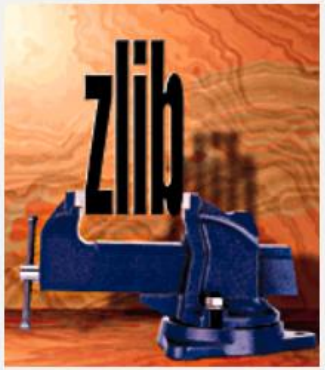
Κατηγορία αλγόριθμων συμπίεσης, οι οποίοι σε γενικές γραμμές κάνουν τα εξής:

- Κατασκευάζουν μια δομή λεξικού από συμβολοσειρές που έχουν δει μέχρι την τρέχουσα θέση στο αρχείο εισόδου.
- Αναζητούν στη δομή αυτή για να βρουν τη μεγαλύτερη συμβολοσειρά που ταυτίζεται με τη συμβολοσειρά η οποία ξεκινά από την τρέχουσα θέση.
- Επιστρέφουν ένα κώδικα που αντιστοιχεί σε μια τέτοια ταύτιση.
- Η τρέχουσα θέση μετακινείται μετά τη συμβολοσειρά που έχει ταυτιστεί.

Αλγόριθμοι Lempel-Ziv

Lempel-Ziv and friends.

- LZ77.
- LZ78.
- LZW.
- Deflate / zlib = LZ77 variant + Huffman.



Unix compress, GIF, TIFF, V.42bis modem: LZW.

zip, 7zip, gzip, jar, png, pdf: deflate / zlib.

iPhone, Sony Playstation 3, Apache HTTP server: deflate / zlib.

Αλγόριθμος Lempel-Ziv-Welch

Συμπίεση

Το λεξικό D αντιστοιχεί ακολουθίες χαρακτήρων σε κωδικούς των W bit. Αρχικά το D περιέχει μόνο τις συμβολοσειρές με ένα χαρακτήρα.

Εκτελούμε τα παρακάτω βήματα μέχρι να διαβαστεί ολόκληρο το αρχείο εισόδου:

- Αναζητούμε στο D τη μεγαλύτερη συμβολοσειρά s η οποία αποτελεί πρόθεμα μιας συμβολοσειράς που ξεκινά από την τρέχουσα θέση του αρχείου.
- Τυπώνουμε τον κωδικό που αντιστοιχεί στην s.
- Μετακινούμε την τρέχουσα θέση μετά την s. Έστω c ο επόμενος χαρακτήρας.
- Εισαγάγουμε στη D τη συμβολοσειρά sc και της δίνουμε τον επόμενο διαθέσιμο κωδικό.

Αλγόριθμος Lempel-Ziv-Welch

Στα παραδείγματα υποθέτουμε ότι η είσοδος αποτελείται από ακολουθίες χαρακτήρων σε κωδικοποίηση ASCII των 7 bit. Το λεξικό D αντιστοιχεί ακολουθίες χαρακτήρων σε κώδικες των 8 bit στους οποίους αναφερόμαστε με τις τιμές τους στο δεκαεξαδικό σύστημα.

Χρησιμοποιούμε τους κώδικες από το 81 και μετά για κάθε νέα συμβολοσειρά που ανακαλύπτουμε στο αρχείο εισόδου.

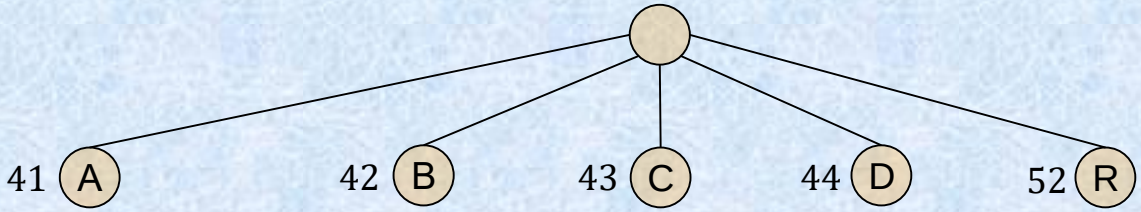
ASCII	Hex	Symbol
32	20	(space)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Symbol
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Symbol
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

Αλγόριθμος Lempel-Ziv-Welch

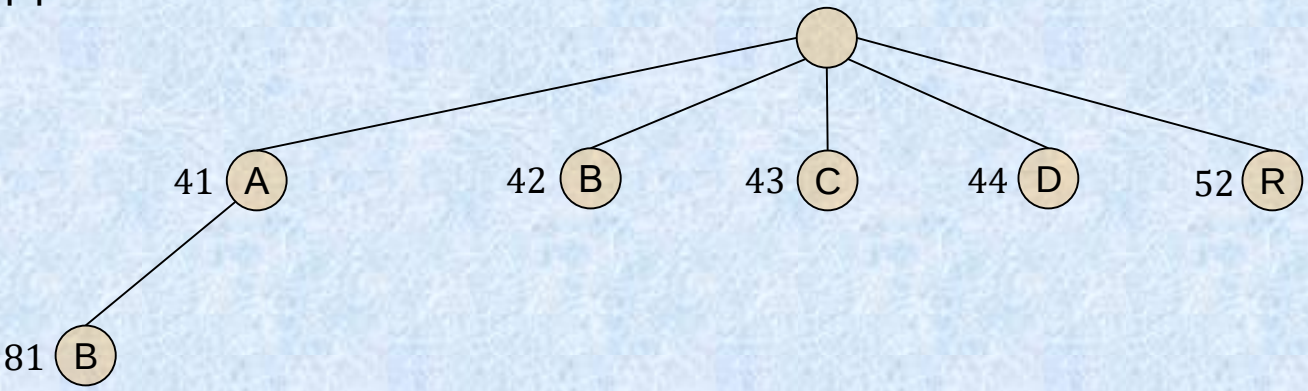
είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A																
κωδικός	41																



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B															
κωδικός	41	42															

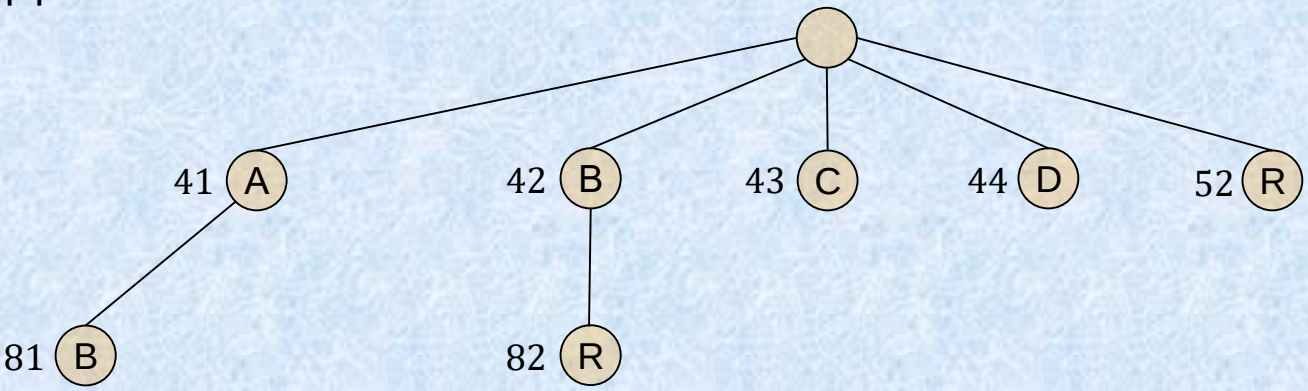
εισαγωγή AB



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R														
κωδικός	41	42	52														

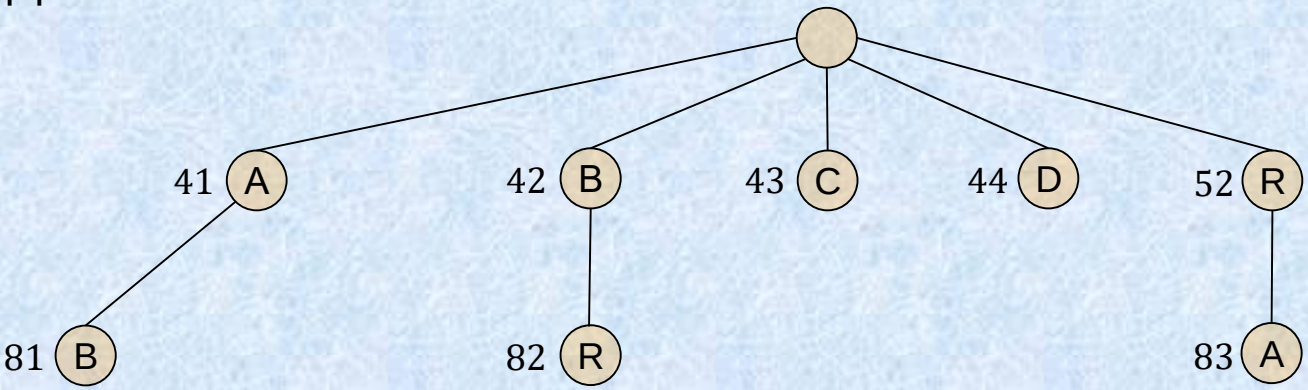
εισαγωγή BR



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A													
κωδικός	41	42	52	41													

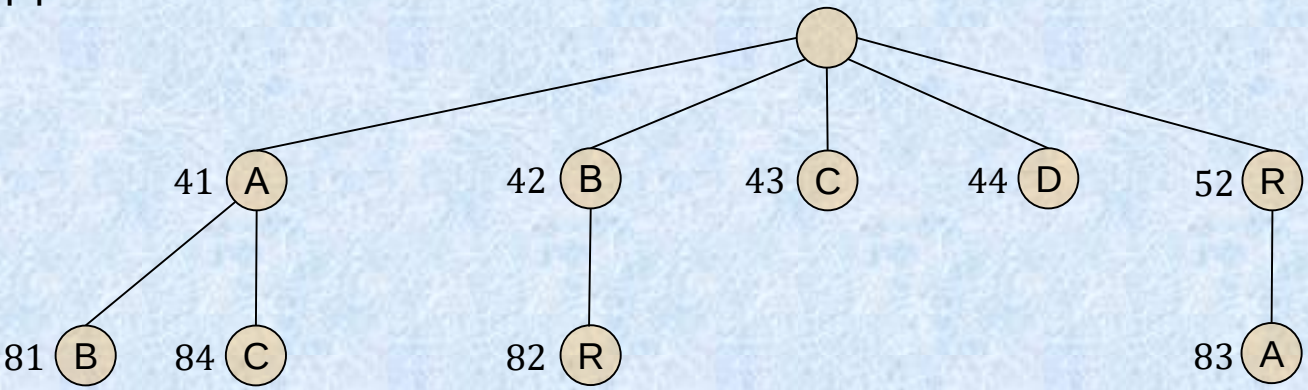
εισαγωγή RA



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C												
κωδικός	41	42	52	41	43												

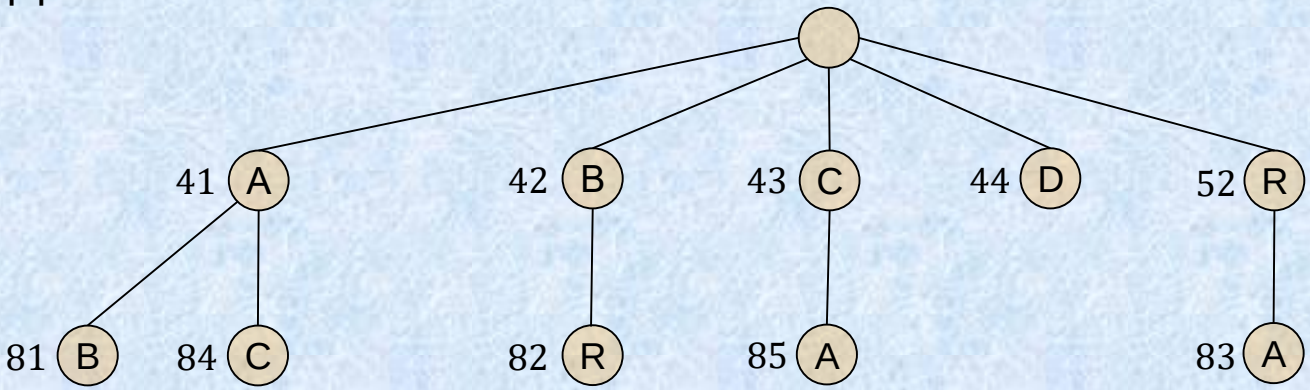
εισαγωγή AC



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C	A											
κωδικός	41	42	52	41	43	41											

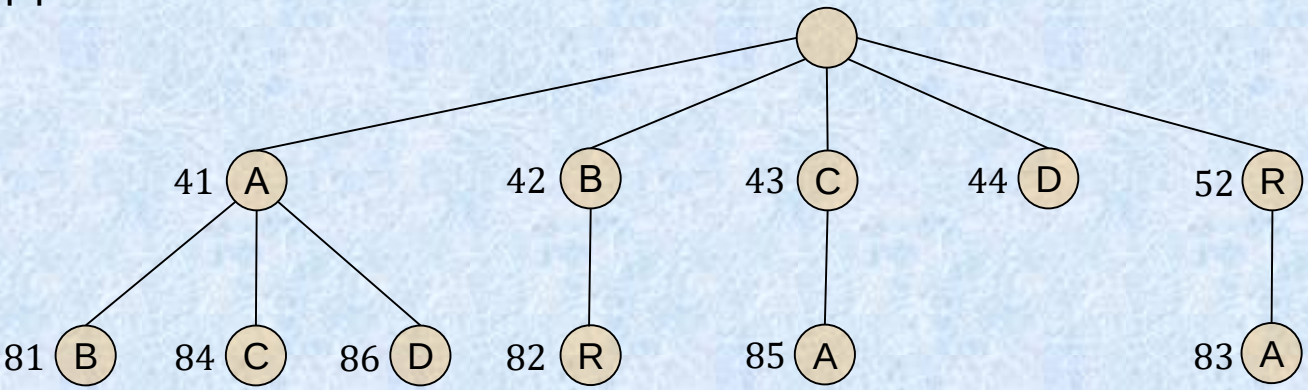
εισαγωγή CA



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C	A	D										
κωδικός	41	42	52	41	43	41	44										

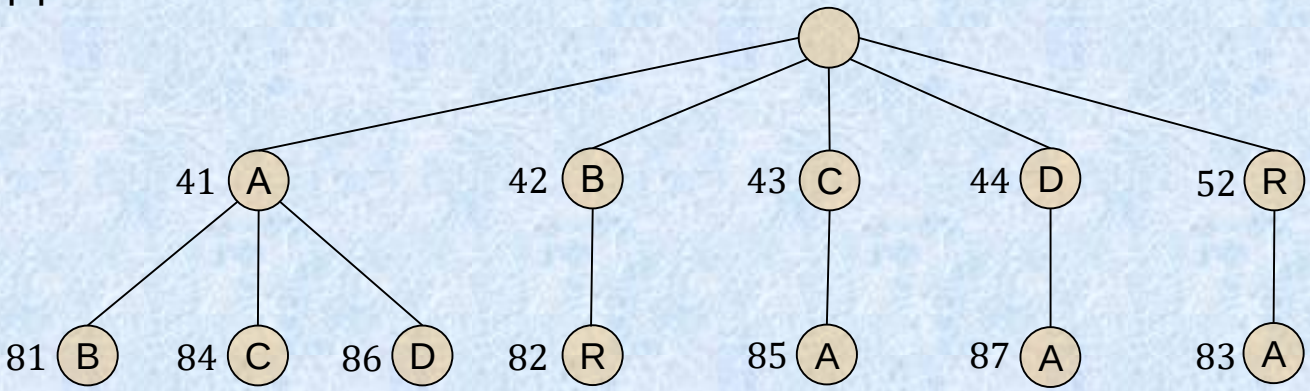
εισαγωγή AD



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C	A	D	AB								
κωδικός	41	42	52	41	43	41	44	81								

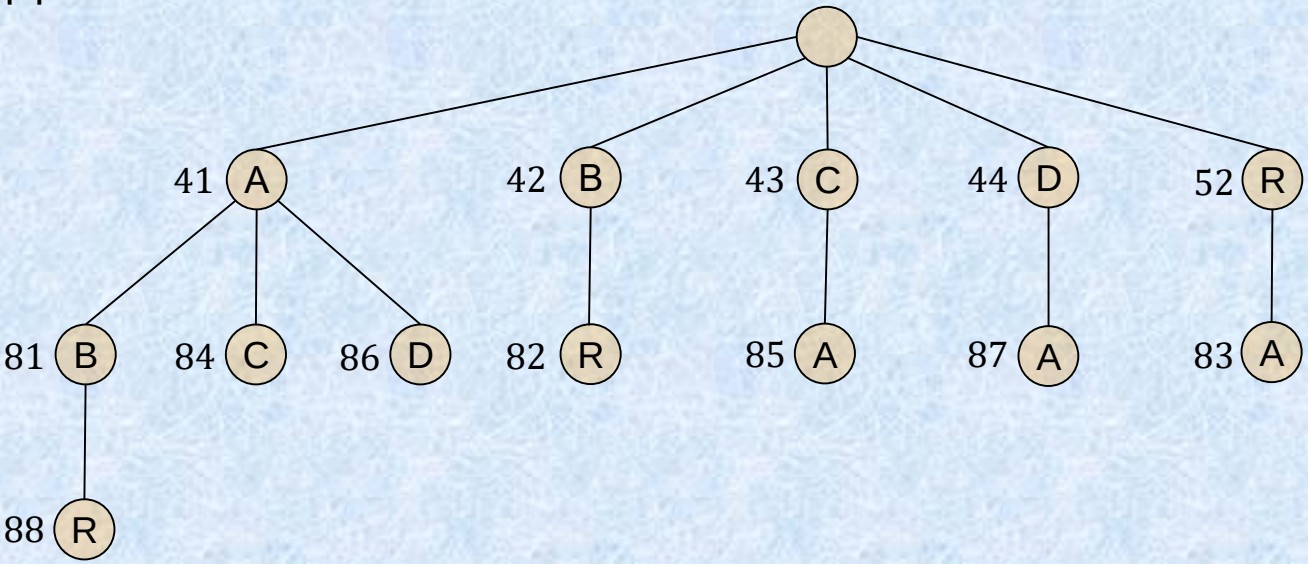
εισαγωγή DA



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C	A	D	AB		RA							
κωδικός	41	42	52	41	43	41	44	81		83							

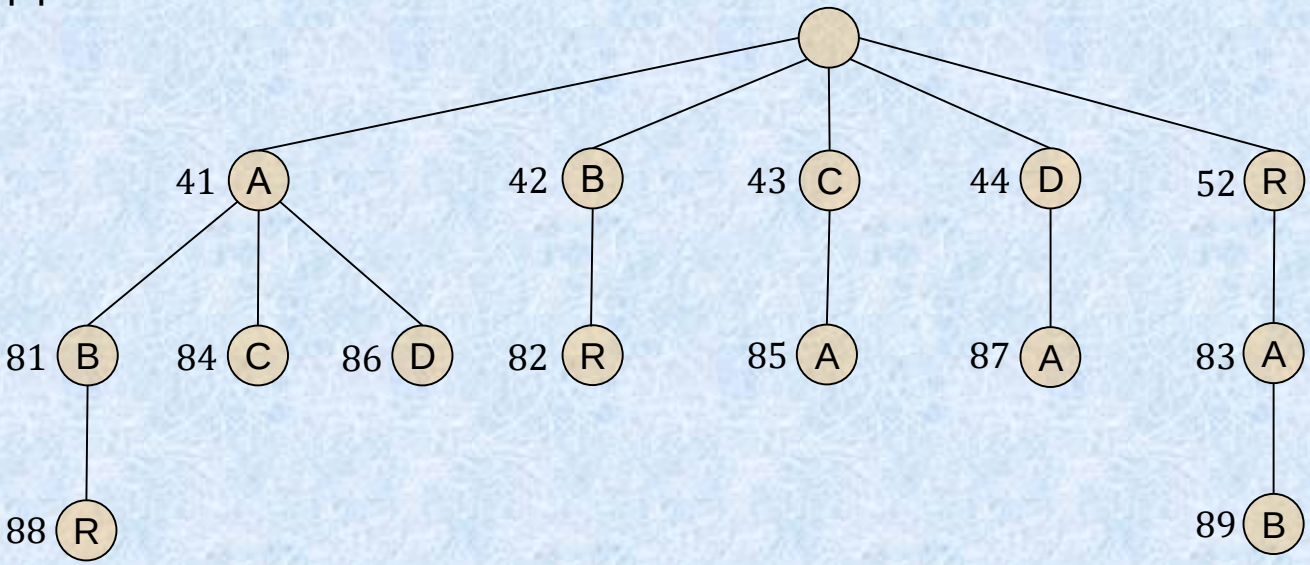
εισαγωγή ABR



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C	A	D	AB		RA		BR				
κωδικός	41	42	52	41	43	41	44	81		83		82				

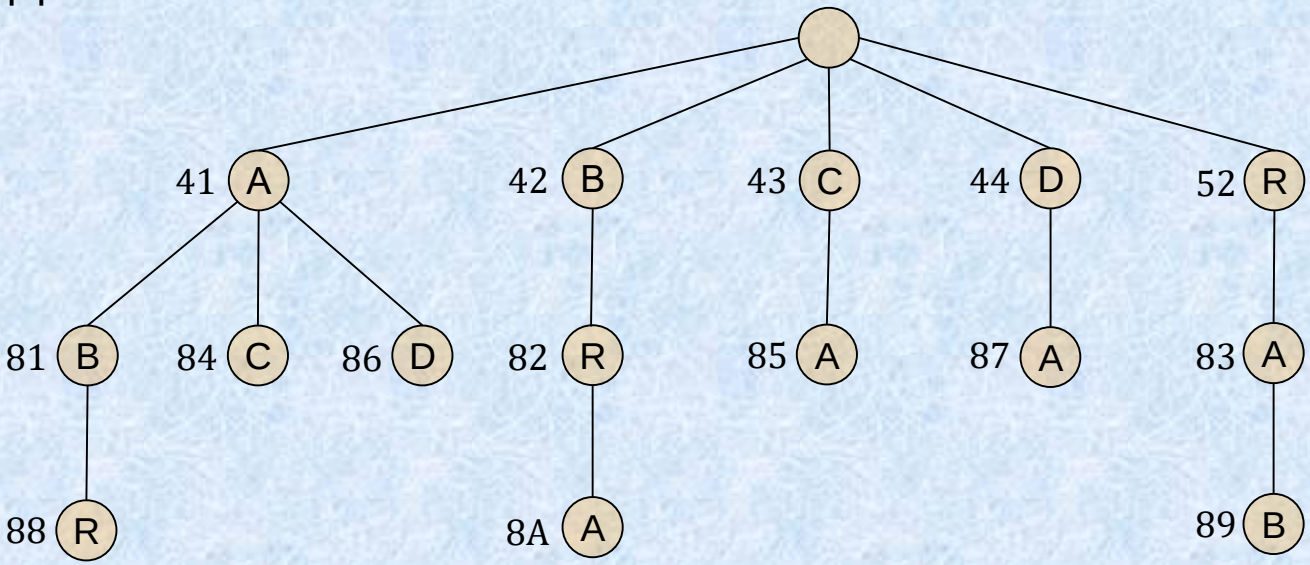
εισαγωγή RAB



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C	A	D	AB		RA		BR		ABR			A
κωδικός	41	42	52	41	43	41	44	81		83		82		88			41

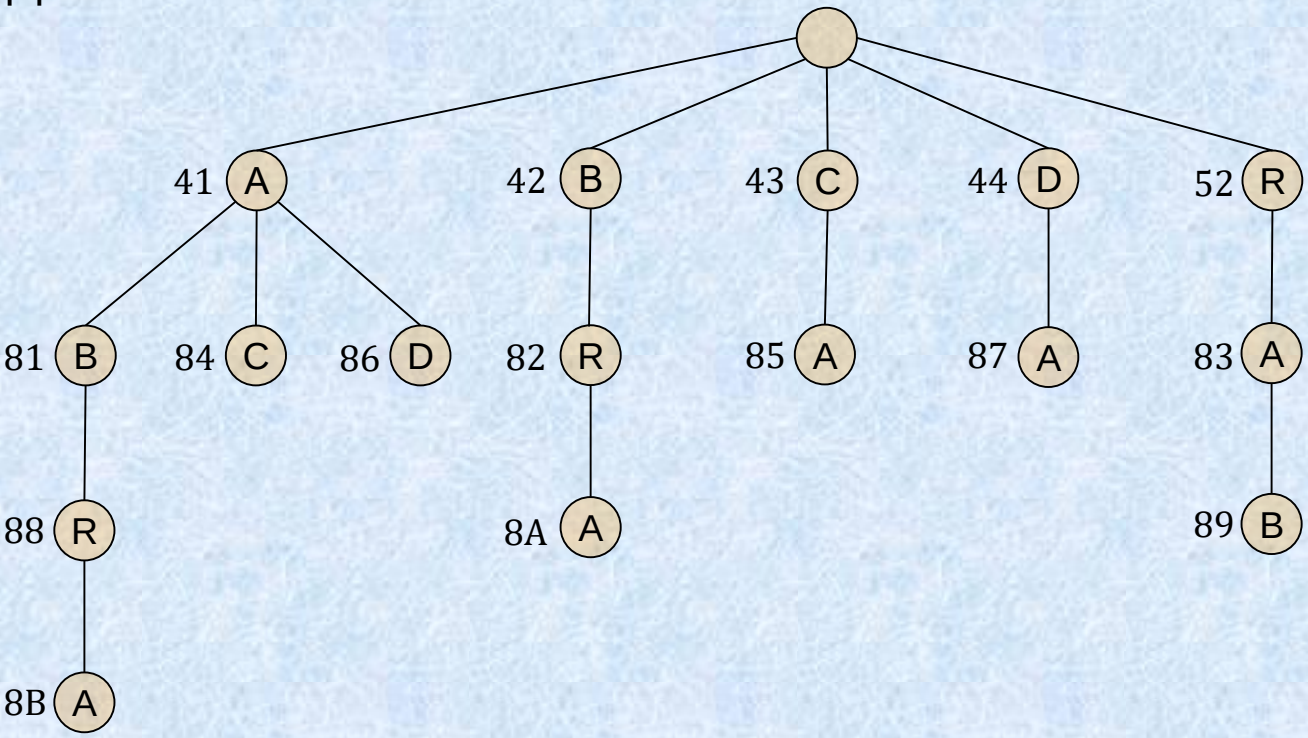
εισαγωγή BRA



Αλγόριθμος Lempel-Ziv-Welch

είσοδος	A	B	R	A	C	A	D	A	B	R	A	B	R	A	B	R	A
ταύτιση	A	B	R	A	C	A	D	AB		RA		BR		ABR			A
κωδικός	41	42	52	41	43	41	44	81		83		82		88			41

εισαγωγή ABRA



Αλγόριθμος Lempel-Ziv-Welch

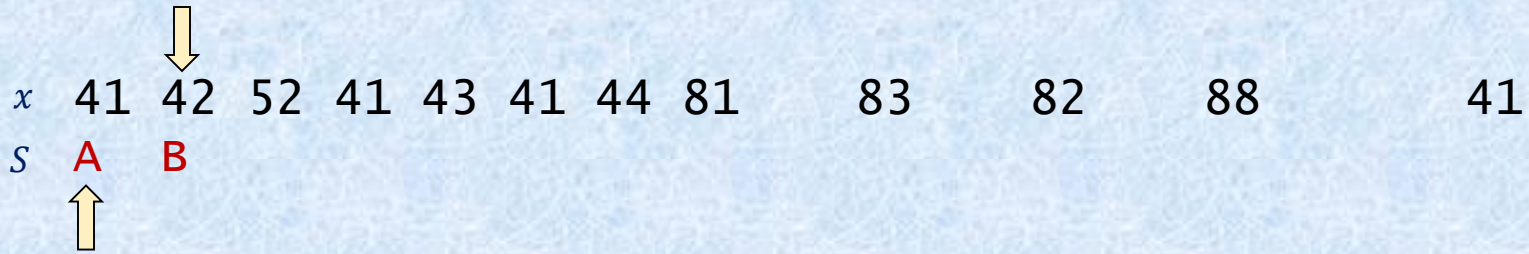
Αποσυμπίεση

Χρησιμοποιούμε ένα λεξικό D' (αντίστροφο του D) το οποίο αντιστοιχεί κωδικούς των W bit σε ακολουθίες χαρακτήρων. Αρχικά το D' περιέχει μόνο τους κωδικούς κάθε ξεχωριστού χαρακτήρα.

Έστω S η τρέχουσα συμβολοσειρά. Αρχικά η S περιέχει το χαρακτήρα που αντιστοιχεί στον πρώτο κωδικό. Εκτελούμε τα παρακάτω βήματα μέχρι να διαβαστούν όλοι οι κωδικοί:

- Τυπώνουμε την τρέχουσα συμβολοσειρά S.
- Διαβάζουμε από την είσοδο τον επόμενο κωδικό x.
- Αναζητούμε στο D' τη συμβολοσειρά S' η οποία αντιστοιχεί στον x.
- Αντιστοιχούμε τον επόμενο διαθέσιμο κωδικό στη συμβολοσειρά $S'c$ όπου c ο πρώτος χαρακτήρας της S.
- Θέτουμε ως τρέχουσα συμβολοσειρά $S \leftarrow S'$.

Αλγόριθμος Lempel-Ziv-Welch

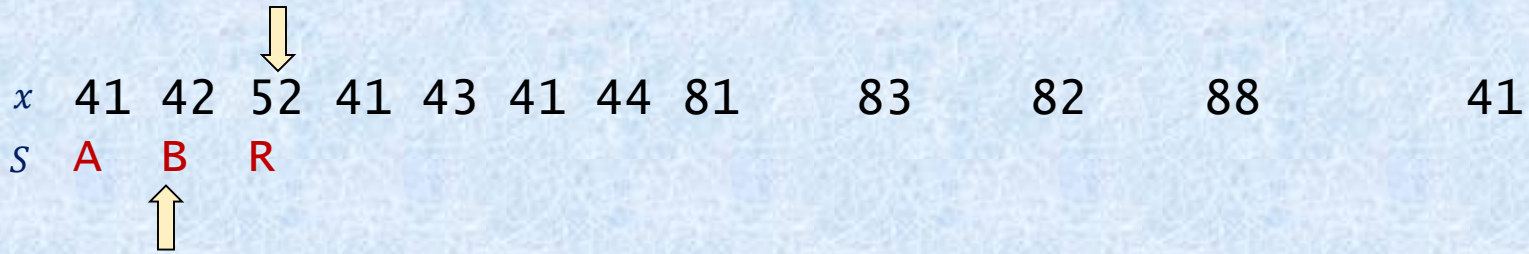


εισαγωγή (81,AB)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	
83	
84	
85	
86	
87	
88	
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

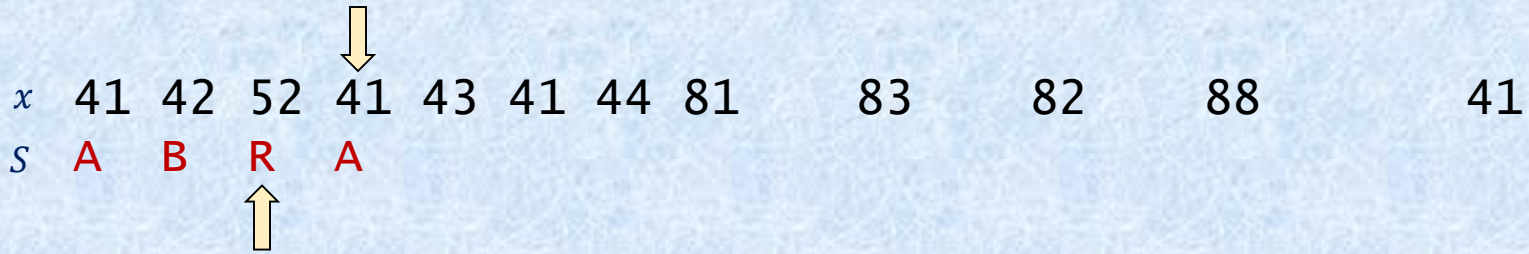


εισαγωγή (82,BR)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	
84	
85	
86	
87	
88	
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

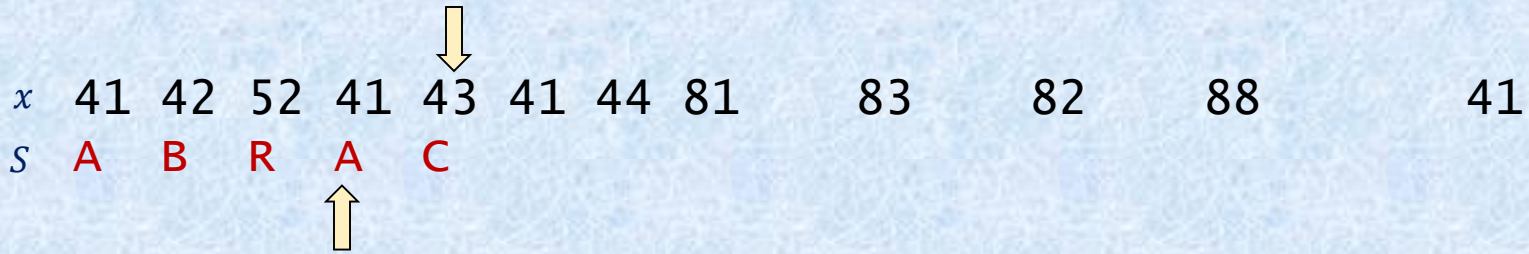


εισαγωγή (83,RA)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	
85	
86	
87	
88	
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

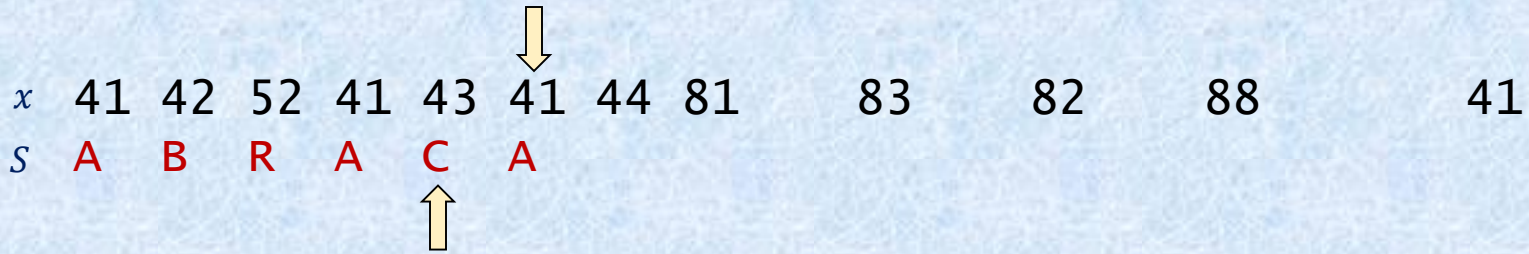


εισαγωγή (84,AC)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	
86	
87	
88	
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

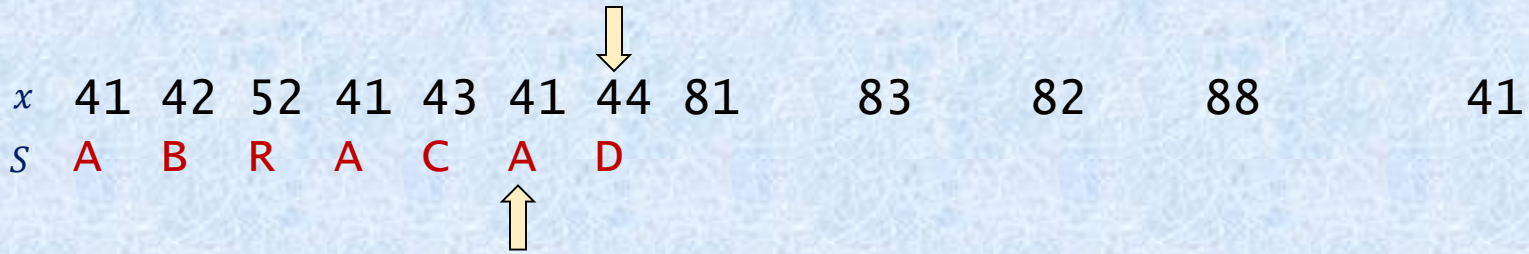


εισαγωγή (85,CA)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	
87	
88	
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

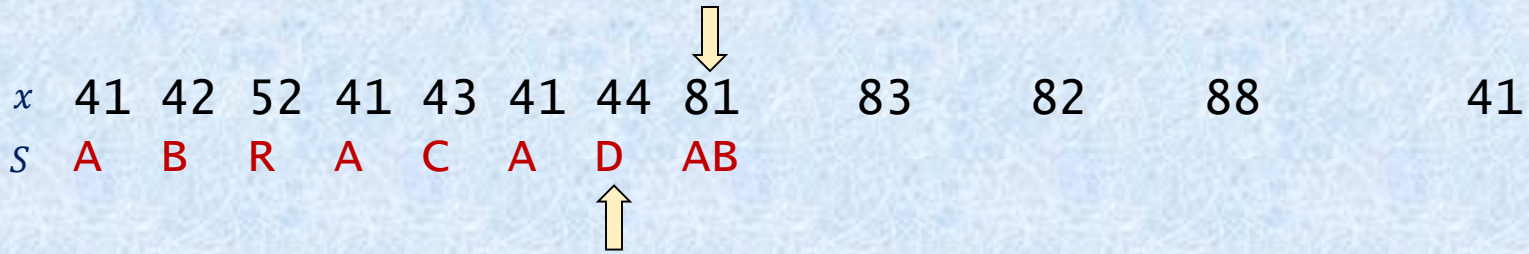


εισαγωγή (86,AD)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	AD
87	
88	
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

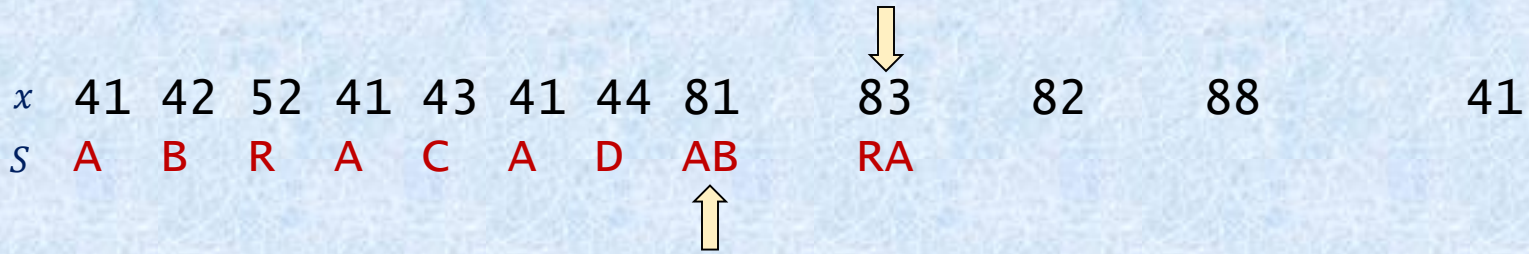


εισαγωγή (87,DA)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	AD
87	DA
88	
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

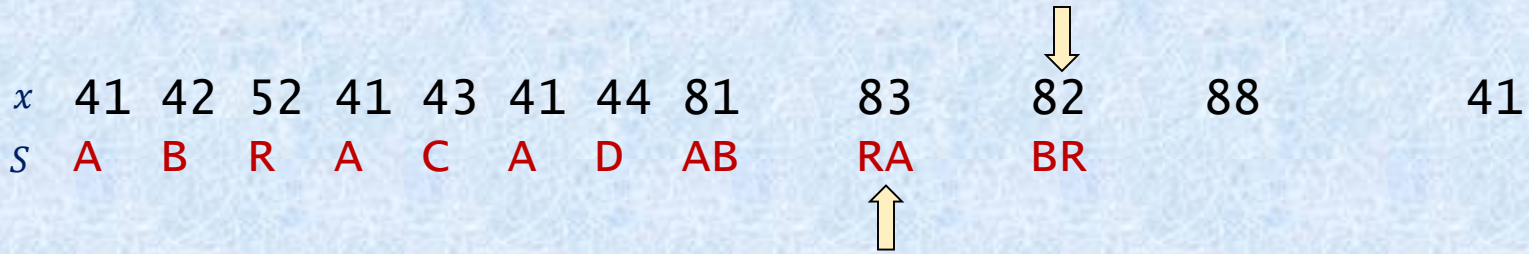


εισαγωγή (88,ABR)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	AD
87	DA
88	ABR
89	
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch

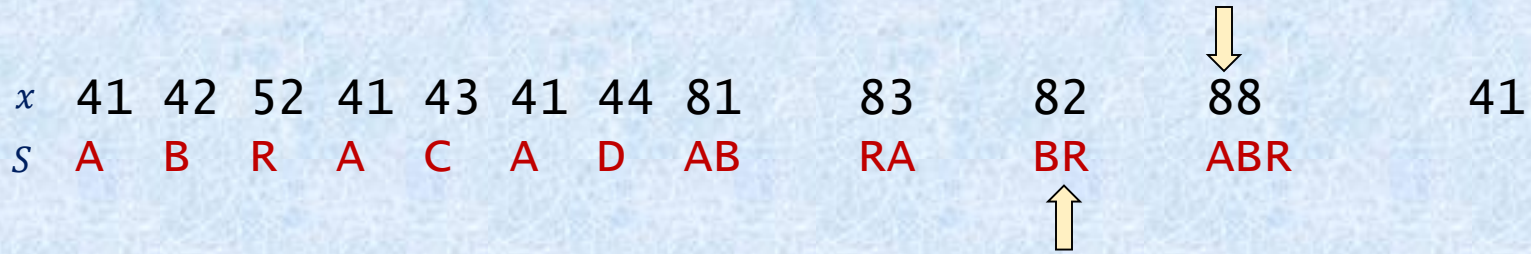


εισαγωγή (89,RAB)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	AD
87	DA
88	ABR
89	RAB
8A	
8B	

Αλγόριθμος Lempel-Ziv-Welch



εισαγωγή (8A,BRA)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	AD
87	DA
88	ABR
89	RAB
8A	BRA
8B	

Αλγόριθμος Lempel-Ziv-Welch

x	41	42	52	41	43	41	44	81	83	82	88	41
S	A	B	R	A	C	A	D	AB	RA	BR	ABR	A

εισαγωγή (8B,ABRA)

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	AD
87	DA
88	ABR
89	RAB
8A	BRA
8B	ABRA

Αλγόριθμος Lempel-Ziv-Welch

x	41	42	52	41	43	41	44	81	83	82	88	41
S	A	B	R	A	C	A	D	AB	RA	BR	ABR	A

κώδικας	συμβολοσειρά
41	A
42	B
43	C
44	D
…	…
52	R

κώδικας	συμβολοσειρά
81	AB
82	BR
83	RA
84	AC
85	CA
86	AD
87	DA
88	ABR
89	RAB
8A	BRA
8B	ABRA

Αλγόριθμος Lempel-Ziv-Welch

Προβληματική περίπτωση

Έστω ότι έχουμε την ακόλουθη συμπίεση

είσοδος	A	B	A	B	A	B	A
ταύτιση	A	B	AB		ABA		
κωδικός	41	42	81		83		

η οποία δίνει τους ακόλουθους κώδικες

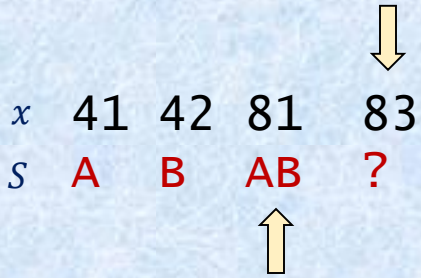
συμβολοσειρά	κώδικας
A	41
B	42

συμβολοσειρά	κώδικας
AB	81
BA	82
ABA	83

Αλγόριθμος Lempel-Ziv-Welch

Προβληματική περίπτωση

Στην αποσυμπίεση συναντάμε το ακόλουθο πρόβλημα



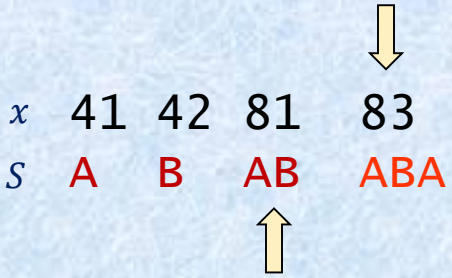
Χρειαζόμαστε τη συμβολοσειρά με κωδικό 83 προτού εισαχθεί στον πίνακα!

κώδικας	συμβολοσειρά
41	A
42	B

κώδικας	συμβολοσειρά
81	AB
82	BA
83	

Αλγόριθμος Lempel-Ziv-Welch

Προβληματική περίπτωση



κώδικας	συμβολοσειρά
41	A
42	B

κώδικας	συμβολοσειρά
81	AB
82	BA
83	

- Συμβαίνει όταν ο επόμενος κώδικας εισόδου είναι αυτός που πρέπει να αναθέσουμε στην επόμενη συμβολοσειρά $S'c$ την οποία αποθηκεύουμε στο D'.

- Μπορεί να διορθωθεί εύκολα: ο χαρακτήρας c πρέπει να είναι ίδιος με τον πρώτο χαρακτήρα της S'.

Αλγόριθμος Lempel-Ziv-Welch

Πρακτικές Υλοποιήσεις

- Για τη συμπίεση μπορούμε να υλοποιήσουμε το λεξικό ως trie το οποίο επιτρέπει την αναζήτηση του μεγαλύτερου προθήματος (`longestPrefixOf`).
- Για την αποσυμπίεση αρκεί ένας πίνακας μεγέθους 2^W.

Η διαχείριση του λεξικού μπορεί να είναι προβληματική. Έχουν προταθεί διάφοροι τρόποι αντιμετώπισης, π.χ.:

- Όταν το λεξικό γίνει πολύ μεγάλο το διαγράφουμε και συνεχίζουμε με ένα νέο λεξικό (GIF)
- Όταν το λεξικό είναι αναποτελεσματικό το διαγράφουμε και συνεχίζουμε με ένα νέο λεξικό (Unix compress)
- Όταν το λεξικό φτάσει ένα ανώτατο όριο μεγέθους διαγράφουμε τα στοιχεία που χρησιμοποιήθηκαν λιγότερο πρόσφατα (British telecom standard)

Αλγόριθμοι συμπίεσης χωρίς απώλειες

year	scheme	bits / char
1967	ASCII	7.00
1950	Huffman	4.70
1977	LZ77	3.94
1984	LZMW	3.32
1987	LZH	3.30
1987	move-to-front	3.24
1987	LZB	3.18
1987	gzip	2.71
1988	PPMC	2.48
1994	SAKDC	2.47
1994	PPM	2.34
1995	Burrows-Wheeler	2.29
1997	BOA	1.99
1999	RK	1.89

data compression using Calgary corpus