

**Αλγόριθμοι και Στοιχεία Πολυπλοκότητας.**

**Ενότητα 9:** Δέντρα.

Διδάσκων: Ηλίας Κ Σάββας, Αναπληρωτής Καθηγητής.

Τμήμα Μηχανικών Πληροφορικής, Τεχνολογικής Εκπαίδευσης.

**Άδειες χρήσης.**

* Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons (C C). **Αναφορά δημιουργού (B Y), Μη εμπορική χρήση (N C), Μη τροποποίηση (N D), 3.0, Μη εισαγόμενο.**
* Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



**Χρηματοδότηση.**

* Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
* Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.



# Σκοποί ενότητας.

Ο αναγνώστης να μπορεί να:

1) αντιληφθεί την έννοια της δομής δέντρου,

2) επιλύει προβλήματα σε δυαδικά δέντρα αναζήτησης.

# ΠΕΡΙΕΧΌΜΕΝΑ ΕΝΌΤΗΤΑΣ.

[Σκοποί ενότητας. 2](#_Toc368164452)

[ΠΕΡΙΕΧΌΜΕΝΑ ΕΝΌΤΗΤΑΣ. 3](#_Toc368164453)

[ΣΧΉΜΑΤΑ. 3](#_Toc368164454)

[ΠΊΝΑΚΕΣ. 3](#_Toc368164455)

[9. ΔΈΝΤΡΑ. 4](#_Toc368164456)

[9.1 Εισαγωγικές έννοιες – Ορισμοί. 4](#_Toc368164457)

[9.2 Διαπεράσεις δυαδικού δέντρου. 6](#_Toc368164458)

[9.3 Υλοποίηση δυαδικού δέντρου με πίνακα. 6](#_Toc368164459)

[9.4 Δυαδικά δέντρα αναζήτησης. 8](#_Toc368164460)

[9.5 Κώδικας Huffman. 10](#_Toc368164461)

# ΣΧΉΜΑΤΑ.

|  |  |
| --- | --- |
| ***Σχήμα 9.1****: Δυαδικό δέντρο.* | 5 |
| ***Σχήμα 9.2****:**Δυαδικό δέντρο.* | 9 |
| ***Σχήμα 9.3****:**Εισαγωγή στοιχείου.* | 9 |
| ***Σχήμα 9.4****: Διαγραφή στοιχείου.* | 9 |
| ***Σχήμα 9.5****: Κωδικοποίηση μεταβλητού μήκους.* | 12 |

# ΠΊΝΑΚΕΣ.

|  |  |
| --- | --- |
| ***Πίνακας 9.1****: Κωδικοποίηση σταθερού μήκους.* | 10 |
| ***Πίνακας 9.2****: Κωδικοποίηση μεταβλητού μήκους.* | 11 |

# 9. ΔΈΝΤΡΑ.

## 9.1 Εισαγωγικές έννοιες – Ορισμοί.

***Δέντρο*** είναι ένα σύνολο κόμβων (ίδιου τύπου) και ακμών, που συνδέουν τους κόμβους, με βάση κάποια σχέση που δημιουργεί την ιεραρχική δομή των κόμβων. Ένας από τους κόμβους αποτελεί και την ***ρίζα*** του δέντρου. Η δομή του δέντρου αναδρομικά ορίζεται ως εξής:

* + Ένας κόμβος αποτελεί από μόνος του ένα δέντρο. Μάλιστα αυτός ο κόμβος αποτελεί και την ρίζα του δέντρου.
	+ Εάν *n* είναι ένας κόμβος και $T\_{1}, T\_{2}, …,T\_{k}$, είναι δέντρα με ρίζες $n\_{1 },n\_{2},…, n\_{k}$, αντίστοιχα, τότε μπορούμε να κατασκευάσουμε ένα νέο δέντρο, με κόμβο γονέα τον *n* των κόμβων, $n\_{1 },n\_{2},…, n\_{k}$. Έτσι, ο κόμβος *n* αποτελεί την ρίζα του νέου δέντρου, τα $T\_{1}, T\_{2}, …,T\_{k}$, αποτελούν τα ***υπό-δέντρα*** της ρίζας, και οι κόμβοι, $n\_{1 },n\_{2},…, n\_{k}$, τα ***παιδιά*** του κόμβου *n*.

***Κενό*** είναι το δέντρο που δεν περιέχει καθόλου κόμβους και καμία ακμή.

***Εσωτερικοί***, καλούνται οι κόμβοι στους οποίους μπορούν και να καταλήγουν, αλλά και να ξεκινούν ακμές. Στην ρίζα δεν καταλήγουν ακμές. Οι κόμβοι στους οποίους μόνο καταλήγουν ακμές, ονομάζονται ***φύλλα*** (ή και τερματικοί κόμβοι). Τα ***υπό-δέντρα*** σχηματίζονται, αρκεί να θεωρήσουμε σαν ρίζα οποιονδήποτε κόμβο του δέντρου.

Εάν $n\_{1 },n\_{2},…, n\_{k}$, είναι μια σειρά κόμβων ενός δέντρου τέτοια ώστε ο κόμβος $n\_{i}$, να είναι ο γονέας του $n\_{i+1}$, για , τότε η ακολουθία αυτή ονομάζεται ***διαδρομή*** (path) από τον $n\_{1}$ κόμβο, προς τον κόμβο $n\_{k }$. ***Μήκος*** (length) μίας διαδρομής, ονομάζεται το σύνολο των περιεχομένων ακμών, (που ισούται με το πλήθος των περιεχομένων κόμβων μείον ένα).

***Ύψος*** (height) κόμβου ενός δέντρου, είναι η μεγαλύτερη διαδρομή από τον κόμβο προς κάποιο φύλλο. Ύψος του δέντρου είναι το ύψος της ρίζας του. Τα φύλλα ενός δέντρου έχουν ύψος μηδέν.

***Επίπεδο*** (level) ενός κόμβου, είναι το μήκος της μοναδικής διαδρομής από την ρίζα προς αυτόν τον κόμβο. Η ρίζα κάθε δέντρου βρίσκεται στο μηδενικό επίπεδο. Ένα κενό δέντρο έχει ύψος -1.

***Βαθμός*** (degree) ενός κόμβου είναι το πλήθος των παιδιών του.

Ειδικές κατηγορίες δέντρων αποτελούν τα ***δυαδικά*** δέντρα, τα ***τριαδικά*** δέντρα, και τα λοιπά.

**ΠΑΡΆΔΕΙΓΜΑ.**



***Σχήμα 9.1****: Δυαδικό δέντρο.*

*Διαδρομή* από *n2* προς *n9,* είναι η ακολουθία, $n\_{2}, n\_{4},n\_{9}$.

Το *μήκος* της διαδρομής αυτής είναι 2.

*Φύλλα* είναι, τα $n\_{8}, n\_{9},n\_{5}, n\_{6}, n\_{7}$.

Το *ύψος* του δέντρου είναι 3, ενώ ύψος του κόμβου *n2* είναι 2, και του *n9* μηδέν.

Ο *βαθμός* του κόμβου *n9,* είναι μηδέν, ενώ του κόμβου *n2* είναι 2.

Επίπεδο του κόμβου *n2* είναι ένα, και του *n9* είναι τρία.

***Δυαδικά Δέντρα***, είναι τα δέντρα που ο κάθε κόμβος μπορεί να έχει βαθμό το πολύ δύο, δηλαδή ένα αριστερό και ένα δεξιό παιδί. Εάν όλοι οι κόμβοι έχουν από δύο παιδιά, πλην ίσως αυτών του τελευταίου επιπέδου, αλλά και εδώ, όλοι οι κόμβοι να βρίσκονται όσο πιο αριστερά, τότε το δυαδικό δέντρο ονομάζεται ***πλήρες***, (πλήρες είναι το δυαδικό δέντρο του *Σχήματος 9.1*).

Σε ένα δυαδικό δέντρο, στο επίπεδο μηδέν υπάρχουν = 1 κόμβοι. Στο επίπεδο 1 το σύνολο των κόμβων είναι . Επομένως το μέγιστο δυνατό σύνολο των κόμβων ενός πλήρους δυαδικού δέντρου ύψους *k*, είναι . Επομένως το δέντρο του *Σχήματος 9.1*, θα μπορούσε να έχει το πολύ κόμβους , (ύψος 3).

***Πρόταση***: Για ένα δυαδικό δέντρο *n* κόμβων, και ύψους *k*, ισχύει **.**

*Απόδειξη*:

Ο ελάχιστος αριθμός κόμβων σε ένα πλήρες δυαδικό δέντρο ύψους k, είναι , εάν υποθέσουμε ότι στη χειρότερη περίπτωση έχει συμπληρωμένους όλους τους κόμβους μέχρι το επίπεδο k-1, και έχει μόνο ένα κόμβο στο επίπεδο k.

Τότε το πλήθος αυτό είναι:

 .

Επίσης, αφού το μέγιστο πλήθος κόμβων αυτού του δέντρου είναι:

 

.

## 9.2 Διαπεράσεις δυαδικού δέντρου.

Διαπέραση δέντρου είναι η διαδικασία επίσκεψης των κόμβων του. Εάν θεωρήσουμε τις διαδικασίες:

1. Επίσκεψη ρίζας δέντρου.
2. Επίσκεψη αριστερού υπό-δέντρου, και
3. επίσκεψη δεξιού υπό-δέντρου,

τότε, διαπέραση σημαίνει την επίσκεψη κάθε κόμβου, μία φορά κατά μία συγκεκριμένη σειρά. Έτσι διακρίνουμε τους παρακάτω τρόπους διαπέρασης:

* ***Προ-διατεταγμένη*** (preorder), όπου η σειρά είναι 1 – 2 – 3,
* ***Ενδο-διατεταγμένη*** (inorder), όπου η σειρά είναι 2 – 1 – 3, και τέλος,
* ***Μετά-διατεταγμένη*** (postorder), όπου η σειρά είναι 2 – 3 – 1.

## 9.3 Υλοποίηση δυαδικού δέντρου με πίνακα.

Έστω *T(n)*, ο πίνακας που θα αναπαραστήσει ένα δυαδικό δέντρο ύψους *k*, και πλήθους κόμβων *n*. Κατ’ αρχήν πρέπει , δηλαδή, το πλήθος των στοιχείων του πίνακα, να είναι όσο και το μέγιστο πλήθος των κόμβων του δέντρου.

Στο πρώτο στοιχείο του πίνακα τοποθετείται η ρίζα του δέντρου. Στη συνέχεια, και στις επόμενες θέσεις του πίνακα, τοποθετούνται με την σειρά, πρώτα το αριστερό παιδί και μετά το δεξιό. Για παράδειγμα, οι κόμβοι του δέντρου του *Σχήματος 9.1*, θα τοποθετούταν ως εξής: $[n\_{1}, n\_{2}, n\_{3}, n\_{4},n\_{5}, n\_{6}, n\_{7}, n\_{8}, n\_{9}]$.

Τώρα, το κυρίαρχο θέμα που προκύπτει για το κάθε στοιχείο με δείκτη *i* του πίνακα *T*,είναι, ποιό είναι το αριστερό και ποιό το δεξί παιδί του, καθώς και ποιος είναι ο γονέας του. Παρατηρούμε τα ακόλουθα:

* Αν *i* = 1, τότε είναι η ρίζα του δέντρου.
* Αν , τότε ο γονέας του είναι το στοιχείο με δείκτη, το ακέραιο πηλίκο $\frac{i}{2}$ .
* Αν , τότε το στοιχείο αποτελεί φύλλο του δέντρου, (δεν υπάρχουν παιδιά).
* Αν , τότε αριστερό παιδί του είναι το 2*i*, και δεξί του το 2*i* + 1.

**Αναδρομικός Αλγόριθμος Προ-διατεταγμένης Διαπέρασης Δυαδικού Δέντρου.**

(Τ = Δέντρο, n = πλήθος κόμβων).

ΠΔΔΔ(r),

root = r.

Εάν root <=n τότε,

Επίσκεψη(Τ(root)),

 left = 2r, /\* Αριστερό παιδί \*/,

ΠΔΔΔ(left),

 right = 2r + 1, /\* Δεξιό παιδί \*/,

 ΠΔΔΔ(right),

Τέλος Εάν.

Τέλος ΠΔΔΔ.

**Υλοποίηση σε Pascal,**

program tree1;

const n=9;

var t:array[1..n] of char; {Δυαδικό Δέντρο}

i: integer;

procedure pddd(r: integer);

{Προδιατεταγμένη Διαπέραση}

var root, left, right: integer;

begin

root: = r;

if root <= n then

begin

write(t[root],' ');

left: = 2 \* r; pddd(left);

right: = 2 \* r + 1; pddd(right);

end;

end; {pddd}

begin {main}

for i: =1 to n do

begin

write(i:5,'-->');

readln(t[i]);

writeln;

end;

writeln; writeln;

pddd(1);

end.

**Υλοποίηση σε C.**

#include <stdio.h>

#define N 10

void pddd(int r);

int T[N];

main()

{

 int i=1;

 do {

printf("\n\n Input node -->");

scanf("%d", &T[i]);

i++;

 } while (i < N);

 printf("\n\n\n Preorder: ");

 pddd(1);

 printf("\n\n\n");

}

void pddd(int r)

{

 int root=r, left, right, node;

 if (root < N) {

node = T[root];

printf(" %d ", node);

left = 2 \* root;

pddd(left);

right = 2 \* root + 1;

pddd(right);

 }

}

##

## 9.4 Δυαδικά δέντρα αναζήτησης.

***Δυαδικό Δέντρο Αναζήτησης (binary search trees)***,είναι ένα δυαδικό δέντρο με κόμβους διατεταγμένους με τέτοιο τρόπο, ώστεη τιμή του στοιχείου του κάθε κόμβου, να είναι μεγαλύτερη από τις τιμές όλων των στοιχείων των κόμβων του αριστερού του υπό-δέντρου, και μικρότερη από αυτές των στοιχείων του δεξιού του υπό-δέντρου. Τέτοιο δέντρο αποτελεί το δέντρο του *Σχήματος 9.1*.

***Βασικό πλεονέκτημα*** αυτής της δομής, είναι ότι συνδυάζει γρήγορες αναζητήσεις, ενώ παράλληλα, επιτρέπει την εύκολη εισαγωγή και διαγραφή στοιχείων.

* ***Αλγόριθμος Αναζήτησης***:
	+ Σύγκριση του στοιχείου της ρίζας με το υπό αναζήτησης στοιχείο. Εάν είναι ίσα, τότε ο αλγόριθμος τελειώνει (επιτυχής αναζήτηση).
		- Εάν το υπό αναζήτηση στοιχείο είναι μεγαλύτερο, τότε συνεχίζουμε με το δεξιό υπό-δέντρο, αλλιώς με το αριστερό, (θέτοντας σαν ρίζα την ρίζα του αριστερού ή δεξιού υπό-δέντρου αντίστοιχα).
	+ Το προηγούμενο βήμα επαναλαμβάνεται, μέχρι ή να βρεθεί το στοιχείο ή την συνάντηση κενού υπό-δέντρου (ανεπιτυχής αναζήτηση).
* ***Αλγόριθμος Εισαγωγής***:
	+ Σύγκριση του στοιχείου της ρίζας με το υπό εισαγωγή στοιχείο. Εάν είναι ίσα, τότε ο αλγόριθμος τελειώνει (δεν υπάρχει λόγος εισαγωγής).
		- Εάν το υπό εισαγωγή στοιχείο είναι μεγαλύτερο, τότε συνεχίζουμε με το δεξιό υπό-δέντρο, αλλιώς με το αριστερό, (θέτοντας σαν ρίζα την ρίζα του αριστερού ή δεξιού υπό-δέντρου αντίστοιχα).
	+ Το προηγούμενο βήμα επαναλαμβάνεται μέχρι την συνάντηση κενού υπό-δέντρου, και εισάγεται σε αυτό ακριβώς το σημείο, δηλαδή σαν ρίζα αυτού του κενού υπό-δέντρου.
* Ο αλγόριθμος εισαγωγής, δημιουργεί το δυαδικό δέντρο αναζήτησης, το οποίο αν προσπελασθεί με την ενδο-διατεταγμένη διάταξη, εμφανίζει ταξινομημένα τα περιεχόμενα-κόμβους του. Αυτή η μέθοδος ταξινόμησης, ονομάζεται **heap sort**, και η πολυπλοκότητά της είναι, , δηλαδή όσο και της quick sort.
* ***Αλγόριθμος Διαγραφής***:
	+ Κατ’ αρχήν, εκτελείται ο αλγόριθμος αναζήτησης, και στην περίπτωση μη εύρεσης του προς διαγραφή στοιχείου, ο αλγόριθμος τελειώνει. Εάν όμως βρεθεί στον κόμβο *ni* τότε:
	+ Εάν ο *ni* δεν έχει παιδιά απλώς διαγράφεται, και την θέση του καταλαμβάνει ένα κενό υπό-δέντρο.
	+ Εάν ο *ni* έχει μόνο ένα παιδί, τότε ο *ni* διαγράφεται, και την θέση του καταλαμβάνει το μοναδικό παιδί του.
	+ Εάν ο *ni* έχει δύο παιδιά, τότε:
		- Σύμφωνα με την ενδο-διατεταγμένη διάταξη (in order), βρίσκεται ο αμέσως επόμενος κόμβος του *ni*, έστω *nj*, και διαγράφεται ο *ni*, και τη θέση του καταλαμβάνει ο *nj*. (Φυσικά διαγράφεται και ο n*j* από την παλιά του θέση).

**ΠΑΡΆΔΕΙΓΜΑ.**



***Σχήμα 9.2****:**Δυαδικό δέντρο.*

*Αναζήτηση* του 4: $n\_{1}, n\_{2}, n\_{4}, n\_{9}$ .

*Εισαγωγή* του 59: $n\_{1}, n\_{3}, n\_{6}$, και στη συνέχεια εισαγωγή σαν δεξιό υπό-δέντρο του $n\_{6}$. Δηλαδή,

 

***Σχήμα 9.3****:**Εισαγωγή στοιχείου.*

*Διαγραφή* του 17: $n\_{1}, n\_{2}, n\_{5}$, και στη συνέχεια διαγραφή του $n\_{2}$, και τοποθέτηση του $n\_{5}$, στη θέση του. Δηλαδή,



***Σχήμα 9.4****: Διαγραφή στοιχείου.*

* **ΠΟΛΥΠΛΟΚΌΤΗΤΑ.** Η πολυπλοκότητα και των τριών αλγορίθμων, ουσιαστικά εξαρτάται από την πολυπλοκότητα της αναζήτησης. Εάν υποτεθεί ότι το ύψος του δέντρου είναι *k*, τότε ισχύει ότι **.** Επομένως, και ο αριθμός αναζητήσεων , περιορίζεται από την σχέση ****. Οπότε η πολυπλοκότητα εκφράζεται με τον τύπο **,** και με μέση συνάρτηση πολυπλοκότητας την ****.

## 9.5 Κώδικας Huffman.

Ένας από τους πιο διαδεδομένους κώδικες αναπαράστασης χαρακτήρων, είναι ο κωδικός ASCII, (American Standard Code for Information Interchange). Ο ASCII, όπως και οι περισσότεροι κώδικες, αναπαριστούν όλους τους χαρακτήρες με σταθερό αριθμό από bits. Αυτός ο τρόπος κωδικοποίησης είναι ιδιαίτερα αποτελεσματικός, γιατί κάθε συμβολοσειρά από bits μπορεί εύκολα να διασπασθεί στους επιμέρους χαρακτήρες του, αλλά από την άλλη μεριά, έχει το μειονέκτημα ότι δεν δημιουργεί συμβολοσειρές ελάχιστου μήκους. Για παράδειγμα, ας υποτεθεί ότι υπάρχει ένα αλφάβητο μόνο τεσσάρων γραμμάτων, έστω <α, β, γ, δ>. Με μία σταθερή αναπαράσταση, θα απαιτούταν 2 bits. Έστω:

α 🡪 00,

β 🡪 01,

γ 🡪 10,

δ 🡪 11.

Επομένως, για την κωδικοποίηση της συμβολοσειράς <α α β α δ β α γ β α> θα χρειαζόταν 2\*10 = 20 bits, <α α β α δ β α γ β α> 🡺 <00000100110100100100> όπως φαίνεται στον *Πίνακα 9.1*.

***Πίνακας 9.1****: Κωδικοποίηση σταθερού μήκους.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| α | α | β | α | δ | β | α | γ | β | α |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

Η ελαχιστοποίηση όμως του μήκους των συμβολοσειρών, αποτελεί ένα σημαντικό πρόβλημα, όπου κωδικοποιήσεις σταθερού μήκους δεν προσφέρονται για την επίλυσή του, σε αντίθεση με τις κωδικοποιήσεις Huffman. Μία σημαντική παρατήρηση, η οποία οφείλεται στους Άραβες του ενάτου μΧ αιώνα, και η οποία αρχικά χρησιμοποιήθηκε για την αποκρυπτογράφηση κειμένων, ήταν το εφαλτήριο των κωδικοποιήσεων Huffman, (πρώτος που αντιλήφθηκε ότι η συχνότητα εμφάνισης των γραμμάτων, μπορεί να χρησιμοποιηθεί και σαν εργαλείο, για το σπάσιμο των κρυπτογραφικών κωδικών, ήταν ο Άραβας επιστήμονας Αλ-Κιντί, γνωστός και σαν «φιλόσοφος των Αράβων», και ο οποίος έγραψε 290 επιστημονικά βιβλία). Σε κάθε γλώσσα, η εμφάνιση των γραμμάτων της αλφαβήτου δεν είναι ισόποση. Υπάρχουν γράμματα που η συχνότητα εμφάνισής τους είναι μεγάλη, αλλά και γράμματα που η αντίστοιχη συχνότητά τους είναι μικρή. Επομένως, αν κωδικοποιηθούν τα γράμματα με μεγάλη συχνότητα με μικρό μήκος, ενώ τα γράμματα με μικρή συχνότητα με μεγαλύτερο μήκος, τότε ίσως, το μήκος της τελικής συμβολοσειράς, είναι μικρότερο από αυτό της κωδικοποίησης σταθερού μήκους. Για παράδειγμα, έστω ότι η συχνότητα εμφάνισης των γραμμάτων στην αλφάβητο <α, β, γ, δ> είναι:

α 🡪 60%, β 🡪 30%, γ 🡪 5%, και δ🡪 5%.

Αυτό δείχνει ότι πρέπει να χρησιμοποιηθεί το μικρότερο μήκος κωδικοποίησης για το <α> και το μεγαλύτερο για τα <γ, δ>. Έστω, ότι η κωδικοποίηση είναι η ακόλουθη:

α 🡪 0,

β 🡪 10,

γ 🡪 110,

δ 🡪 111.

Τότε, η συμβολοσειρά <α α β α δ β α γ β α> με την νέα μορφή κωδικοποίησης, θα έπαιρνε την μορφή που φαίνεται στον *Πίνακα 9.2*. Επομένως, θα απαιτούσε 17 bits για την κωδικοποίησή της, αντί των 20 bits της σταθερής κωδικοποίησης, δηλαδή προσφέρει ένας κέρδος της τάξης του 15%.

***Πίνακας 9.2****: Κωδικοποίηση μεταβλητού μήκους.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| α | α | β | α | δ | β | α | γ | β | α |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |

Γενικεύοντας, το όφελος σε μια συμβολοσειρά *Ν* χαρακτήρων, μπορεί εύκολα να προσδιορισθεί από τις συχνότητες των χαρακτήρων, και τα μήκη που τους αντιστοιχήθηκαν. Το αναμενόμενο μήκος μίας συμβολοσειράς *Ν* χαρακτήρων, θα είναι (πάντα για το υποθετικό αλφάβητο των τεσσάρων χαρακτήρων):

N \* (0,6 \* 1 + 0,3 \* 2 + 0,05 \* 3 + 0,05 \* 3) = 1,5N,

σε αντίθεση με 2*N* που θα απαιτούσε η σταθερή κωδικοποίηση, δηλαδή ένα όφελος της τάξης του 25%. Οπότε το πρόβλημα ανάγεται στην εύρεση αλγορίθμων, που θα αναπαραστήσουν με τον βέλτιστο τρόπο ένα αλφάβητο, με την προϋπόθεση βέβαια ότι η συχνότητα εμφάνισης των χαρακτήρων είναι γνωστές.

Ένα σημαντικό πρόβλημα που πρέπει να επιλυθεί, πριν την ανάπτυξη αλγορίθμων κωδικοποίησης μεταβλητού μήκους, είναι το κατά πόσο θα είναι δυνατή η αποκωδικοποίηση μίας κωδικοποιημένης συμβολοσειράς. Εάν η κωδικοποίηση ήταν σταθερού μήκους, θα αρκούσε η αποκοπή πλήθους bits όσων και το σταθερό μήκος τους, και εν συνεχεία η αποκωδικοποίηση τους. Όμως στην μεταβλητού μήκους κωδικοποίηση, κάτι τέτοιο δεν είναι δυνατόν, και το πρόβλημα έγκειται στο πως θα μπορούσε να αποκωδικοποιηθούν δύο χαρακτήρες, που και οι δύο ξεκινούν με τα ίδια ψηφία. Εάν για παράδειγμα ήταν α 🡪 01 και β 🡪 011, τότε θα ήταν αδύνατο να αποκωδικοποιηθεί μία συμβολοσειρά, που έστω ότι άρχιζε με τα ψηφία 011. Θα μπορούσε να ήταν το <α>, και μετά κάποιος άλλος χαρακτήρας που θα άρχιζε από 1, αλλά θα μπορούσε να άρχιζε και με <β>. Επομένως, είναι σημαντικότατο να μην υπάρχει κανένας χαρακτήρας, που η κωδικοποίησή του να είναι όμοια με την αρχή της κωδικοποίησης κάποιου άλλου χαρακτήρα. Αυτή η μέθοδος ονομάζεται ***προθεματική κωδικοποίηση***, δηλαδή η κωδικοποίηση όπου καμία κωδικό-λέξη δεν αποτελεί πρόθεμα καμίας άλλης.

Τα δυαδικά δένδρα προσφέρουν το κατάλληλο υπόβαθρο για την υλοποίηση της προθεματικής κωδικοποίησης, όπου τα φύλλα αποτελούν τις κωδικό-λέξεις, και οι αριστεροί κλάδοι αναπαριστούν το 0, ενώ οι δεξιοί το 1. Η κωδικοποίηση μπορεί να επιτευχθεί ακολουθώντας τον δρόμο από την ρίζα προς τα φύλλα, και φθάνοντας σε κάποιο φύλλο, τότε η διαδοχή των 1 και 0 οδηγούν στην κωδικό-λέξη του χαρακτήρα του φύλλου (*Σχήμα 9.5*). Οπότε, εάν , είναι η συχνότητα εμφάνισης του χαρακτήρα x, μίας αλφάβητου *A*, και , το μήκος της κωδικό-λέξης του, με βάση το δυαδικό δένδρο αναπαράστασης *T*, τότε το συνολικό μήκος *S*(*N*), μίας συμβολοσειράς *N* χαρακτήρων δίνεται από το τύπο:



Επομένως, το ζητούμενο είναι η ελαχιστοποίηση του *S*(*N*). Είναι βέβαια δυνατόν, η προτεινόμενη βέλτιστη κωδικοποίηση να μην είναι και η μοναδική. Ένας τέτοιος αλγόριθμος αναπτύχθηκε από τον David Huffman, στην διδακτορική διατριβή του το 1952, και ονομάσθηκε κωδικοποίηση Huffman, (αυτή η μέθοδος χρησιμοποιείται και από την διαδικασία pack του Unix).



***Σχήμα 9.5****: Κωδικοποίηση μεταβλητού μήκους.*

Ο αλγόριθμος Huffman σε γενικές γραμμές λειτουργεί ως εξής: Αρχικά συνενώνει δύο χαρακτήρες του αλφάβητου, προς ένα νέο υπέρ-χαρακτήρα, του οποίου η συχνότητα είναι ίση με το άθροισμα των χαρακτήρων που τον δημιούργησαν (γ + δ, *Σχήμα 9.5*). Τώρα το αλφάβητο αποτελείται κατά ένα χαρακτήρα λιγότερο (<γ δ> + β, <γ δ β> + α). Στην συνέχεια, αυτό επαναλαμβάνεται μέχρι να μην έχει μείνει κανένας χαρακτήρας. Το δένδρο δημιουργείται από κάτω προς τα πάνω, (από τα φύλλα προς την ρίζα), οπότε η διαδικασία πρέπει να αρχίζει με τους χαρακτήρες που έχουν τις μικρότερες συχνότητες. Όταν η διαδικασία ολοκληρωθεί, η κωδικό-λέξη κάθε χαρακτήρα είναι έτοιμη. Ένας άλλος τρόπος δημιουργίας του δυαδικού δένδρου είναι ξεκινώντας από την ρίζα. Αρχικά, σαν αριστερό παιδί τοποθετείται ο χαρακτήρας με την μεγαλύτερη συχνότητα, και αναδρομικά συνεχίζει με τον ίδιο τρόπο, δημιουργώντας το δεξιό υπό-δένδρο.

Τέλος ένατης ενότητας.

 