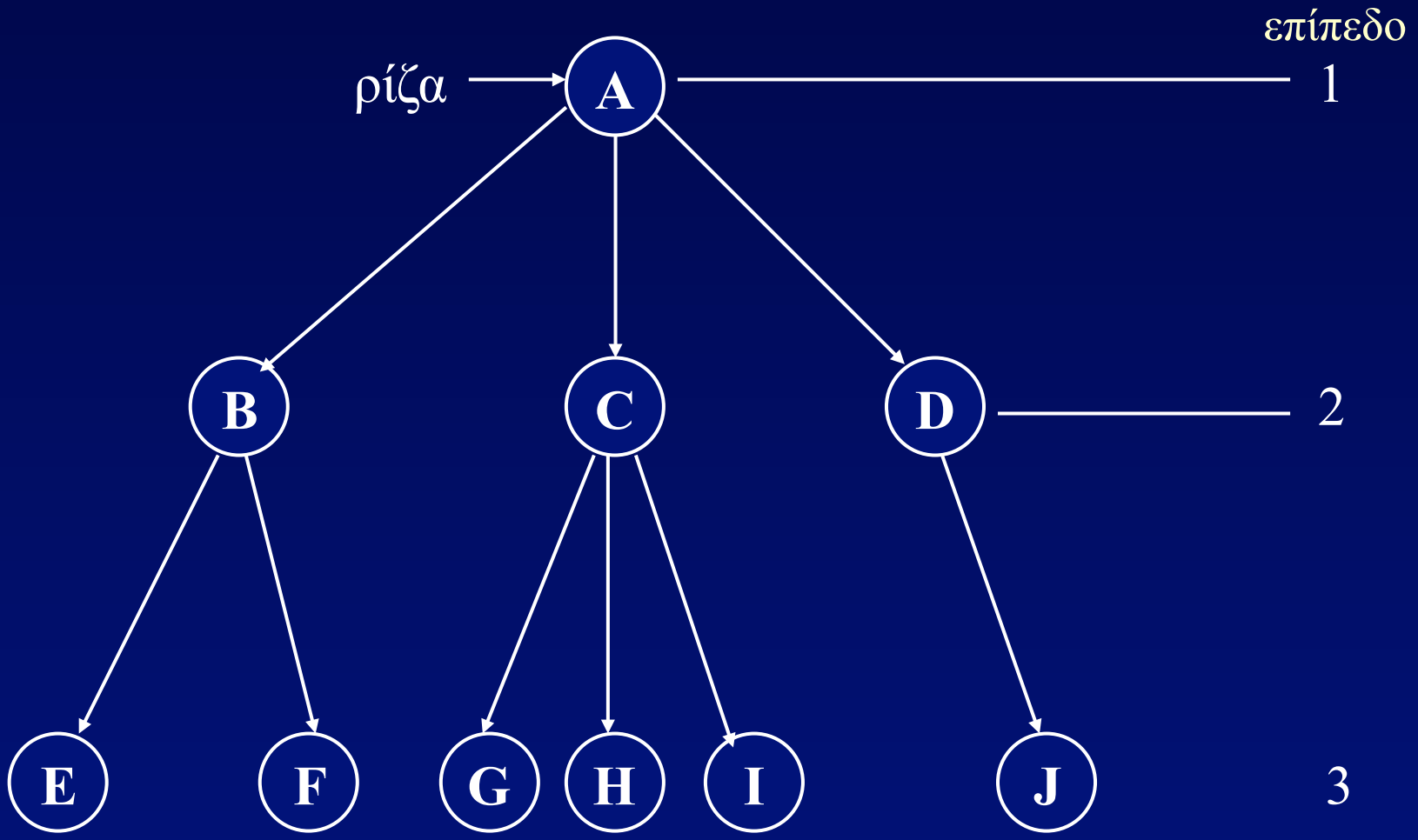
Δέντρα (Trees)

Ορισμός: Ενα δέντρο είναι ένα πεπερασμένο μη κενό σύνολο κόμβων τέτοιο ώστε:

1. Υπάρχει ένας μοναδικός κόμβος, που καλείται ρίζα, ο οποίος δεν έχει προηγούμενο.
2. Οι υπόλοιποι κόμβοι χωρίζονται σε n>= 0 ξένα μεταξύ τους υποσύνολα, $T_1, T_2, \ldots, T_n$, κάθε ένα από τα οποία είναι ένα δέντρο. Τα $T_1, T_2, \ldots, T_n$ καλούνται υπόδεντρα (subtrees) της ρίζας.

Ας σημειωθεί ότι χαρακτηριστική ιδιότητα του δέντρου είναι ότι η ρίζα είναι εκείνος ο κόμβος που δεν έχει εισερχόμενα τόξα και ξεκινώντας από αυτόν μπορούμε να καταλήξουμε σε ένα οποιοδήποτε κόμβο ακολουθώντας μια μοναδική διαδρομή από τόξα. Ο αριθμός των υποδέντρων ενός κόμβου καλείται βαθμός του κόμβου αυτού.

Ο αριθμός των υποδέντρων ενός κόμβου καλείται βαθμός του κόμβου αυτού. Ας θεωρήσουμε το δέντρο του παρακάτω Σχήματος. Το δέντρο αποτελείται από 10 κόμβους {A, B, C, D, E, F, G, H, I, J}.



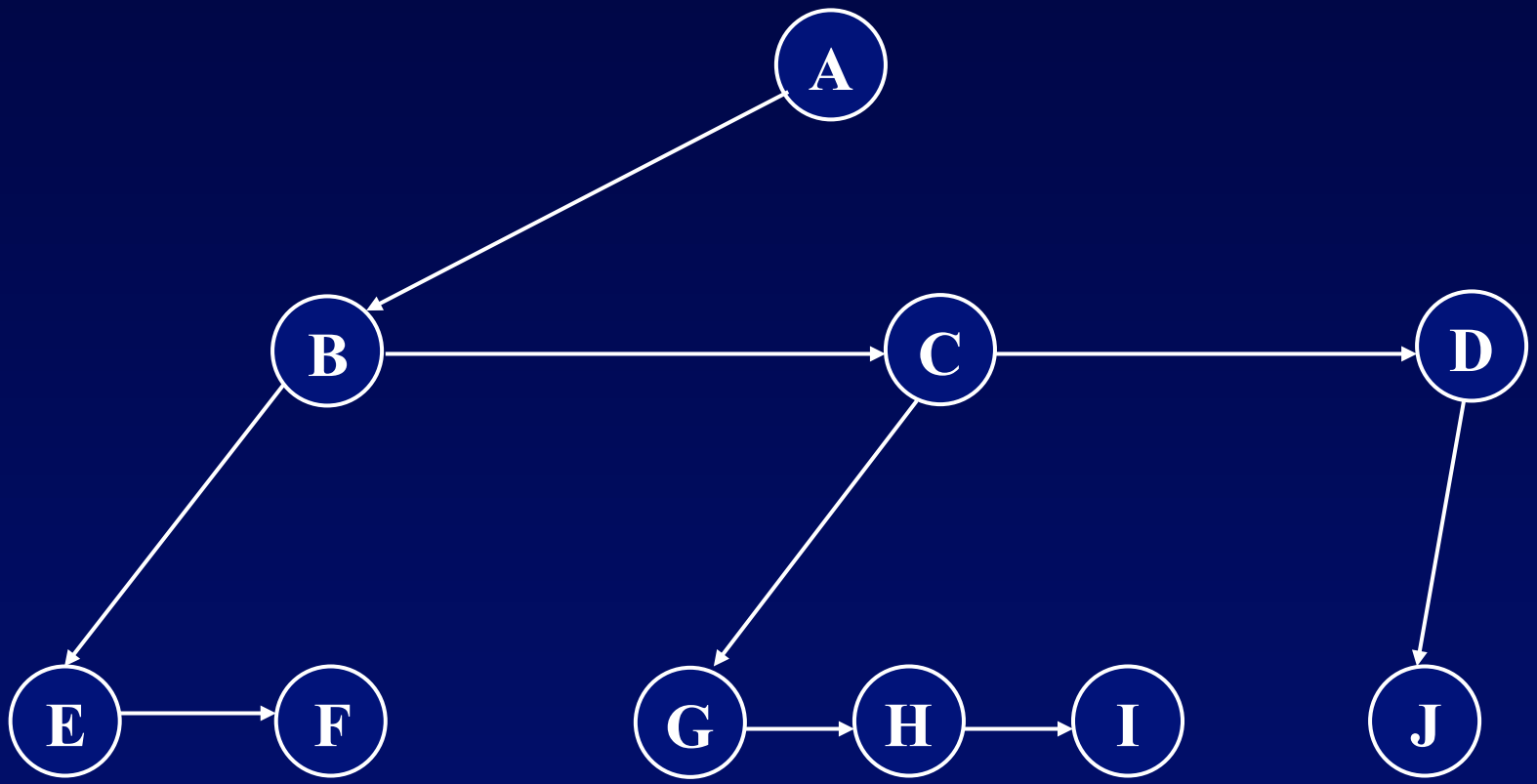
- βαθμός κόμβου
- βαθμός δέντρου
- τερματικοί κόμβοι (φύλλα)
- εσωτερικοί κόμβοι
- παιδί, πατέρας
- πρόγονος, απόγονος
- ύψος, βάθος

Ενα δυαδικό δέντρο είναι ένα πεπερασμένο σύνολο κόμβων, το οποίο είτε είναι κενό ή :

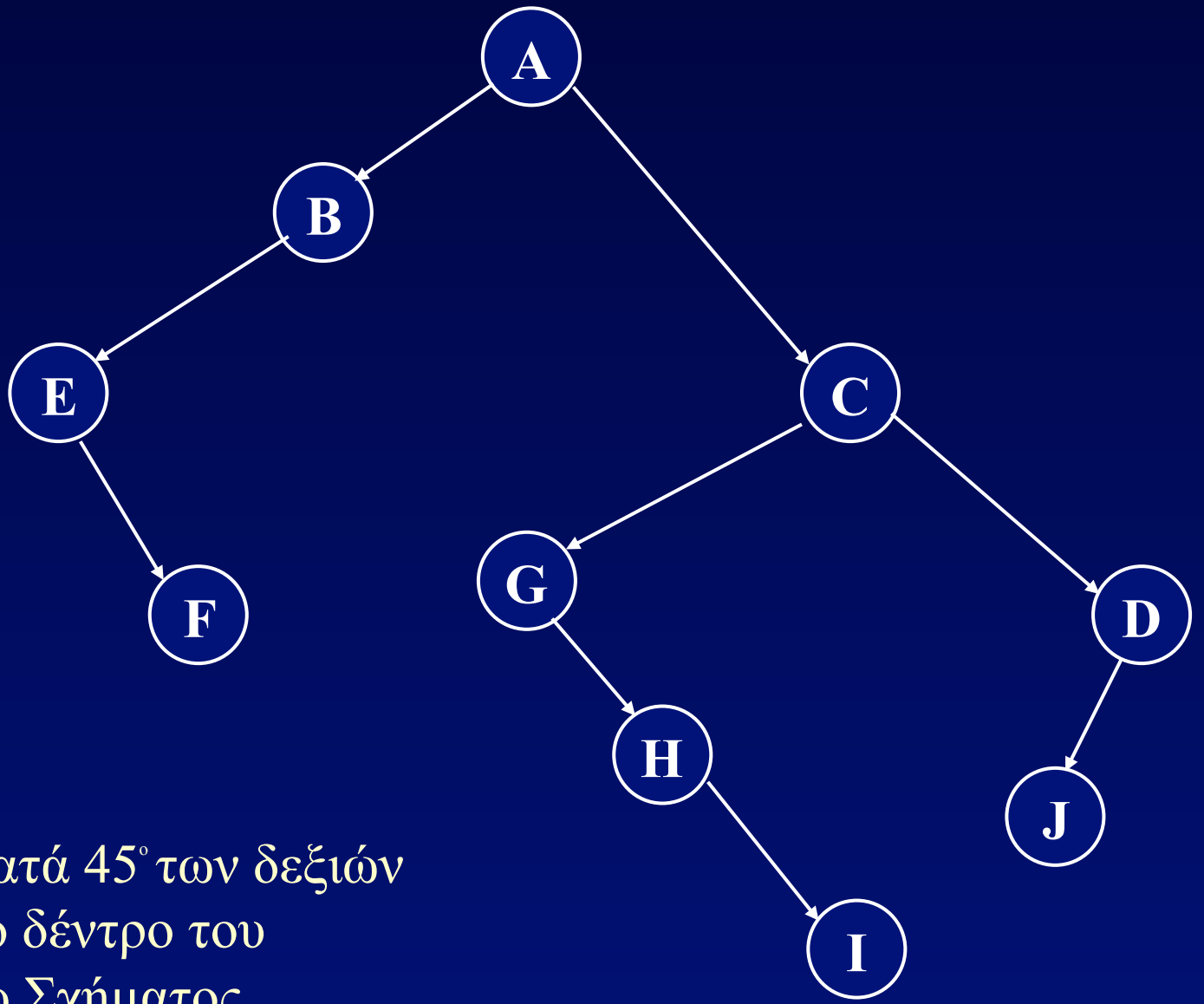
1. Υπάρχει ένας ειδικός κόμβος που καλείται ρίζα, και

2. Οι υπόλοιποι κόμβοι χωρίζονται σε δύο ξένα μεταξύ τους υποσύνολα, Α και Δ, κάθε ένα από τα οποία είναι ένα δυαδικό δέντρο. Το Α είναι το αριστερό υπόδεντρο της ρίζας και το Δ είναι το δεξί υπόδεντρο της ρίζας.

Κάθε δέντρο μπορεί να παρασταθεί σαν ένα δυαδικό δέντρο.



Μετατροπή του δέντρου του παραπάνω σχήματος σε ένα δυαδικό δέντρο



Στροφή κατά 45° των δεξιών τόξων στο δέντρο του παραπάνω Σχήματος

Πρόταση

(i) Ο μέγιστος αριθμός κόμβων στο επίπεδο i ενός δέντρου βαθμού d είναι d^{i-1}, i >= 1 και

(ii) Ο μέγιστος αριθμός κόμβων ενός δέντρου βαθμού d και ύψους h είναι $(d^h - 1)/(d - 1)$.

Απόδειξη

(i) Η απόδειξη θα γίνει με επαγωγή. Αν i = 1, τότε $d^{i-1} = d^0 = 1$ πράγμα που ισχύει. Εστω ότι ο μέγιστος αριθμός των κόμβων στο k επίπεδο είναι d^{k-1}. Το μέγιστο πλήθος των κόμβων στο k + 1 επίπεδο είναι $d \cdot d^{k-1} = d^k$.

(ii) Ο μέγιστος αριθμός των κόμβων σε ένα δυαδικό δέντρο ύψους h είναι:

$$\sum_{i=1}^{h} (\text{μέγιστο αριθμόκόμβων στο επίπεδο } i) =$$

$$\sum_{i=1}^{h} d^{i-1} = 1 + d + d^2 + \ldots + d^{h-1} = \frac{d^h - 1}{d - 1}$$

Πόρισμα

(i) Ο μέγιστος αριθμός κόμβων στο επίπεδο i ενός δυαδικού δέντρου είναι 2^{i-1}, i >= 1 και

(ii) Ο μέγιστος αριθμός κόμβων ενός δυαδικού δέντρου ύψους h είναι $2^h - 1$.

Πρόταση

Ενα δυαδικό δέντρο με n κόμβους έχει ύψος τουλάχιστον $\lceil \log_2(n+1) \rceil$

Απόδειξη

Αν h είναι το ύψος ενός δυαδικού δέντρου, τότε από το παραπάνω πόρισμα έχουμε $2^h - 1 >= n$ ή $h >= \log_2(n + 1)$. Αρα το ελάχιστο ύψος του δυαδικού δέντρου είναι $\lceil \log_2(n+1) \rceil$

Πρόταση

Για ένα μη κενό δυαδικό δέντρο, Τ, αν n_0 είναι το πλήθος των τερματικών κόμβων και n_2 είναι το πλήθος των εσωτερικών κόμβων βαθμού 2, τότε $n_0 = n_2 + 1$.

Απόδειξη

$$n = n_0 + n_1 + n_2 \quad (1)$$

όπου Β είναι το πλήθος των κλαδιών. Επίσης $n = B + 1$. Αλλά

$$B = n_1 + 2n_2$$

άρα $n = 1 + n_1 + 2n_2$ (2)

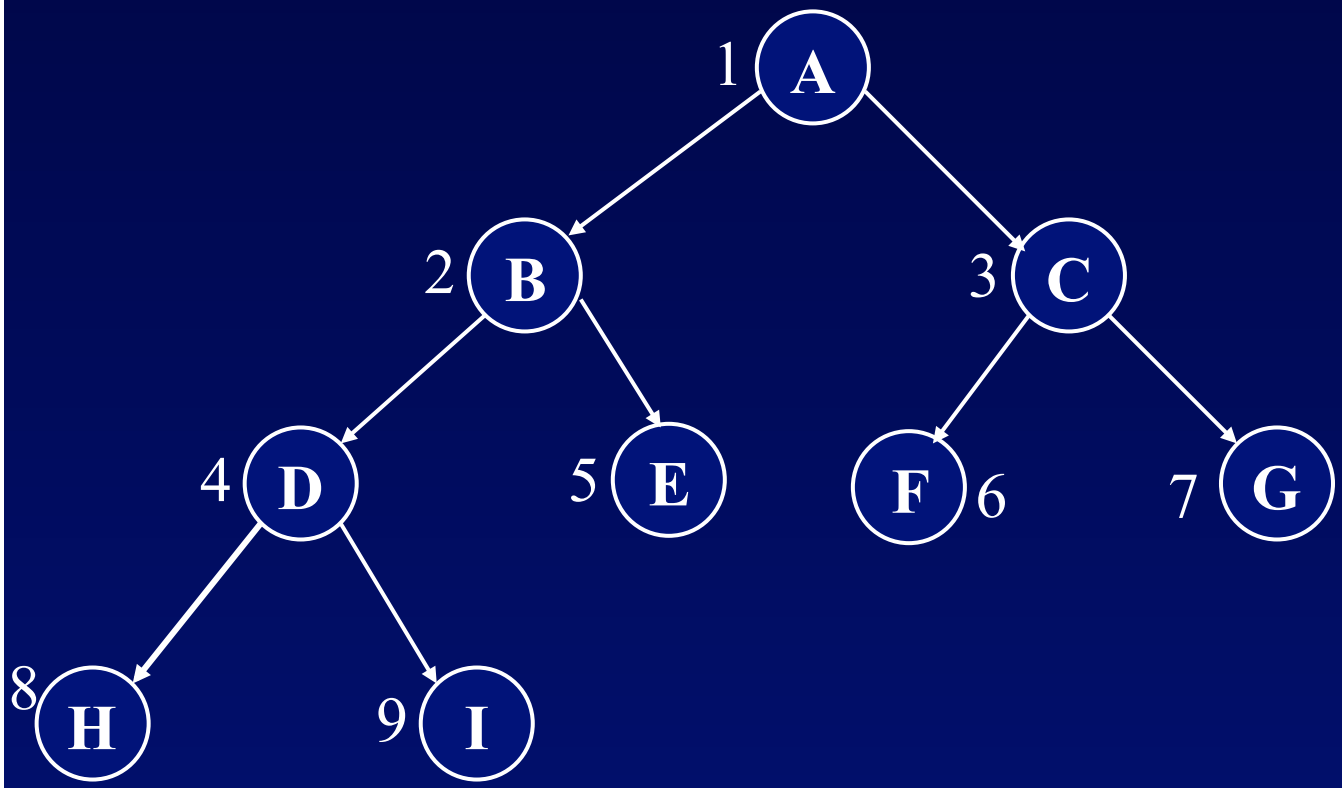
Από τις (1) και (2) έχουμε :

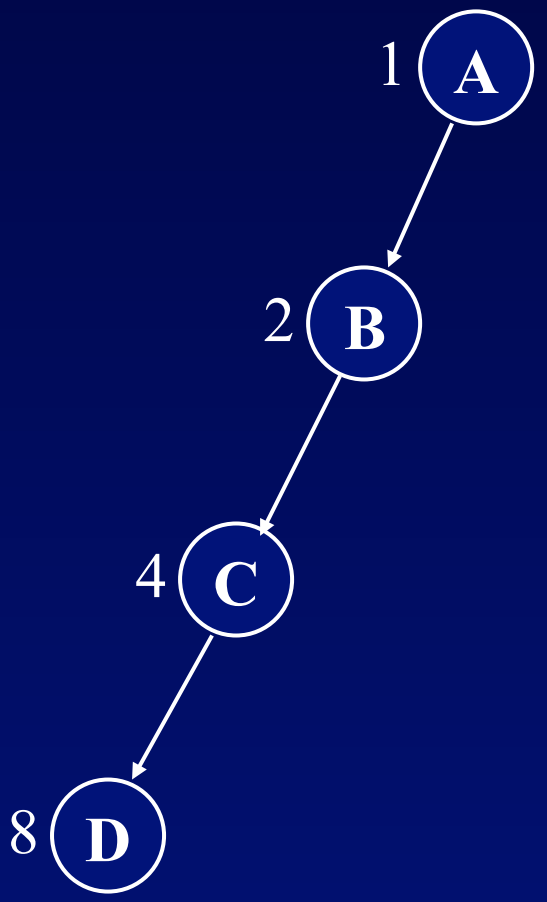
$$n_0 = n_2 + 1.$$

Για τον πλήρη ορισμό του ΑΤΔ δυαδικό δέντρο απομένει να ορισθούν οι βασικές πράξεις του που είναι:

1. Δημιουργία
2. Κενό
3. Πατέρας
4. Αριστερό παιδί
5. Δεξί παιδί
6. Διαγραφή υποδέντρων
7. Αντικατάσταση υποδέντρων
8. Συγχώνευση
9. Ανάκτηση
10. Ενημέρωση
11. Αναζήτηση

Υλοποίηση δυαδικών δέντρων με πίνακα





Πρόταση

Αν ένα πλήρες δυαδικό δέντρο με n κόμβους (δηλ. βάθος = $\lceil \log_2(n+1) \rceil$) παριστάνεται ακολουθιακά όπως παραπάνω, τότε για οποιοδήποτε κόμβο με δείκτη i, 1<= i<= n έχουμε:

(1) Ο pateras(i) είναι στη θέση $\lfloor i/2 \rfloor$ αν i<>1. Οταν i = 1, τότε δεν υπάρχει πατέρας, γιατί το i είναι η ρίζα.

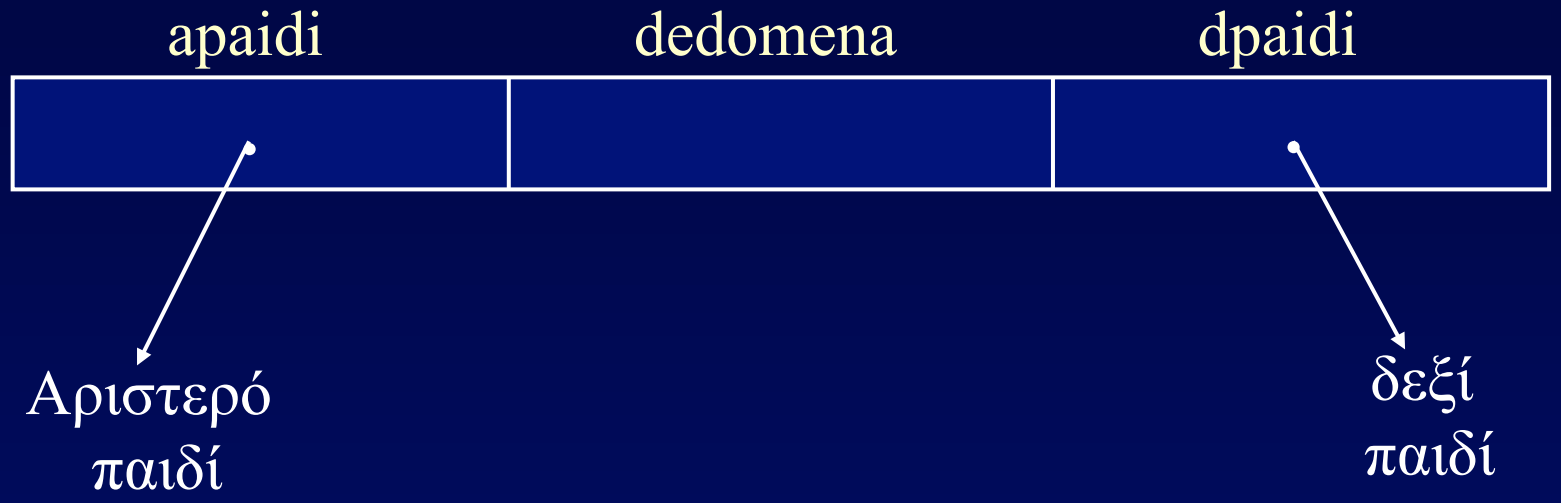
(2) Το apaidi(i) είναι στη θέση 2i αν 2i <= n. Αν 2i > n, τότε ο κόμβος δεν έχει αριστερό παιδί.

(3) Το dpaidi(i) είναι στη θέση 2i + 1, αν 2i + 1 <= n. Αν 2i + 1 > n, τότε ο κόμβος δεν έχει δεξί παιδί.

Η παράσταση αυτή είναι ιδανική για ένα πλήρες δυαδικό δέντρο, ωστόσο αρκετός χώρος μνήμης παραμένει ανεκμετάλευτος στα μη πλήρη δυαδικά δέντρα.

	dentro
1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I

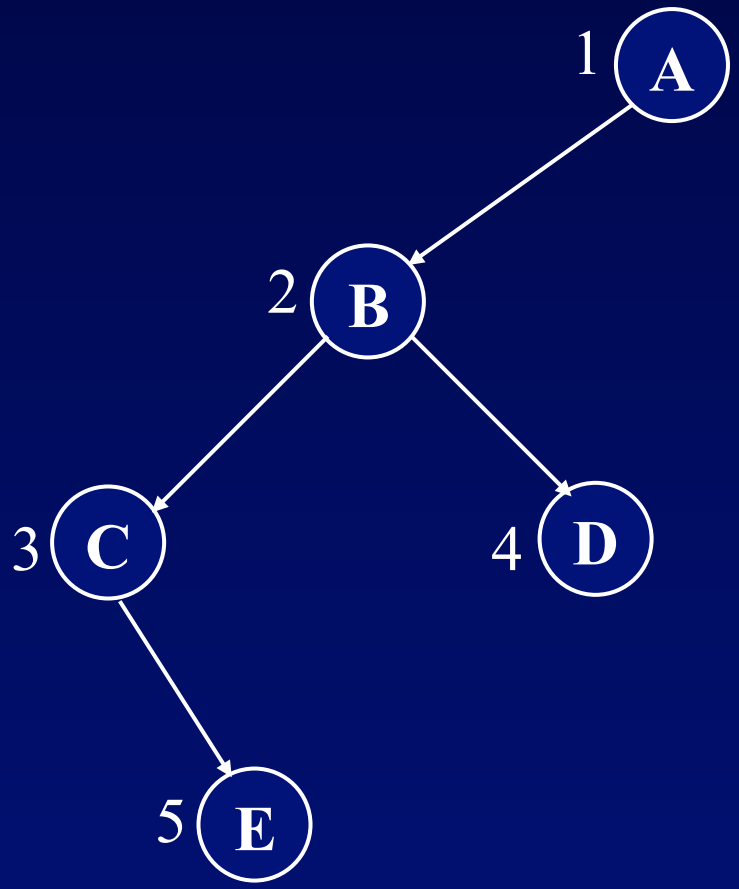
	dentro
1	A
2	B
3	-
4	-
5	-
6	-
7	-
8	D
9	-



Ενα δυαδικό δέντρο τώρα μπορεί να παρασταθεί σαν ένας πίνακας τέτοιων εγγραφών. Εχουμε λοιπόν τις παρακάτω δηλώσεις :

```
#define plithos ...
typedef ... typos_stoixeiou;
typedef int typos_deikti;

typedef struct
        {typos_stoixeiou dedomena;
         typos_deikti apaidi,dpaidi;
        } typos_komvou;

typedef typos_komvou pinakas_komvou[plithos];
```

Συνδεδεμένη παράσταση δυαδικού δέντρου με πίνακες :



τιμές δείκτη	dedomena	apaidi	dpaidi
1	A	2	0
2	B	3	4
3	C	0	5
4	D	0	0
5	E	0	0

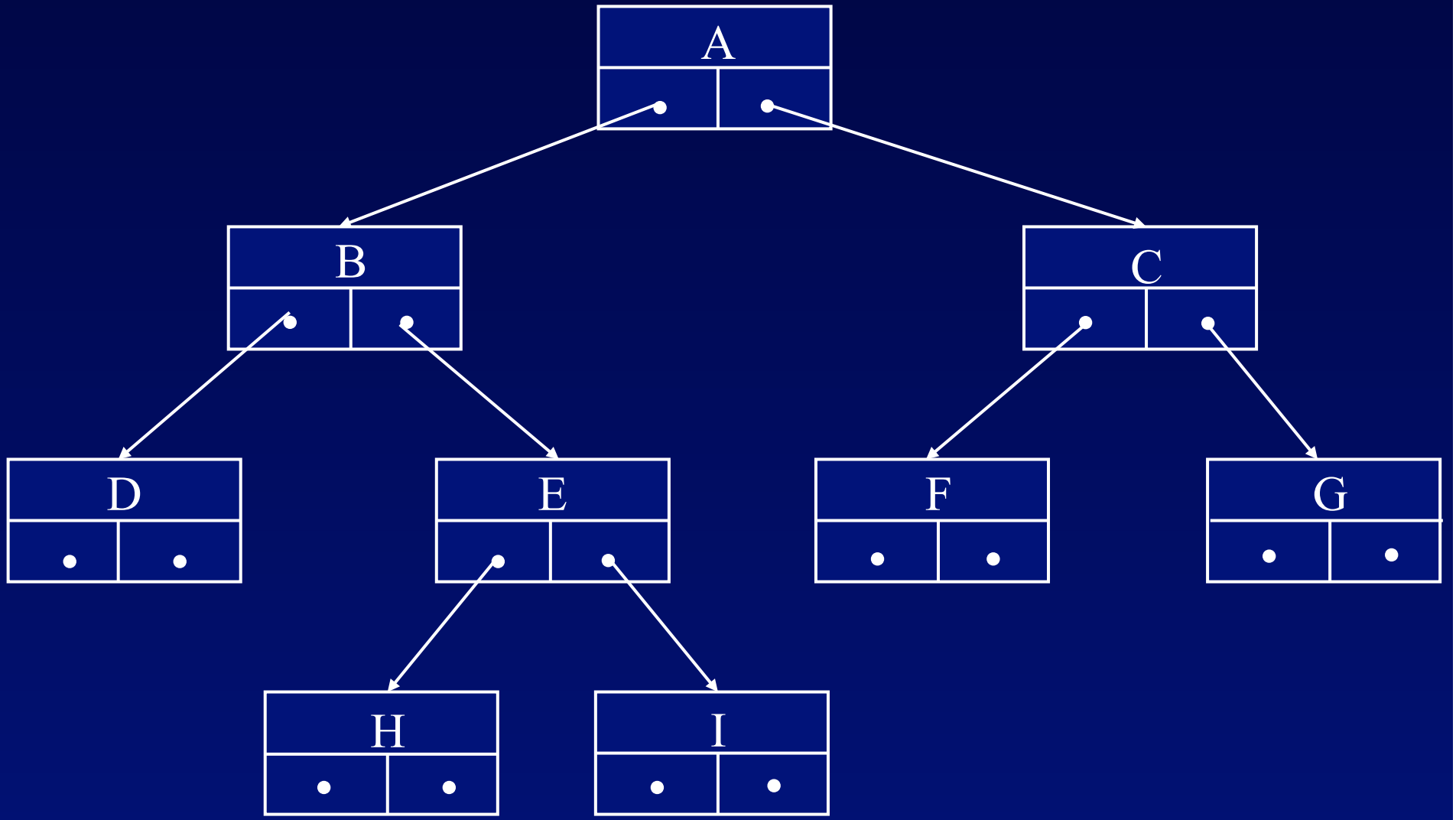
Υλοποίηση δυαδικών δέντρων με λίστες :

```
typedef ... typos_stoixeiou;
typedef struct typos_komvou *typos_deikti;
typedef struct typos_komvou
{
        typos_stoixeiou dedomena;
        typos_deikti  apaidi,dpaidi;
};

typos_deikti riza;
```

```
void dimiourgia_dentro(typos_deikti *riza)
        /*Προ: Καμμία.
         Μετά: Εχει δημιουργηθεί ένα κενό δυαδικό δέντρο
        στο  οποίο δείχνει η riza.*/
{
        *riza = NULL;
}
```

```
int keno_dentro(typos_deikti riza)
/*      Προ:   Εχει δημιουργηθεί το δέντρο στο οποίο δείχνει η riza
        Μετά: Το υποπρόγραμμα επιστρέφει την τιμή true ή false
        ανάλογα με το αν το δέντρο είναι κενό ή όχι.*/
{
        return (riza == NULL);
}
```

Συνδεδεμένη παράσταση δυαδικού δέντρου με λίστες :



Διαδρομή δυαδικού δέντρου

Μια από τις βασικές επεξεργασίες ενός δυαδικού δέντρου είναι η επίσκεψη κάθε κόμβου του μια μόνο φορά.

1. Επίσκεψη της ρίζας.
2. Διαδρομή του αριστερού υποδέντρου της.
3. Διαδρομή του δεξιού υποδέντρου της.

Τα τρία αυτά βήματα μπορούν να εκτελεστούν με οποιαδήποτε διάταξη. Αν συμβολίσουμε τα παραπάνω βήματα με:

Κ : Επίσκεψη ενός κόμβου.

Α : Διαδρομή αριστερού υποδέντρου του.

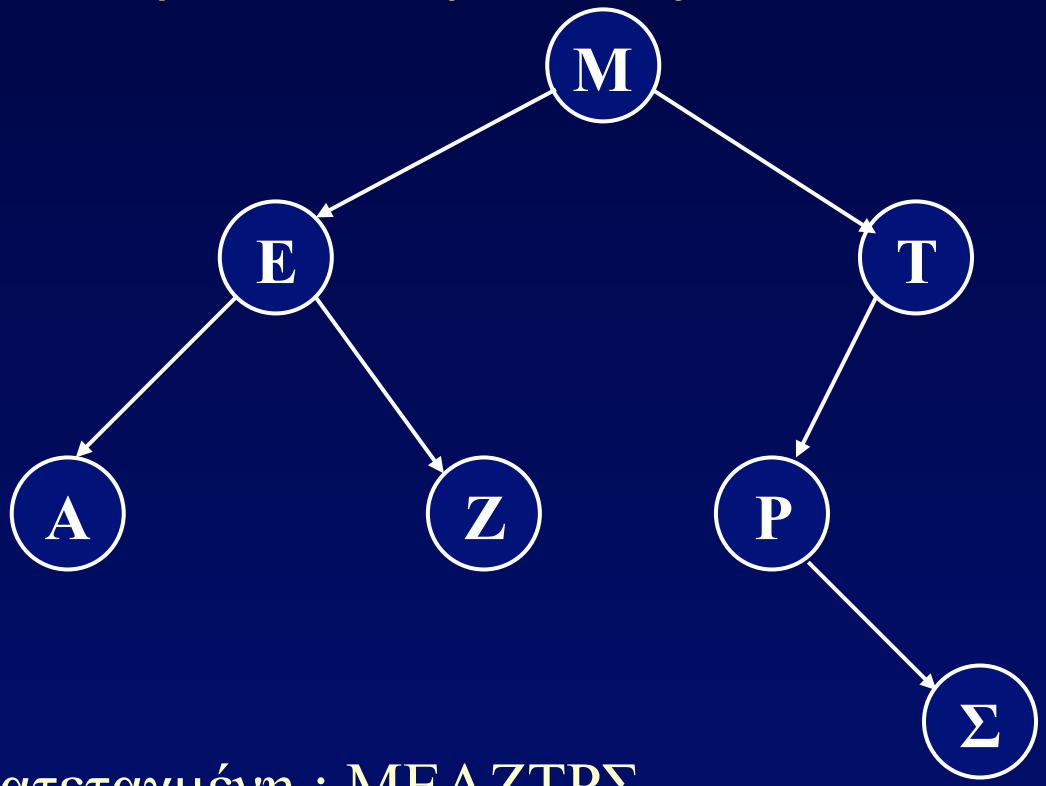
Δ : Διαδρομή δεξιού υποδέντρου του.

Τότε υπάρχουν έξι διατάξεις για την επίσκεψη των κόμβων ενός δυαδικού δέντρου που είναι οι:

ΚΑΔ ΑΚΔ ΑΔΚ ΚΔΑ ΔΚΑ ΔΑΚ

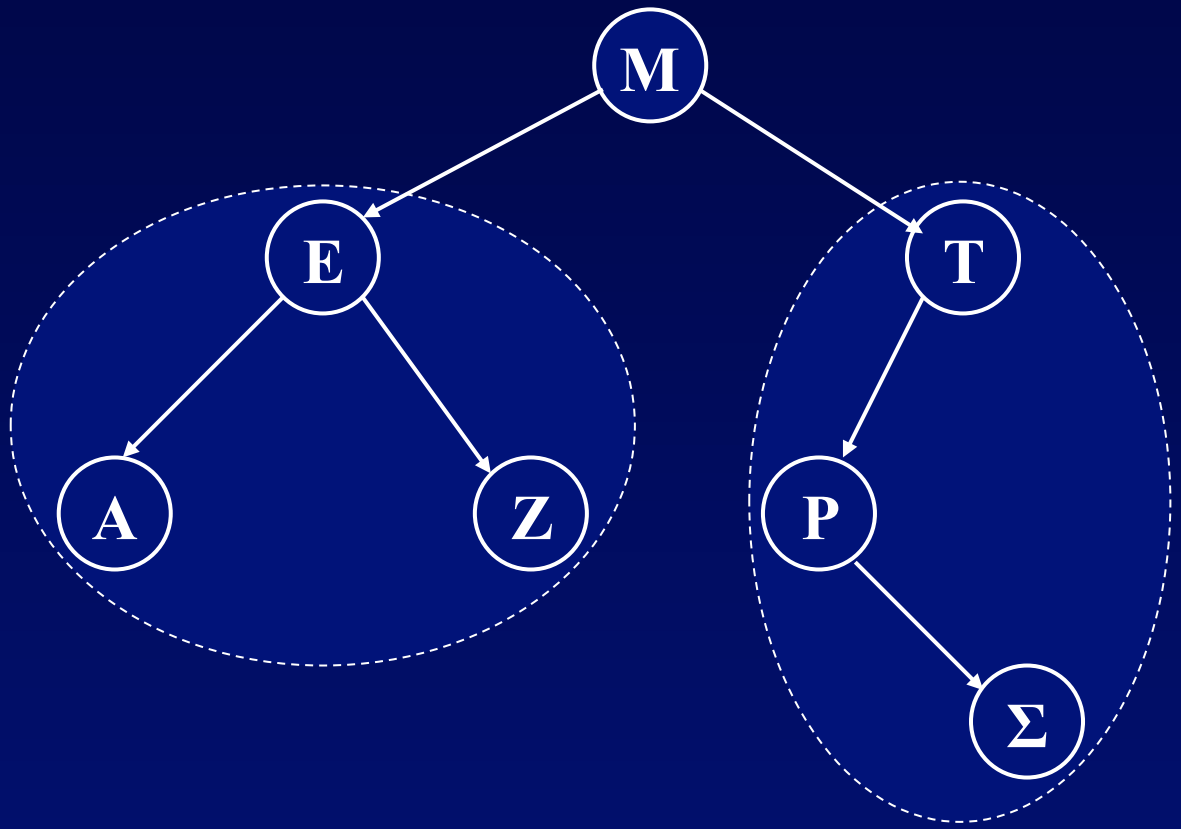
ΚΑΔ : προδιατεταγμένη διαδρομή.
ΑΚΔ : ενδοδιατεταγμένη διαδρομή.
ΑΔΚ : μεταδιατεταγμένη διαδρομή.

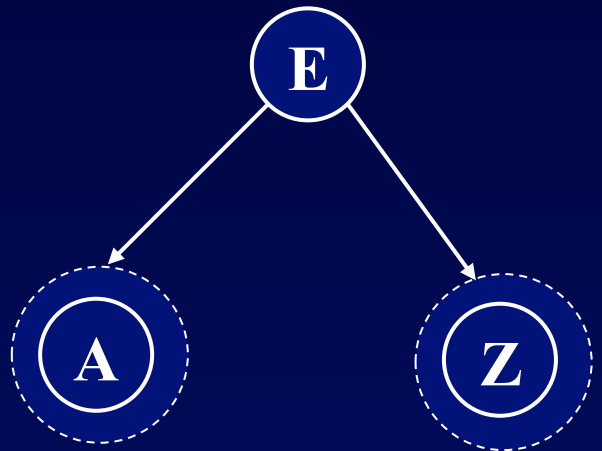
Παράσταση των τριών διατάξεων ενός δυαδικού δέντρου:

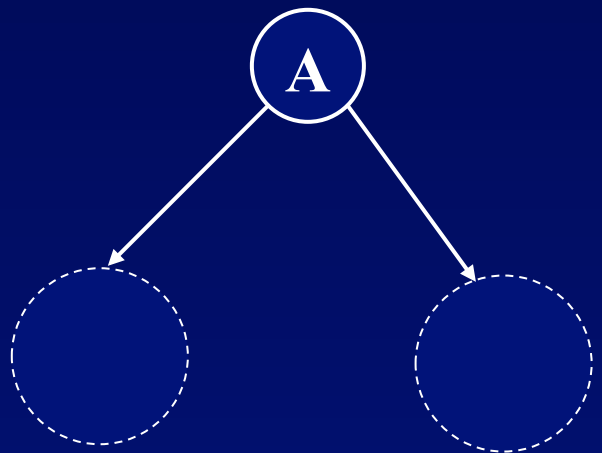


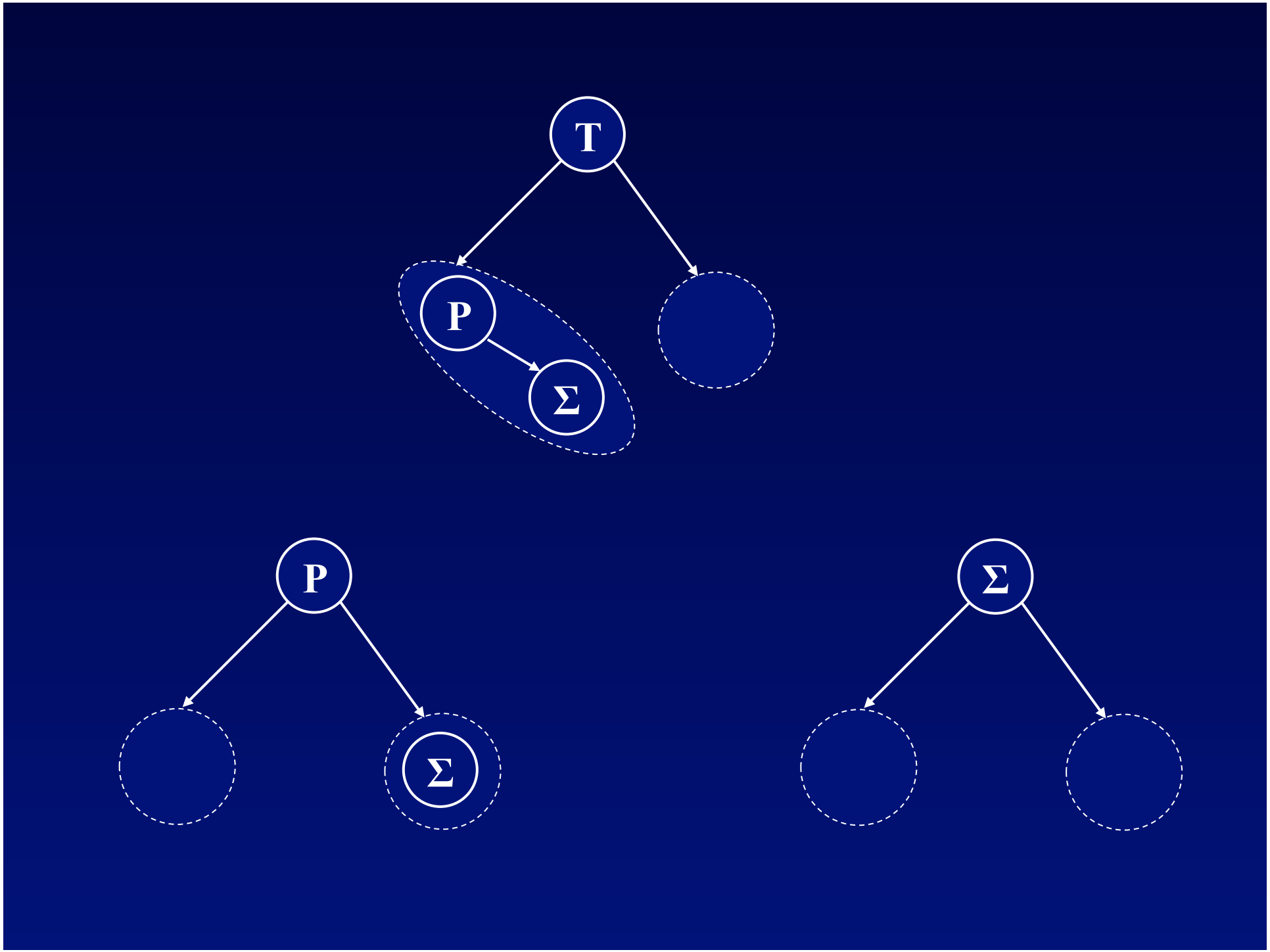
προδιατεταγμένη : ΜΕΑΖΤΡΣ
ενδοδιατεταγμένη : ΑΕΖΜΡΣΤ
μεταδιατεταγμένη : ΑΖΕΣΡΤΜ

Μεταδιατεταγμένη διαδρομή




E
A
Z

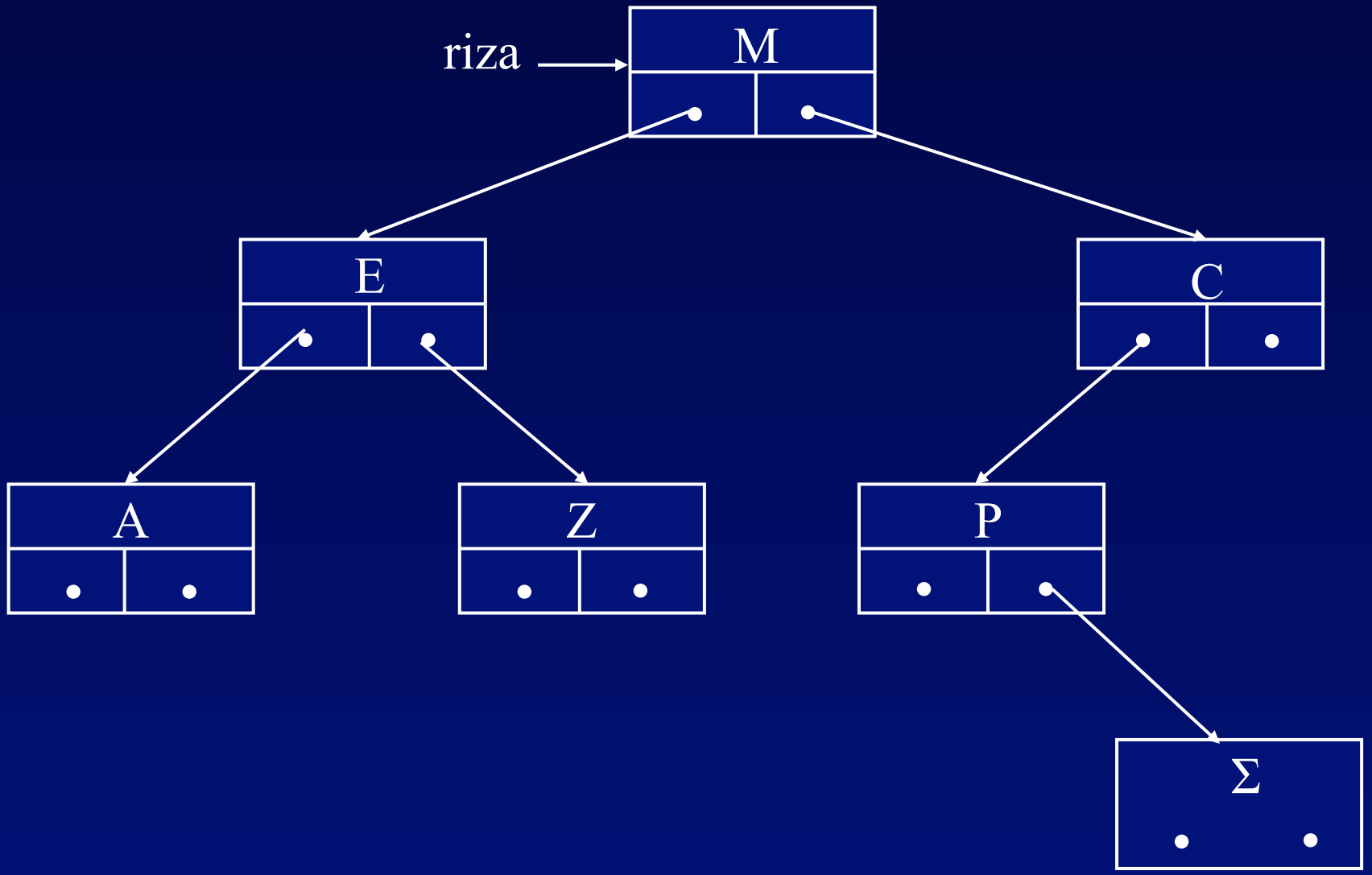

A


T
P
Σ
P
Σ
Σ

Αναδρομικό Υποπρόγραμμα

```
void endodiataksi(typos_deikti riza)
{       /* Αναδρομικό υποπρόγραμμα. Διατρέχει με
        ενδοδιατεταγμένη διαδρομή ένα δυαδικό δέντρο. */

        if (!keno_dentro(riza))
        {/* Επίσκεψη αριστερού υπόδεντρου */
                endodiataksi(riza->apaidi);
                /* Επίσκεψη της ρίζας */
                episkepsi(riza);
                /* Επίσκεψη δεξιού υπόδεντρου */
                endodiataksi(riza->dpaidi);
        }
}
```

Ας υποθέσουμε ότι έχουμε το δυαδικό δέντρο:



και ότι η episkeyi τυπώνει το περιεχόμενο ενός κόμβου. Η εντολή

endodiataksi (riza)

καλεί τη διαδικασία endodiataksi προκειμένου να εκτελέσει την ενδοδιατεταγμένη διαδρομή του παραπάνω δέντρου. Η ενέργεια αυτής της διαδικασίας εμφανίζεται αναλυτικά στον ακόλουθο πίνακα :

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Μ	Κλήση της διαδικασίας με δείκτη στη ρίζα (Ε) του αριστερού υποδέντρου.	
Ε	Κλήση της διαδικασίας με δείκτη στη ρίζα (Α) του αριστερού υποδέντρου.	
Α	Κλήση της διαδικασίας με δείκτη (NULL) στη ρίζα του αριστερού υποδέντρου.	
Κενό	Κενό δέντρο, επιστροφή στον κόμβο-πατέρα.	
Α	Εκτύπωση του περιεχομένου του κόμβου	Α

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Α	Κλήση της διαδικασίας με δείκτη NULL στη ρίζα του δεξιού υποδέντρου.	
Κενό	Κενό δέντρο, επιστροφή στον κόμβο-πατέρα.	
Α	Επιστροφή στον κόμβο πατέρα.	
Ε	Εκτύπωση του περιεχομένου του του κόμβου.	Ε
Ε	Κλήση της διαδικασίας με δείκτη στη ρίζα (Ζ) του δεξιού υποδέντρου.	

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Z	Κλήση της διαδικασίας με δείκτη (NULL) στη ρίζα του αριστερού υποδέντρου.	
Κενό	Κενό δέντρο, επιστροφή στον κόμβο-πατέρα.	
Z	Εκτύπωση του περιεχομένου του κόμβου.	Z
Z	Κλήση της διαδικασίας με δείκτη (NULL) στη ρίζα του δεξιού υποδέντρου.	

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Κενό	Κενό δέντρο, επιστροφή στον κόμβο πατέρα.	
Z	Επιστροφή στον κόμβο-πατέρα.	
E	Επιστροφή στον κόμβο-πατέρα.	
M	Εκτύπωση του περιεχομένου του κόμβου.	M
M	Κλήση της διαδικασίας με δείκτη στη ρίζα (T) του δεξιού υποδέντρου.	

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Z	Κλήση της διαδικασίας με δείκτη στη ρίζα (P) του αριστερού υποδέντρου.	
P	Κλήση της διαδικασίας με δείκτη (NULL) στη ρίζα του αριστερού υποδέντρου.	
Κενό	Κενό δέντρο, επιστροφή στον κόμβο πατέρα.	
P	Εκτύπωση του περιεχομένου του κόμβου	P

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Ρ	Κλήση της διαδικασίας με δείκτη στη ρίζα (Σ) του δεξιού υποδέντρου.	
Σ	Κλήση της διαδικασίας με δείκτη (NULL1) στη ρίζα του αριστερού υποδέντρου.	
Κενό	Κενό δέντρο, επιστροφή στον κόμβο-πατέρα.	
Σ	Εκτύπωση του περιεχομένου του κόμβου	Σ

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Σ	Κλήση της διαδικασίας με δείκτη (NULL) του δεξιού υποδέντρου.	
Κενό	Κενό δέντρο, επιστροφή στον κόμβο πατέρα.	
Σ	Επιστροφή στον κόμβο-πατέρα	
Ρ	Επιστροφή στον κόμβο-πατέρα	
Τ	Εκτύπωση του περιεχομένου του κόμβου.	Τ

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
Τ	Κλήση της διαδικασίας με δείκτη (NULL) του δεξιού υποδέντρου.	
Κενό	Κενό δέντρο, επιστροφή στον κόμβο-πατέρα.	
Τ	Επιστροφή στον κόμβο-πατέρα	
Μ	Τέλος διαδικασίας	

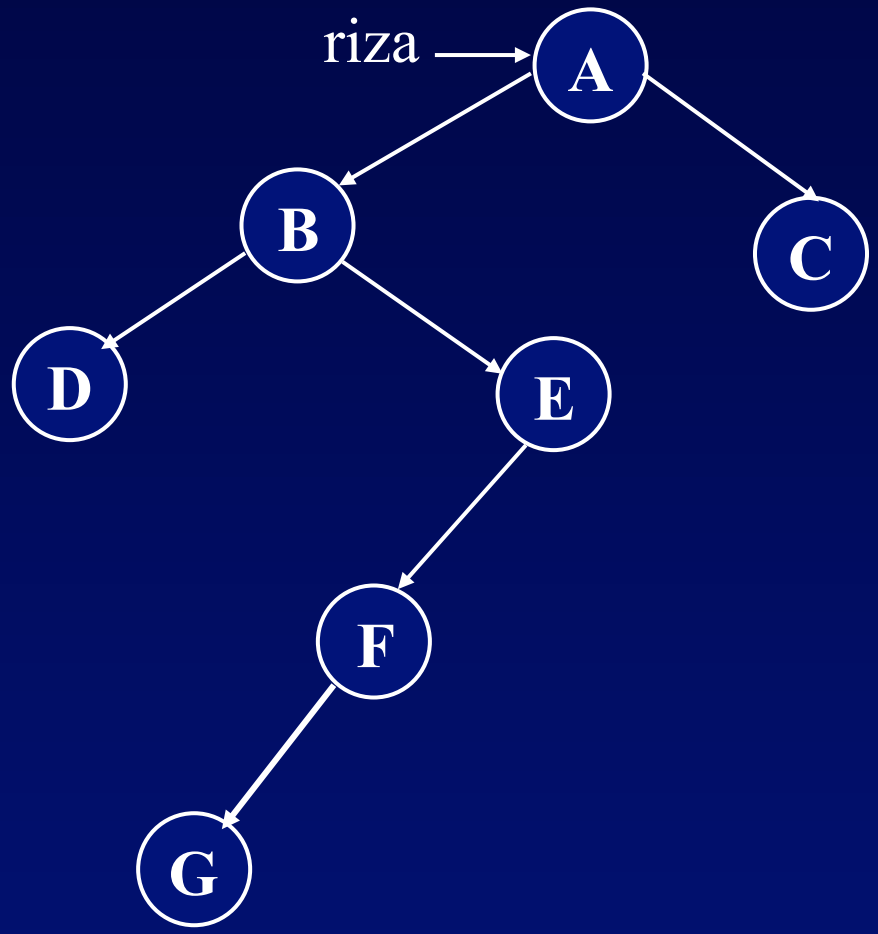
Είναι φανερό ότι η προδιατεταγμένη και μεταδιατεταγμένη διαδρομή προκύπτουν αν απλά αλλάξουμε την σειρά των εντολών στη διαδικασία endodiataksi, όπως φαίνεται παρακάτω :

```
void prodiataksi(typos_deikti riza)
{/* προδιατεταγμένη  διαδρομή */

        if (!keno_dentro(riza))
        {
                episkepsi(riza);
                prodiataksi(riza->apaidi);
                prodiataksi(riza->dpaidi);
        }
}
```

συνέχεια

```
void metadiataksi(typos_deikti riza)
{       /* μεταδιατεταγμένη διαδρομή */
        if (!keno_dentro(riza))
        {
                metadiataksi(riza->apaidi);
                metadiataksi(riza->dpaidi);
                episkepsi(riza);
        }
}
```

Μη Αναδρομικά Υποπρογράμματα



Μη αναδρομικός αλγόριθμος για την ενδοδιατεταγμένη διαδρομή

1. trexon = riza;

2. Οσο η στοίβα δεν είναι κενή και trexon != NULL να εκτελούνται:
   α. Να διατρέχονται οι κόμβοι του αριστερού υποδέντρου και να τοποθετούνται οι διευθύνσεις τους σε μια στοίβα.
   β. Αν η στοίβα δεν είναι κενή τότε {αριστερό υπόδεντρο κενό}
      (i) Εξαγωγή διεύθυνσης από στοίβα
      (ii) Επεξεργασία του περιεχομένου του.
      (iii) Διαδρομή του δεξιού υποδέντρου του.

Η υλοποίηση του παραπάνω αλγορίθμου είναι η ακόλουθη:

```
typedef typos_deiktitypos_stoixeiou_stoivas;
typedef struct typos_komvou_stoivas *typos_deikti_stoivas;
typedef struct typos_komvou_stoivas
{
        typos_stoixeiou_stoivas dedomena;
        typos_deikti_stoivas   epomenos;
};
typedef typos_deikti_stoivas typos_stoivas;

void endodiataksi (typos_deikti riza)
{/* Επαναληπτικό πρόγραμμα ενδοδιάταξης */
        typos_stoivas stoiva;
        typos_deikti trexon;
```

συνέχεια

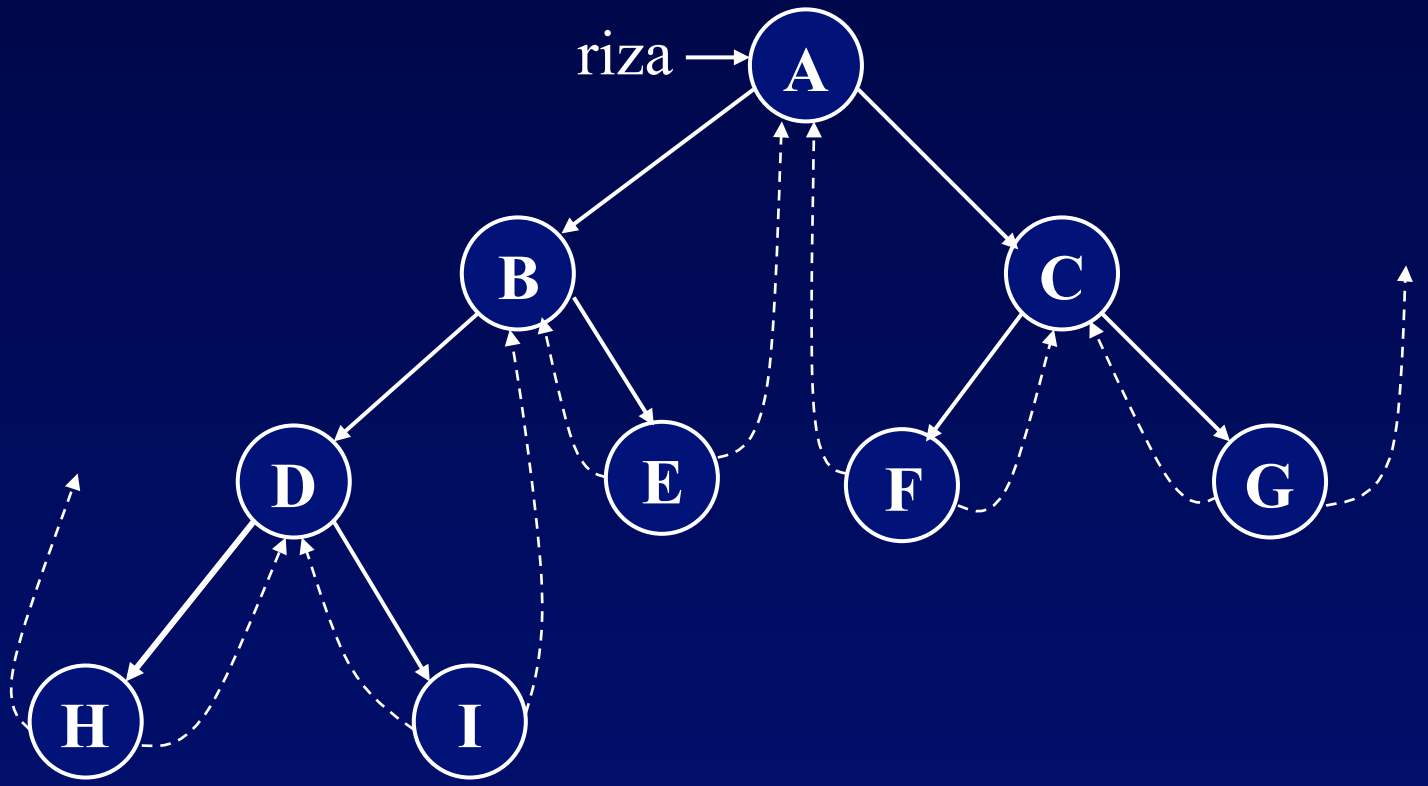
```
dimiourgia(&stoiva);
trexon = riza;
do
{/* Διατρέχονται οι κόμβοι των αριστερών κλαδιών */
        while (trexon!=NULL)
        {       othisi(&stoiva,trexon);
                trexon = trexon->apaidi;
        }
        /* Έλεγχος αν η στοίβα είναι κενή */
        if (!keni(stoiva))
        /* Το αριστερό υπόδεντρο είναι κενό */
        {       exagogi(&stoiva,&trexon);
                /* Επίσκεψη της ρίζας */
                episkepsi(trexon);
                /* Επίσκεψη δεξιού υπόδεντρου */
                trexon = trexon->dpaidi;
```

συνέχεια

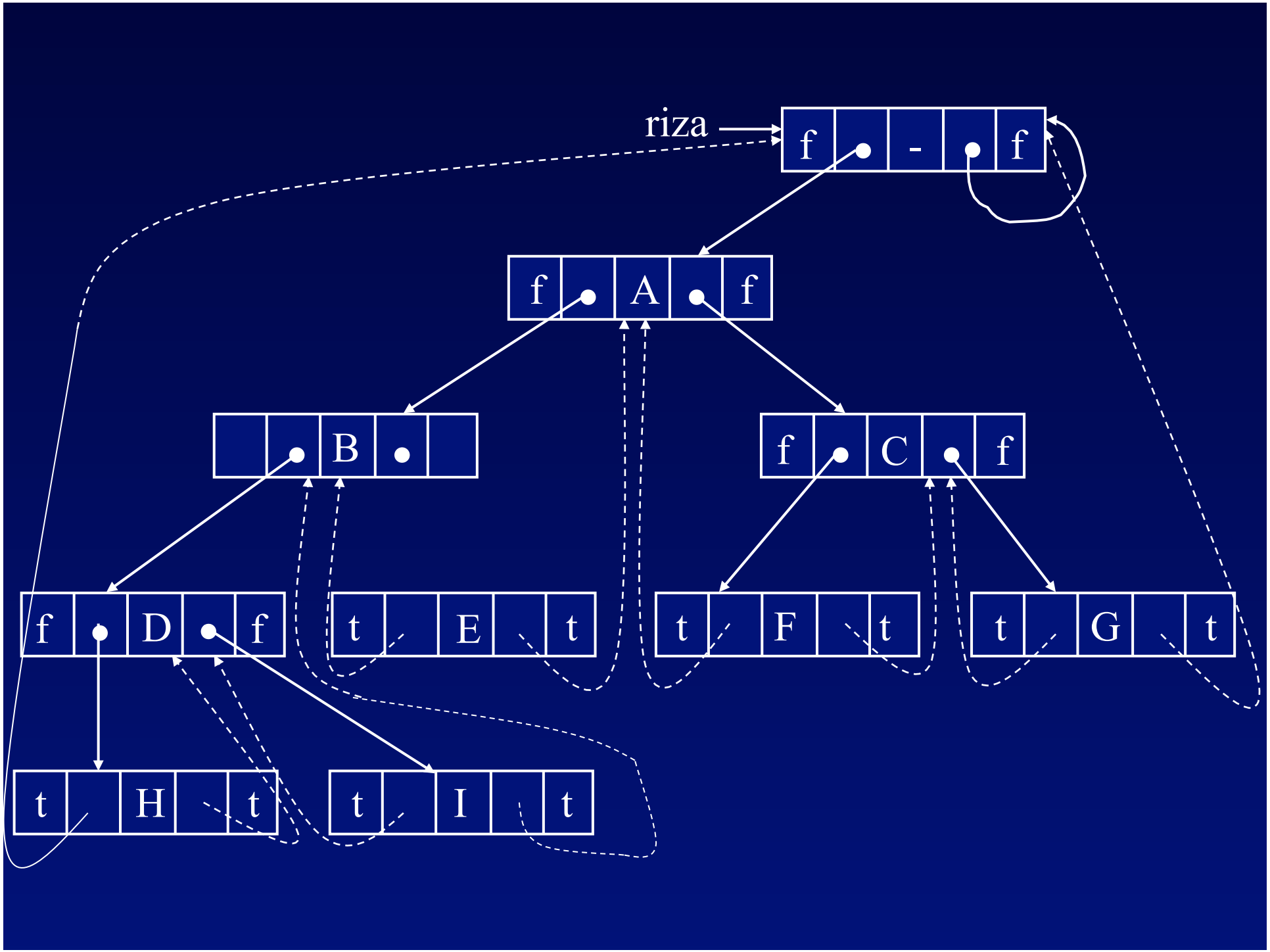
```
            }
        }
        while ((!keni(stoiva)) || (trexon!=NULL));
}
```

Ο χρόνος εκτέλεσης του παραπάνω υποπρογράμματος είναι μεγαλύτερος από εκείνο του αναδρομικού, πράγμα που έρχεται σε αντίθεση με το γεγονός ότι τα αναδρομικά υποπρογράμματα είναι συνήθως αργότερα από τα μη αναδρομικά.

Δυαδικά Δέντρα με κλωστές

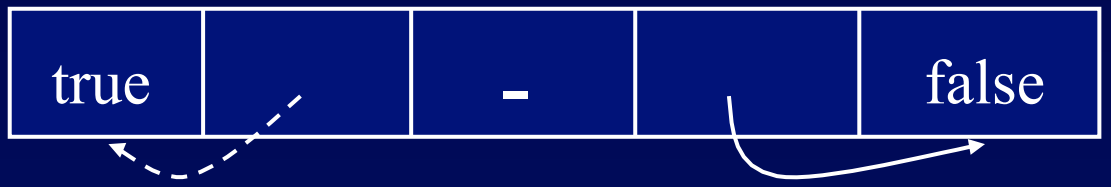


```
typedef ... typos_stoixeiou;

typedef struct typos_komvou *typos_deikti;
typedef struct typos_komvou
{         typos_stoixeiou  dedomena;
           typos_deikti  apaidi,dpaidi;
           int  aklosti,dklosti;
};

typos_deikti riza;
```


riza
f
-
f
f
A
f
B
f
C
f
f
D
f
t
E
t
t
F
t
t
G
t
t
H
t
t
I
t

Το παραπάνω σχήμα αποτελεί την παράσταση δυαδικού δέντρου με κλωστές, f = false, t = true.

Ενα κενό δέντρο με κλωστές παριστάνεται με τον κόμβο κεφαλή του όπως δείχνει το ακόλουθο σχήμα:



Για ένα οποιοδήποτε κόμβο x ενός δυαδικού δέντρου με κλωστές παρατηρούμε ότι αν x->dklosti = true, τότε το πεδίο x->dpaidi περιέχει τη διεύθυνση του ενδοδιατεταγμένου επόμενου του x. Διαφορετικά, ο ενδοδιατεταγμένος επόμενος του x βρίσκεται αν ακολουθηθεί το μονοπάτι των αριστερών κλαδιών του δεξιού παιδιού του x μέχρις ότου συναντηθεί ένας κόμβος για τον οποίο ισχύει aklosti = true. Το ακόλουθο υποπρόγραμμα εντοπίζει τον ενδοδιατεταγμένο επόμενο ενός κόμβου χωρίς τη χρήση στοίβας.

```
typos_deikti endo_epomenos(typos_deikti trexon)
{/* Μετά: Η συνάρ. επιστρέφει τον ενδοδιατεταγμένο
        επόμενο του κόμβου που δείχνει ο trexon σε
        ένα δυαδικό δέντρο με κλωστές. */

        typos_deikti prosorinos;

        if (trexon->dklosti) /* Ο κόμβος είναι φύλλο */

                return trexon->dpaidi;
```

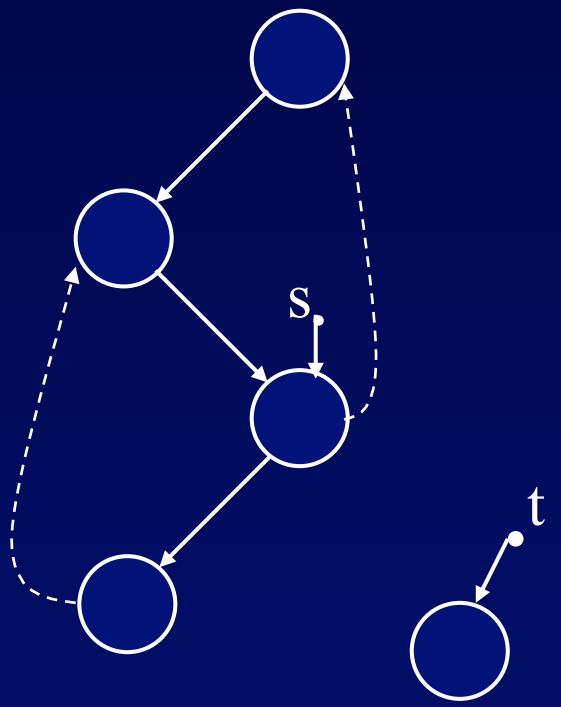
συνέχεια

```
        else /* Ακολούθησε το αριστερό μονοπάτι του δεξιού παιδιού */
        {       prosorinos = trexon->dpaidi;
                while (!(prosorinos->aklosti))
                        prosorinos = prosorinos->apaidi;
                return prosorinos;
        }
}
```

```
void endo_diadromi(typos_deikti riza)
{/* Μετά: Επίσκεψη των κόμβων ενός δυαδικού δέντρου με
        κλωστές με την ενδοδιατεταγμένη διαδρομή. */
         typos_deikti prosorinos;
         prosorinos=riza; /* κεφαλή, οχι πραγματική ρίζα */
         do
         {
                  prosorinos=endo_epomenos(prosorinos);
                  if (prosorinos!=riza)
                           episkepsi(prosorinos);
         }
         while (prosorinos != riza);
}
```

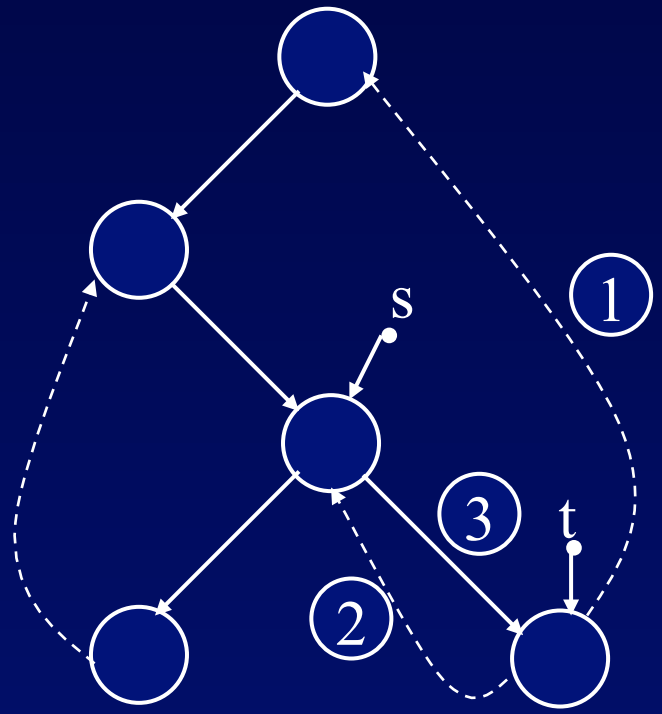
Πολυπλοκότης : O(n)

Ο s δεν έχει δεξί παιδί

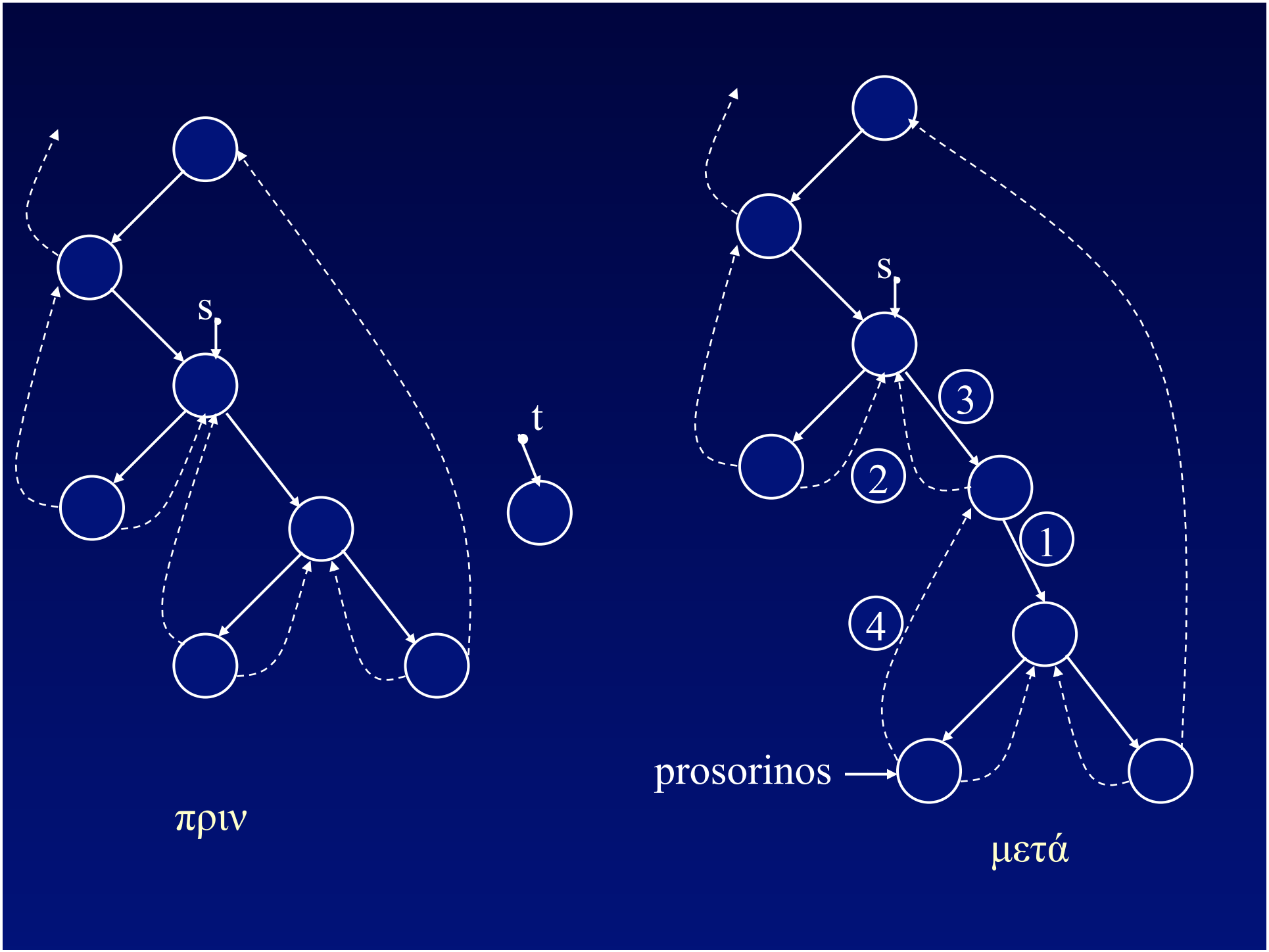


πριν

Ο s έχει δεξί παιδί



μετά


s
t
πριν
s
3
2
1
4
prosorinos
μετά

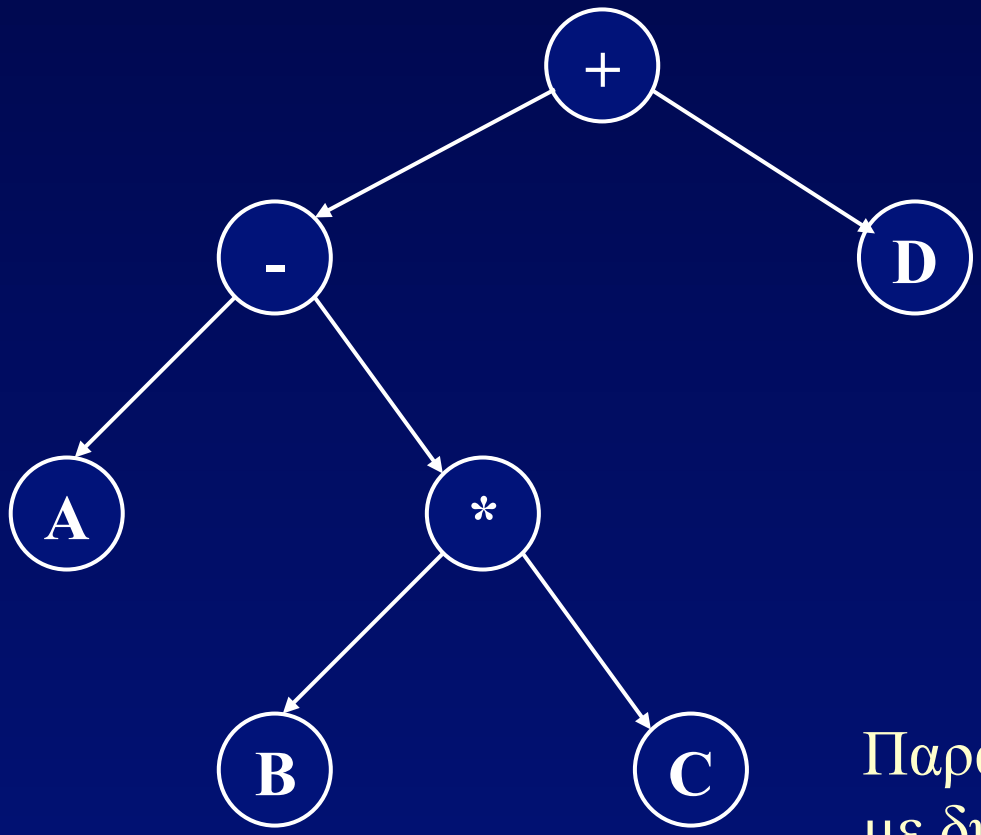
```
int eisagogi_dexi(typos_deikti s,typos_deikti t)
{/*     Προ : Ο κόμβος s δεν είναι ο κόμβος κεφαλή του
                δυαδικού δέντρου με κλωστές.
       Μετά: Αν ο κόμβος s δεν είναι ο κόμβος κεφαλή του
       δυαδικού δέντρου με κλωστές, τότε ο κόμβος t
       εισάγεται σαν το δεξι παιδί του κόμβου s σε
       ένα δυαδικό δέντρο με κλωστές και επιστρέφεται 1
       αλλιώς επιστρέφεται Ο */

       typos_deikti prosorinos;

       if (!kefali(s))

       {       t->dpaidi = s->dpaidi;

               t->dklosti = s->dklosti;
```

συνέχεια

```
            t->apaidi = s;
            t->aklosti = 1; /* apaidi είναι μια κλωστή */
            s->dpaidi = t; /* προσάρτηση του t στο s */
            s->dklosti = 0;
            if (!(t->dklosti)) /* ο s έχει δεξί παιδί */
            {       prosorinos = endo_epomenos(t);
                    prosorinos->apaidi = t;
            }

            return 1;
        }
        else
            return 0;
}
```

Ανομοιογενή δυαδικά δέντρα (δέντρα παράστασης)

Συχνά τα δεδομένα που περιέχονται σε διαφορετικούς κόμβους δεν είναι όλα του ίδιου τύπου.

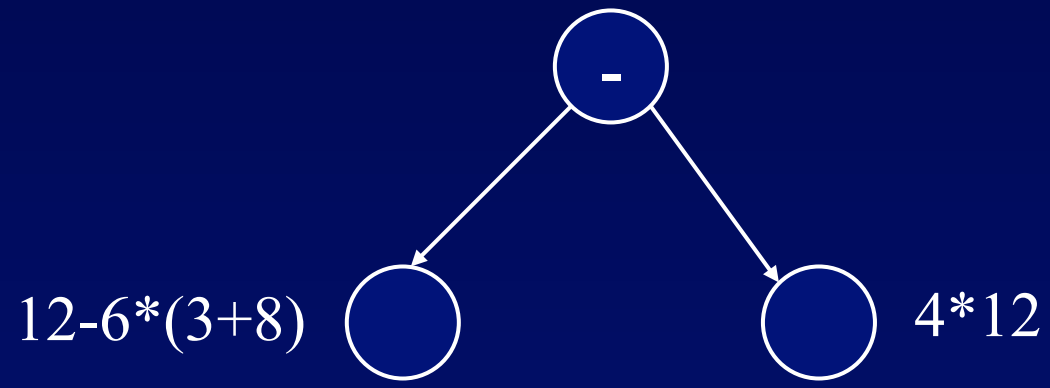


Παράσταση της A-B*C+D με δυαδικό δέντρο

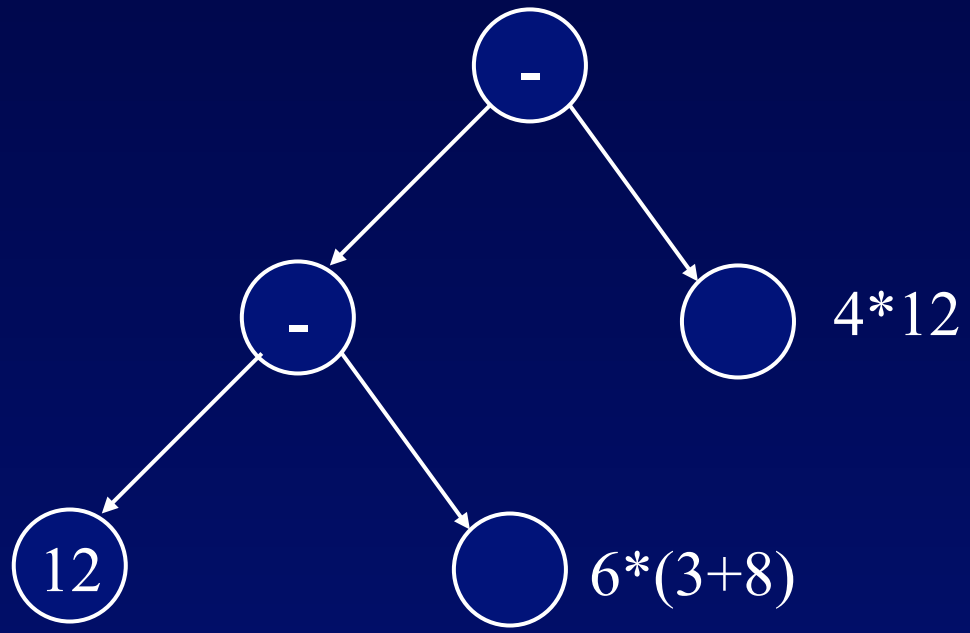
Για τη μετατροπή μιας ενδοδιατεταγμένης παράστασης στο αντίστοιχο δυαδικό δέντρο, το οποίο ονομάζεται **δέντρο παράστασης**, θα πρέπει σε κάθε βήμα να αναζητείται ο τελεστής εκείνος ο οποίος θα εκτελεστεί τελευταίος (έχει τη μικρότερη ιεραρχία).

Δημουργία δυαδικού δέντρου παράστασης της 12-6*(3+8)-4*12

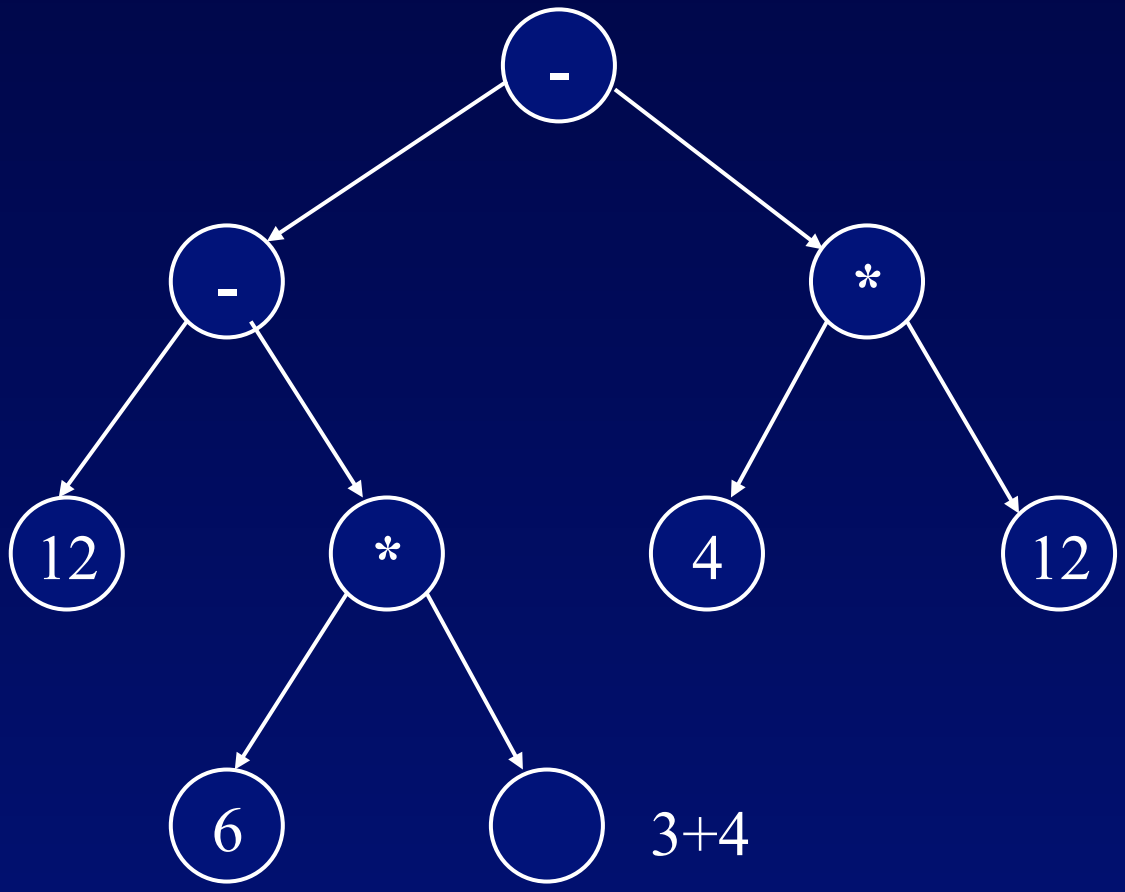
1ο Βήμα



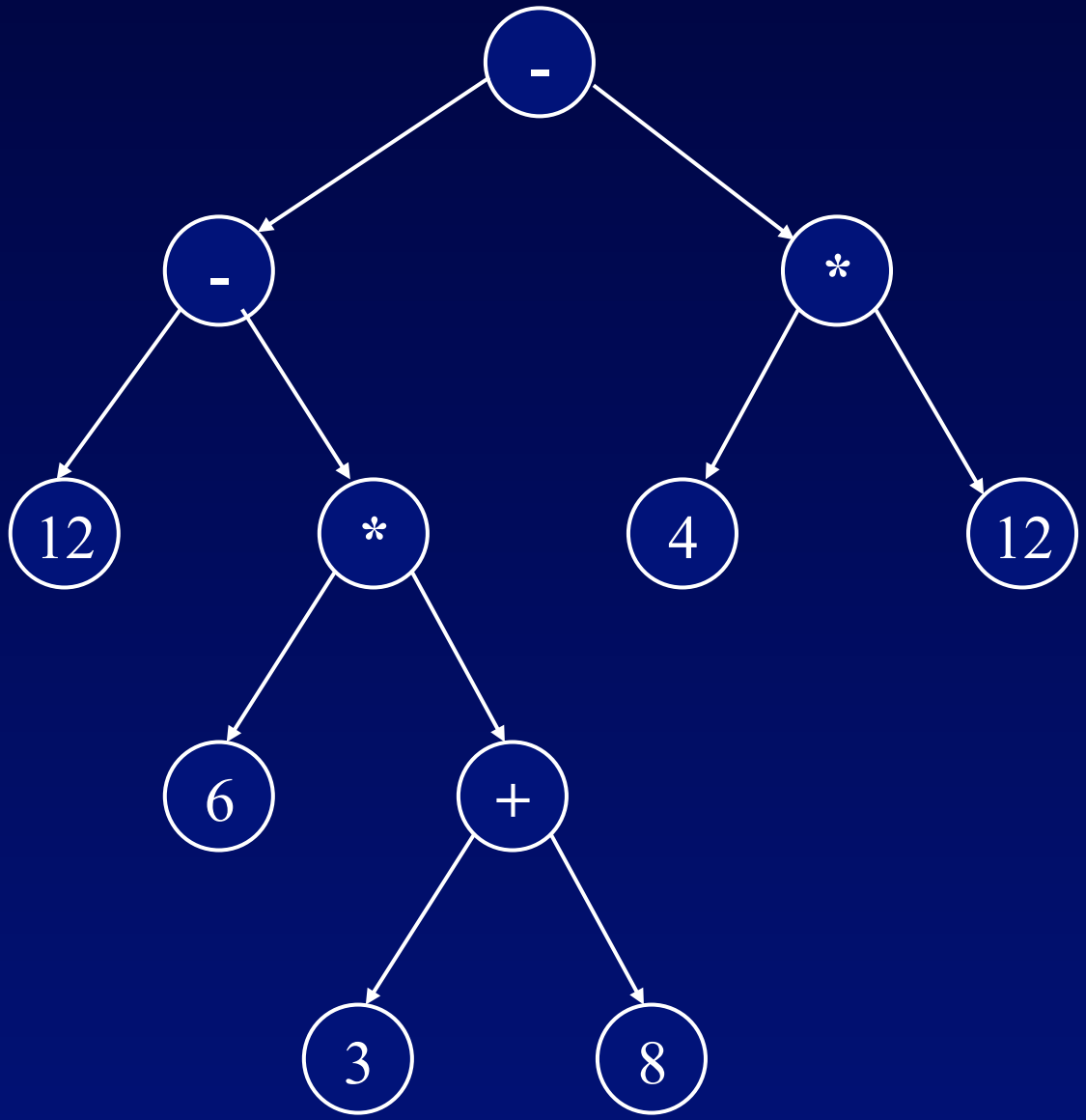
2ο Βήμα



3ο Βήμα



3ο Βήμα



Για την υλοποίηση ενός τέτοιου δέντρου θα χρησιμοποιηθεί μια μεταβαλλόμενη εγγραφή για την παράσταση κάθε κόμβου του. Εχουμε λοιπόν τους ορισμούς :

```
typedef enum { oros,telest } typos_dedomenon;
typedef struct typos_komvou *typos_deikti;
typedef struct typos_komvou
{       typos_dedomenon simadi;
        float arithmos;
        char  xaraktiras;
        typos_deikti apaidi,dpaidi;
};

typos_deikti riza;
```

Στους παραπάνω ορισμούς παρατηρούμε ότι μόνο οι κόμβοι που περιέχουν τελεστές έχουν πεδία δεικτών apaidi και dpaidi για τους υπόλοιπους δεν χρειάζονται γιατί είναι φύλλα. Στη συνέχεια παρουσιάζεται ένα αναδρομικό υποπρόγραμμα που υπολογίζει την τιμή της παράστασης.

```
float ypologismos_dentrou(typos_deikti riza)
{/* Επιστρέφει την τιμή της παράστασης η οποία είναι
        καταχωρημένη στο ανομοιογενές δυαδικό δέντρο riza.
 */

        float oros1,oros2;
        char  symbolo;
```

συνέχεια

```
        switch (riza->simadi)
        {
                case oros: /* Ενας μοναδικός όρος */
                        return riza->arithmos;
                case telest:
                        oros1 = ypologismos_dentrou(riza->apaidi);
                        oros2 = ypologismos_dentrou(riza->dpaidi);
                        symbolo = riza->xaraktiras;
                        return telestis(symbolo,oros1,oros2);
        }
}
```

Δυαδικά Δέντρα Αναζήτησης (Binary Search Trees)

Ορισμός : Ενα δυαδικό δέντρο αναζήτησης t είναι ένα δυαδικό δέντρο, το οποίο είναι κενό ή κάθε κόμβος του έχει ένα περιεχόμενο και:

(i) όλα τα περιεχόμενα στο αριστερό υποδέντρο του t είναι μικρότερα (αριθμητικά ή αλφαβητικά) από το περιεχόμενο της ρίζας του t.

(ii) όλα τα περιεχόμενα στο δεξί υποδέντρο του t είναι μεγαλύτερα από το περιεχόμενο της ρίζας του t.

(iii) το αριστερό και το δεξί υποδέντρο του t είναι επίσης δυαδικά δέντρα αναζήτησης.

Οι βασικές πράξεις που ορίζουν τον ΑΤΔ δυαδικό δέντρο αναζήτησης είναι οι ακόλουθες:

1. **Δημιουργία** : Δημιουργεί ένα κενό ΔΔΑ.
2. **Κενό** : Ελέγχει αν το ΔΔΑ είναι κενό.
3. **Αναζήτηση** : Επιστρέφει τη θέση του κόμβου με δεδομένο περιεχόμενο.
4. **Εισαγωγή** : Εισάγει ένα κόμβο στο ΔΔΑ έτσι ώστε το προκυπτόμενο δέντρο να είναι ένα ΔΔΑ.
5. **Διαγραφή** : Διαγράφει ένα κόμβο με δεδομένο περιεχόμενο έτσι ώστε το προκυπτόμενο δέντρο να είναι ένα ΔΔΑ.

Αναζήτηση

Αναδρομικό υποπρόγραμμα :

```
void anazitisi_dentrou(typos_deikti riza,
                       typos_stoixeiou stoixeio,
                       typos_deikti *prosorinos,
                       int *vrethike)
{/* Μετά: Αν το περιεχόμενο ενός κόμβου του ΔΔΑ riza
        είναι ίσο με stoixeio τοτε το *prosorinos
        δείχνει τον κόμβο αυτό και η *vrethike
        είναι 1,αλλιώς το *vrethike είναι 0 και
         το *prosorinos είναι NULL. */
```

συνέχεια

```
if (keno_dentro(riza))
{       *prosorinos = NULL;
        *vrethike = 0;
}
else
{       if (riza->dedomena == stoixeio)
        {       *prosorinos = riza;
                *vrethike = 1;
        }
        else
        {       if ( stoixeio < riza->dedomena )
                /* αναζήτηση αριστερού υπόδεντρου */
                        anazitisi_dentrou(riza->apaidi,
                        stoixeio,prosorinos,vrethike);
```

συνέχεια

```
                else
                /* αναζήτηση δεξιού υπόδεντρου */
                        anazitisi_dentrou(riza->dpaidi,
                        stoixeio,prosorinos,vrethike);
                }
        }
}
```

Μη Αναδρομικό υποπρόγραμμα :

```
void anazitisi_dentrou(typos_deikti riza, typos_stoixeiou stoixeio,
        typos_deikti *prosorinos,int *vrethike)
{
        *prosorinos = riza;
        *vrethike = 0;
        while ( !(*vrethike) && ((*prosorinos) !=  NULL))
                if (stoixeio<((*prosorinos)->dedomena))
                        *prosorinos = (*prosorinos)->apaidi;
                else
                        if (stoixeio > ((*prosorinos)->dedomena))
                                *prosorinos=(*prosorinos)->dpaidi;
                        else
                                *vrethike = 1;
}
```

Πολυπλοκότης : O(n)

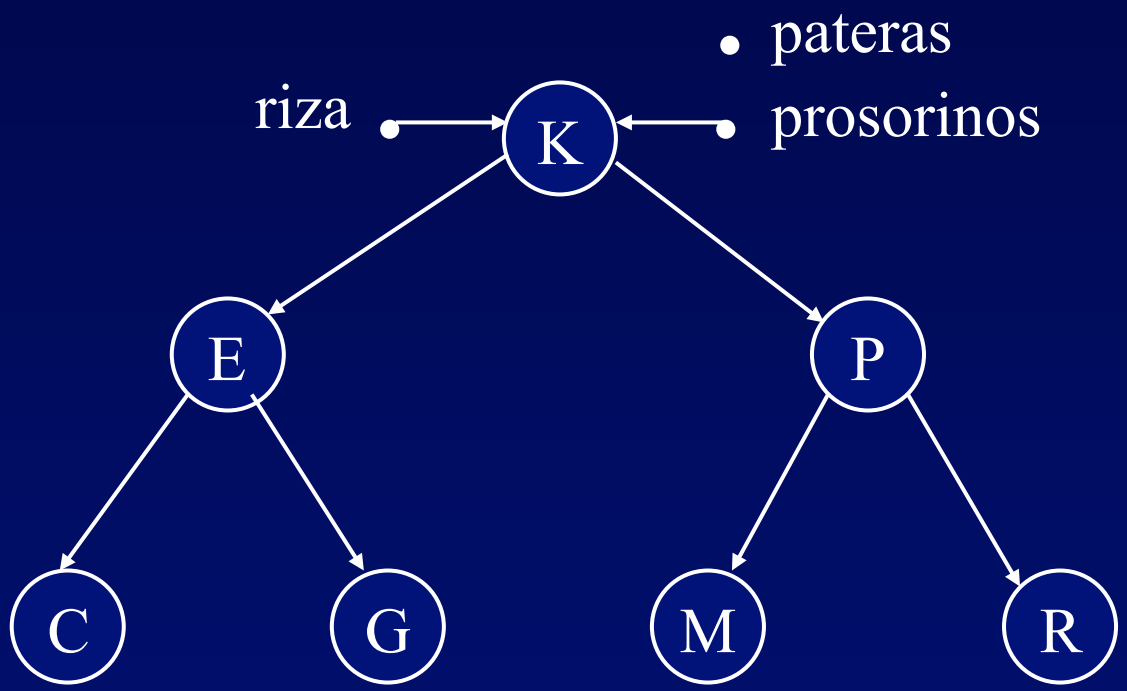
Εισαγωγή κόμβου

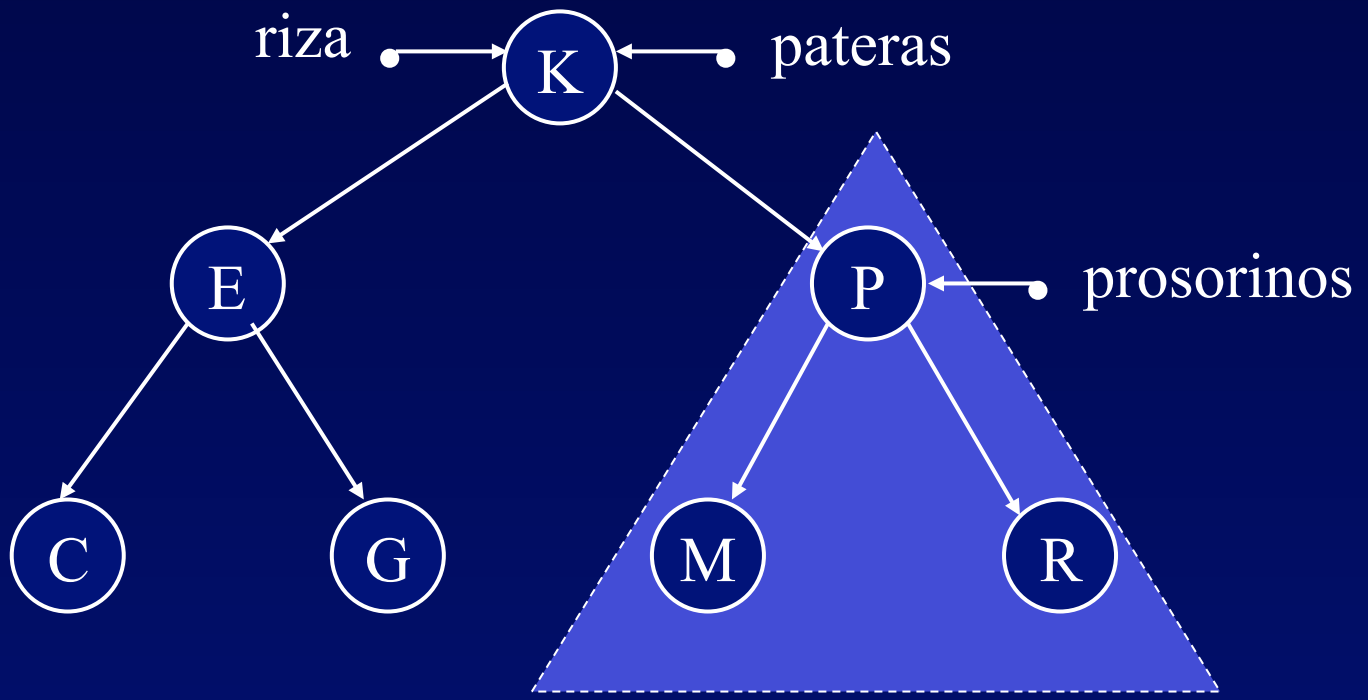
```
int eisagogi_dentro(typos_deikti *riza,typos_stoixeiou stoixeio)
{/* Μετά: Αν το stoixeio δεν ανήκει στο ΔΔΑ τότε
        εισάγεται και επιστρέφεται 1, αλλιώς επιστρέφεται 0. */

         int eisagogi;
         if (keno_dentro(*riza))
         {*riza = (typos_deikti) malloc(sizeof(struct typos_komvou));
                  (*riza)->dedomena = stoixeio;
                  (*riza)->apaidi   = NULL;
                  (*riza)->dpaidi   = NULL;
                  eisagogi = 1;
         }
```

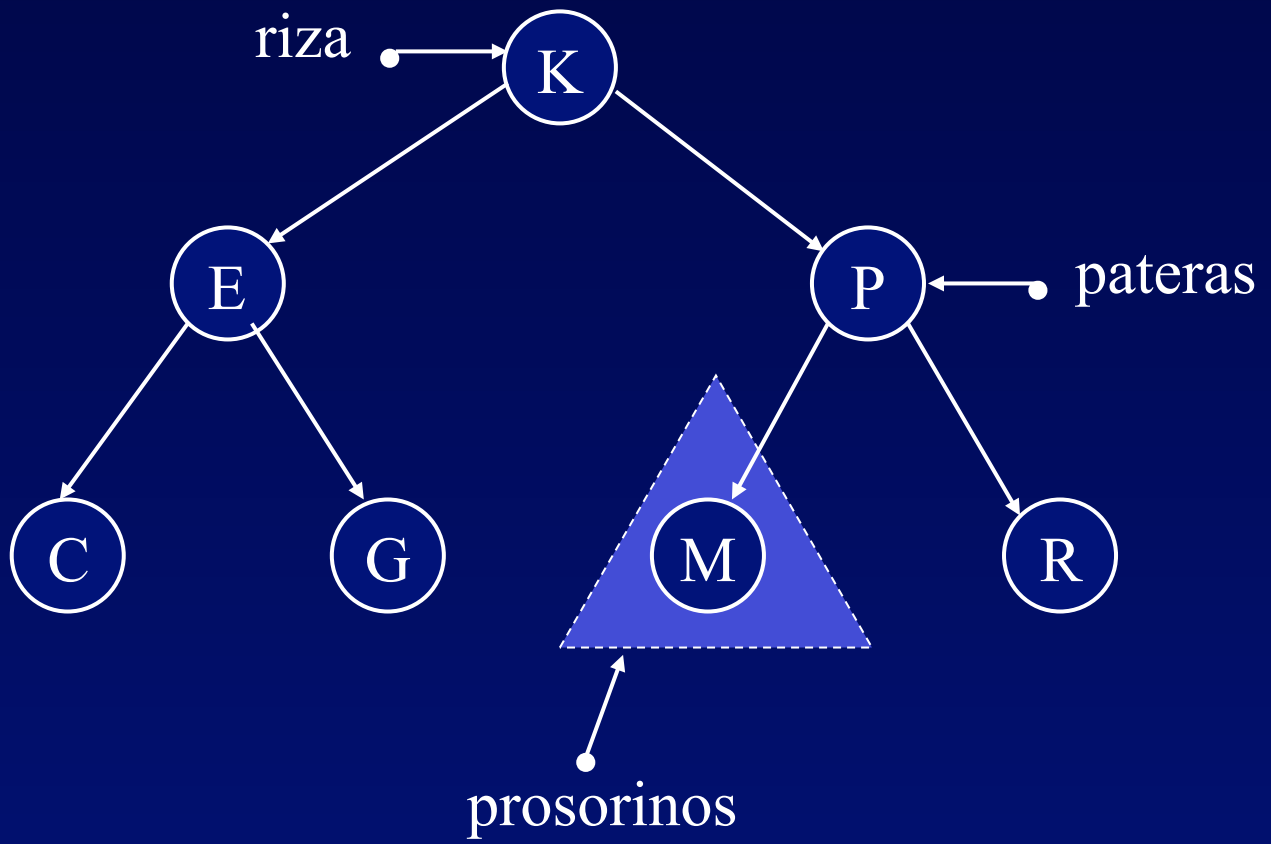
συνέχεια

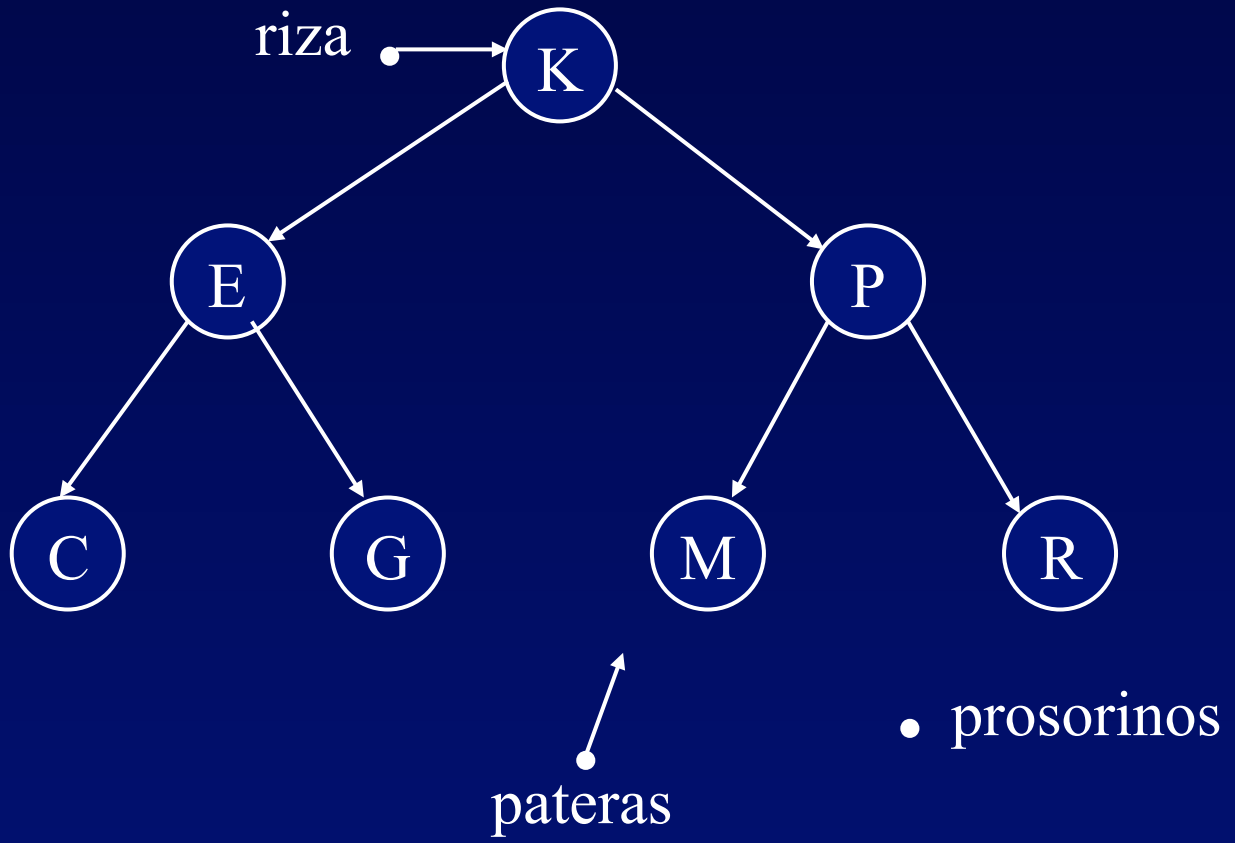
```
        else
                if (stoixeio < (*riza)->dedomena )
                /* εισαγωγή στο αριστερό υπόδεντρο */
                        eisagogi = eisagogi_dentro(
                                &((*riza)->apaidi),stoixeio);
                else
                        if (stoixeio > (*riza)->dedomena )
                        /* εισαγωγή στο δεξιό υπόδεντρο */
                                eisagogi = eisagogi_dentro(
                                        &((*riza)->dpaidi),stoixeio);
                        else
                        /* Ο κόμβος είναι ήδη στο δέντρο */
                                eisagogi = 0;
        return eisagogi;
}
```

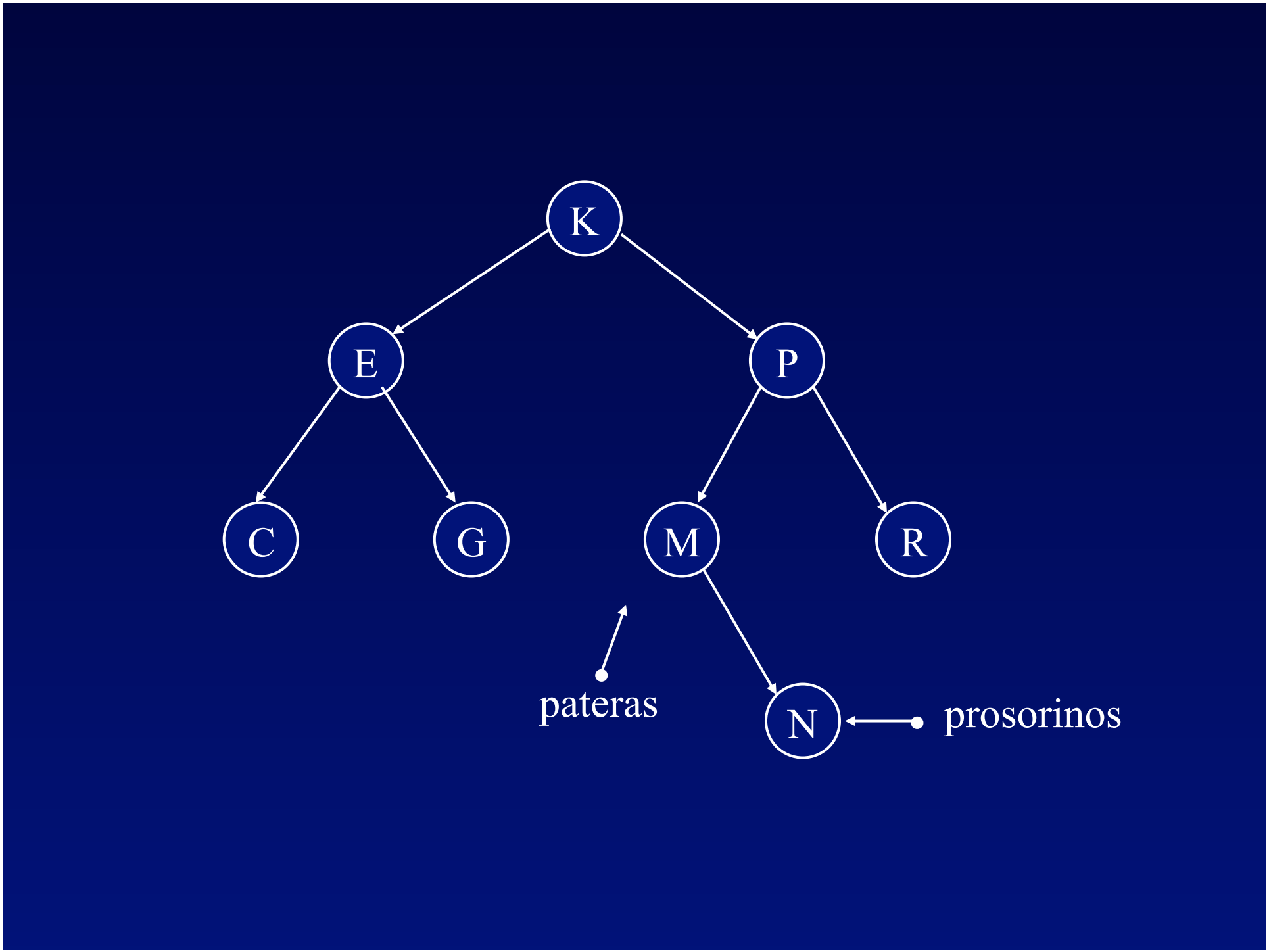
Ας υποθέσουμε ότι επιθυμούμε την εισαγωγή του N στο ακόλουθο δυαδικό δέντρο αναζήτησης :




riza
K
pateras
E
P
prosorinos
C
G
M
R


riza
K
E
P
pateras
C
G
M
R
prosorinos


riza
K
E
P
C
G
M
R
pateras
prosorinos


K
E
P
C
G
M
R
pateras
N
prosorinos

```
typos_deikti anazitisi_patera(typos_deikti riza,
                              typos_stoixeiou stoixeio,
                              typos_deikti *pateras, int *vrethike)
{       /* Μετά: Αν το ΔΔΑ riza δεν είναι κενό και το stoixeio δεν
        ανήκει σε αυτό τοτε το *vrethike είναι 0 και το *pateras
        είναι η διεύθυνση του κόμβου που θα ήταν
        πατέρας του stoixeio. Αν το riza δεν είναι κενό και το stoixeio
        ανήκει στο δέντρο και δεν βρίσκεται στη ρίζα του,τότε το
        *vrethike είναι 1 και το *pateras είναι η διεύθυνση του
        κόμβου πατέρα του. Αν το stoixeio βρίσκεται στη ρίζα
        του δέντρου τότε το *vrethike είναι 1 και το *pateras
        είναι NULL. Αν το δέντρο είναι κενό τότε το *vrethike
        είναι 0 και το *pateras είναι NULL. Αν το stoixeio ανήκει
        στο δέντρο τότε η συνάρτηση επιστρέφει τη διεύθυνση
        του κόμβου που περιέχει το stoixeio, αλλιώς η συνάρτηση
        επιστρέφει NULL. */
```

συνέχεια

```
        typos_deikti prosorinos;
        prosorinos = riza;
        *pateras = NULL;*vrethike = 0;
        while (!(*vrethike) && (prosorinos!=NULL))
        {       if ( stoixeio < prosorinos->dedomena )
                {       *pateras = prosorinos;
                        prosorinos = prosorinos->apaidi;}
                else
                        if ( stoixeio > prosorinos->dedomena )
                        {       *pateras = prosorinos;
                                prosorinos = prosorinos->dpaidi;}
                        else
                                *vrethike = 1; /* ο κόμβος υπάρχει */
        }
        return prosorinos;
}
```

```
int eisagogi_dentro(typos_deikti *riza, typos_stoixeiou
stoixeio)
{       /* Μετά: Αν το stoixeio δεν ανήκει στο ΔΔΑ τότε
        εισάγεται και επιστρέφεται 1, αλλιώς
        επιστρέφεται 0. */

        typos_deikti prosorinos,pateras;
        int vrethike;
        anazitisi_patera(*riza,stoixeio,&pateras,&vrethike);
        if (vrethike)
                return 0;
        else
        {       prosorinos = (typos_deikti) malloc(sizeof(struct
                                typos_komvou));
                prosorinos->dedomena = stoixeio;
                prosorinos->apaidi   = NULL;
                prosorinos->dpaidi   = NULL;
```

συνέχεια

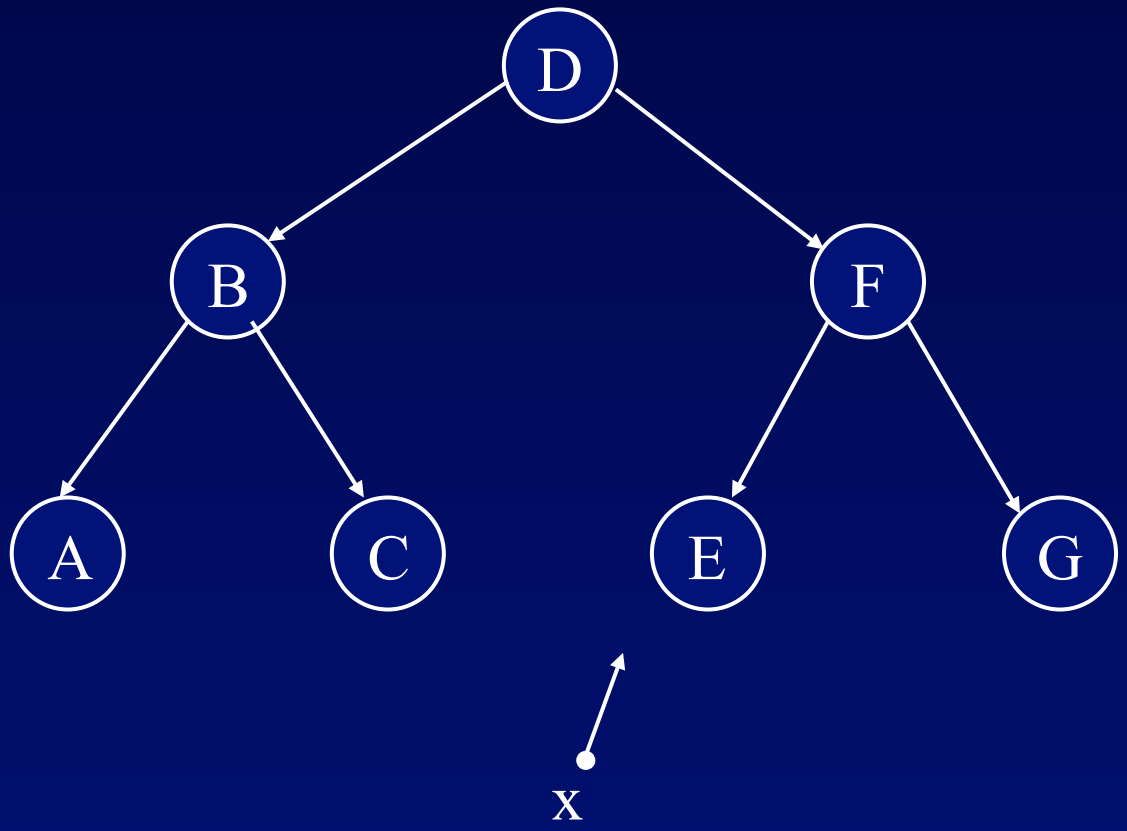
```
            if ( pateras == NULL ) /* Κενό Δέντρο */
                    *riza = prosorinos;
            else
                    if ( stoixeio < pateras->dedomena )
                            pateras->apaidi = prosorinos;
                    else
                            pateras->dpaidi = prosorinos;
             return 1;
        }
}
```

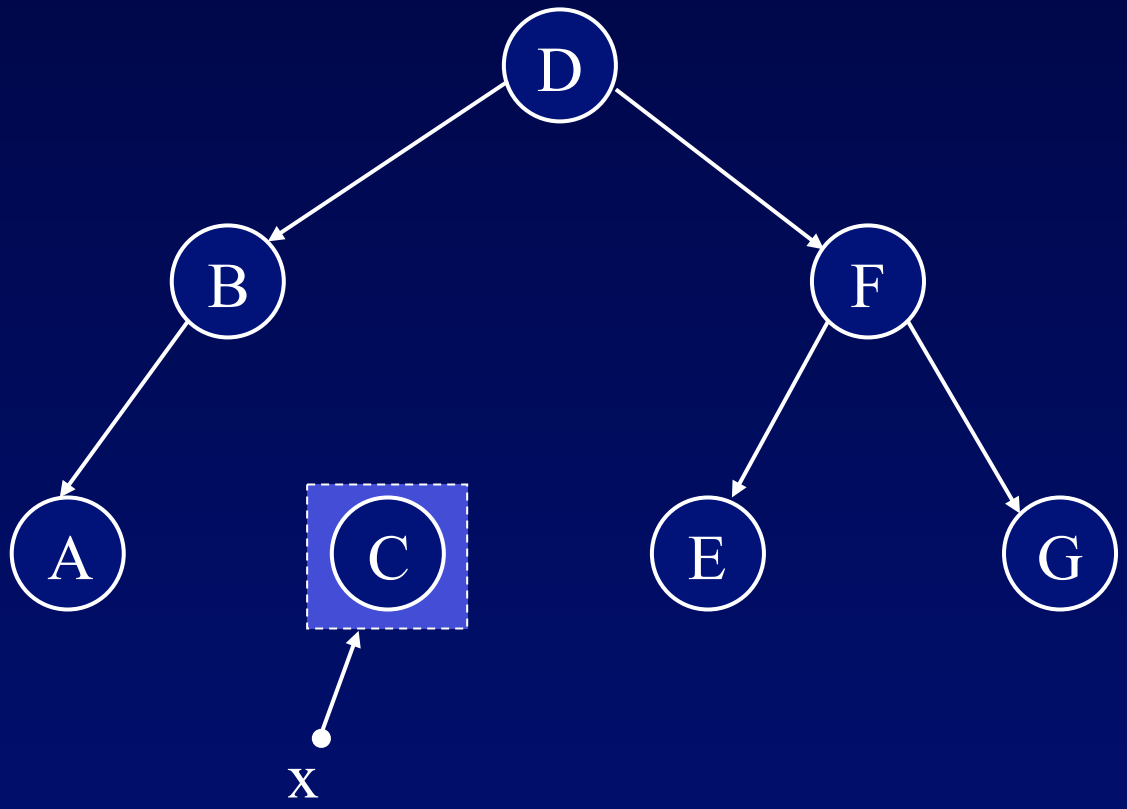
Διαγραφή κόμβου

Για τη διαγραφή ενός κόμβου x από ένα ΔΔΑ διακρίνουμε τρεις περιπτώσεις:

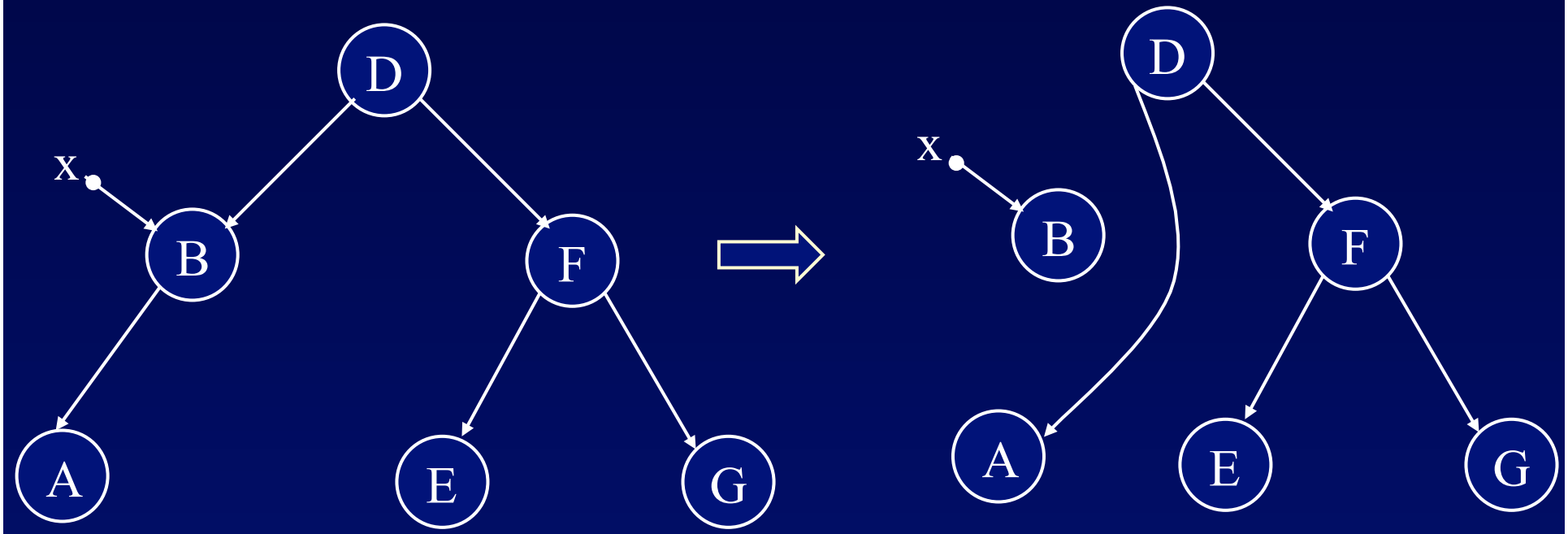
1. Ο κόμβος x είναι φύλλο.
2. Ο κόμβος x έχει ένα παιδί.
3. Ο κόμβος x έχει δύο παιδιά.

1. Ο κόμβος x είναι φύλλο

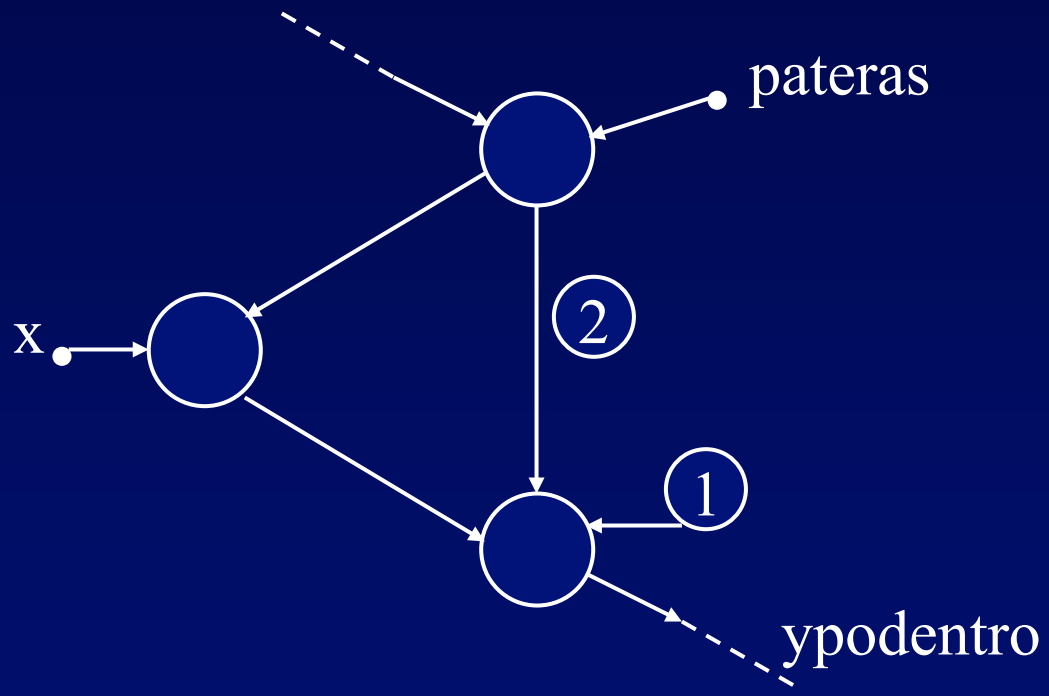



D
B
F
A
C
E
G
x

2. Ο κόμβος x έχει ένα παιδί

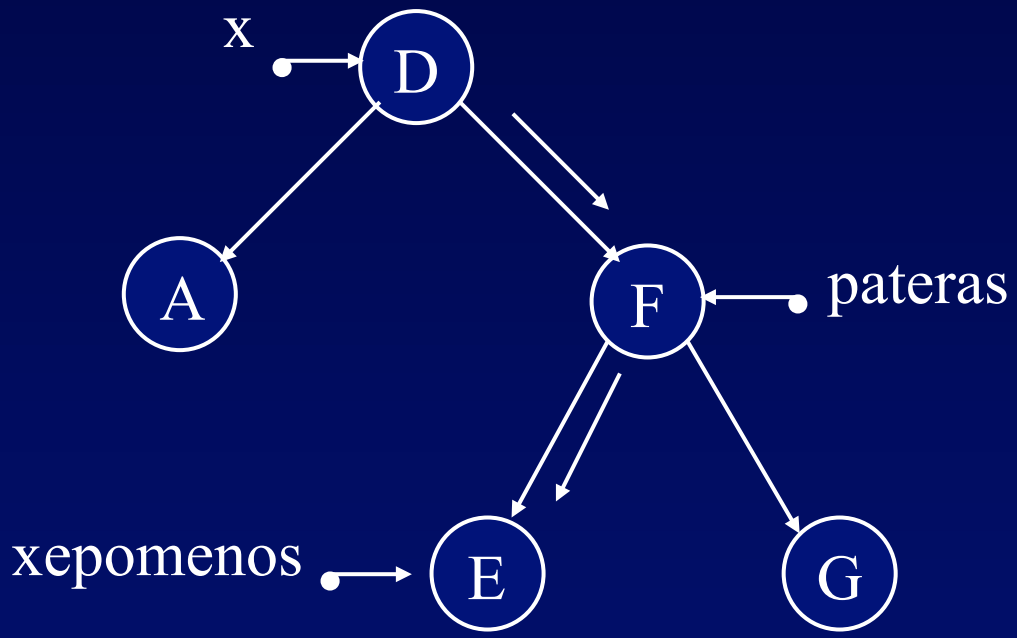


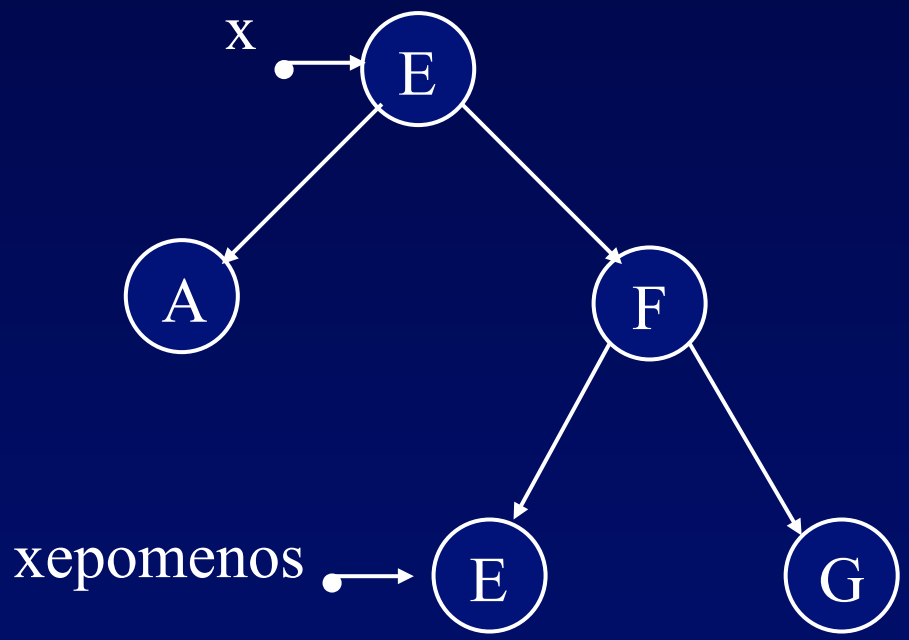
Διαγραφή κόμβου με το πολύ ένα υποδέντρο

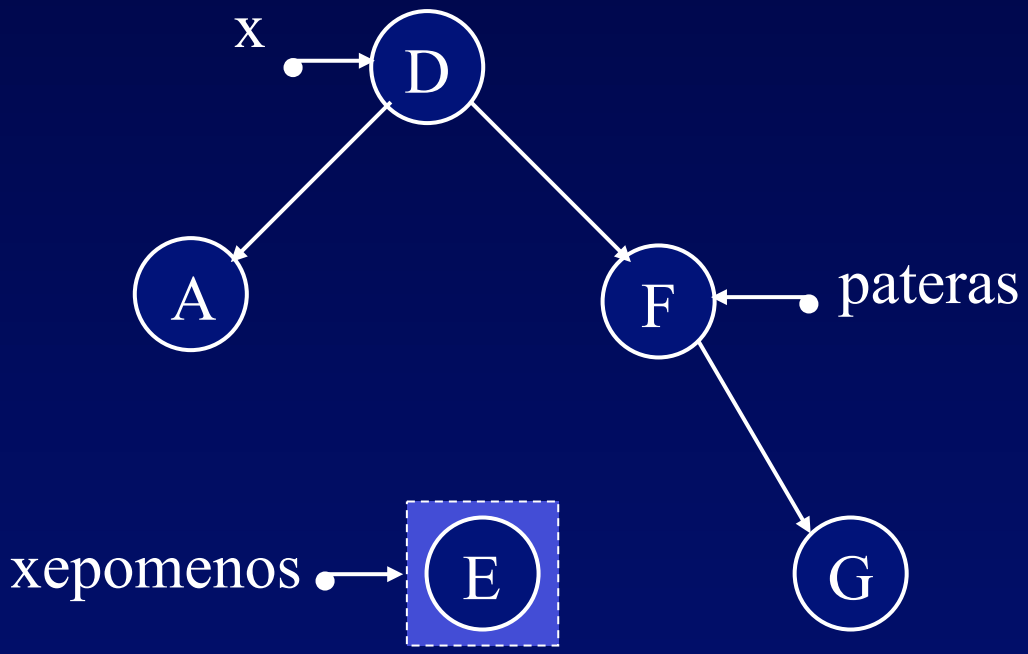


```
ypodentro = x .dpaidi;
{ο x δεν έχει αριστερό υπόδεντρο}
if (ypodentro == NULL)
        ypodentro = x .apaidi;
if (pateras == NULL)
        /*διαγράφεται η ρίζα*/
        riza = ypodentro;
else
        if (pateras .apaidi == x)
                pateras .apaidi = ypodentro;
        else
                pateras .dpaidi = ypodentro;
```

3. Ο κόμβος x έχει δύο παιδιά




x
E
A
F
xepomenos
E
G


x
D
A
F
pateras
xepomenos
E
G

Μη Αναδρομικό

```
void diagrafi_dentrou (typos_deikti *riza, typos_stoixeiou stoixeio)
/*Εντοπίζει τον κόμβο με περιεχόμενο stoixeio και το διαγράφει
    από το ΔΔΑ*/
        typos_deikth   x;    /*δείχνει τον κόμβο
                                με περιεχόμενο stoixeio*/
        typos_deikth   pateras; /*ο πατέρας του x ή του xepomenos*/
        typos_deikth   xepomenos; /*ο ενδοδιατεταγμένος
                                    επόμενος του x*/
        typos_deikth   ypodentro; /*δείχνει το υποδέντρο του x*/
        char           vrethike;  /*true αν το περιεχόμενο της stoixeio
                                    βρίσκεται στο ΔΔΑ*/
```

συνέχεια

```
{
        anazitisi_patera (*riza, stoixeio, x, &pateras, &vrethike);
        if (!vrethike)
                printf ( "Ο κόμβος δεν υπάρχει στο ΔΔΑ" );
        else
        {
                if ((x->apaidi != NULL) && (x->dpaidi != NULL))
                { /*ο κόμβος έχει δύο παιδιά*/
                  /*εντοπισμός του ενδοδιατεταγμένου επόμενου
                    κόμβου και του πατέρα του*/
                        xepomenos = x->dpaidi;
                        pateras = x;
                        while (xepomenos->apaidi != NULL)
                        { /*μονοπάτι αριστερών υποδέντρων*/
                                pateras = xepomenos;
```

συνέχεια

```
                    xepomenos := xepomenos^.apaidi
            }
            /*ανταλλαγή των περιεχομένων των xepomenos
            και και αλλαγή της τιμής του x ώστε να δείχνει
            τον xepomenos που πρόκειται να διαγραφτεί*/
            x->dedomena = xepomenos->dedomena;
            x = xepomenos;
} /*Προχώρησε για τις περιπτώσεις όπου ο κόμβος
  έχει 0 ή 1 παιδί*/
ypodentro = x ->dpaidi;
if (ypodentro ==NULL)
        ypodentro = x ->apaidi;
if (pateras == NULL) /*διαγράφεται η ρίζα*/
        riza = ypodentro;
```

συνέχεια

```
            else
                    if (pateras->apaidi == x)
                            /*αριστερό παιδί του πατέρα*/
                            pateras->apaidi = ypodentro;
                    else
                            pateras->dpaidi = ypodentro;
             free(x);
        }
} /*diagrafi_dentrou*/
```

Αναδρομικό

```
int diagrafi(typos_deikti *riza, typos_stoixeiou stoixeio)
{       /* Μετά: Αν το stoixeio ανήκει στο ΔΔΑ τότε διαγράφεται και η
συνάρτηση επίστρέφει 1,αλλιώς 0. */
        int diagraf;
        if (keno_dentro(*riza)) /* Δεν υπάρχει στο ΔΔΑ */
                diagraf = 0;
        else
                if ((*riza)->dedomena < stoixeio )
                diagraf = diagrafi (&((*riza)->dpaidi),stoixeio);
        else
                if ((*riza)->dedomena > stoixeio )
                        diagraf = diagrafi ( &((*riza)->apaidi),stoixeio);
                else
                        {diagrafi_komvou(riza);
                         diagraf = 1;}
        return diagraf;
}
```

Το υποπρόγραμμα diagrafi-komvou(riza) διαγράφει τον κόμβο που δείχνει η riza.

```
void diagrafi_komvou(typos_deikti *riza)
{       /* Προ : Το *riza είναι ένα μη κενό ΔΔΑ.Μετά: Αν ο κόμβος *riza είχε 1
ακριβώς παιδί τότε διαγράφθηκε το περιεχόμενο της ρίζας από το ΔΔΑ και το παιδί
έγινε   ρίζα. Αν ο κόμβος *riza δεν είχε κανένα παιδί τότε έγινε κενό ΔΔΑ. Αν ο
κόμβος  *riza είχε 2 παιδιά τότε ιαγράφθηκε ο ενδοδιατεταγμένος επόμενός
        του και το περιεχόμενό του αντιγράφθηκε στον *riza */

        typos_deikti prosorinos;
        prosorinos = *riza;
        if ((prosorinos->dpaidi) == NULL )
        {       (*riza) = prosorinos->apaidi;
                free(prosorinos);
        }
        else
                if ((prosorinos->apaidi) == NULL )
                {       (*riza) = prosorinos->dpaidi;
                        free(prosorinos);}
                else
                        andalagi(&((*riza)->dpaidi),*riza);
}
```

Οπου το υποπρόγραμμα andalagi(p,q) εντοποίζει τον ενδοδιατεταγμένο επόμενο του κόμβου που δείχει 0 p.

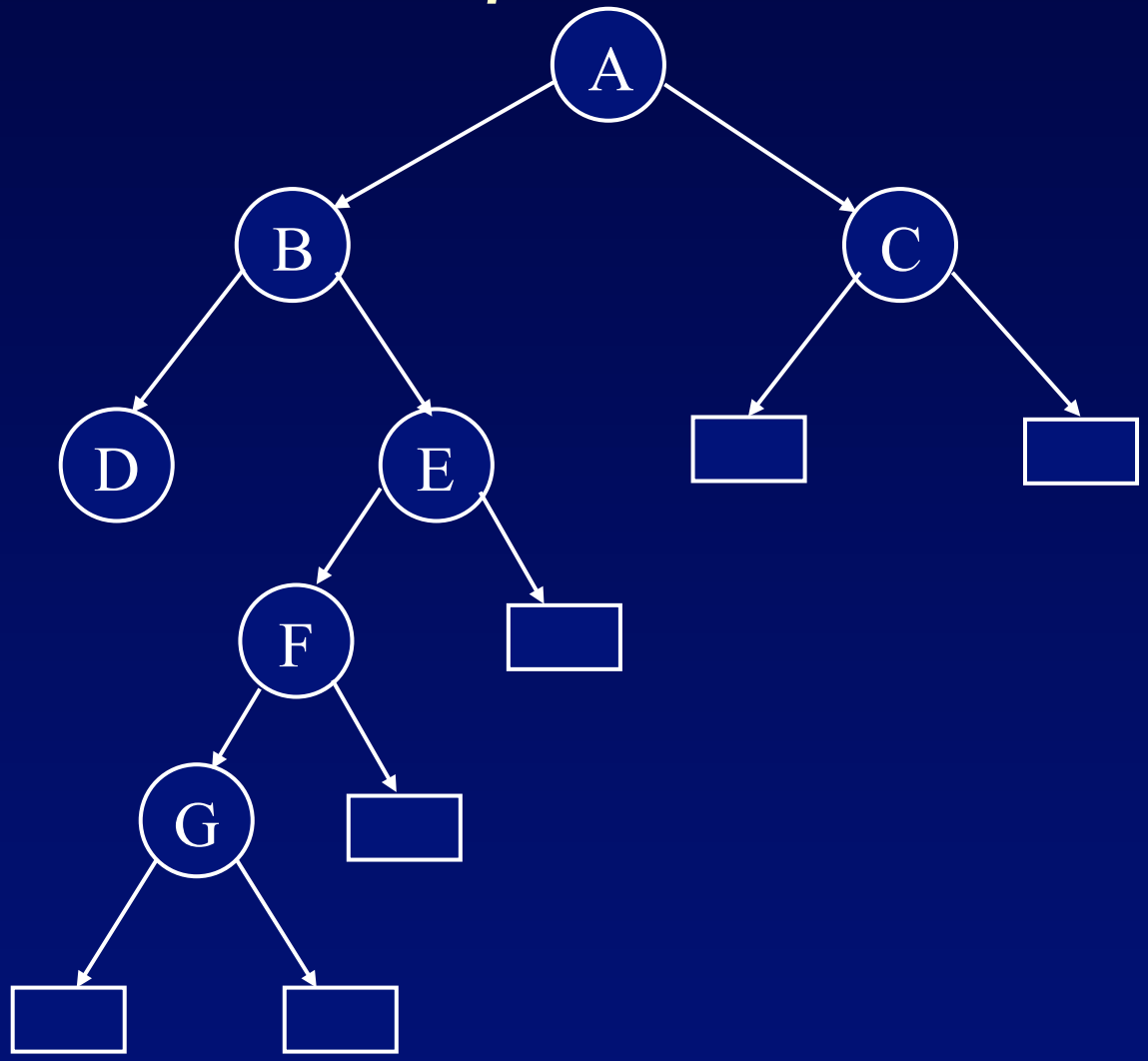
```
void andalagi(typos_deikti *epomenos, typos_deikti riza)
{/*     Προ : Το riza δείχνει κάποιο κόμβο ΔΔΑ και το
        *epomenos το δεξί παιδί του.
        Μετά: Το περιεχόμενο του κόμβου που έχει διεύθυνση
                riza έγινε ίσο με το περιεχόμενο του
        ενδοδιατεταγμένου  επομενού του και απελευθερώθηκε ο
χώρος που   καταλαμβανόταν από τον ενδοδιατεταγμένο
επόμενο. */

        /* εύρεση του ενδοδιατεταγμένου επόμενου */
        if (((*epomenos)->apaidi) != NULL)
                andalagi(&((*epomenos)->apaidi),riza);
```

συνέχεια

```
        else
                /* βρέθηκε ο ενδοδιατεταγμένος επόμενος */
        {
                riza->dedomena=(*epomenos)->dedomena;
                free(*epomenos);
         /* αντικατάσταση του *epomenos με το δεξί παιδί του */
                *epomenos = (*epomenos)->dpaidi;
        }
}
```

Εφαρμογές δυαδικών δέντρων : Κώδικες Huffman

Επεκταμένο Δυαδικό Δέντρο



$$l_G = 1 + 1 + 1 + 1 = 4$$

$$E = 3 + 3 + 5 + 5 + 5 + 4 + 3 + 2 + 2 = 27$$

$$I = 1 + 1 + 2 + 2 + 3 + 4 = 13$$

Πρόταση : Αν s_i είναι το πλήθος των εξωτερικών κόμβων στο επίπεδο i , τότε :

$$E = \sum_{i=2}^{h+1} s_i (i - 1)$$

Πρόταση : Αν s_i είναι το πλήθος των εξωτερικών κόμβων στο επίπεδο i , τότε :

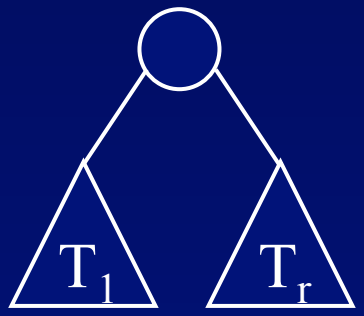
$$I = \sum_{i=2}^{h} n_i (i - 1)$$

μέση απόσταση ενός εσωτερικού κ κόμβ $= \frac{I(T)}{n}$

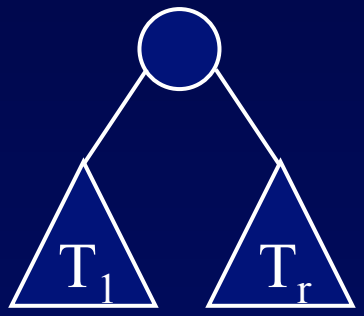
μέση απόσταση ενός εξωτερικού κκόμβο $= \frac{E(T)}{n+1}$

$I(\square) = 0$

$$I\left(T = \right) = I(T_l) + I(T_r) + n - 1$$



$$E(\square) = 0$$

$$E\left(T = \right.$$  $$\left. \right) = E(T_l) + E(T_r) + n + 1$$

Πρόταση : Για ένα δυαδικό δέντρο Τ με n εσωτερικούς κόμβους ισχύει :

$$D(T) = 2n$$

όπου

$$D(T) = E(T) - I(T)$$

Εχουμε ότι :

$$E(T) = I(T) + 2n$$

πράγμα που σημαίνει ότι αρκεί να μελετηθούν οι ιδιότητες μόνο του E(T) ή του I(T).

Ο μέσος αριθμός συγκρίσεων S(n) σε μια επιτυχημένη αναζήτηση είναι : $S(n) = 1 + \frac{I}{n}$

όπου υποτίθεται ότι όλοι οι κόμβοι έχουν την ίδια πιθανότητα εμφάνισης.

Ο μέσος αριθμός συγκρίσεων U(n) σε μια αποτυχημένη αναζήτηση είναι : $$U(n) = \frac{E}{n+1}$$

Συνδιάζοντας τις ανωτέρω σχέσεις εύκολα προκύπτει :

$$S(n) = \left(1 + \frac{1}{n}\right) U(n) - 1$$

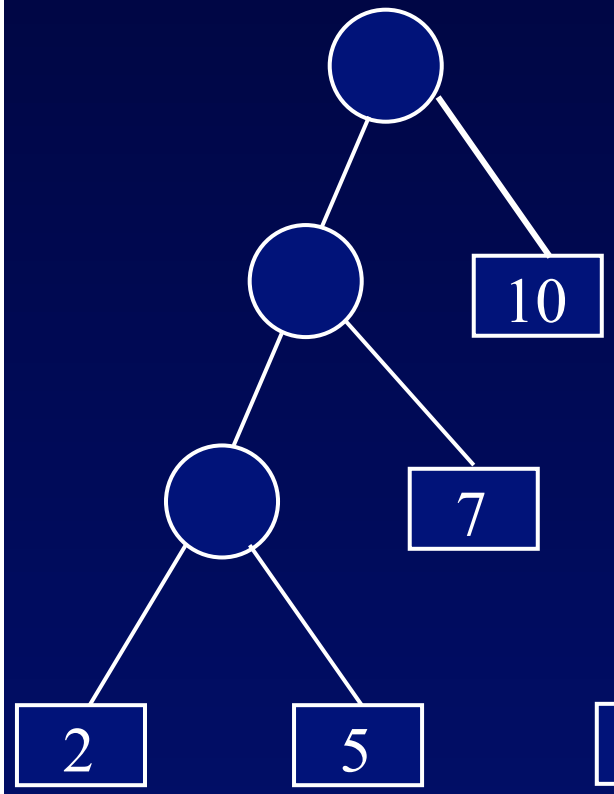
Συνεπώς ο μέσος αριθμός συγκρίσεων για μια επιτυχημένη αναζήτηση είναι περίπου ο ίδιος με εκείνον για μια αποτυχημένη αναζήτηση. Γνωρίζοντας δηλαδή ότι ένα στοιχείο βρίσκεται στη λίστα, η αναζήτησή του με τη μέθοδο των συγκρίσεων των κλειδιών του δεν προσφέρει μεγάλη βοήθεια.

Ποιά είναι όμως η ελάχιστη και μέγιστη δυνατή τιμή του I από όλα τα δυαδικά δέντρα με n εσωτερικούς κόμβους ;

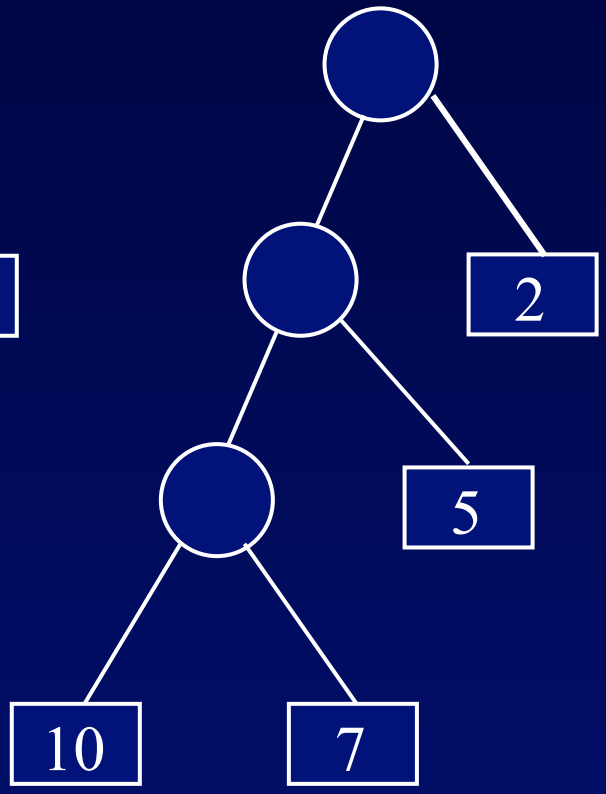
$$I = 0 + 1 + \ldots + (n-2) + (n-1) = n(n-1)/2. \qquad \mathbf{O(n^2)}$$

Η ελάχιστη τιμή του I επιτυγχάνεται για ένα πλήρες δυαδικό δέντρο :

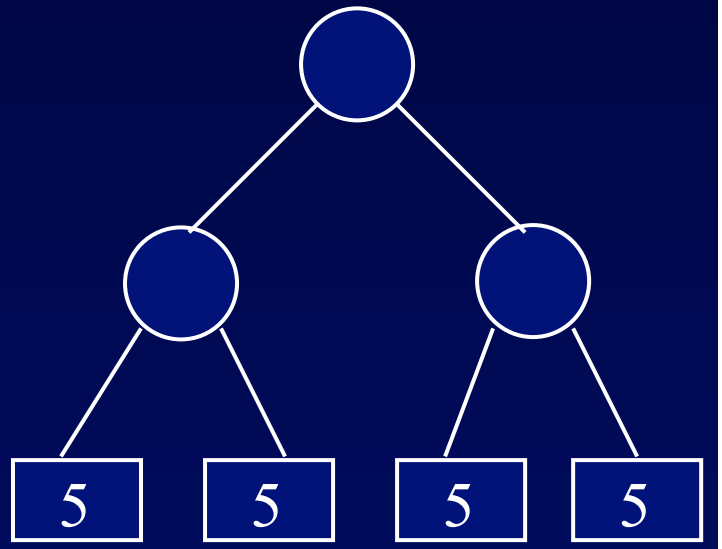
$$I_{min} = O(n\log_2 n)$$



E = 45



E = 63



E = 48

$$E = \sum_{1 \le i \le n} w_i I_i$$

Ποιός είναι ο καλύτερος κώδικας για το σύνολο των χαρακτήρων $\{C_1, C_2, \ldots, C_n\}$;

Αλγόριθμος Huffman

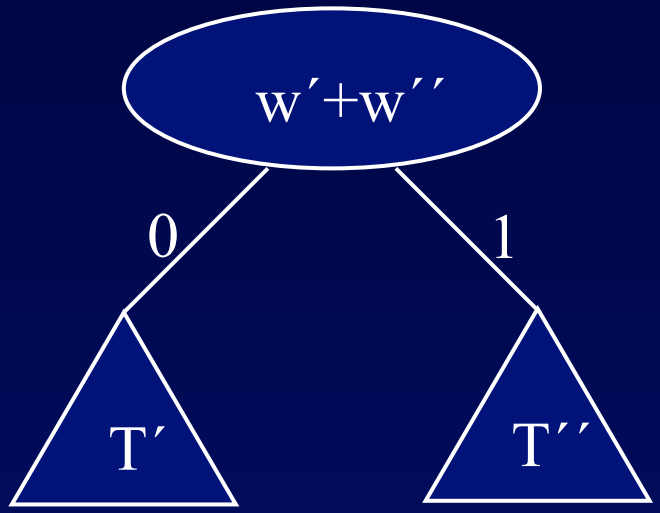
1. Δημιουργία μιας λίστας από δυαδικά δέντρα με ένα κόμβο που περιέχουν τα βάρη w_1, w_2, ..., w_n ταξινομημένα (π.χ. σε αύξουσα σειρά), ένα για κάθε χαρακτήρα C_1, C_2, ..., C_n.

2. Τα παρακάτω βήματα εκτελούνται n-1 φορές:

a. Εύρεση δύο δέντρων Τ΄ και Τ΄΄ της λίστας με ρίζες τα μικρότερα βάρη w΄ και w΄΄.

b. Αντικατάσταση των δέντρων αυτών με ένα δυαδικό δέντρο του οποίου η ρίζα είναι w΄ + w΄΄ υποδέντρα του τα Τ΄ και Τ΄΄ δίνοντας τις τιμές 0 και 1 για τους δείκτες του αριστερού και δεξιού υποδέντρου του, αντίστοιχα.

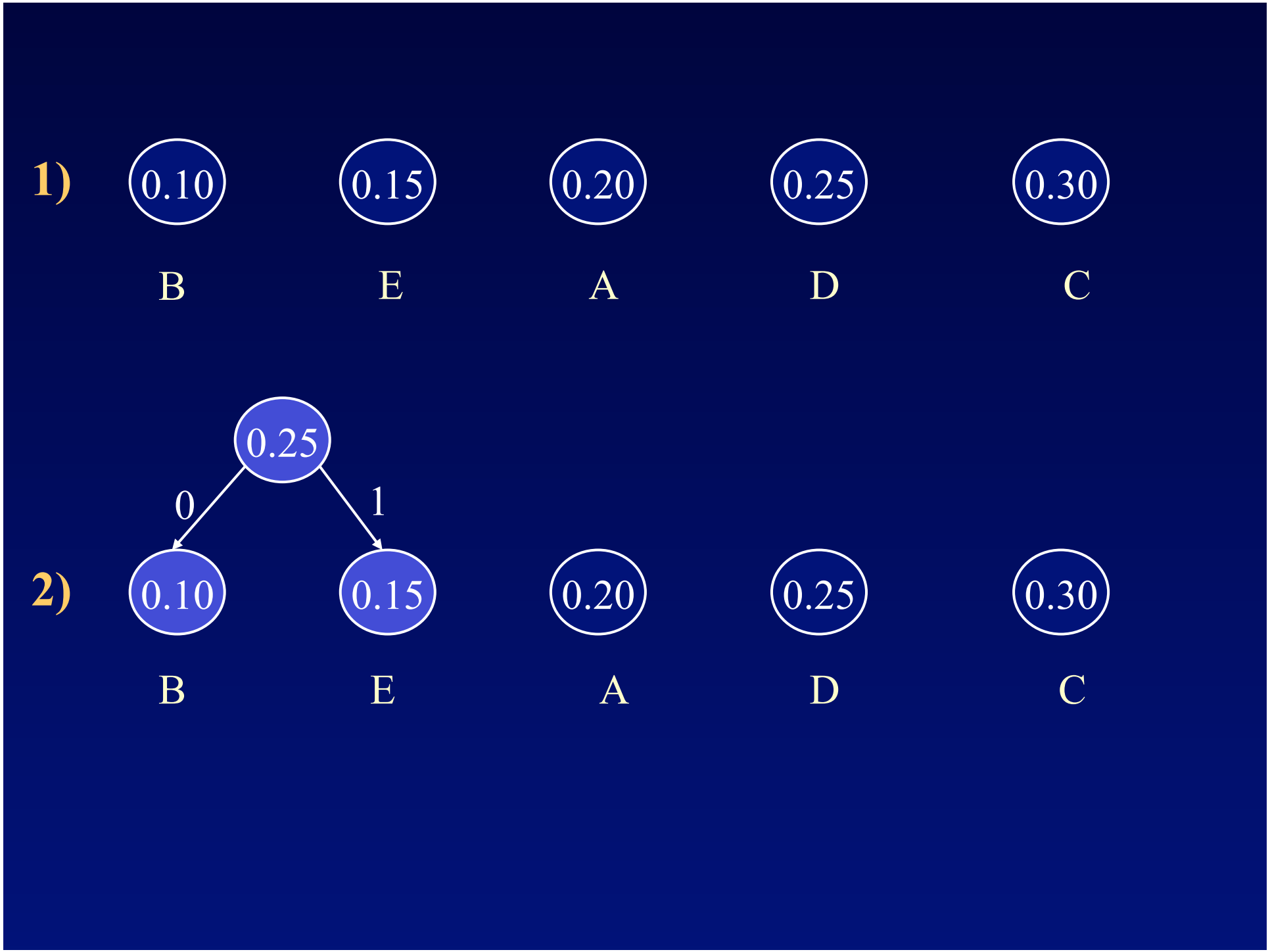
3. Ο κώδικας για τον χαρακτήρα C_i είναι εκείνη η διψήφια σειρά χαρακτήρων που ξεκινά από τη ρίζα του τελικού δυαδικού δέντρου και καταλήγει στο φύλλο C_i.

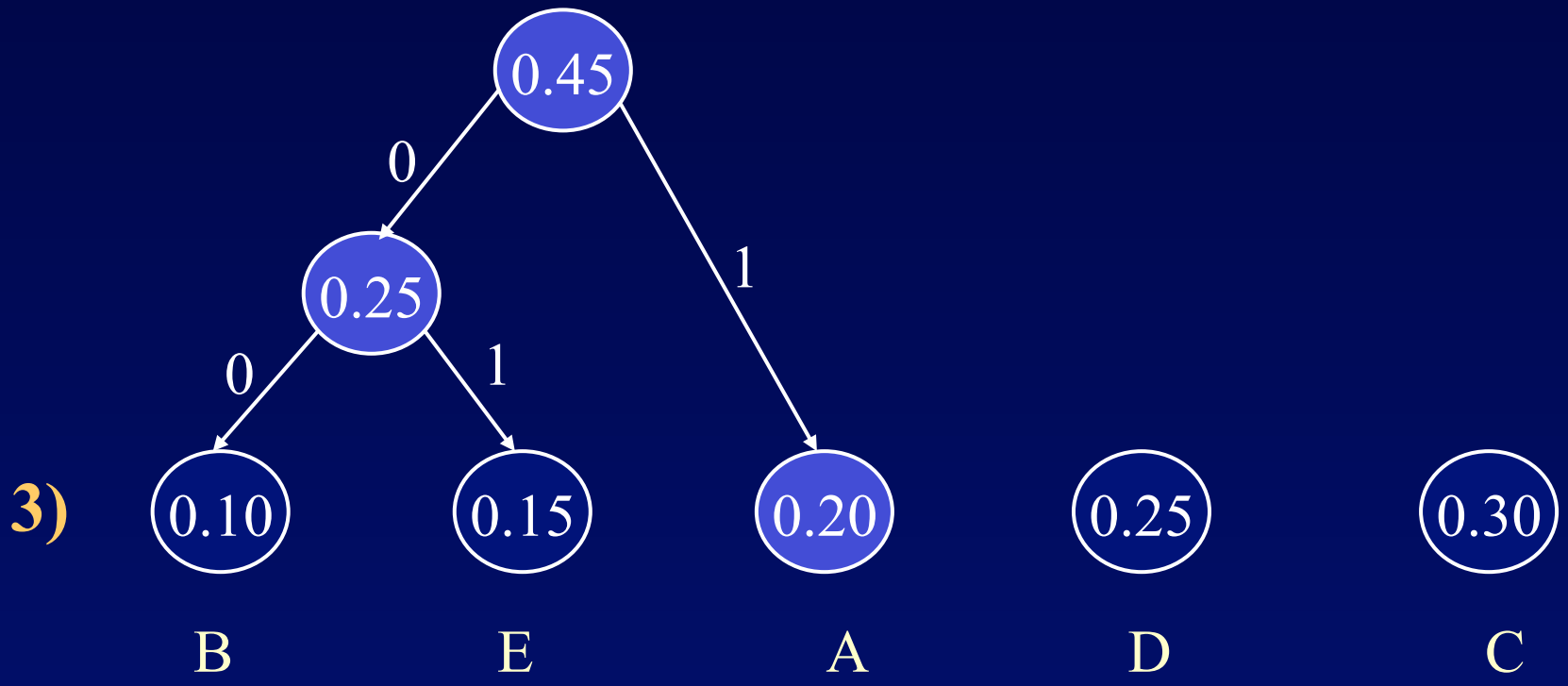


Στη συνέχεια ας υποθέσουμε ότι έχουμε τον παρακάτω πίνακα:

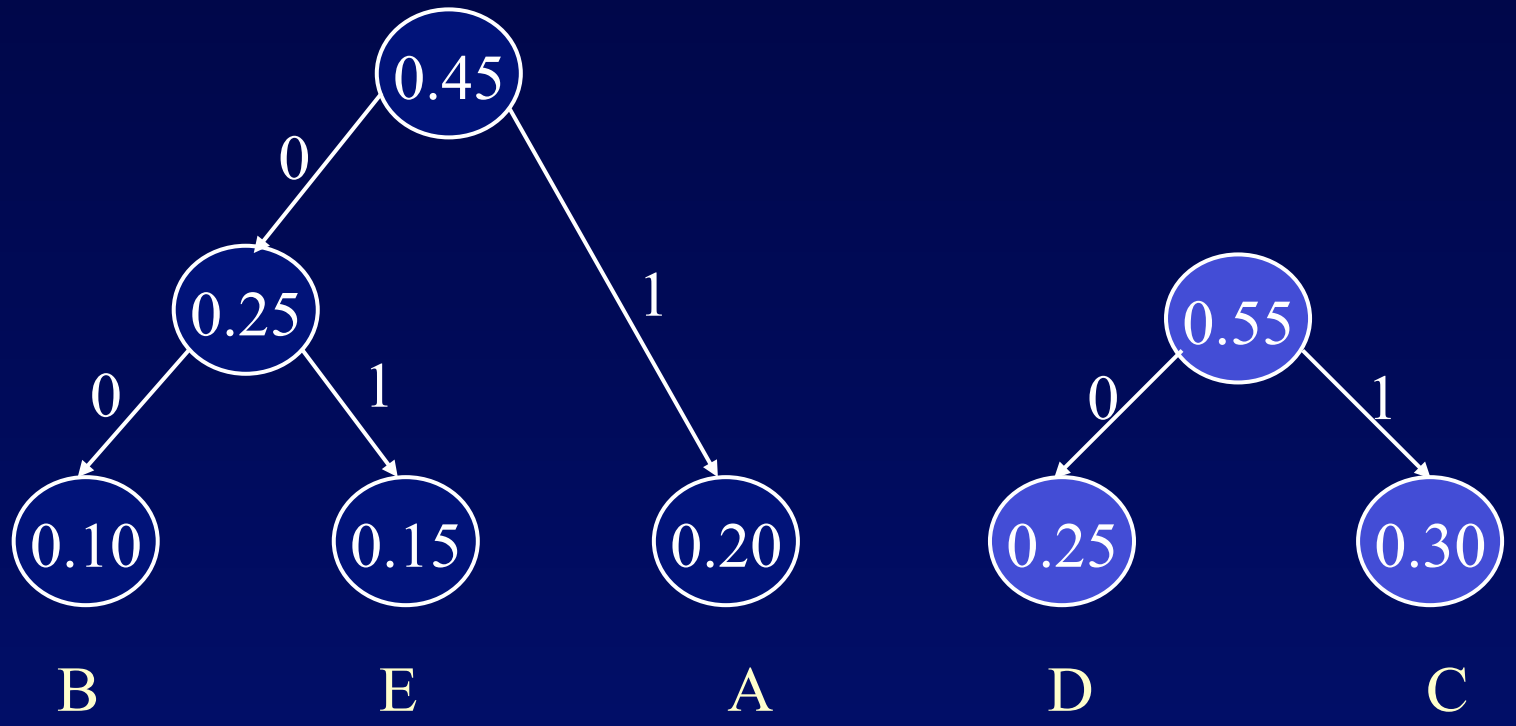
Χαρακτήρας	A	B	C	D	E
βάρος	0.2	0.1	0.3	0.25	0.15

Τότε η εφαρμογή του αλγορίθμου του Huffman παράγει τις παρακάτω διαδοχικές φάσεις δημιουργίας του δυαδικού δέντρου αποκωδικοποίησης:

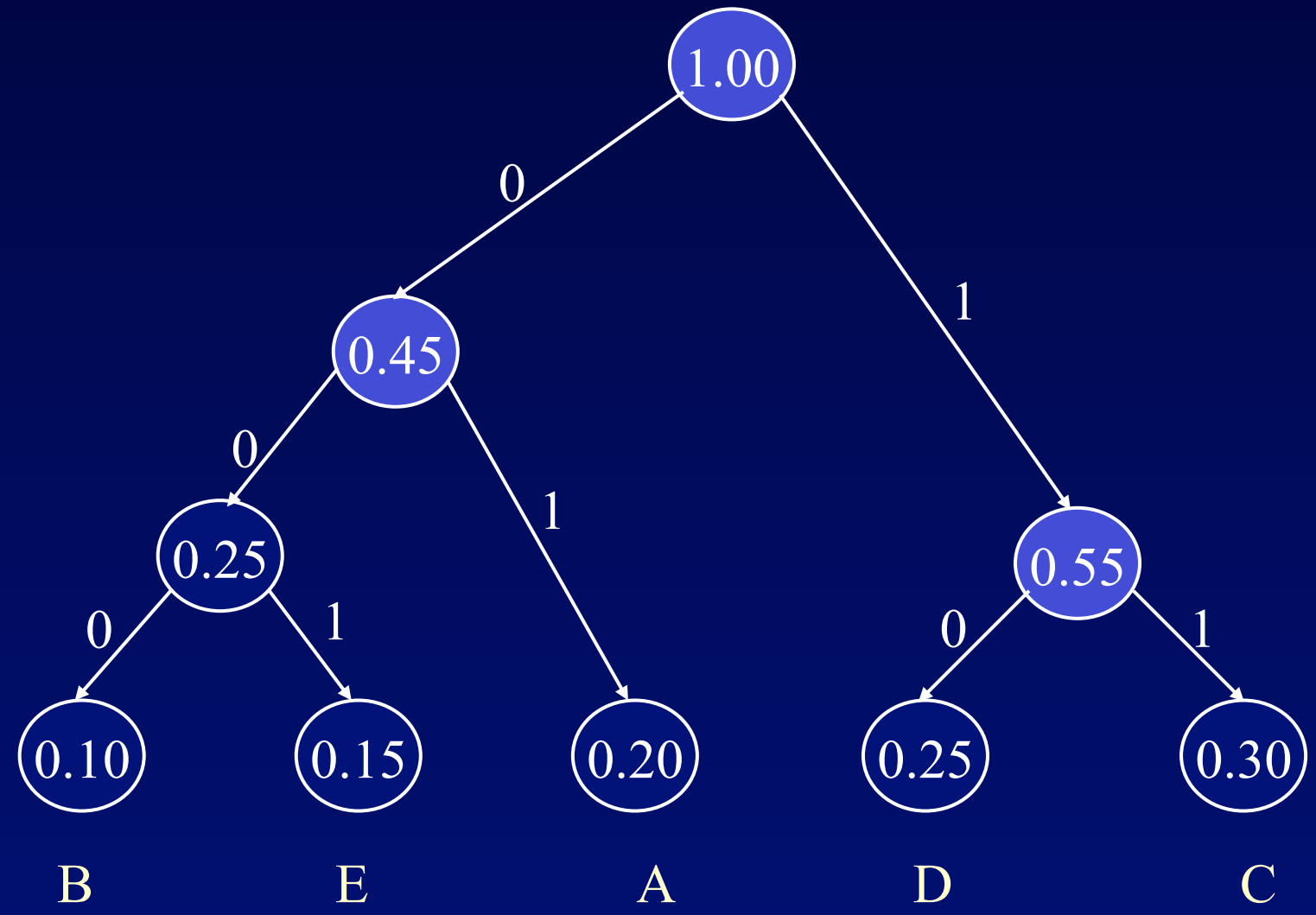

1)
0.10
0.15
0.20
0.25
0.30
B
E
A
D
C
0.25
0
1
2)
0.10
0.15
0.20
0.25
0.30
B
E
A
D
C


0.45
0
1
0.25
0
1
3)
0.10
0.15
0.20
0.25
0.30
B
E
A
D
C

3)



5)



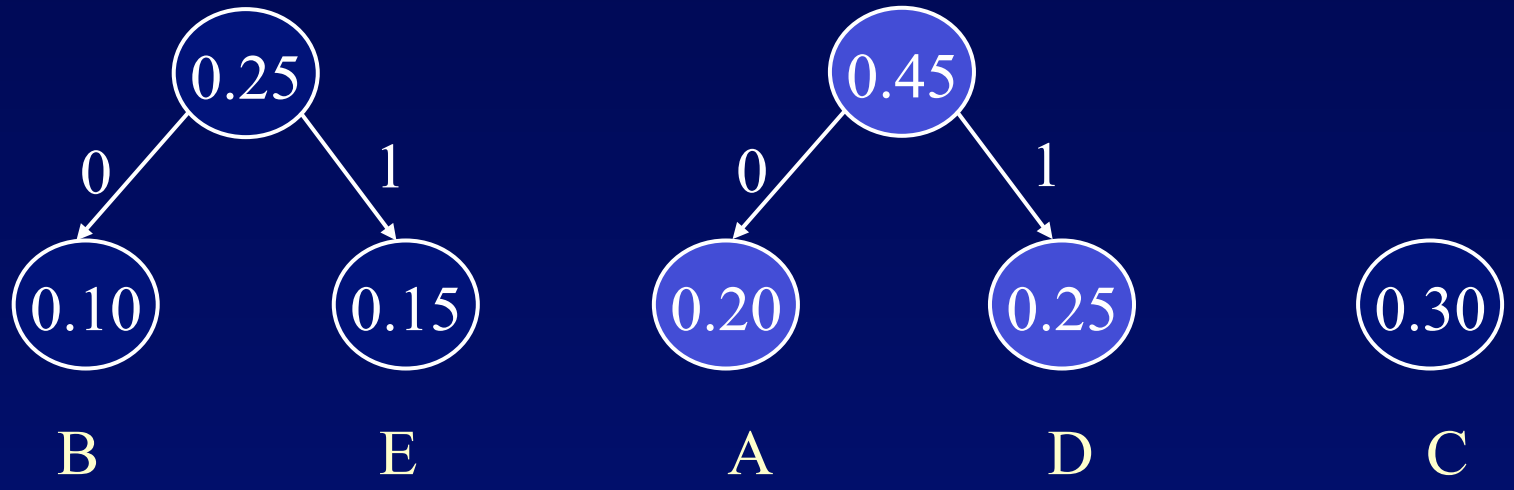
Οι κώδικες Huffman που λαμβάνονται από το παραπάνω δέντρο είναι:

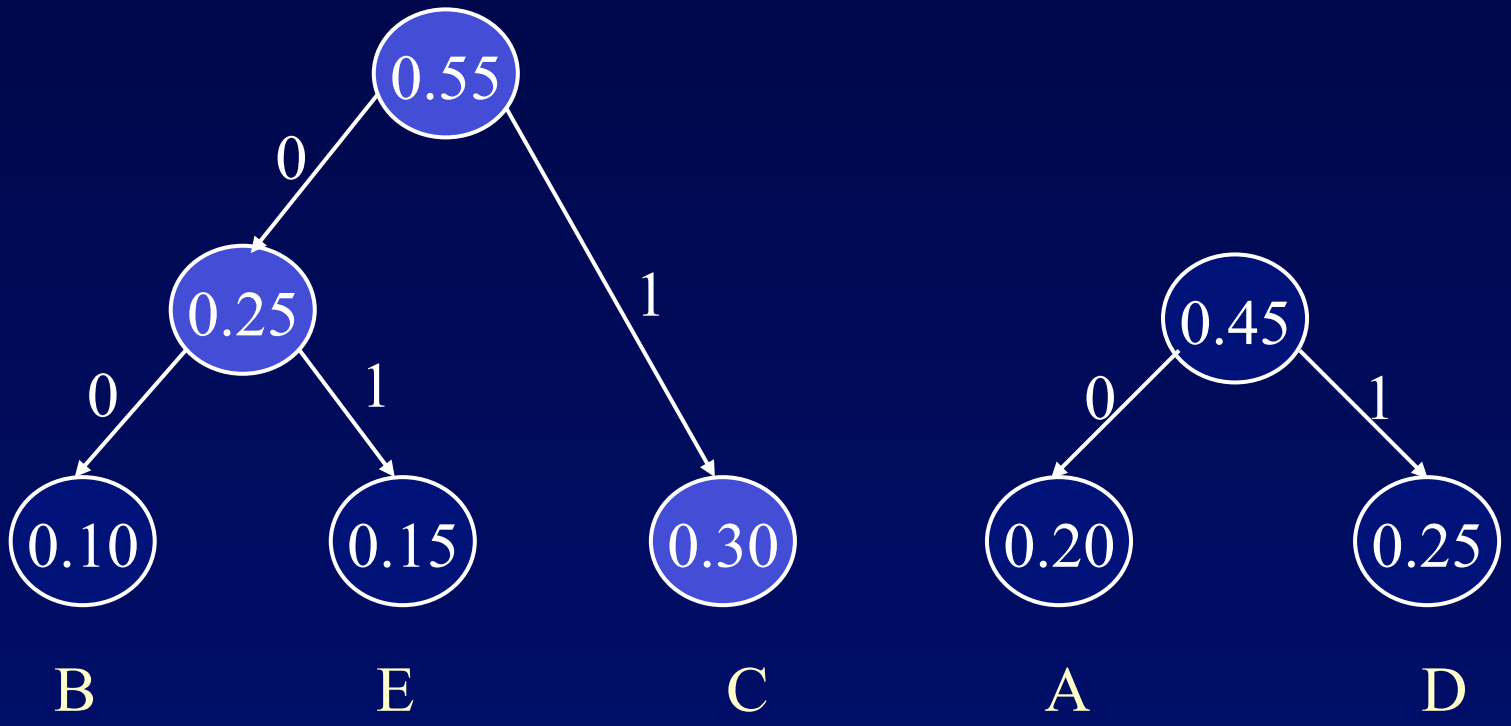
Χαρακτήρας	Κώδικας Huffmann
A	01
B	000
C	11
D	10
E	000

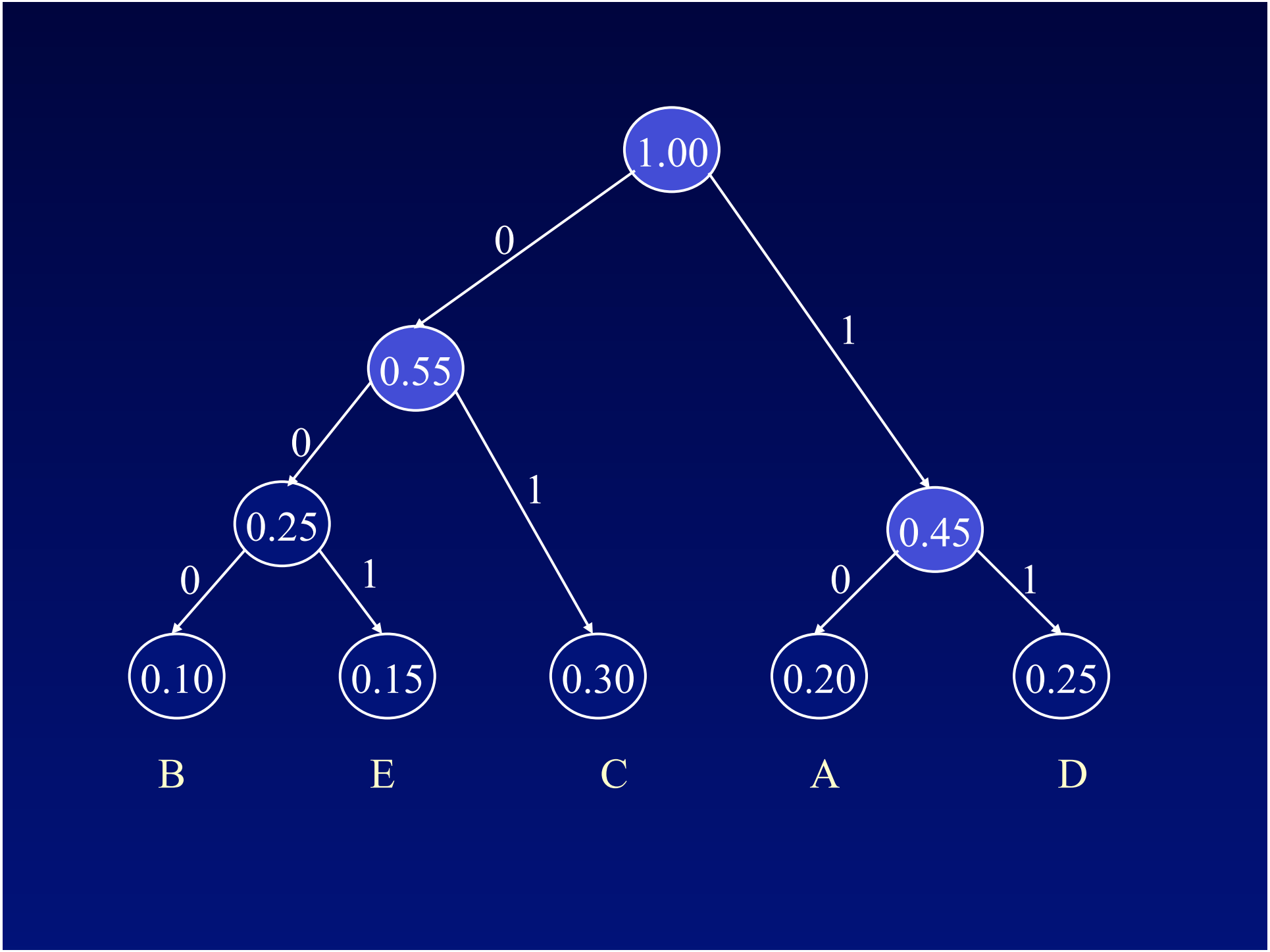
Επίσης η ποσότητα $E = \sum_{1 \le i \le n} w_i I_i$ είναι ίση με :

$$E = 0.1*3+0.15*3+0.2*2+0.25*2+0.3*2 = 2.25$$

Ας σημειωθεί ότι είναι δυνατή μια διαφορετική αντιστοιχία κωδικών στους χαρακτήρες του παραδείγματος με την ίδια τιμή για την Ε. Πράγματι, στη δεύτερη φάση δημιουργίας του δυαδικού δέντρου θα μπορούσαμε να προχωρήσουμε όπως φαίνεται παρακάτω:




0.55
0
1
0.25
0
1
0.10
0.15
0.30
B
E
C
0.45
0
1
0.20
0.25
A
D


1.00
0
1
0.55
0
1
0.25
0.45
0
1
0
1
0.10
0.15
0.30
0.20
0.25
B
E
C
A
D

Στην περίπτωση αυτή οι κώδικες που αντιστοιχούν στους χαρακτήρες δίνονται από τον παρακάρω πίνακα :

Χαρακτήρας	Κώδικας Huffmann
A	10
B	000
C	01
D	11
E	001

Αλγόριθμος αποκωδικοποίησης Huffman

1. Αρχικά ένας δείκτης p τοποθετείται στη ρίζα του δέντρου Huffman.

2. Οσο δεν έχει βρεθεί το τέλος του μηνύματος να εκτελούνται τα παρακάτω:

α. Εστω x είναι το επόμενο bit στη σειρά χαρακτήρων.

β. Αν x == 0 τότε

p = p .apaidi

διαφορετικά

p = p .dpaidi

γ. Αν ο p δείχνει σ' ένα φύλλο τότε:

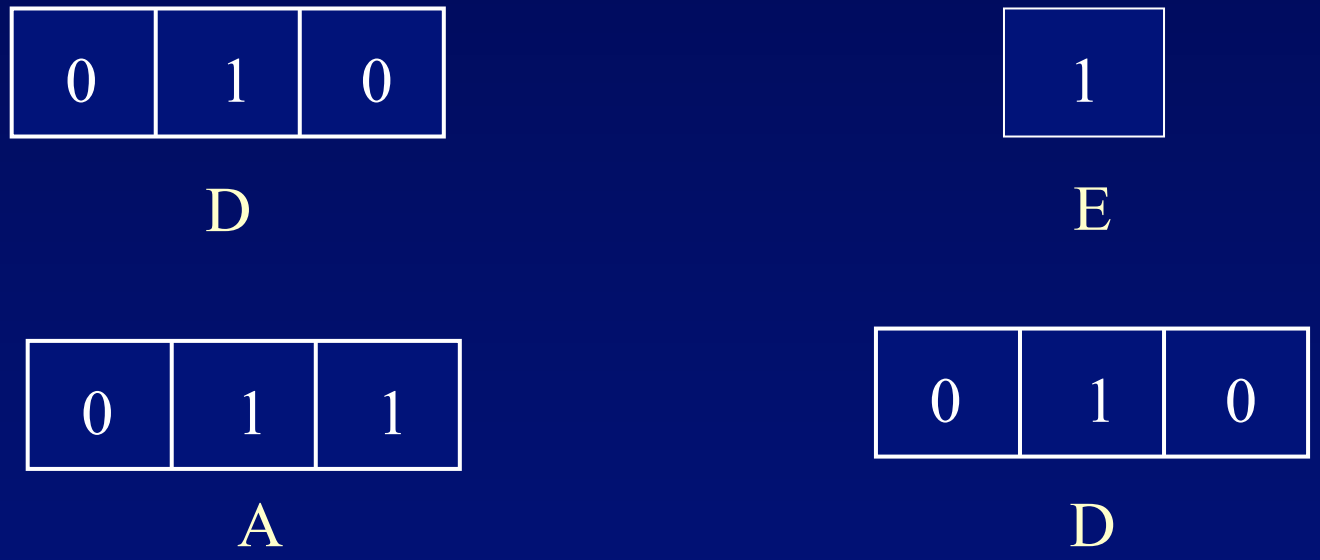
i. Να εκτυπωθεί ο χαρακτήρας αυτού του φύλλου

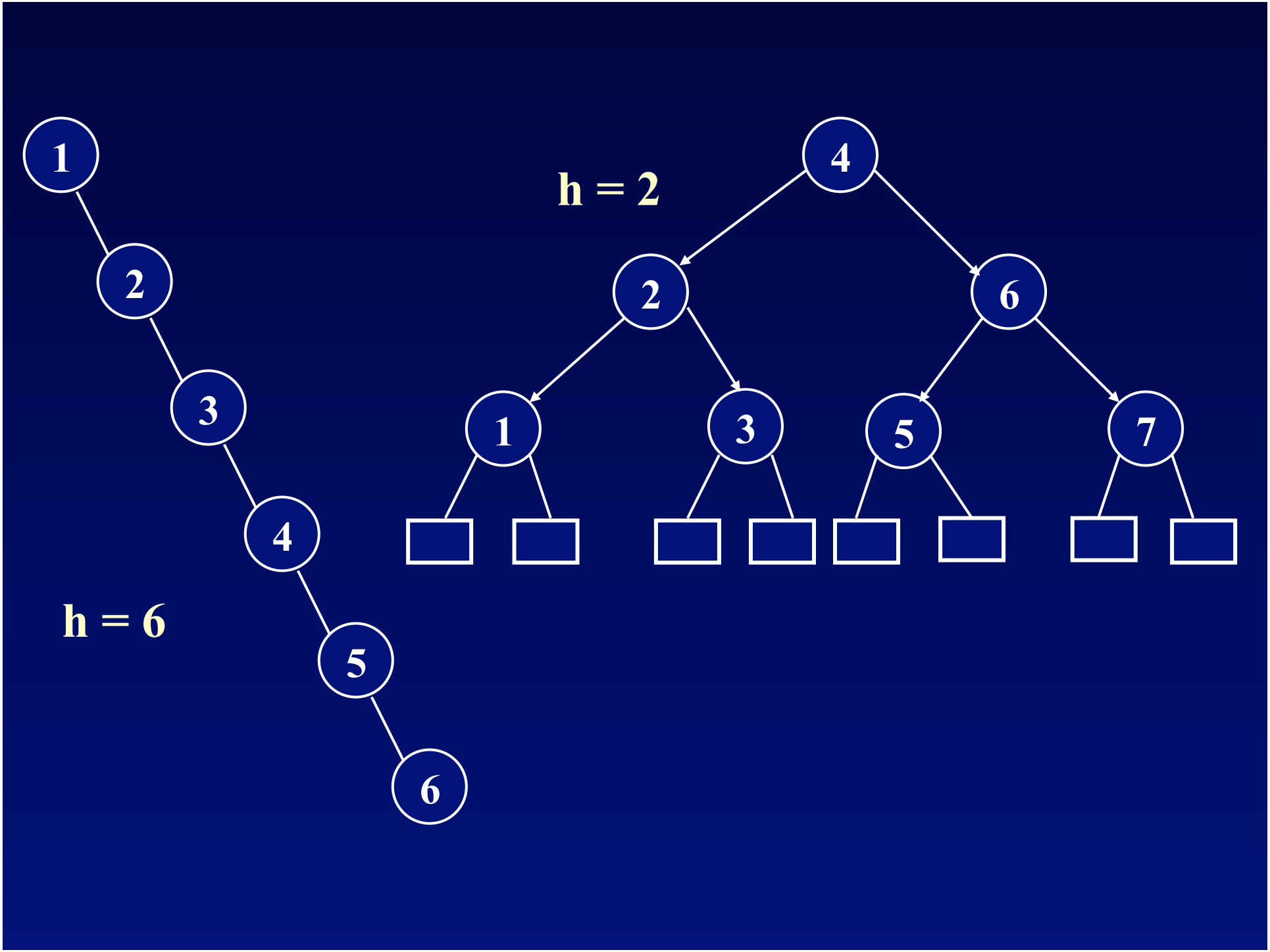
ii. p = riza

Για παράδειγμα, ας υποτεθεί ότι το μήνυμα

0 1 0 1 0 1 1 0 1 0

έχει ληφθεί και κωδικοποιήθηκε με τη δημιουργία του δεύτερου δέντρου Huffman. Τότε η εκάστοτε διαδρομή του p είναι αυτή που παριστάνεται με τα ορθογώνια στο παρακάτω σχήμα :




h = 2
h = 6
1
2
3
4
5
6
4
2
6
1
3
5
7

Το ύψος h ενός ΔΔΑ εξαρτάται από το σχήμα του.

Ας υποθέσουμε ότι οι κόμβοι έχουν φθάσει με μια τυχαία σειρά, τότε πόσες περισσότερες συγκρίσεις, κατά μέσο όρο, απαιτούνται σε μια αναζήτηση του προκυπτόμενου ΄τυχαίου΄ ΔΔΑ από εκείνες που απαιτεί ένα τέλεια ισοζυγισμένο ΔΔΑ;

$$S(n) = \left(1 + \frac{1}{n}\right) U(n) - 1 \qquad \textbf{(1)}$$

$$S(n) = 1 + \frac{U(0) + U(1) + \ldots + U(n-1)}{n} \qquad \textbf{(2)}$$

(1),(2) $\Rightarrow$ $(n+1)U(n) = 2n + U(0) + U(1) + \ldots + U(n-1)$ **(3)**

n := n+1 $n\,U(n-1) = 2(n-1) + U(0) + U(1) + \ldots + U(n-2)$ **(4)**

(3) - (4) $U(n) = U(n-1) + \dfrac{2}{n+1}$ **(5)**

$$U(n) = 0$$

n=1, $$U(1) = U(0) + \frac{1}{2} \quad \Rightarrow \quad U(1) = \frac{2}{2}$$

n=2, $$U(2) = U(1) + \frac{2}{3} \quad \Rightarrow \quad U(1) = \frac{2}{2} + \frac{2}{3}$$

• • •

• • •

• • •

$$U(n) = 2\left[\frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n+1}\right]$$

ή $$U(n) = 2\,H_{n+1} - 2 \qquad \textbf{(6)}$$

όπου $H_{n+1} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n+1}$ αρμονικός αριθμός

$$H_n = \ln(n) + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \in \qquad (7)$$

$$0 \prec \in \prec \frac{1}{252n^6} \quad , \qquad \gamma \cong 0.577215665$$

σταθερά του Euler

(6) $\xrightarrow{(7)}$ $U(n) = 2\,H_{n+1} - 2 \cong 2\ln(n)$

$$\ln(n) = (\ln 2)(\log_2 n)$$

$U(n) \cong 2(\ln 2)(\log_2 n)$ ή $U(n) \cong 1.39 \log_2 n$

όπου $H_{n+1} = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n+1}$ αρμονικός αριθμός

$$H_n = \ln(n) + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \in \qquad \textbf{(7)}$$

$0 \prec \in \prec \frac{1}{252n^6}$, $\gamma \cong 0.577215665$

σταθερά του Euler

(6) $\xrightarrow{(7)}$ $U(n) = 2\,H_{n+1} - 2 \cong 2\ln(n)$

$$\ln(n) = (\ln 2)(\log_2 n)$$

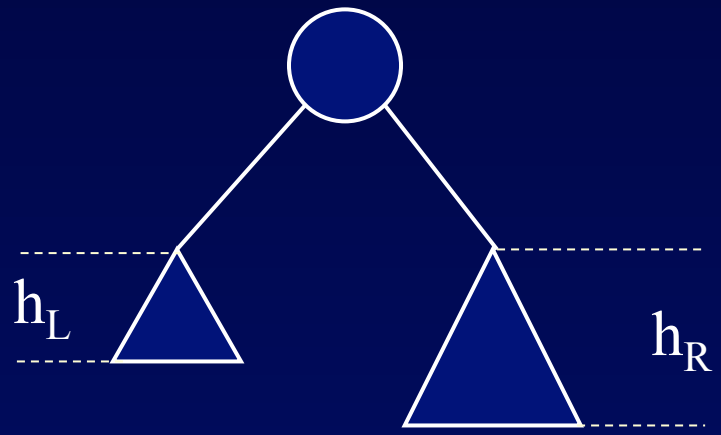
$$U(n) \cong 2\,(\ln 2)(\log_2 n)$$

ή $\quad U(n) \cong 1.39 \log_2 n$

Το μέσο κόστος για τη μη ισοζύγιση ενός ΔΔΑ είναι περίπου 39% περισσότερες συγκρίσεις.

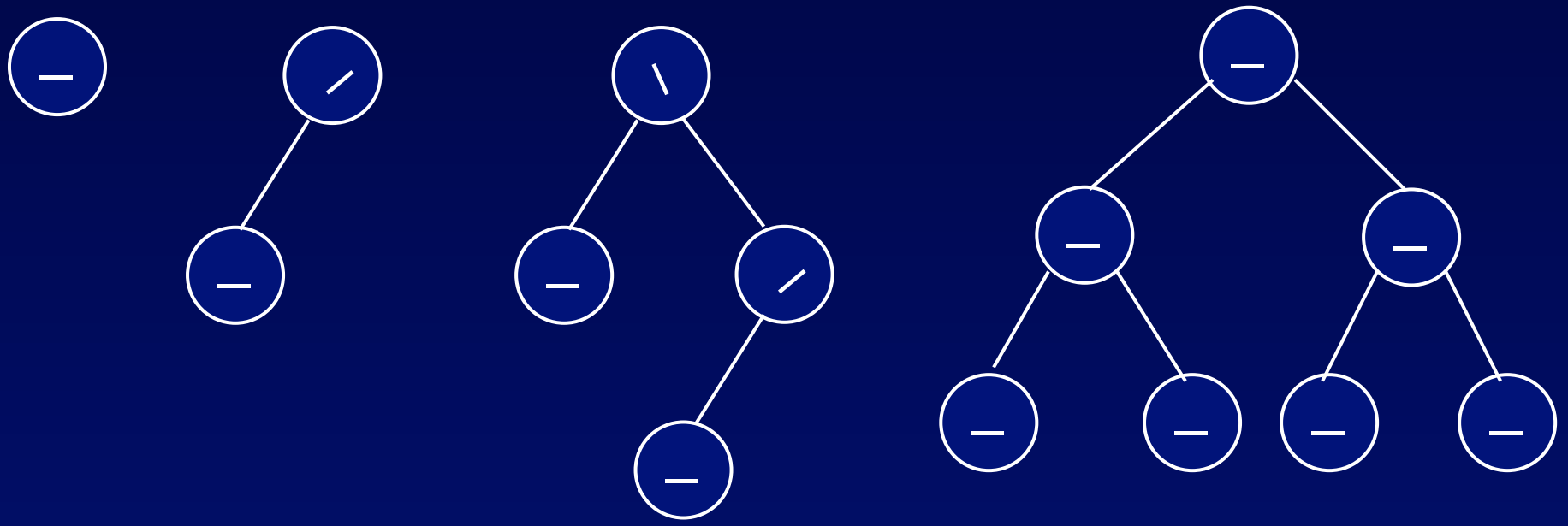
AVL δέντρα

$$|h_L - h_R| \leq 1$$

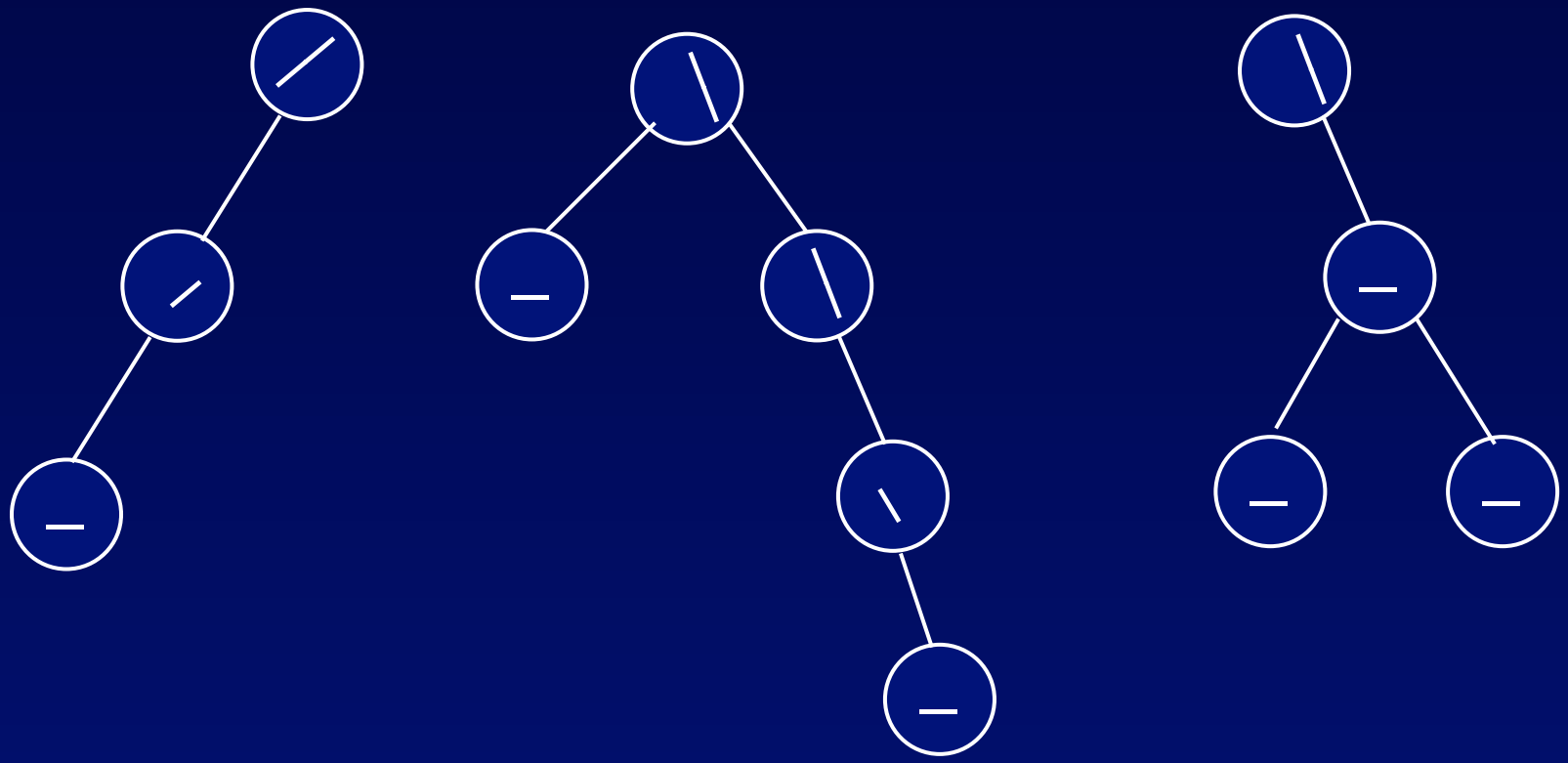


G.M. Adelson_Velkii και E.M. Landis 1962

AVL Δέντρα



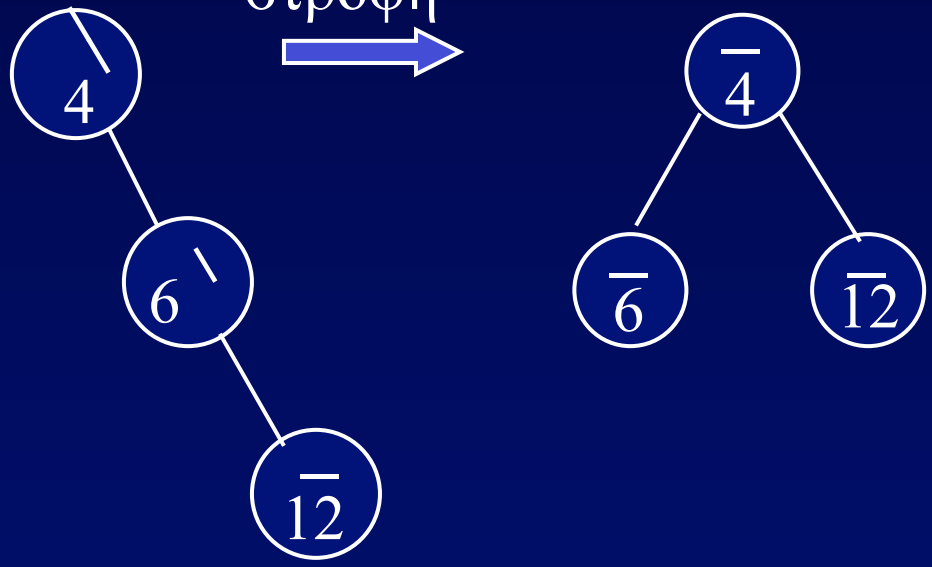
Μη AVL Δέντρα



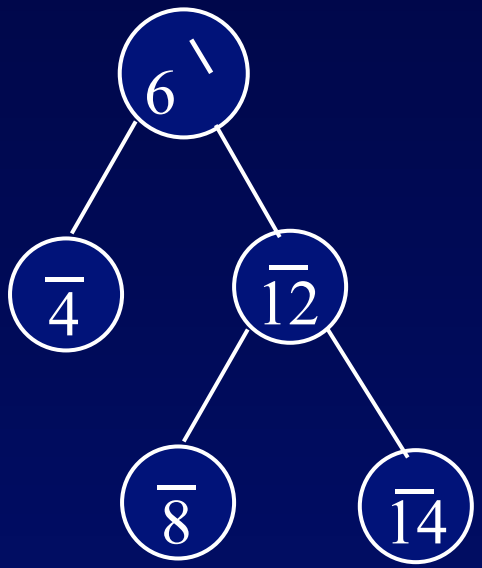
Εισαγωγή κόμβου

4, 6 :

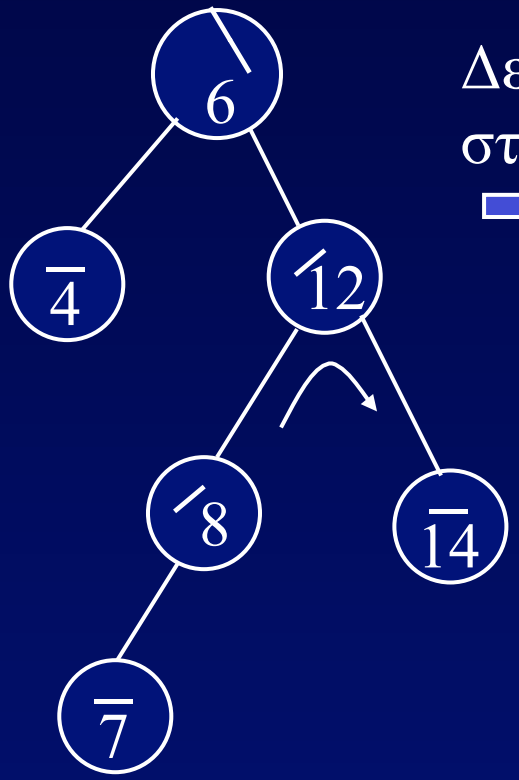
12 :



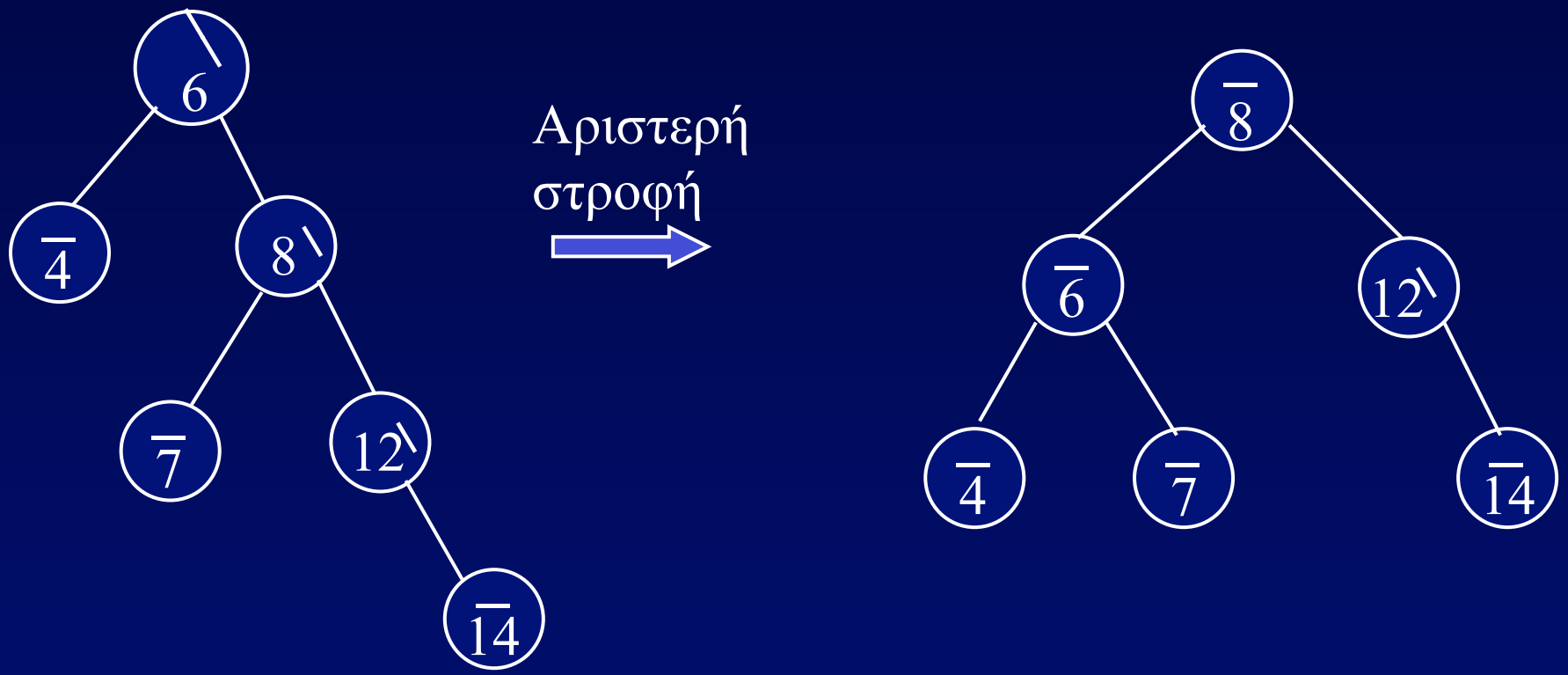
8, 14 :



7 :



Δεξιά στροφή


6
4
8
7
12
14
Αριστερή
στροφή
8
6
12
4
7
14

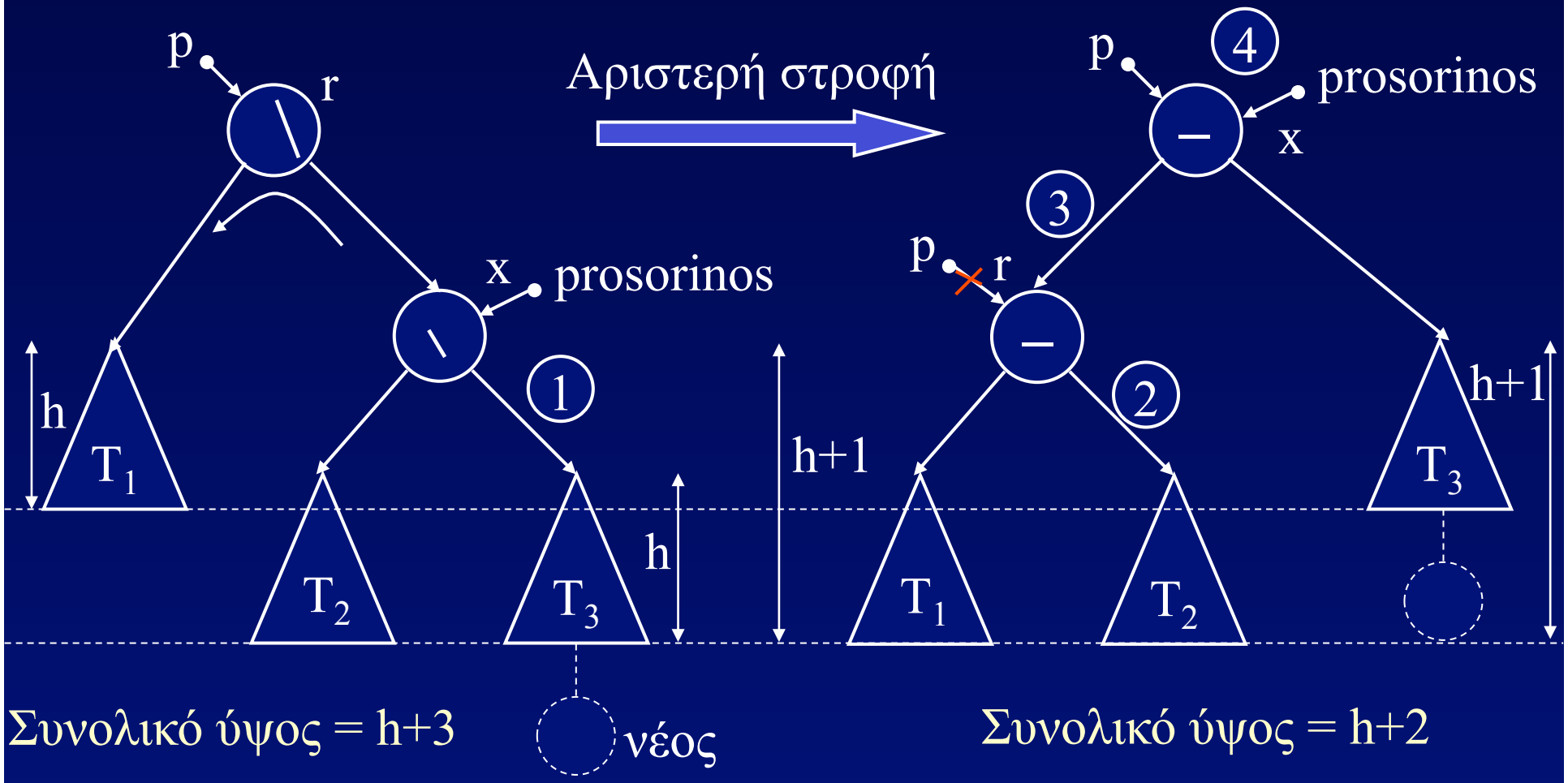
LL : ο νέος κόμβος Υ εισάγεται στο αριστερό υποδέντρο του αριστερού υποδέντρου του Α.

LR : ο νέος κόμβος Υ εισάγεται στο δεξί υποδέντρο του αριστερού υποδέντρου του Α.

RR : ο νέος κόμβος Υ εισάγεται στο δεξί υποδέντρο του δεξιού υποδέντρου του Α.

RL : ο νέος κόμβος Υ εισάγεται στο αριστερό υποδέντρο του δεξιού υποδέντρου του Α.

Περίπτωση 1 : Δεξία Υψηλό

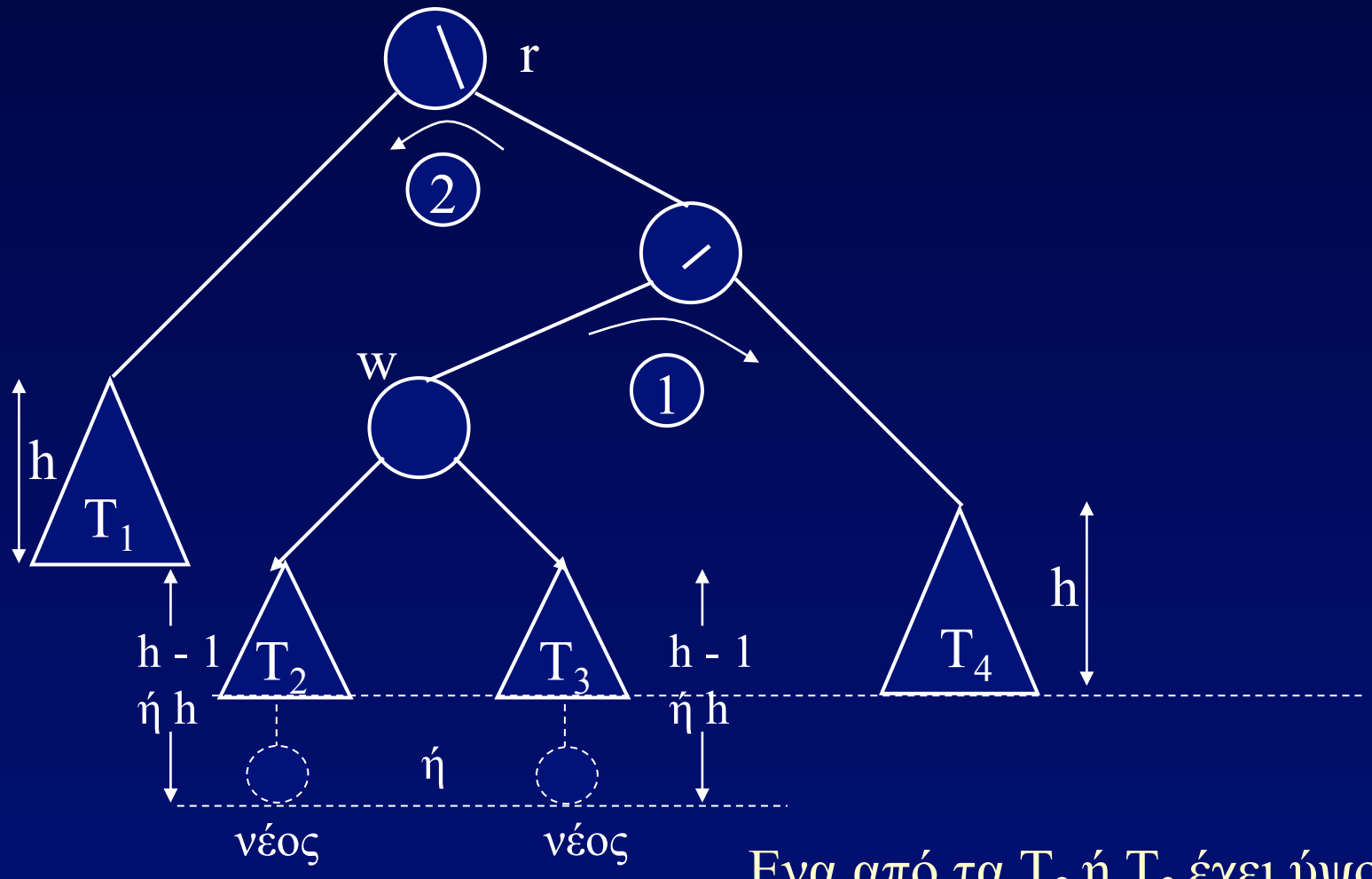


```
void aristeri_peristrofi (typos_deikti *p);
{ /*Ο p δείχνει τη ρίζα του υποδέντρου που περιστρέφεται*/
        typos_deikti prosorinos;

        if (keno_dentro(*p)) /*Είναι αδύνατη η περιστροφή*/
                printf ( "Κενό δέντρο" ) /*ενός κενού δέντρου*/
        else
                if (keno_dentro((*p)->dpaidi))
                        /*Είναι αδύνατο να γίνει ρίζα ένα κενό
                        υποδέντρο*/
                        printf ( "Κενό δεξί υποδέντρο" )
                else
                {
                        prosorinos = (*p)->dpaidi;
```

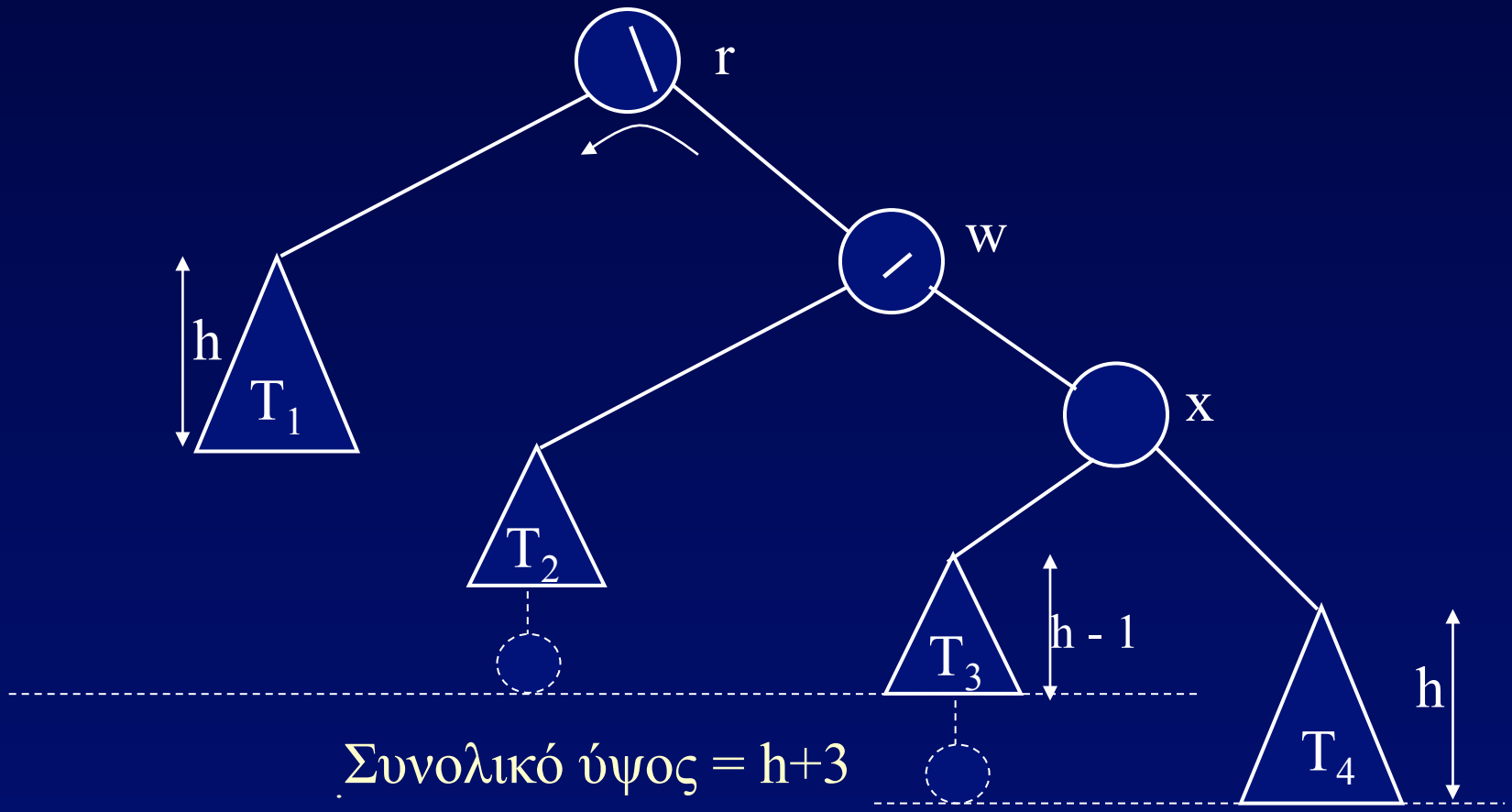
συνέχεια

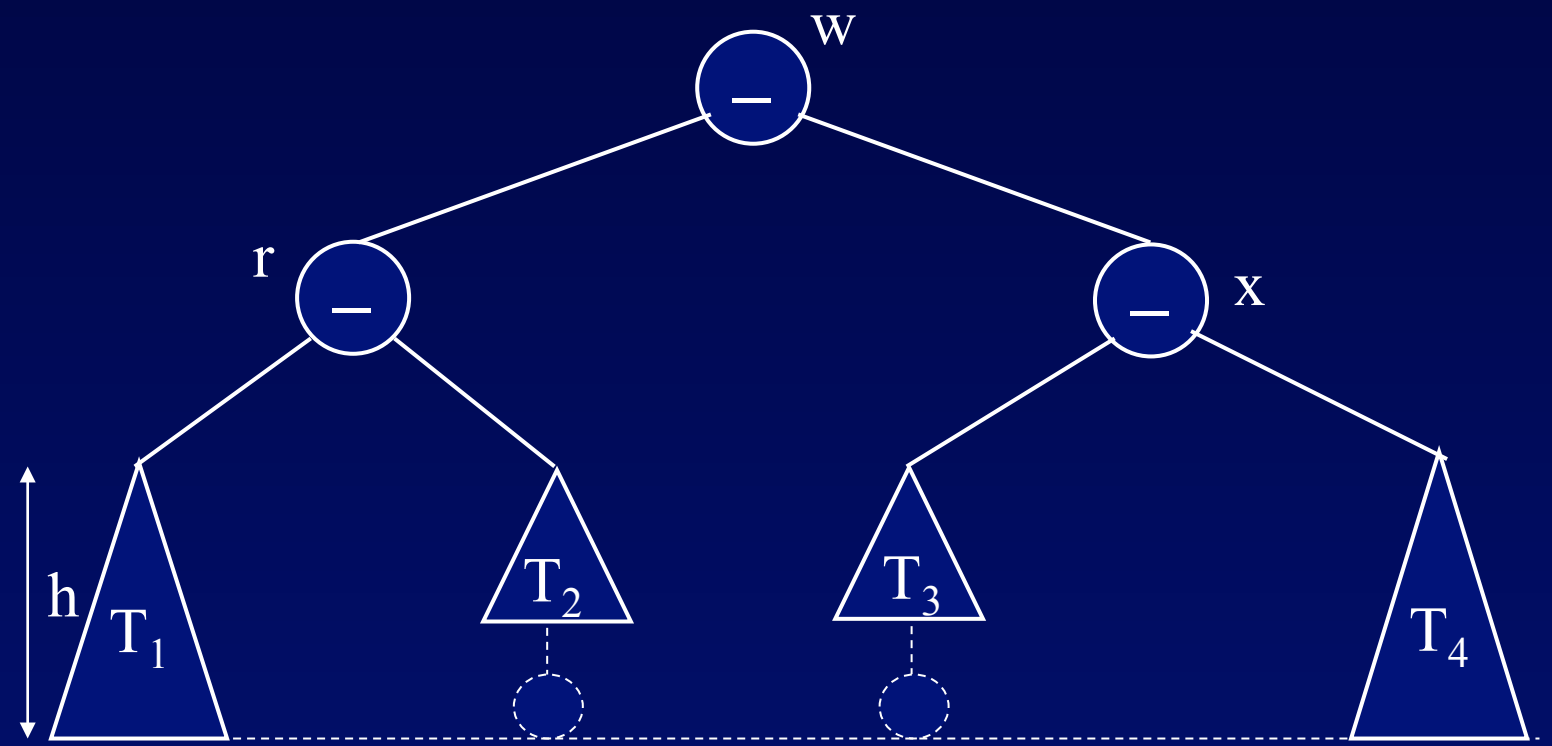
```
                (*p)->dpaidi = prosorinos->apaidi;
                prosorinos->apaidi :=* p;
                *p = prosorinos;
        }
} /*aristeri_peristrofi*/
```

Περίπτωση 2 :Αριστερά Υψηλό



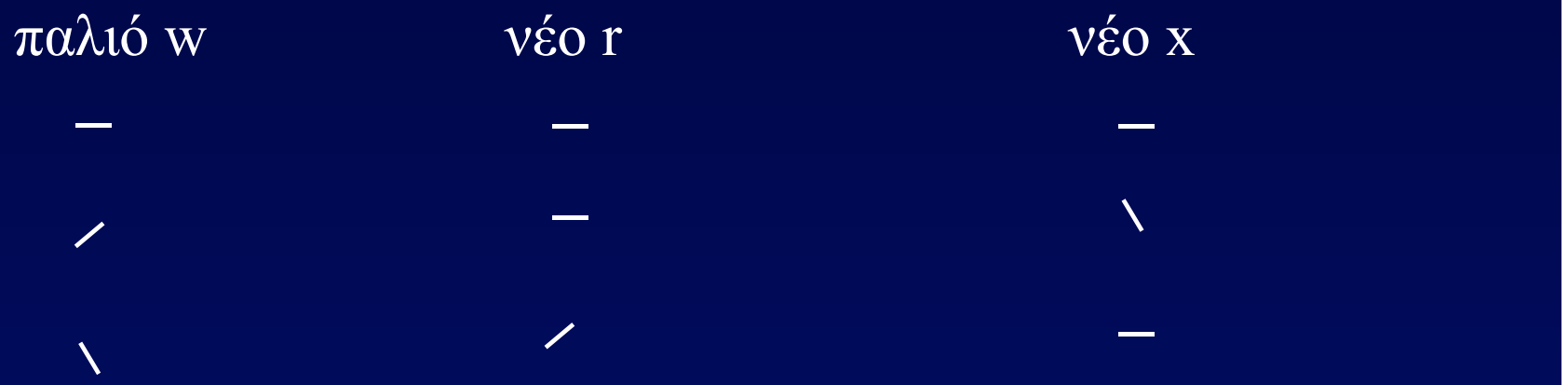
Ενα από τα T_2 ή T_3 έχει ύψος h. Συνολικό ύψος = h+3


r
w
x
h
T_1
T_2
T_3
h - 1
T_4
h
Συνολικό ύψος = h+3



Συνολικό ύψος = h+2

Αποκατάσταση ισοζύγισης (διπλή RL περιστροφή). Οι αριθμοί στους κύκλους δηλώνουν την προτεραιότητα των ενεργειών.


παλιό w
νέο r
νέο x

Η αποκατάσταση της ισοζύγισης με τις RR και RL περιστροφές υλοποιείται με το παρακάτω υποπρόγραμμα.

```
void dexi_varos (typos_deikti          *riza);
{
        typos_deikti x;  {δείκτης στο δεξί υποδέντρο της ρίζας}
        typos_deikti w;

       x = (*riza)->dpaidi;
       switch (x->pi)
       {
              case DY:  {απλή RR περιστροφή}
                     (*riza)->pi = IY;
                     x->pi = IY;
                     aristeri_peristrofi(riza);
                     ypsilotero = false
                     break;
```

συνέχεια

```
case IY:
        /*Δεν συμβαίνει η περίπτωση αυτή*/
        printf ( “Λάθος” );
        break;
case AY:/*διπλή RL περιστροφή*/
        w = x->apaidi;
        switch (w->pi)
        {
                case IY: riza->pi=IY; x->pi= IY break;
                case AY: riza->pi=IY; x->pi= DY break;
                case DY: riza->pi=AY; x->pi=IY break;
        }
        w->pi=IY;
        dexia_peristrofi(&x);
```

συνέχεια

```
                    /*ο x δείχνει τώρα τον w*/
                    riza->dpaidi = x;
                    aristeri_peristrofi(*riza);
                    ypsilotero = false;
                }
            }
}/*dexi_varos*/
```

Το υποπρόγραμμα εισαγωγής ενός κόμβου σε ένα AVL δέντρο είναι το ακόλουθο :

```
void avl_eisagogi (typos_deikti *riza, typos_deikti prosorinos,
                   typos_deikti  stoixeio, char *ypsilotero)
{
        char     ypsilotero_ypodentro; /* Εχει αυξηθεί το ύψος ενός
                                          υποδέντρου*/
         typos_deikti    temp;
```

συνέχεια

```
{       if (keno_dentro(riza))
        {       *riza = prosorinos; (*riza)->apaidi = NULL;
                (*riza)->dpaidi = NULL; (*riza)->pi = IY;
                ypsilotero = true;}
        else
                if (prosorinos->dedomena == stoixeio)
                        printf( "Ο κόμβος υπάρχει ήδη στο ΔΔΑ");
                else
                        if (prosorinos->dedomena < stoixeio)
                        {   /*εισαγωγή στο αριστερό υποδέντρο*/
                                temp=(*riza)->apaidi;
                                avl_eisagogi (&temp, prosorinos, stoixeio,
                                     ypsilotero_ypodentro);
                                (*riza)->apaidi=temp;
```

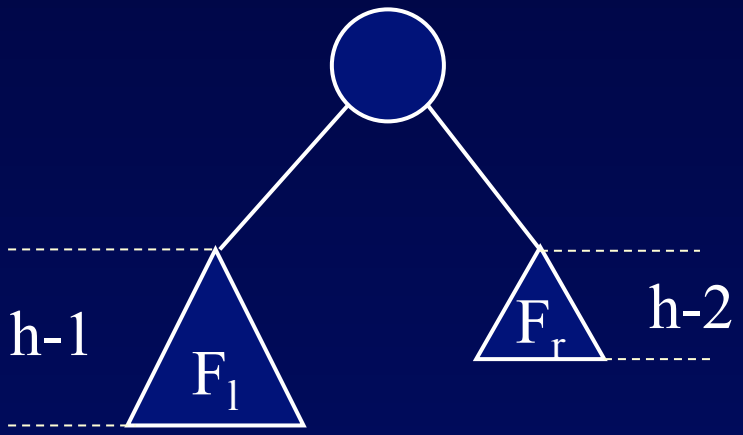
συνέχεια

```
if (ypsilotero_ypodentro)
        switch (pi)
        {
        case AY: aristero_varos(*riza);
        break;
        case IY: pi=AY; ypsilotero=true;
        break;
        case DY: pi=IY;ypsilotero=false;
        break;
        }
else
        ypsilotero = false;
}
```

συνέχεια

```
                else
                {/*εισαγωγή στο δεξί υποδέντρο*/
                        avl_eisagogi (dpaidi, prosorinos, stoixeio,
                        ypsilotero_ypdodentro);
                        if (ypsilotero_ypodentro)
                                switch (pi)
                        case AY:pi=IY;ypsilotero=false;
                                break;
                        case IY: pi=DY;ypsilotero=true;
                                break;
                        case DY: dexi_varos(*riza);
                                break;
                }
        }
        ypsilotero = false;
    }
}/*avl_eisagogi*/
```

Το ύψος ενός AVL δέντρου



$$|F_h| = |F_{h-1}| + |F_{h-2}| + 1$$

$$|F_0| = 1 \qquad |F_1| = 2$$

$$|F_h| + 1 = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^{k+2} - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^{h+2}$$

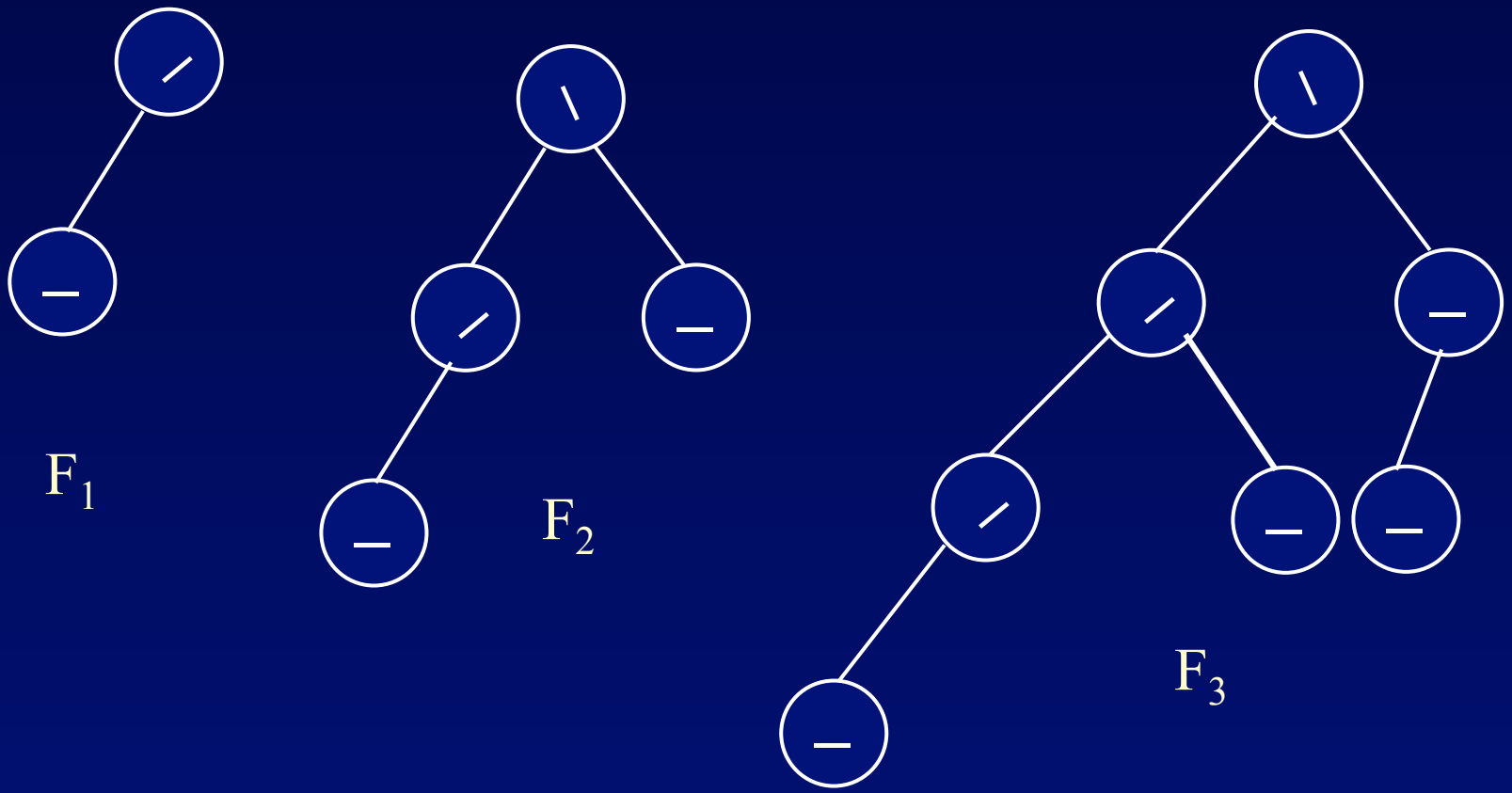
$$h \cong 1.44 \log_2 |F_h|$$

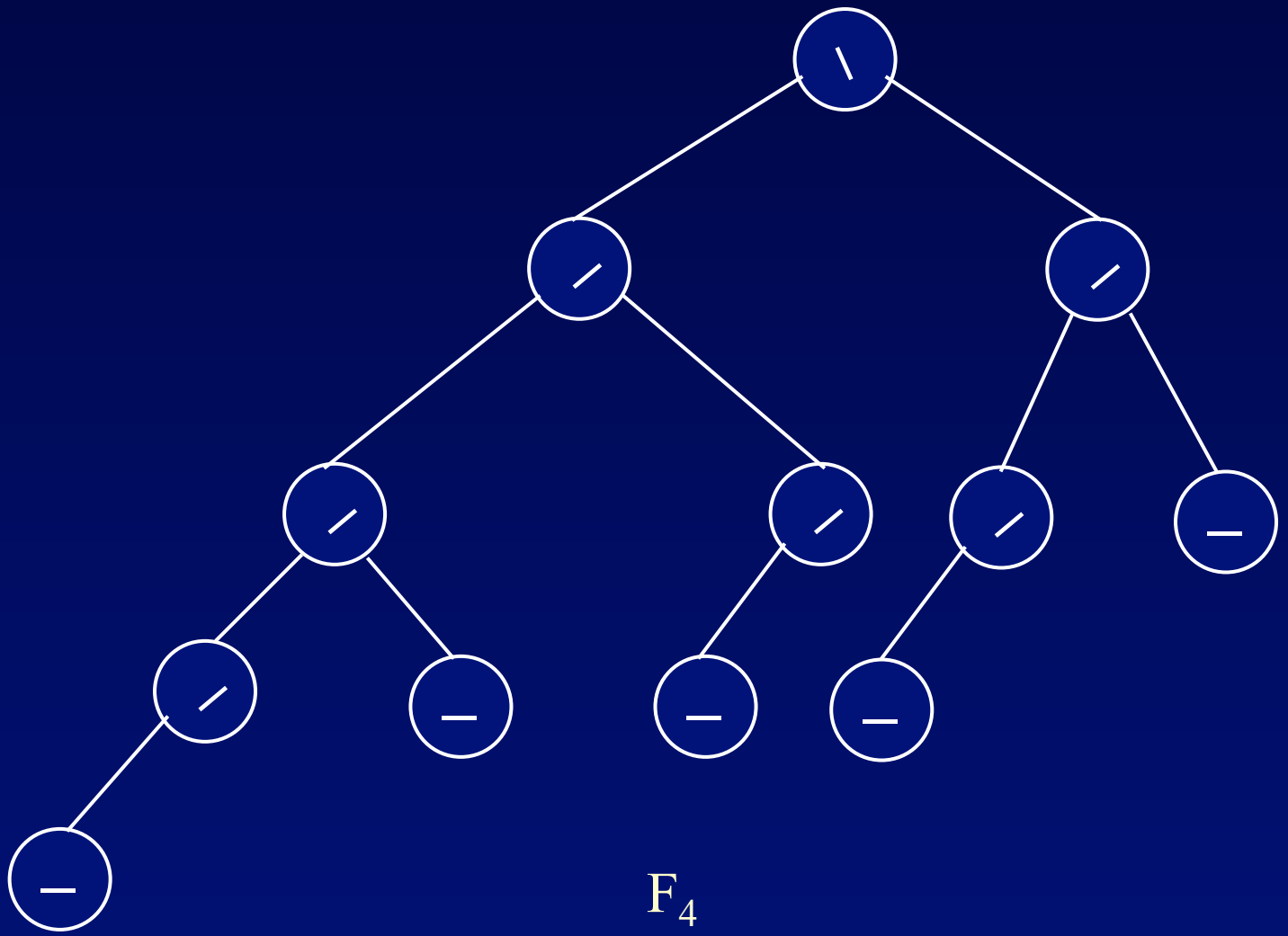
ή

$$h \cong 1.44 \log_2 n$$

Επομένως οι αλγόριθμοι για την επεξεργασία των AVL δέντρων απαιτούν το πολύ 44% περισσότερο χρόνο από το βέλτιστο. Στην πράξη όμως έχει βρεθεί ότι ο χρόνος αυτός είναι πολύ λιγότερος.

Δέντρα Fibonacci





F_4