ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών

Δομές Δεδομένων και Τεχνικές Προγραμματισμού

Ενότητα 6: ΑΤΔ Δένδρο, ΑΤΔ Δυαδικό Δένδρο Αναζήτησης (ΔΔΑ)

Ιωάννης Κοτρώνης

Σχολή Θετικών Επιστημών

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Σκοποί ενότητας

- Ορίσει τους ΑΤΔ Δένδρο και Δυαδικό Δένδρο Αναζήτησης
- Να δείξει σημαντικές ιδιότητες του δένδρου, όπως ελάχιστο ύψος, κόστος αναζήτησης.
- Να δείξει τυπικές υλοποιήσεις
- Να αναπτύξει την εφαρμογή Huffman
- Να δείξει τα AVL δένδρα για καλύτερη ισορρόπηση.

Περιεχόμενα ενότητας

- ΑΤΔ Δένδρο, ορισμός και χρήση
- Ιδιότητες και σχέση αριθμού κόμβων με ύψος
- ΑΤΔ Δυαδικό Δένδρο Αναζήτησης
- Πράξεις, υλοποιήσεις και πολυπλοκότητα
- Αναδρομικοί και μη αναδρομικοί αλγόριθμοι
- Διαδρομές προ-, ενδο-, μετα.
- AVL δένδρα

ΑΤΔ Δένδρο

Δέντρα (Trees) - Ιεραρχική Δομή

Εφαρμογές

Δομή – Οργάνωση Αρχείων

Οργανογράμματα

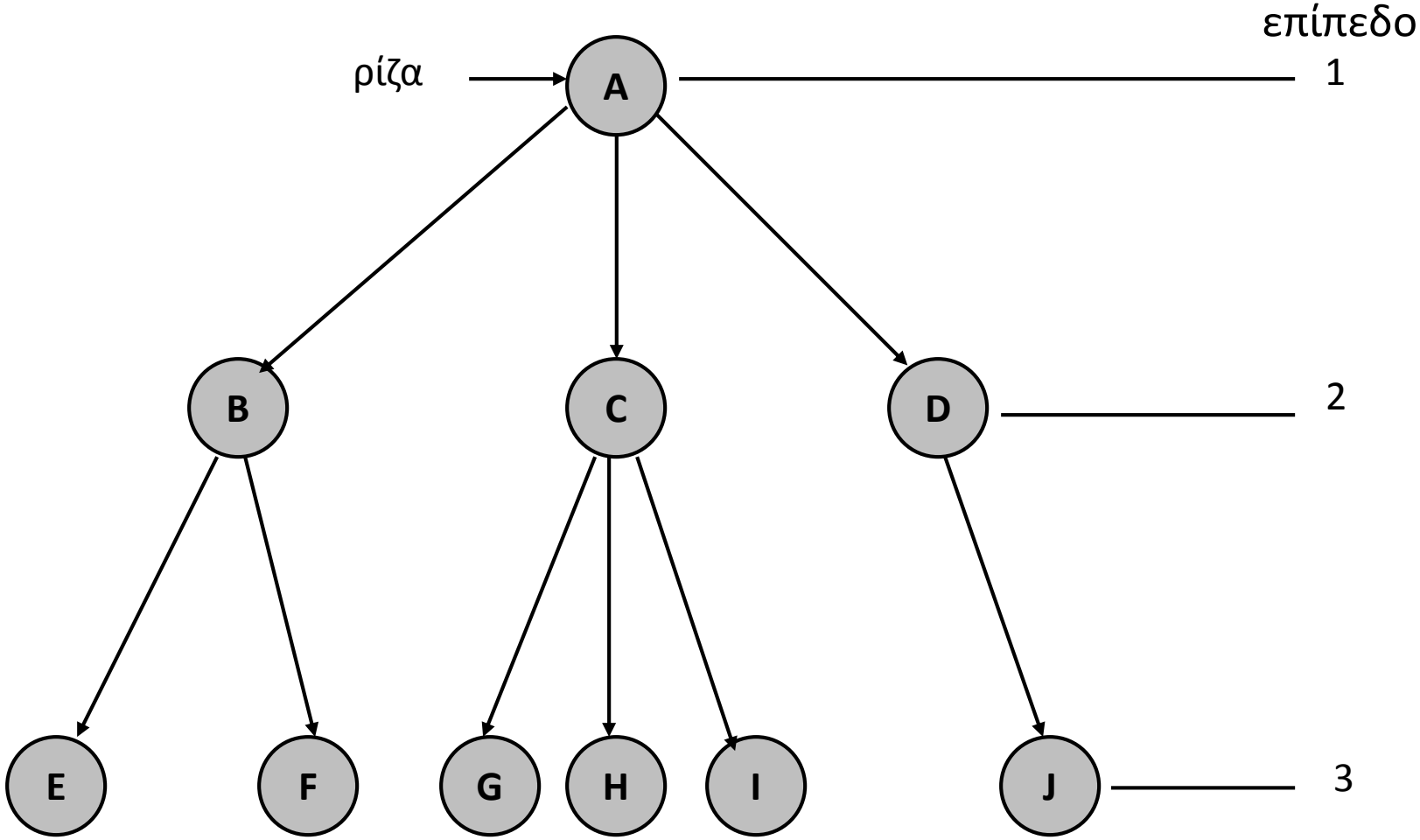
Ορισμός (αναδρομικός ορισμός):

Ένα δέντρο είναι ένα πεπερασμένο σύνολο κόμβων το οποίο είτε είναι κενό είτε μη κενό σύνολο τέτοιο ώστε:

1. Υπάρχει ένας μοναδικός κόμβος, που καλείται ρίζα.

2. Οι υπόλοιποι κόμβοι χωρίζονται σε n>=0 ξένα μεταξύ τους υποσύνολα, T_1, T_2, ... ,T_n, κάθε ένα από τα οποία είναι ένα δέντρο. Τα T_1, T_2, ... ,T_n καλούνται υπόδεντρα (subtrees) της ρίζας.

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}.



Νέοι Όροι

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

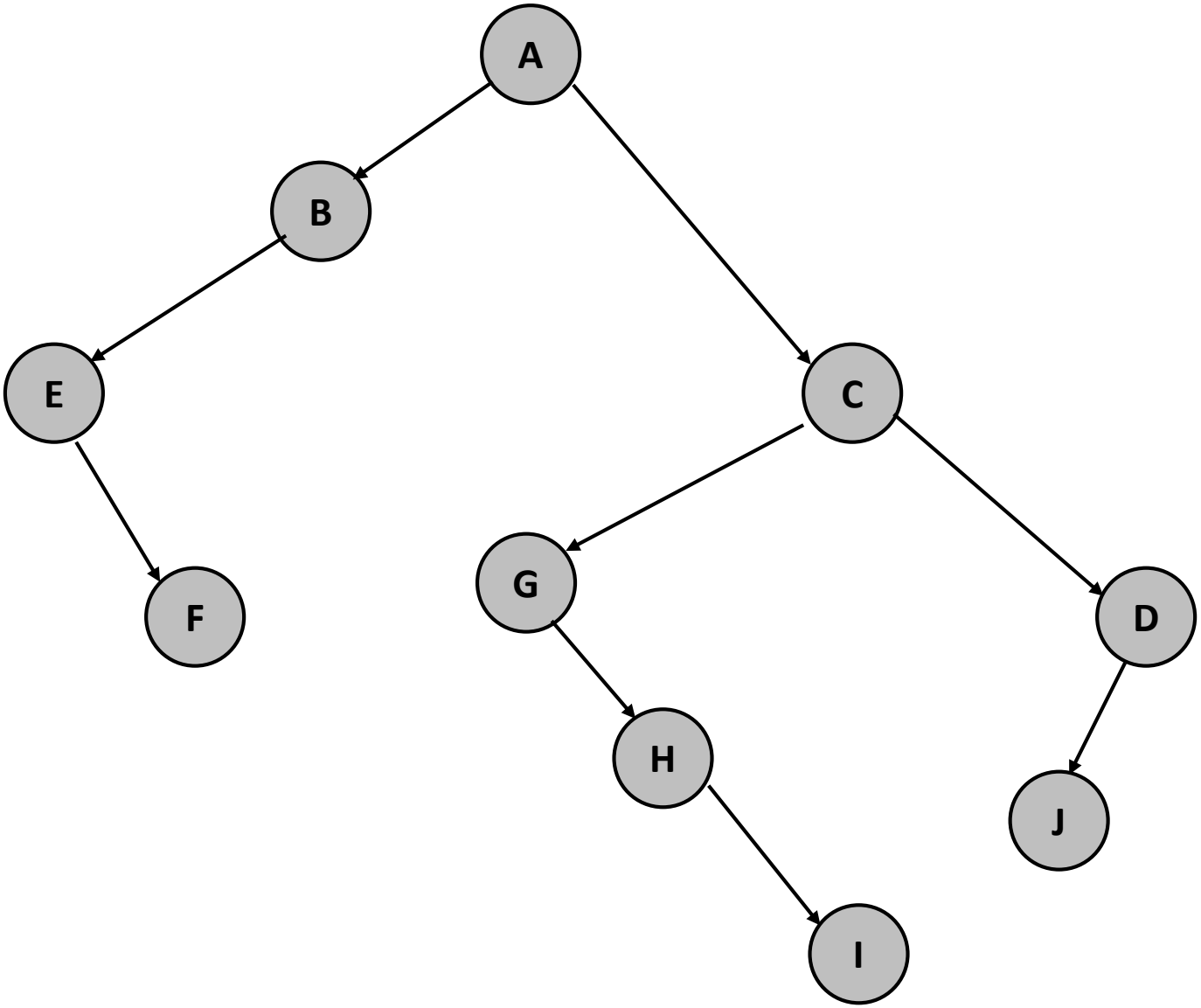
Δυαδικό δέντρο

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.


A
B
E
C
F
G
D
H
J
I

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. Έστω 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 δέντρο ύψους h είναι:

$$\sum_{i=1}^{h} (\text{μέγιστος αριθμός στο επίπεδο } i) =$$

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

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

$$\left\lceil \log_2 (n+1) \right\rceil$$

Απόδειξη

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

άνω φράγμα (ακέραιος) $\log_2 (n+1)$

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

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

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

Αλλά επίσης

$$B = n_1 + 2n_2$$

άρα $n = 1 + n_1 + 2n_2$ (2)

Από τις (1) και (2) έχουμε :

$$n_0 = n_2 + 1.$$

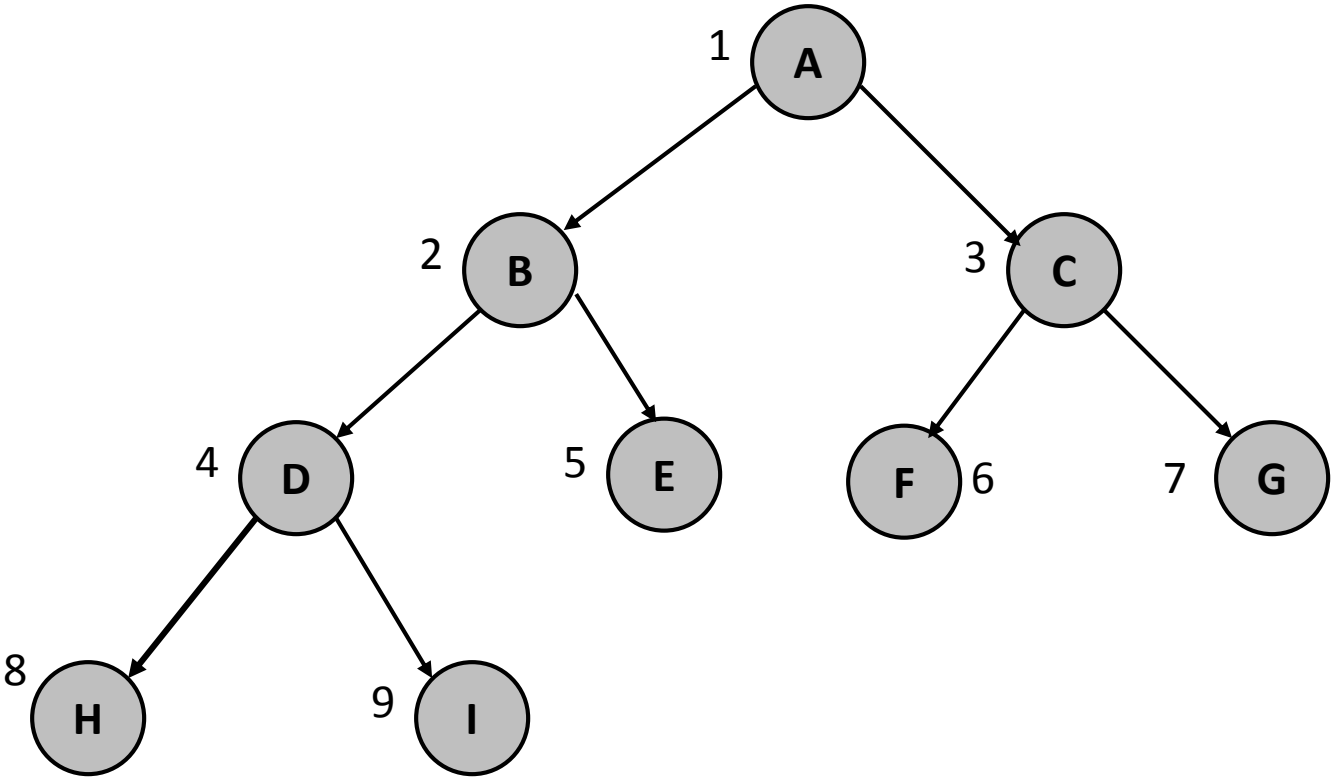
Για τον πλήρη ορισμό του ΑΤΔ δυαδικό δέντρο απομένει να ορισθούν οι βασικές πράξεις του που είναι:

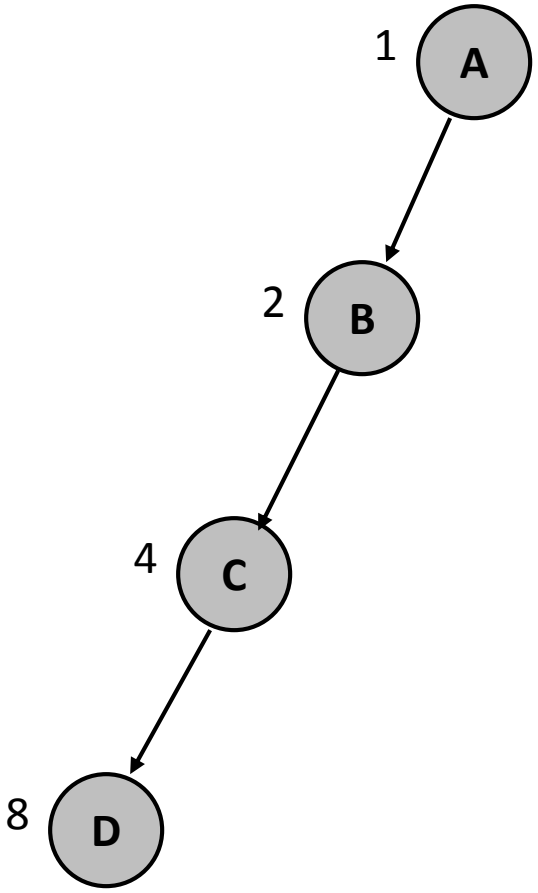
1. Δημιουργία
2. Κενό
3. Αριστερό παιδί
4. Δεξί παιδί
5. Ανάκτηση Δεδομένων (περιεχόμενο)
6. Αλλαγή Δεδομένων

---

7. Πατέρας (γονέας)
8. Διαγραφή υποδέντρων
9. Αντικατάσταση υποδέντρων
10. Συγχώνευση
11. Αναζήτηση

Α. Απεικόνιση (σχεδιασμός) δυαδικών δέντρων με πίνακα





Αριθμούμε τις δυνητικές θέσεις των κόμβων από 1 έως 2^h-1, όπου h το ύψος (βάθος) του δυαδικού δένδρου.

Συναρτήσεις πρόσβασης

Πρόταση

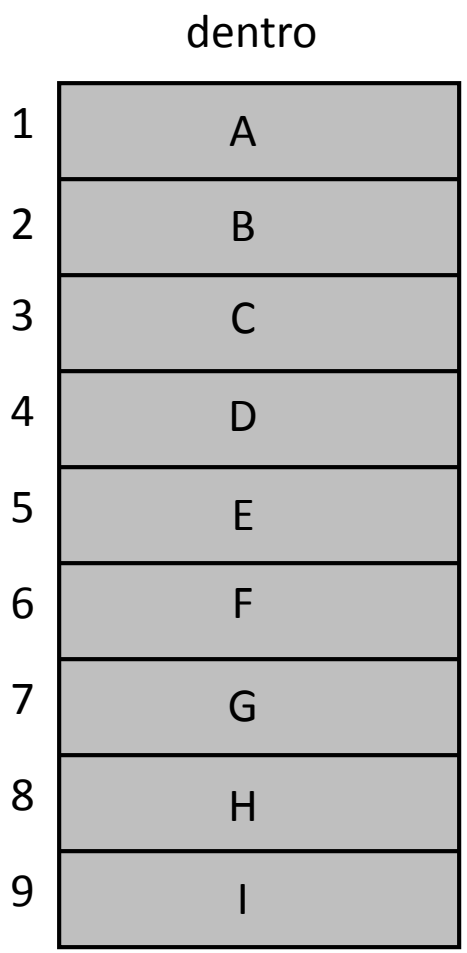
Αν ένα πλήρες δυαδικό δέντρο με n κόμβους (δηλ. βάθος =$\lfloor \log_2(n+1) \rfloor$) παριστάνεται ακολουθιακά όπως παραπάνω, τότε για οποιονδήποτε κόμβο με δείκτη i, 1<= i<= n έχουμε:

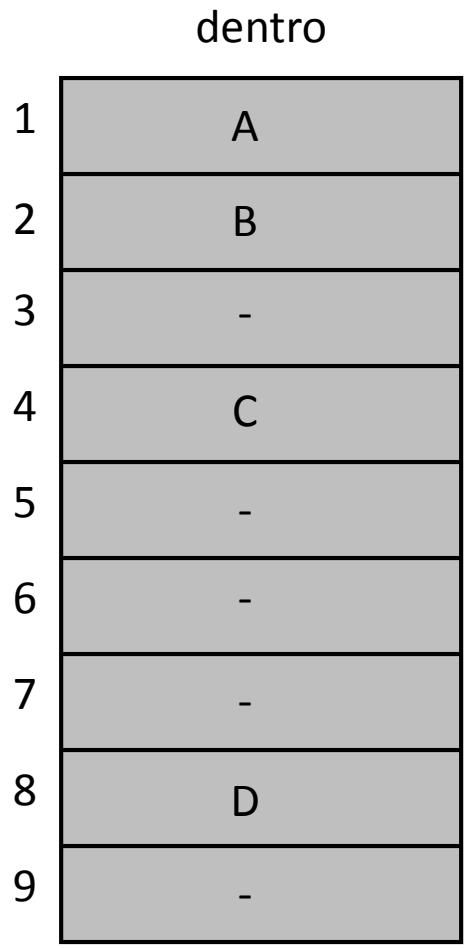
(1) Ο pateras(i) είναι στη θέση αν i< >1. Όταν i = 1, τότε δεν υπάρχει πατέρας, γιατί το i είναι η ρίζα.

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

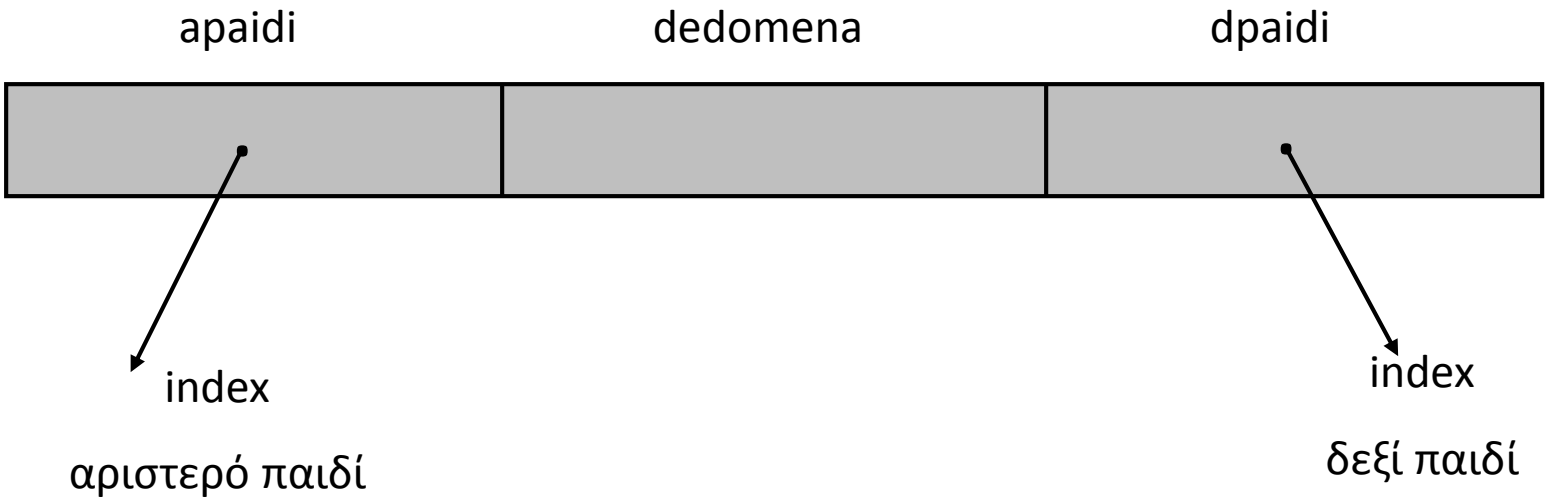
(3) Το dpaidi(i) είναι στη θέση 2i + 1, αν 2i + 1 <= n. Αν 2i + 1 > n, τότε ο κόμβος δεν έχει δεξί παιδί.

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





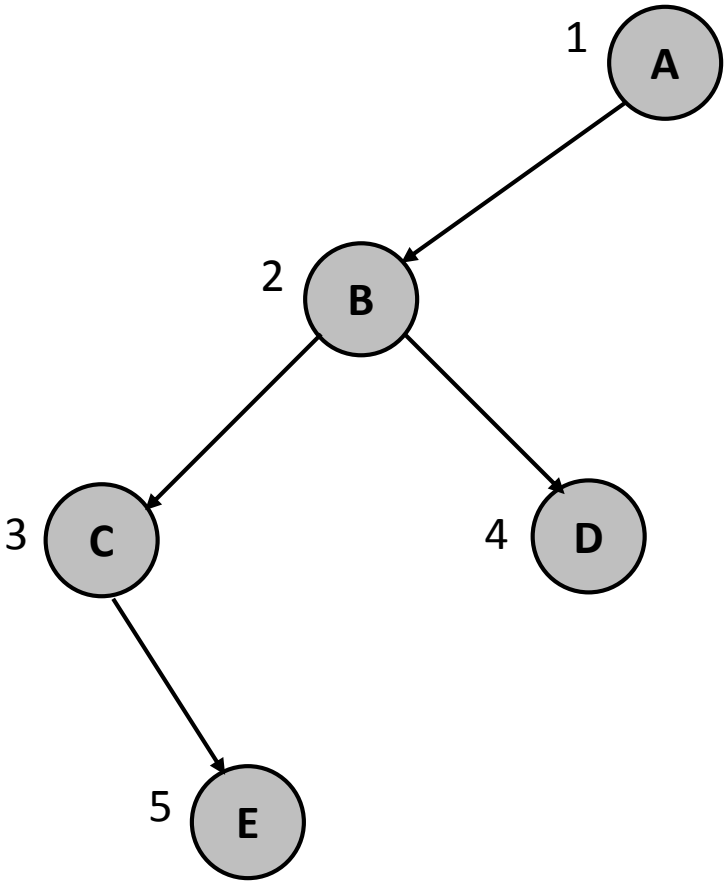
Β. Συνδεδεμένη παράσταση δυαδικού δέντρου με πίνακες



Ένα δυαδικό δέντρο τώρα μπορεί να παρασταθεί ως ένας πίνακας τέτοιων εγγραφών. Έχουμε λοιπόν τις παρακάτω δηλώσεις :

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

typedef int typosSyndesis;
             /* όχι pointer αλλά index */

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

typedef typos_komvou pinakas_komvou[plithos];
```

Συνδεδεμένη παράσταση δυαδικού δέντρου με πίνακες :



Pinakas_komvou Tree1;

Το πρώτο στοιχείο του Tree1 δεν χρησιμοποιείται.
Η ρίζα είναι στο στοιχείο 1 (δες συναρτήσεις)

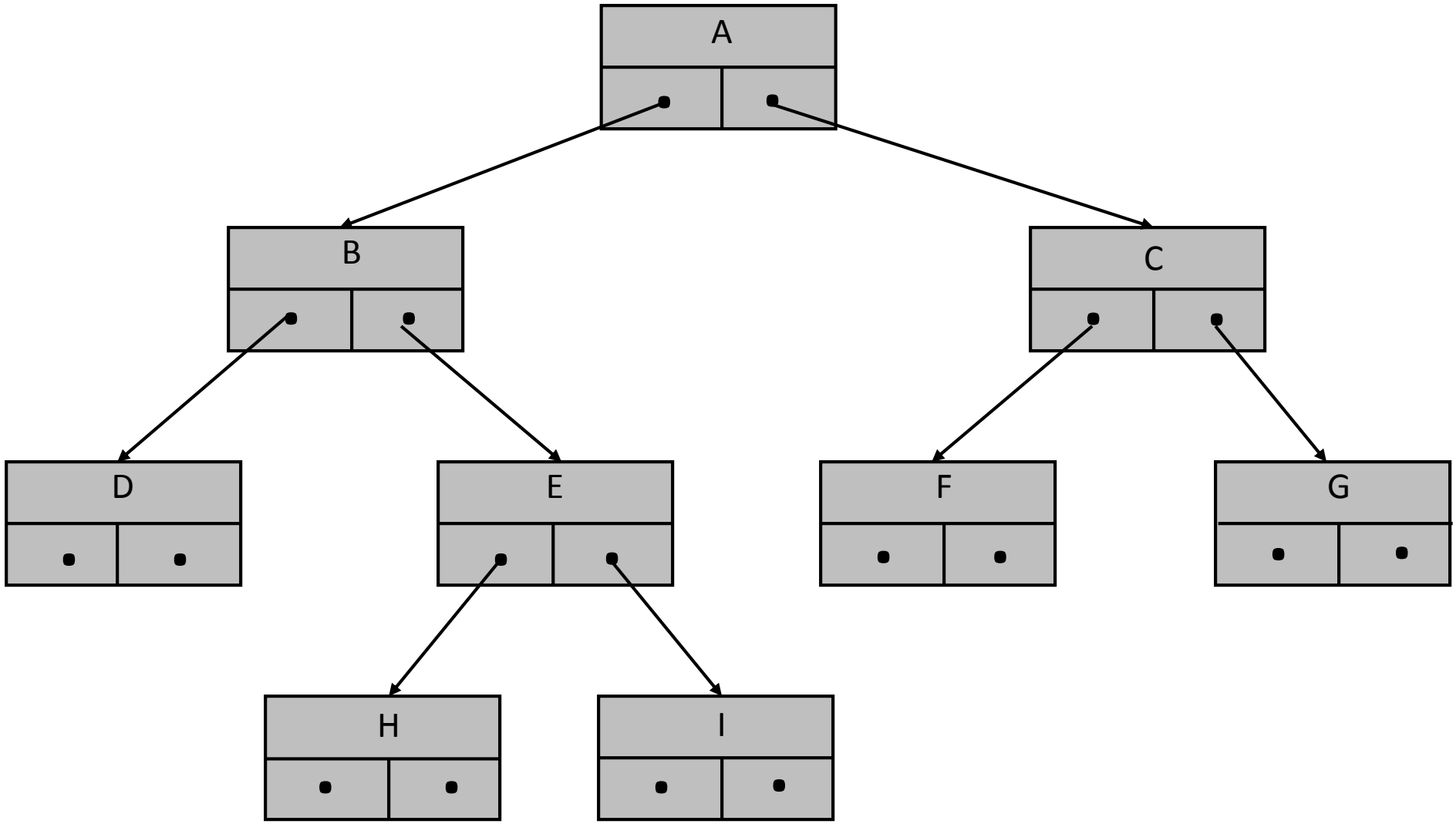
	dedomena	apaidi	dpaidi
1	A	2	-1
2	B	3	4
3	C	-1	5
4	D	-1	-1
5	E	-1	-1

Γ. Υλοποίηση δυαδικών δέντρων με δείκτες (pointers)

```
typedef ... typos_stoixeiou;
typedef struct typos_komvou *typos_deikti;

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

typos_deikti riza;
```

Στην συνέχεια όλες οι υλοποιήσεις θα είναι με δείκτες

Συνδεδεμένη παράσταση δυαδικού δέντρου με δείκτες :



```
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);
}
```

Διαδρομή δυαδικού δέντρου

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

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.

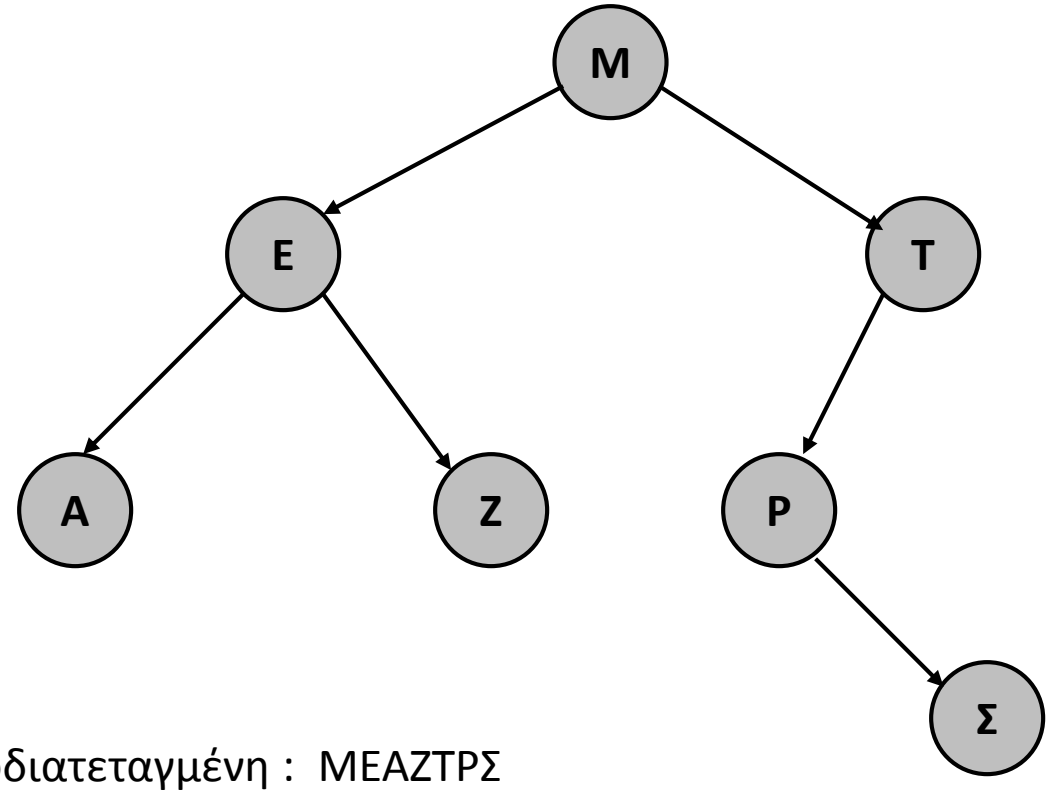
Τότε υπάρχουν έξι διατάξεις για την επίσκεψη των κόμβων ενός δυαδικού δέντρου που είναι οι:

ΚΑΔ ΑΚΔ ΑΔΚ ΚΔΑ ΔΚΑ ΔΑΚ

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

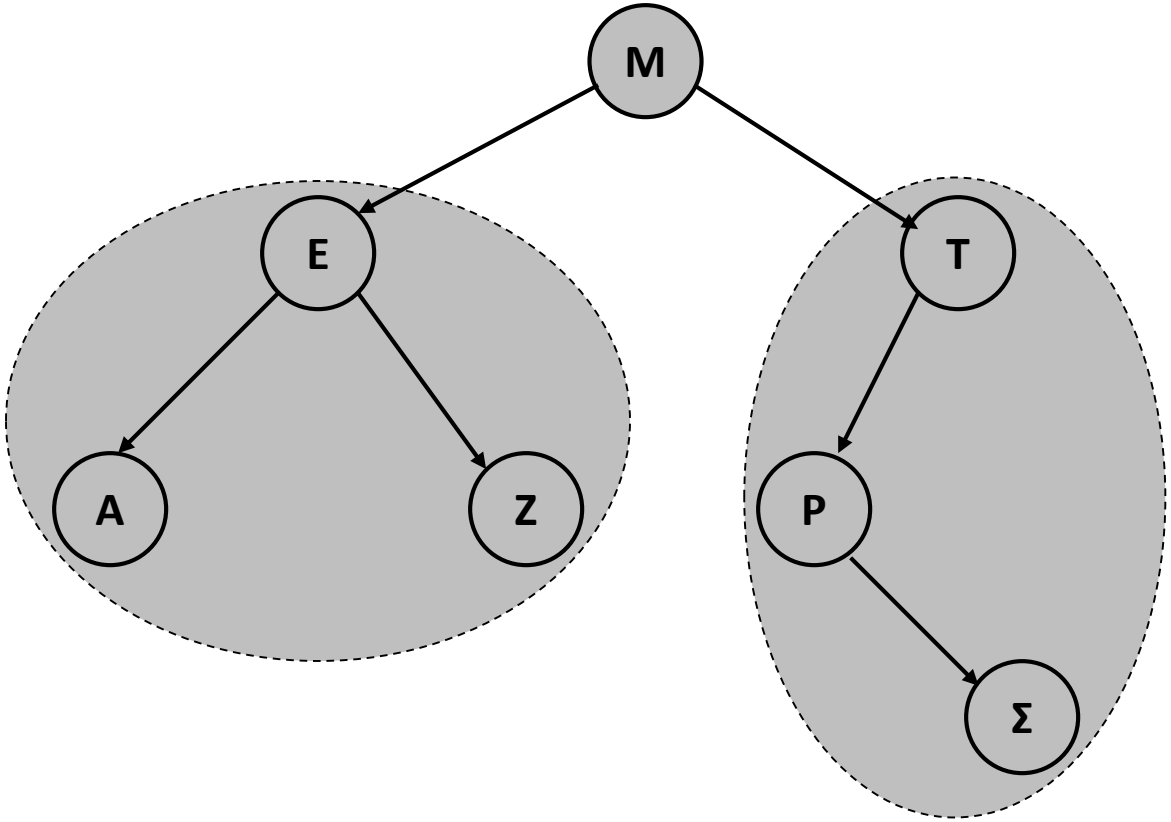
Οι άλλες 3 (ΔΑ) δεν έχουν ενδιαφέρον (συμμετρικές).

Οι τρεις διατάξεις επίσκεψης ενός δυαδικού δέντρου:

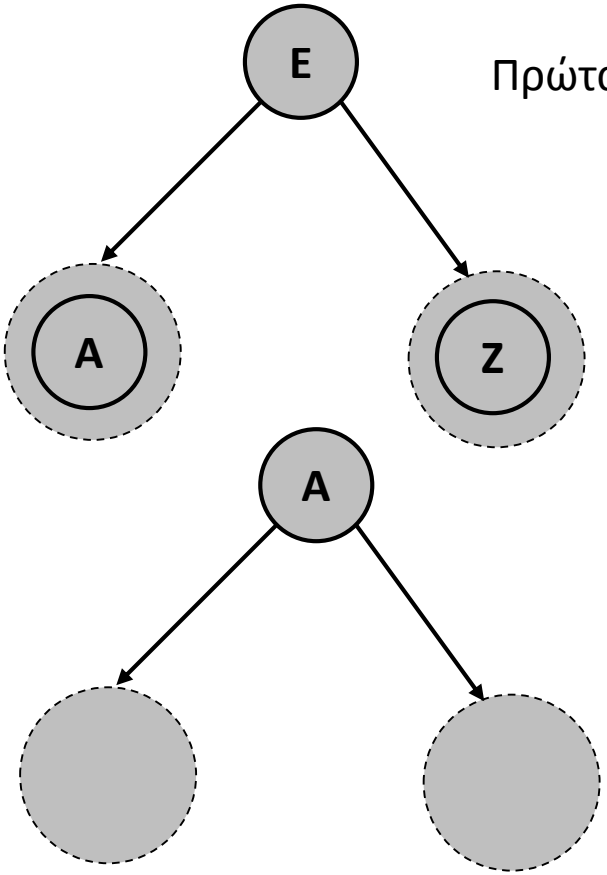


προδιατεταγμένη : ΜΕΑΖΤΡΣ
ενδοδιατεταγμένη : ΑΕΖΜΡΣΤ
μεταδιατεταγμένη : ΑΖΕΣΡΤΜ

Μεταδιατεταγμένη διαδρομή Α Δ Κ



Πρώτα επίσκεψη Αριστερού Υποδένδρου του Μ (ρίζας)

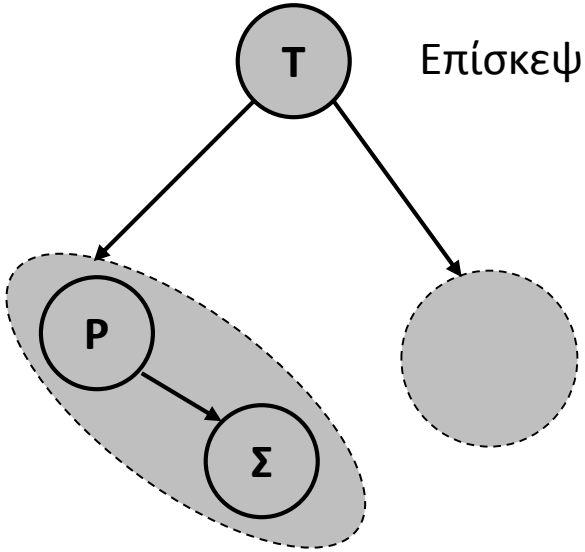


Πρώτα επίσκεψη Αριστερού Υποδένδρου του Ε

Πρώτα επίσκεψη Αριστερού Υποδένδρου του Α, είναι κενό.
Επίσκεψη Δεξιού Υποδένδρου του Α, κενό
Επίσκεψη Α

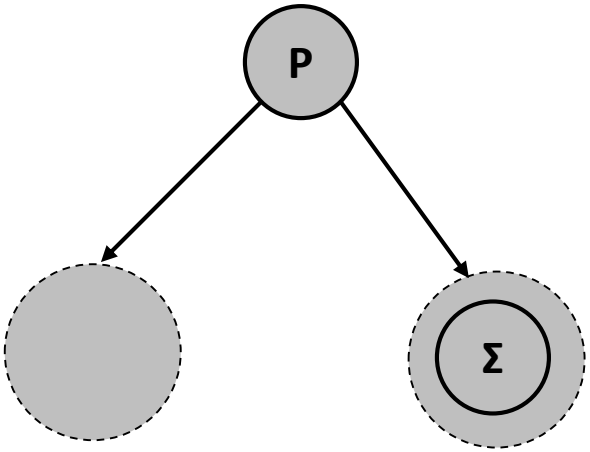
Επίσκεψη Δεξιού Υποδένδρου Ε
Αριστερό Ζ , κενό
Δεξιό Ζ, κενό
Επίσκεψη Ζ

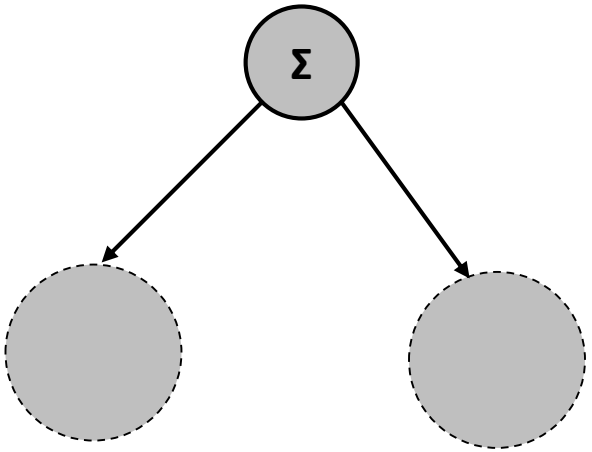
Επίσκεψη Ε



Επίσκεψη Δεξιού Υποδένδρου του Μ (ρίζας)

Επίσκεψη Αριστερού Υ/Δ του Τ
Επίσκεψη Αριστερού Υ/Δ του Ρ, κενό
Επίσκεψη Δεξιού Υ/Δ του Ρ, Σ
Επίσκεψη Αριστερού Υ/Δ του Σ, κενό
Επίσκεψη Δεξιού Υ/Δ του Σ, κενό
Επίσκεψη Σ
Επίσκεψη Ρ
Επίσκεψη Τ
Επίσκεψη Μ





Αναδρομικό Υποπρόγραμμα

Μετα-διατεταταγμένης Διαδρομής

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

  if (!keno_dentro(riza)){
       metadiataksi(riza->apaidi); //αριστερό Υ/Δ
       metadiataksi(riza->dpaidi); //δεξί Υ/Δ
       episkepsi(riza); /* Επίσκεψη της ρίζας */
  }
}
```

όπου η episkepsi τυπώνει το περιεχόμενο ενός κόμβου. Η εντολή

metadiataksi (riza)

καλεί αναδρομικά την metadiataksi προκειμένου να εκτελέσει την μεταδιατεταγμένη διαδρομή του παραπάνω δέντρου.

Αναδρομικό Υποπρόγραμμα

Ενδο-διατεταταγμένης Διαδρομής

```
void endodiataksi(typos_deikti riza){
/* Αναδρομικό υποπρόγραμμα. Διατρέχει με ενδοδιατεταγμένη
διαδρομή ένα δυαδικό δέντρο. */
  if (!keno_dentro(riza)){
       endodiataksi(riza->apaidi); //αριστερό Υ/Δ
       episkepsi(riza); // Επίσκεψη της ρίζας
       endodiataksi(riza->dpaidi); /* δεξί Υ/Δ
  }
}
```

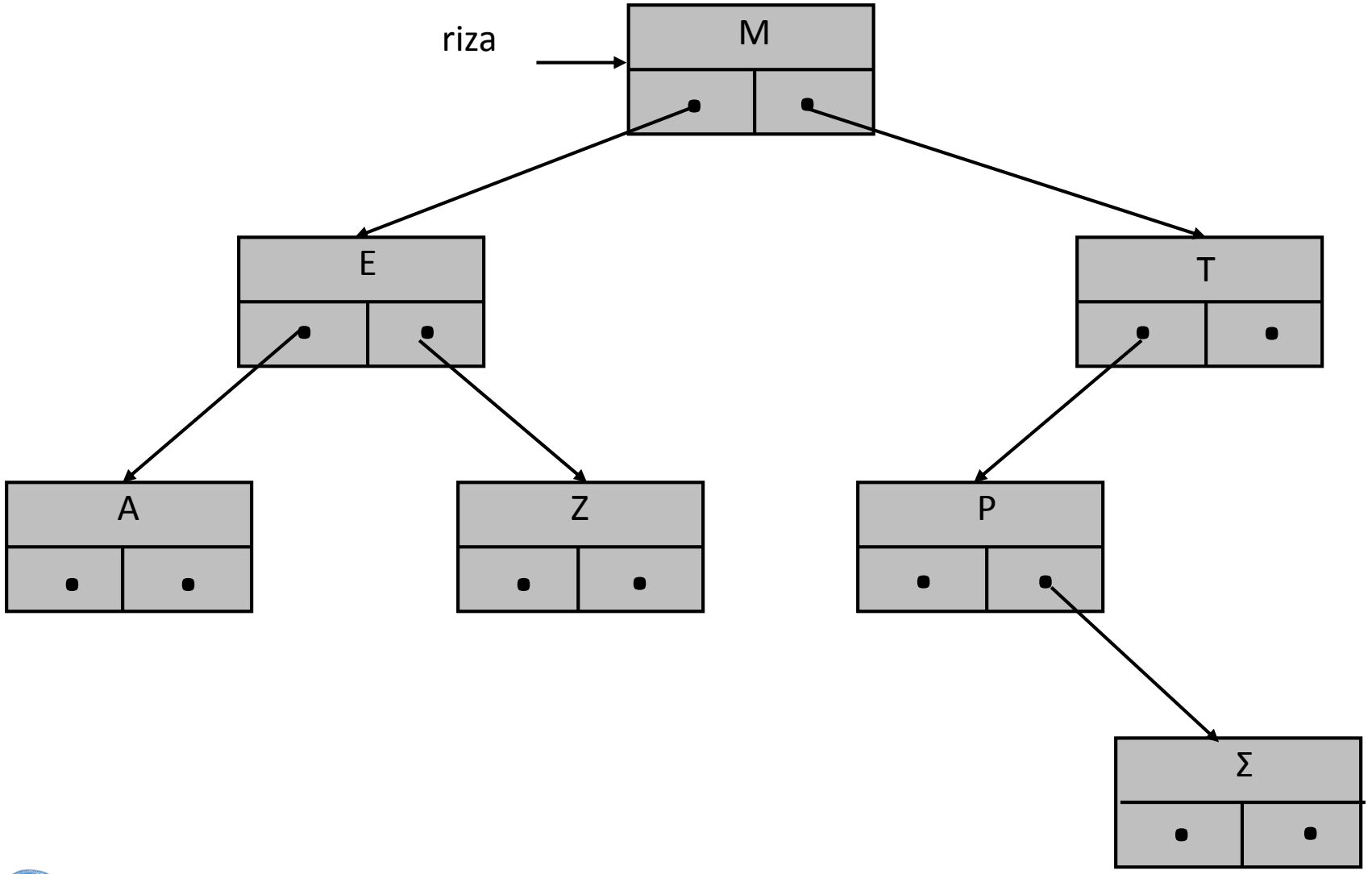
όπου η episkepsi τυπώνει το περιεχόμενο ενός κόμβου. Η εντολή

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:

Ας υποθέσουμε ότι έχουμε το δυαδικό δέντρο:



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

Περιεχόμενο κόμβου	Ενέργεια	Αποτέλεσμα
T	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 (NULLl) 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	
P	Επιστροφή στον κόμβο-πατέρα	
T	Εκτύπωση του περιεχομένου του κόμβου.	T

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

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

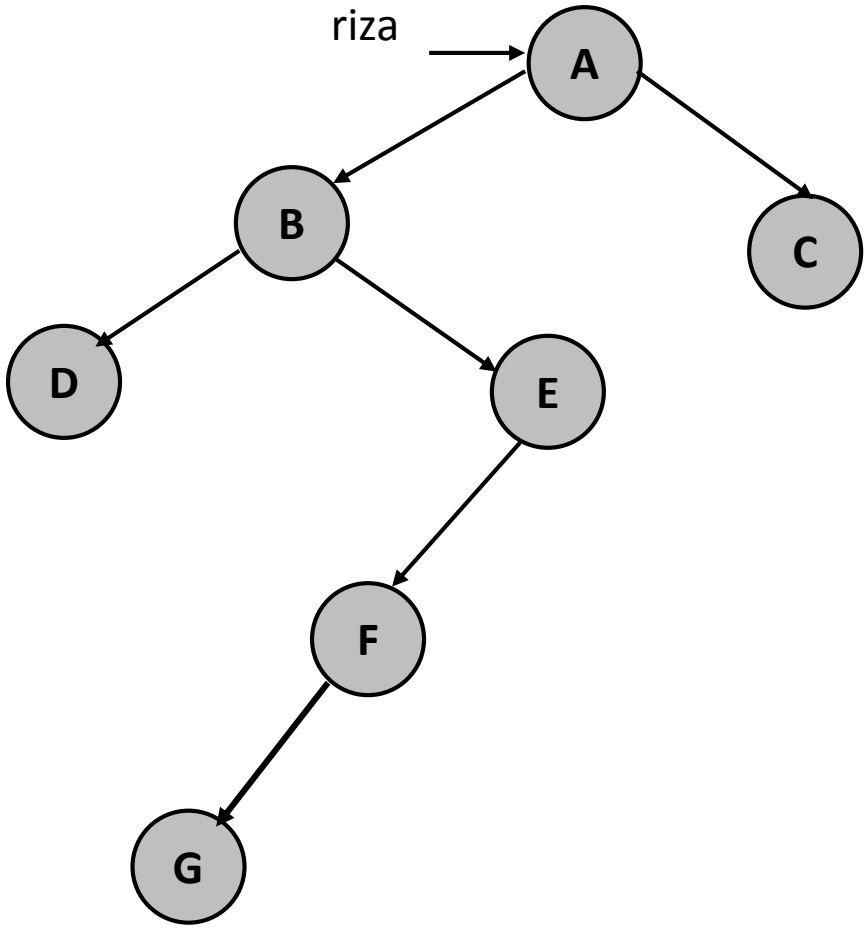
```
void prodiataksi(typos_deikti riza){
/* προδιατεταγμένη  διαδρομή */

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

```
void metadiataksi(typos_deikti riza){
/* μεταδιατεταγμένη διαδρομή */

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

Πολυπλοκότητα O(n)

Περνάμε από κάθε κόμβο είτε ΜΙΑ φορά (φύλλο) είτε ΔΥΟ φορές (εσωτερικός κόμβος).

Μη Αναδρομικά Υποπρογράμματα Διαδρομής



Μη αναδρομικός αλγόριθμος για την ενδοδιατεταγμένη διαδρομή (με χρήση στοίβας)

1. trexon = riza;

2. Όσο η στοίβα δεν είναι κενή και trexon != NULL να εκτελούνται:

α. Να διατρέχονται οι αριστεροί κόμβοι του αριστερού υποδέντρου και να τοποθετούνται οι διευθύνσεις τους σε μια στοίβα. {έως ότου αριστερό υπόδεντρο κενό}

β. Αν η στοίβα δεν είναι κενή τότε

(i) Εξαγωγή διεύθυνσης από στοίβα

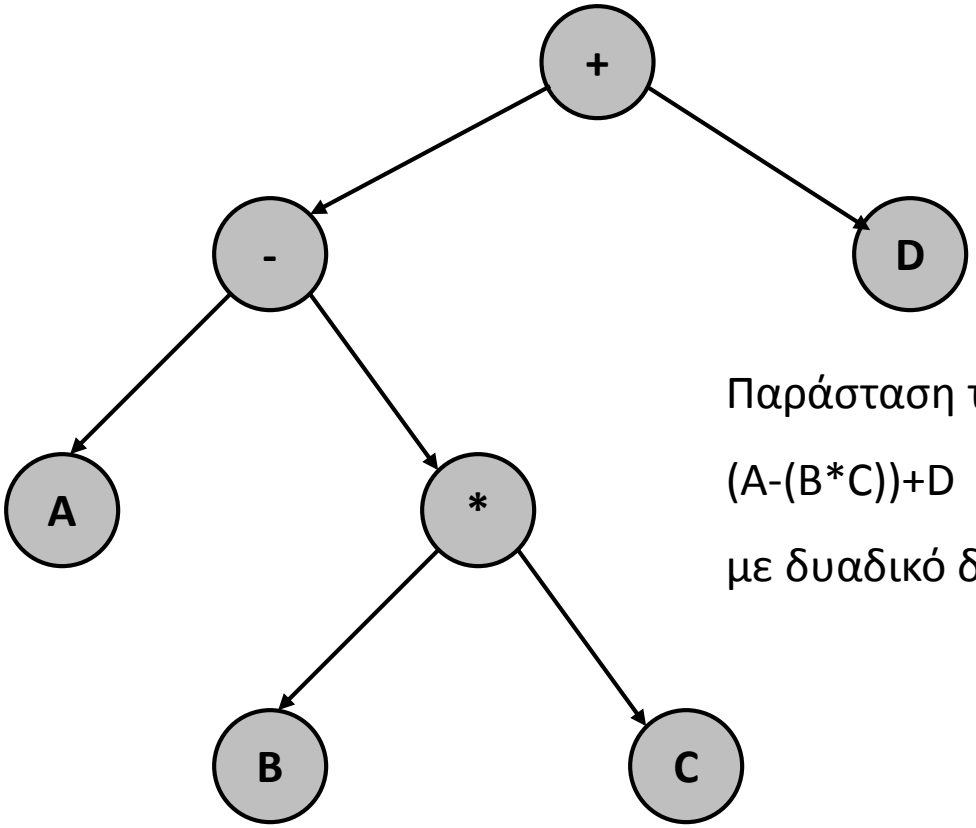
(ii) Επεξεργασία του περιεχομένου του.

(iii) Συνέχισε με δεξί παιδί, trexon= δεξι παιδί

```
void endodiataksi(typos_deikti riza){//Επαναληπτικό
      typos_stoivas stoiva;
      typos_deikti trexon = riza;
      dimiourgia(&stoiva);
      do {//Διατρέχονται οι κόμβοι των αριστερών κλαδιών
         while (trexon!=NULL){
            othisi(&stoiva,trexon);
            trexon = trexon->apaidi;
         }
         /* Το αριστερό υπόδεντρο είναι κενό */
         if (!keni(stoiva)) {
            exagogi(&stoiva,&trexon);
             episkepsi(trexon); /*Επίσκεψη ρίζας */
            trexon = trexon->dpaidi; /*δεξιός κόμβος*/
         }
      while ((!keni(stoiva)) || (trexon!=NULL));
}
```

Ανομοιογενή δυαδικά δέντρα (παραστάσεων-expressions)

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



Παράσταση της

(A-(B*C))+D

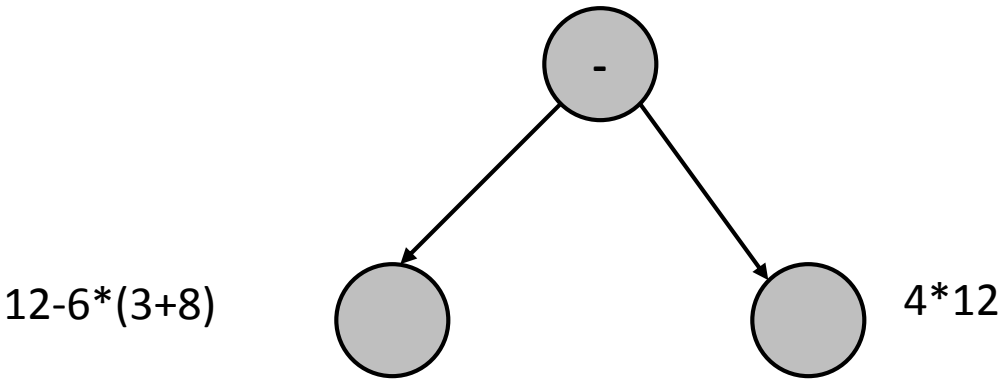
με δυαδικό δέντρο

Για τη μετατροπή μιας ενδοδιατεταγμένης παράστασης (expression) κατασκευάζουμε δυαδικό δέντρο, το οποίο ονομάζεται **δέντρο παράστασης.**

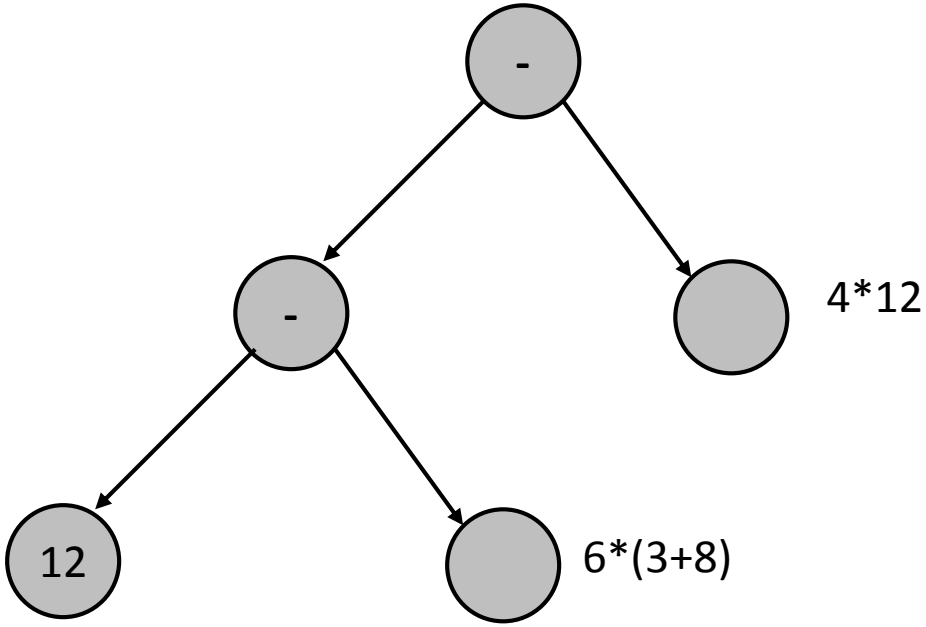
Σε κάθε βήμα να αναζητείται ο τελεστής εκείνος ο οποίος θα εκτελεστεί τελευταίος (έχει τη μικρότερη ιεραρχία).

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

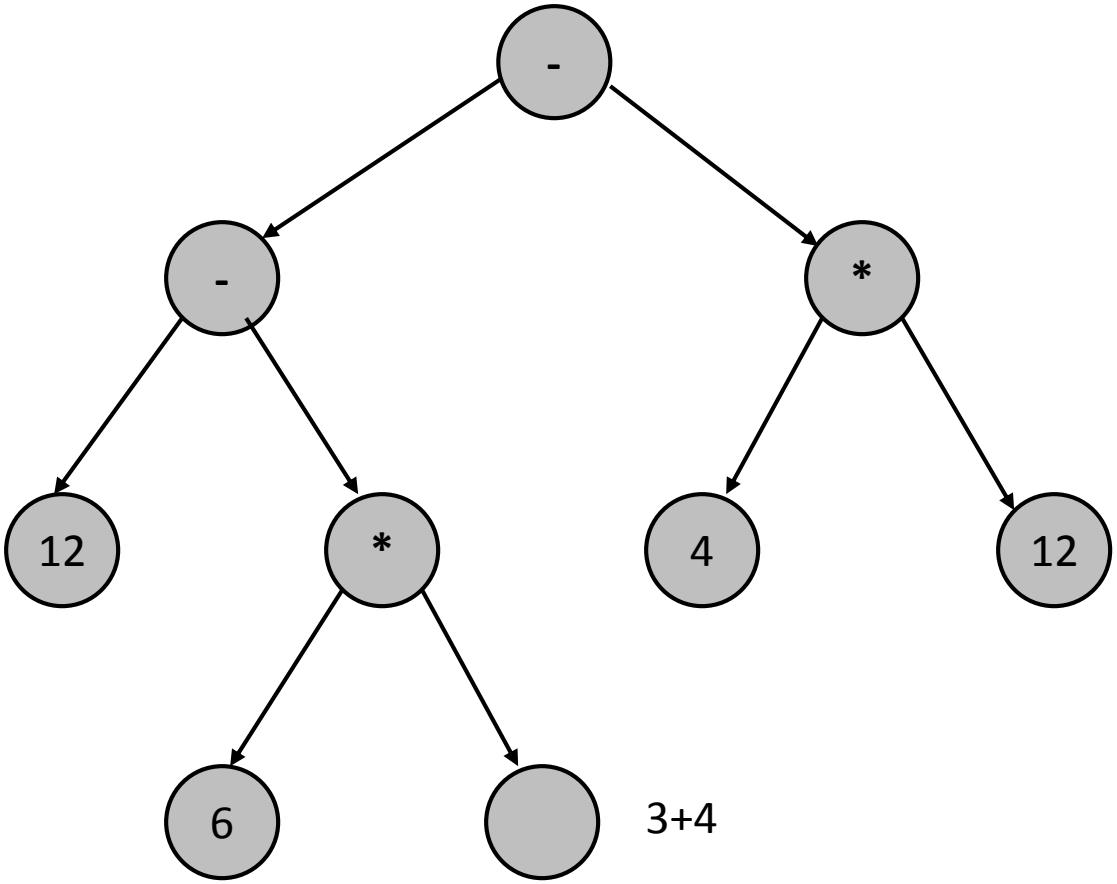
1ο Βήμα (ρίζα ο τελεστής που εκτελείται τελευταίος)



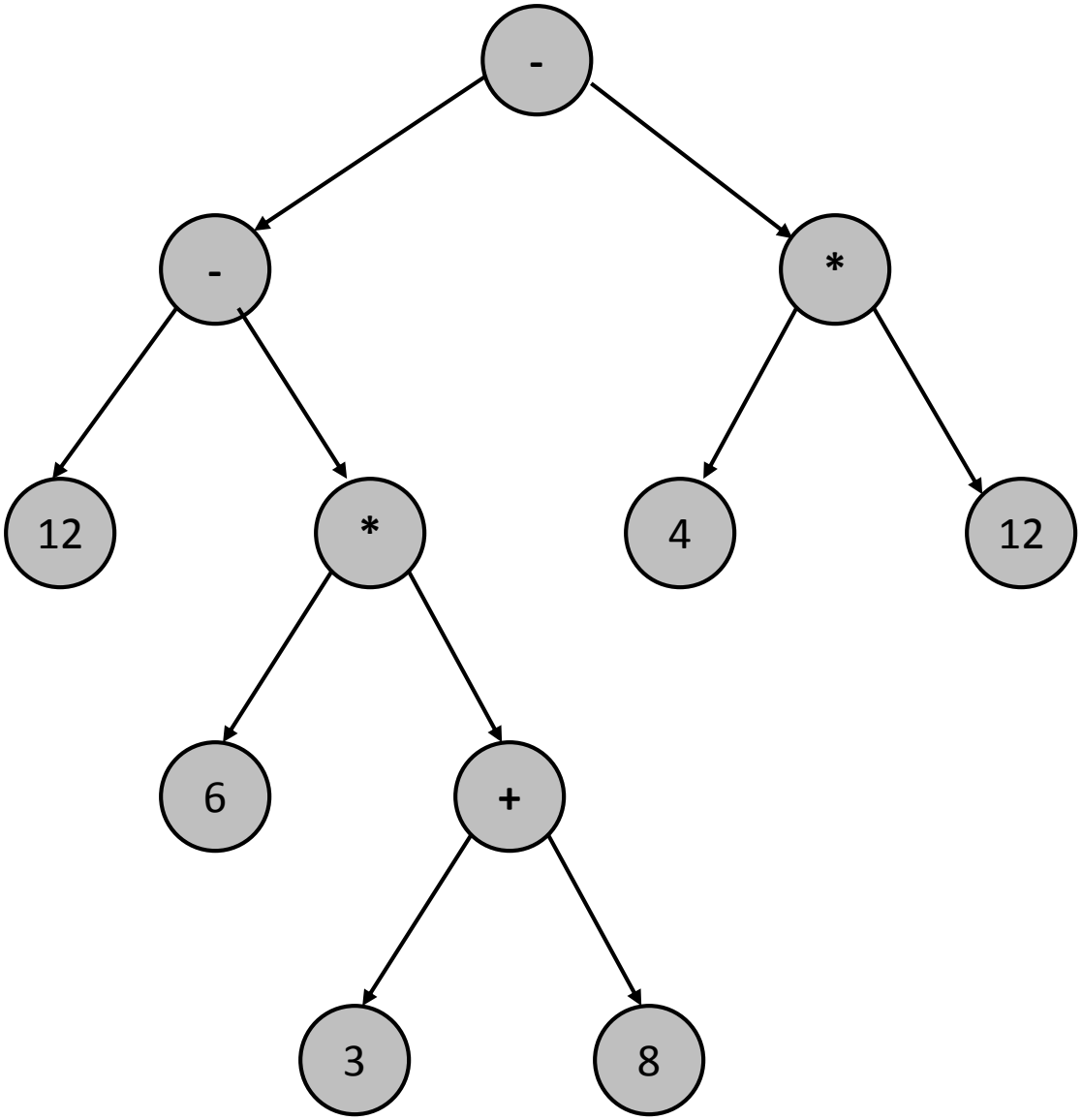
2ο Βήμα (αναδρομικά σε αριστερό και δεξί υποδένδρο)



3ο Βήμα



4ο Βήμα



```
typedef enum {oros,telest } typos_dedomenon;
typedef struct typos_komvou *typos_deikti;

struct telestis {
    char xar;
    typos_deikti apaidi;
    typos_deikti dpaidi;
};
union oros_telestis{
    float arithmos;
    struct telestis xaraktiras;
};
typedef struct typos_komvou {
    typos_dedomenon simadi;
    oros_tel dedomena;
};
```

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){
      float oros1,oros2;
      char  symbolo;

      switch (riza->simadi){
      case oros:return riza->dedomena.arithmos;
      case telestis:
            oros1 = ypologismos_dentrou
                  (riza->dedomena.xaraktiras.apaidi);

            oros2 = ypologismos_dentrou
                  (riza->dedomena.xaraktiras.dpaidi);

            symbolo = riza->dedomena.xaraktiras.xar;

            return telestis(symbolo, oros1, oros2);
      }
}
```

ΑΤΔ Δυαδικό Δένδρο Αναζήτησης

Δυαδικά Δέντρα Αναζήτησης (Binary Search Trees)

Ορισμός : Ένα δυαδικό δέντρο αναζήτησης t είναι ένα δυαδικό δέντρο, το οποίο είτε είναι κενό είτε:

(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 δέντρο να παραμένει 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)
```

Αν το περιεχόμενο ενός κόμβου του ΔΔΑ riza είναι ίσο με stoixeio τοτε
το prosorinos δείχνει τον κόμβο αυτόν και η vrethike είναι 1, αλλιώς το *vrethike
είναι 0 και το prosorinos είναι NULL. */

```
void anazitisi_dentrou(typos_deikti riza,
                       typos_stoixeiou stoixeio,
                       typos_deikti *prosorinos, int *vrethike){

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

Πολυπλοκότητα O(h), όπου h το ύψος του δένδρου

Μη Αναδρομικό υποπρόγραμμα Πολυπλοκότης : O(h):

```
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;
}
```

Εισαγωγή κόμβου

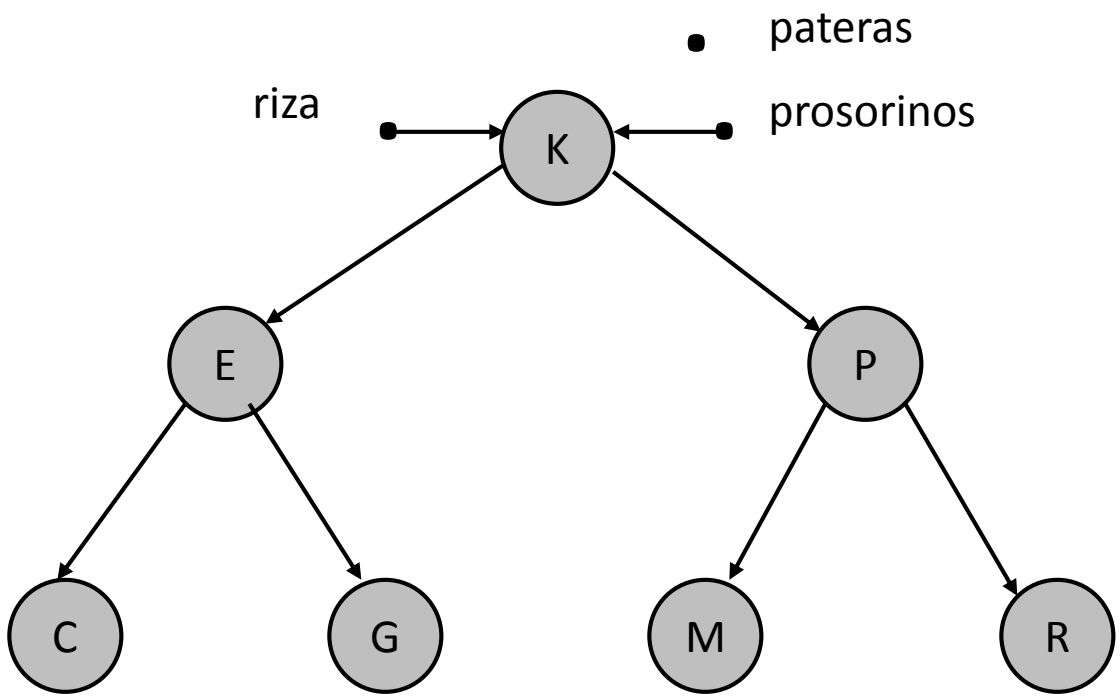
```
int eisagogi_dentro(typos_deikti *riza, typos_stoixeiou stoixeio)

/* Μετά: Αν το stoixeio δεν ανήκει στο ΔΔΑ τότε
        εισάγεται και επιστρέφεται 1, αλλιώς επιστρέφεται 0. */
```

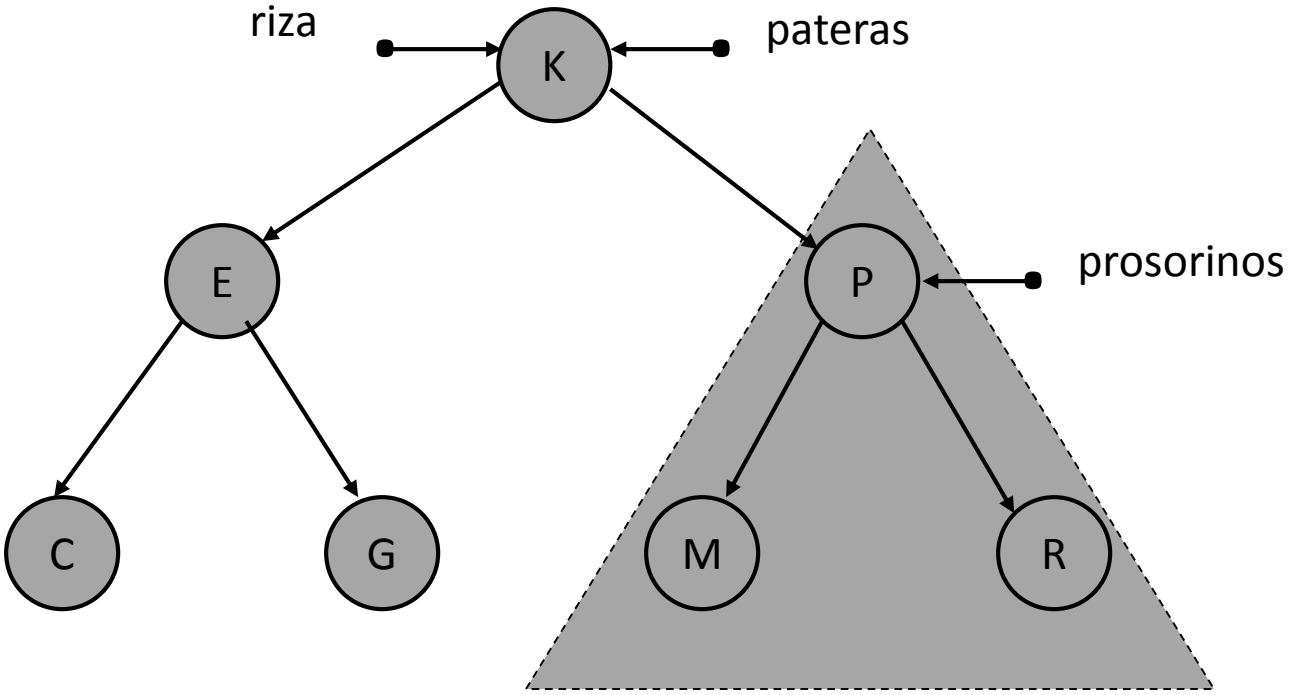
Η εισαγωγή γίνεται πάντα ως νέο φύλλο. Σε θέση ώστε να διατηρείται η ιδιότητα ΔΔΑ.

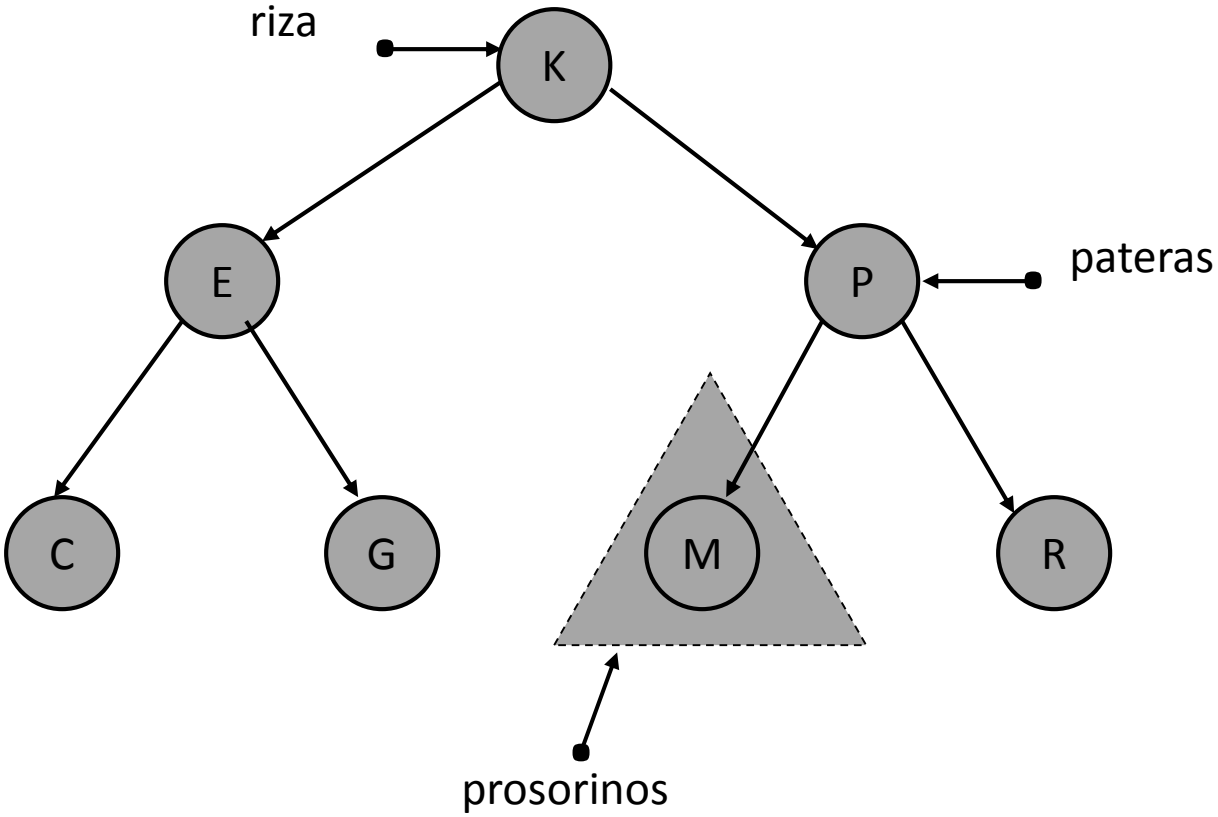
```
int eisagogi_dentro(typos_deikti *riza,
                        typos_stoixeiou stoixeio){
      int eisagogi;
      if (keno_dentro(*riza)){
            *riza = malloc(sizeof(struct typos_komvou));
            (*riza)->dedomena = stoixeio;
            (*riza)->apaidi =(*riza)->dpaidi = NULL;
  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

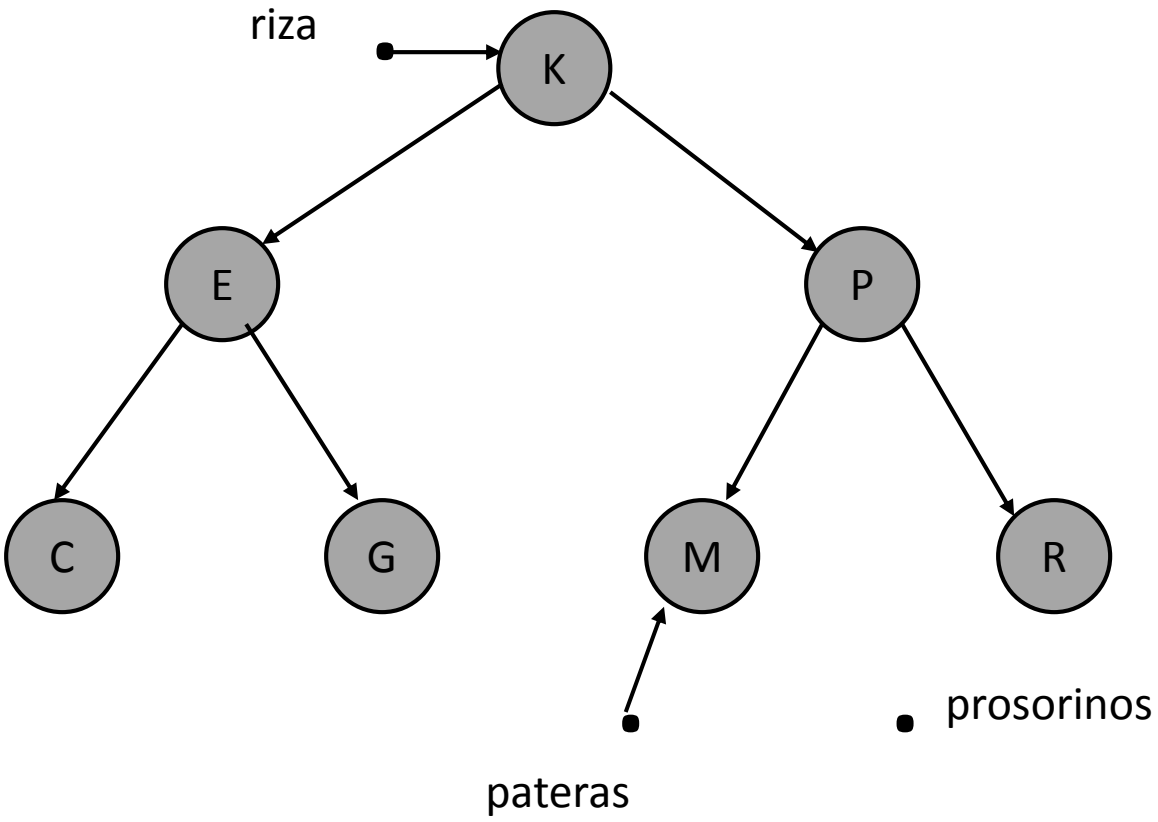
Μη αναδρομική εισαγωγή

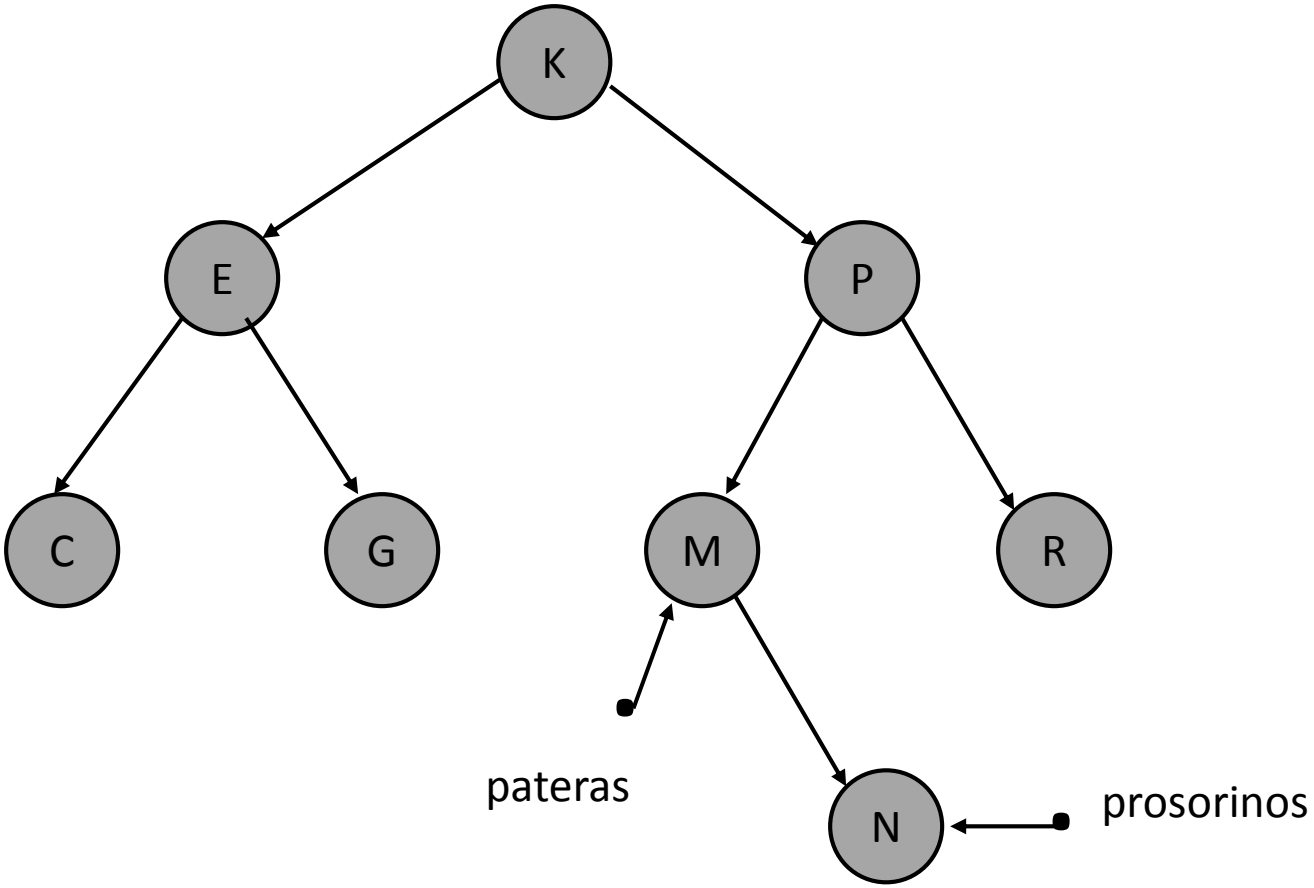
Ας υποθέσουμε ότι επιθυμούμε την εισαγωγή του Ν στο ακόλουθο δυαδικό δέντρο αναζήτησης . Να βρούμε τον πατέρα (γονέα)




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
prosorinos
pateras


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 τότε
        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. */
```

```
typos_deikti anazitisi_patera(typos_deikti riza,
                  typos_stoixeiou stoixeio,
                  typos_deikti *pateras, int *vrethike){

      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. */
```

```
int eisagogi_dentro(typos_deikti *riza,
                    typos_stoixeiou stoixeio){
    typos_deikti prosorinos, pateras;
    int vrethike;

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

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

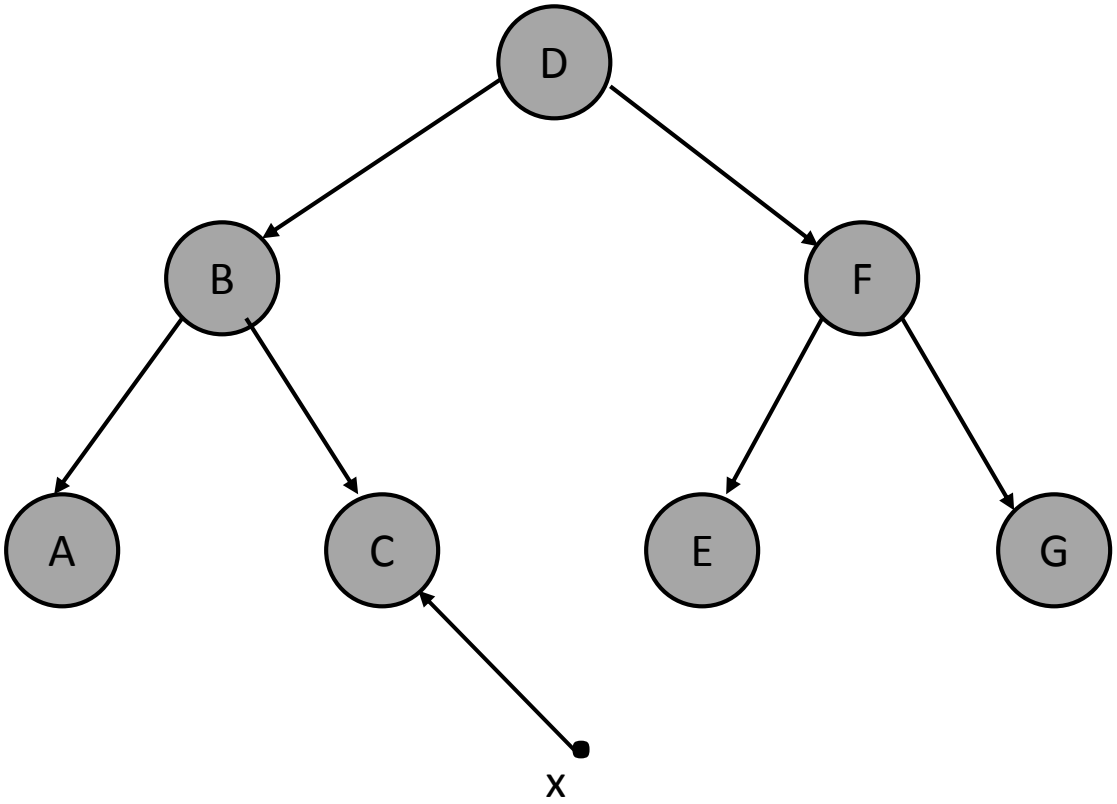
Διαγραφή κόμβου

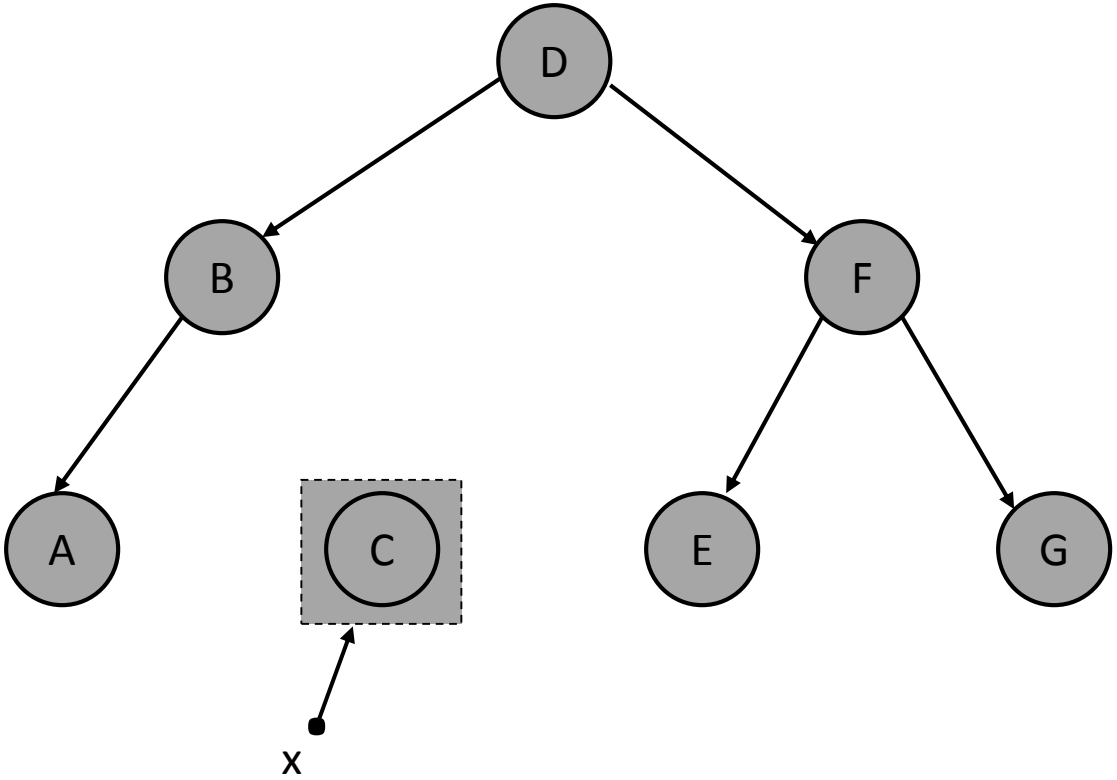
Για τη διαγραφή ενός κόμβου x από ένα ΔΔΑ διακρίνουμε τρεις περιπτώσεις:

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

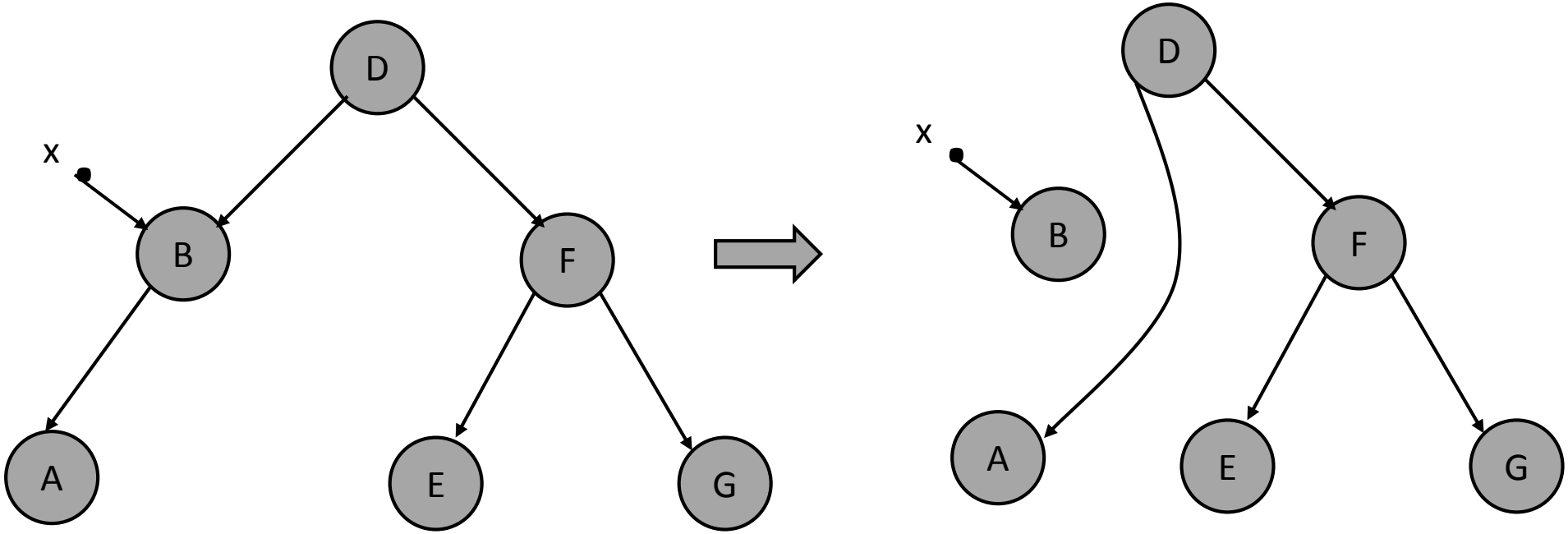
τον διαγράφουμε και τον δείκτη που οδηγεί σε αυτόν NULL




D
B
F
A
C
E
G
x

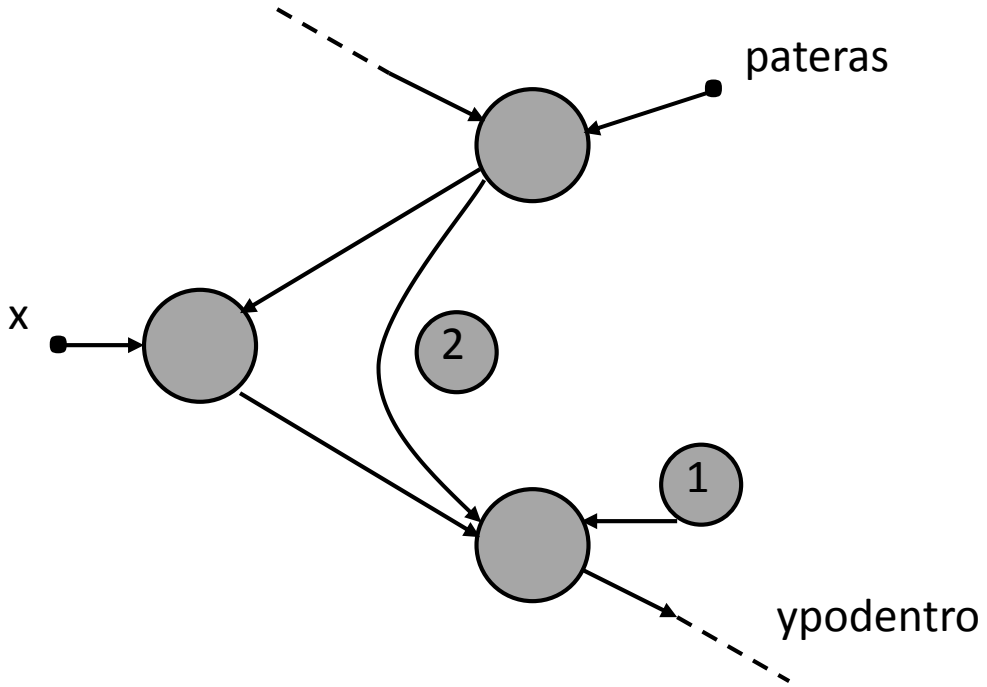
2. Ο κόμβος x έχει ένα παιδί

τον παρακάμπτουμε και τον διαγράφουμε



Διαγραφή κόμβου με το πολύ ένα υποδέντρο

(ενοποίηση 1+2)



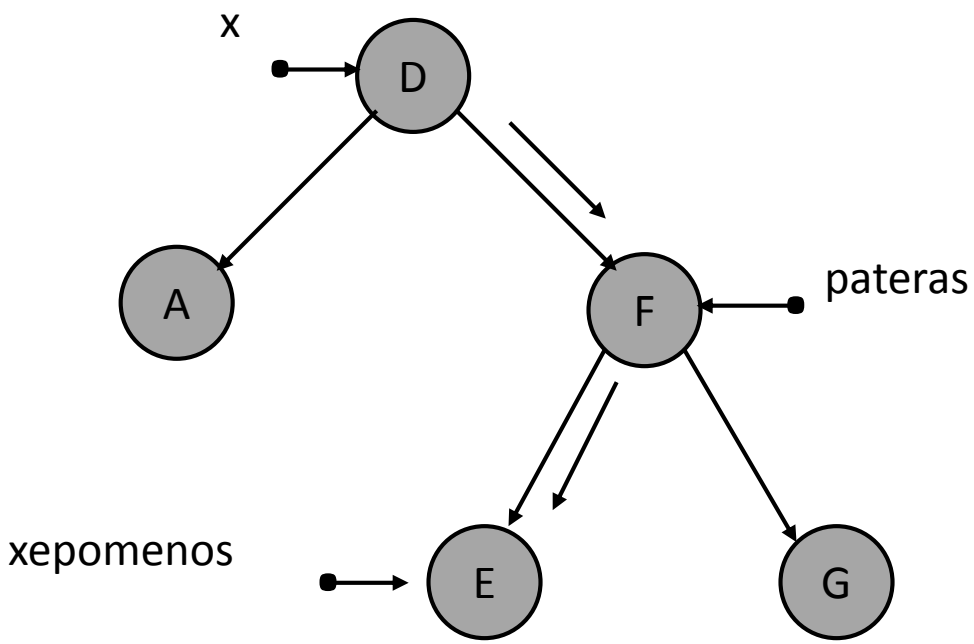
```
        ypodentro = x.dpaidi;
        if (ypodentro == NULL)
                ypodentro = x.apaidi;
/* ο δείκτης ypodentro στο παιδί */

        if (pateras == NULL)/*διαγράφεται η ρίζα*/
                riza = ypodentro;
        else
                if (pateras.apaidi == x)
                pateras.apaidi = ypodentro;
                else
                pateras.dpaidi = ypodentro;
        free(x);
```

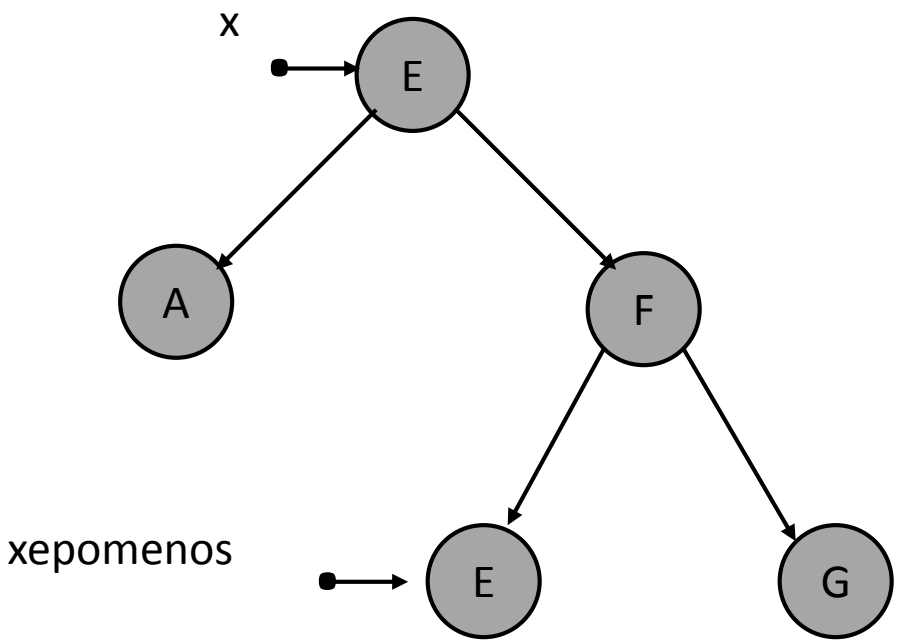
3. Ο κόμβος x έχει δύο παιδιά

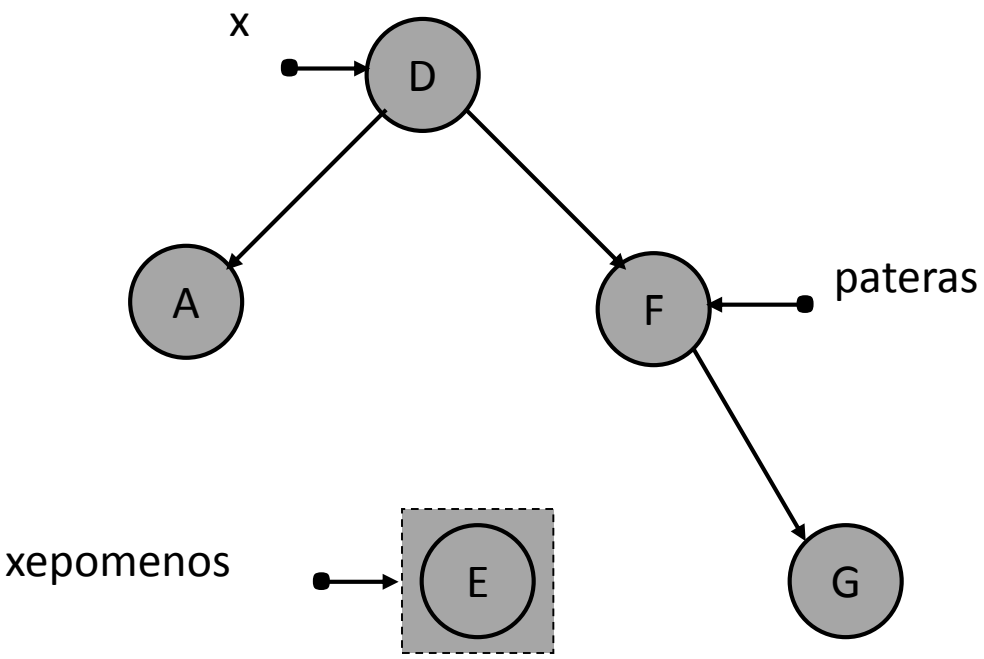
3α. αντικατάσταση τιμής από την αμέσως επόμενη του και

3β. διαγραφή κόμβου προέλευσης τιμής

Πού βρίσκεται η αμέσως επόμενη τιμή του;




x
E
A
F
xepomenos
E
G


x
D
A
F
pateras
xepomenos
E
G

Μη Αναδρομικός

```
int diagrafi_dentrou (typos_deikti *riza,
                      typos_stoixeiou stoixeio)

/*Εντοπίζει τον κόμβο με περιεχόμενο stoixeio και τον
      διαγράφει από το ΔΔΑ*/
{
      typos_deikti x; //δείχνει τον κόμβο που αναζητούμε

      typos_deikti pateras; // ο πατέρας του x
      typos_deikti xepomenos; /*ο επόμενος του x*/
      typos_deikti ypodentro; /* το υποδέντρο του x*/

      int vrethike;  /*true stoixeio βρίσκεται στο ΔΔΑ*/
```

συνέχεια

```
anazitisi_patera(*riza,stoixeio,x,&pateras,&vrethike);
  if (!vrethike) return 0;
 else
 {    if ((x->apaidi != NULL) && (x->dpaidi != NULL)){
      /*ο κόμβος έχει δύο παιδιά, εντοπισμός του
            επόμενου κόμβου και του πατέρα του */

      xepomenos = x->dpaidi;
      pateras = x;
      while (xepomenos->apaidi != NULL){
            /*μονοπάτι αριστερών υποδέντρων*/
            pateras = xepomenos;
            xepomenos = xepomenos->apaidi;
      }
      /* έχει βρεθεί ο επόμενος xepomenos και απόδοση
            των περιεχομένων του στον x */

      x->dedomena = xepomenos->dedomena;
```

συνέχεια

```
/* διαγραφει τον xepomenos */

/* 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 = NULL;
        else if (pateras->apaidi == xepomenos)
              pateras->apaidi = ypodentro;
        else  pateras->dpaidi = ypodentro;

        free(xepomenos);
    }
}
```

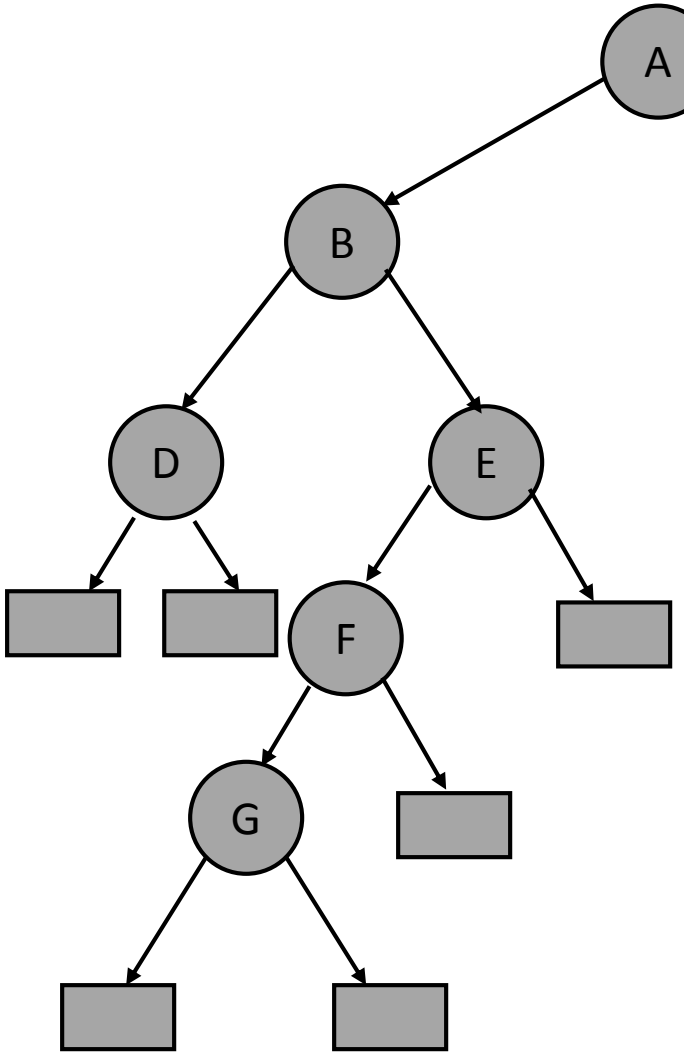
ΑΝΑΔΡΟΜΙΚΟΣ

```
void diagrafi_komvou(typos_deikti *riza,
                          typos_stoixeiou stoixeio){
/* Προ:Υπάρχει ο κόμβος με περιεχόμενο stoixeio στο ΔΔΑ.
   Μετά: Διαγράφτηκε ο κόμβος με περιεχόμενο stoixeio */

  typos_deikti prosorinos, xepomenos;
  if (!keno_dentro (*riza))
      if (stoixeio == (*riza)->dedomena){// Βρέθηκε
            if (keno_dentro((*riza)->apaidi))){
                  /* Η riza δεν έχει αριστερό παιδί */
                  prosorinos = *riza;
                  *riza = (*riza) -> dpaidi;
                  free (prosorinos);
            }else if (keno_dentro((*riza)->dpaidi)))
                  /* Η riza δεν έχει δεξί παιδί */{
                  prosorinos = *riza;
                  *riza = (*riza)->apaidi;
                  free (prosorinos);
            }else /* Η riza έχει δύο παιδιά */
```

```
        {
        /* Εύρεση του ενδοδιατεταγμένου επόμενου */
              xepomenos = (*riza) -> dpaidi;
              while (xepomenos -> apaidi ! = NULL)
                    xepomenos = xepomenos->apaidi;
        /* Απόδοση του περιεχομένου */
              (*riza)->dedomena = xepomenos->dedomena;

        /* διαγραφή του ενδοδιατεταγμένου επόμενου */
              diagrafi_komvou(  &((*riza)-> dpaidi),
                                (*riza)-> dedomena);
        }

    } else /* stoixeio != (*riza) -> dedomena */
     if (stoixeio < (*riza)->dedomena)
          diagrafi_komvou(&((*riza)->apaidi),stoixeio);
     else diagrafi_komvou(&((*riza)->dpaidi),stoixeio);
}
```

Εφαρμογές δυαδικών δέντρων : Κώδικες Huffman

Επεκταμένο Δυαδικό Δέντρο



Προσθέτουμε n+1 εξωτερικούς Κόμβους (ορθογώνιους)

$l_G = 1 + 1 + 1 + 1 = 4$ (μήκος μονοπατιού για τον κόμβο G)

Μεγέθη δένδρου (αθροίσματα μονοπατιών των κόμβων)

E = 3 + 3 + 5 + 5 + 4 + 3 + 2 + 2 = 27 (μήκος εξωτερικού μονοπατιού)

I = 0 + 1 + 1 + 2 + 2 + 3 + 4 = 13 (μήκος εσωτερικού μονοπατιού)

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 n_i είναι το πλήθος των εσωτερικών κόμβων στο επίπεδο i , τότε :

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

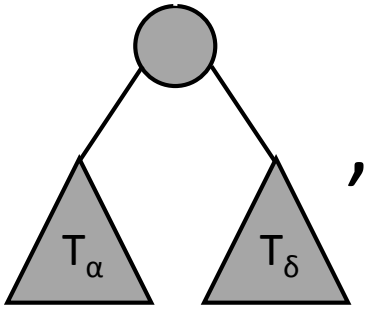
2 Χρήσιμα μέτρα (υποδηλώνουν την μορφή δένδρου)

$$\text{μέση απόσταση ενός εσωτερικού κόμβου} = \frac{I(T)}{n}$$

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

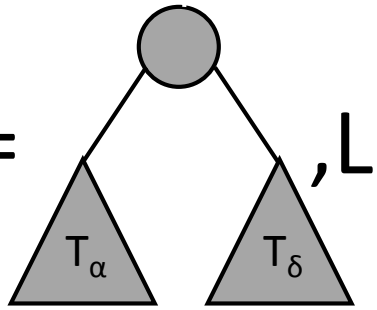
Αναδρομικός Υπολογισμός του I με συνάρτηση I(T, L=0)

$$I(\square, L) = 0$$

$$I\left(T = \right.$$ 

$$\left., L\right) = L\text{-}1 + I(T_\alpha, L+1) + I(T_\delta, L+1)$$

$E(\ \blacksquare\ , L) = L$

$$E\left(T = \right.$$  $$\left., L\right) = E(T_\alpha, L+1) + E(T_\delta, L+1)$$

Πρόταση :

Για ένα δυαδικό δέντρο Τ με n εσωτερικούς κόμβους ισχύει :

$D(T) = E(T) - I(T) = 2n$

Απόδειξη με επαγωγή. Ισχύει για $n=1$, $I(T_1)=0$, $E(T_1)=2$, $E(T_1)-I(T_1)=2$

Έστω ότι ισχύει για $n=k$, δηλαδή $E(T_k)-I(T_k) = 2k$.

Προσθέτουμε έναν εσωτερικό κόμβο ακόμα (k+1 εσωτερικοί) σε επίπεδο L+1. $I(T_{k+1})=I(T_k)+L$;

$E(T_{k+1})=E(T_k)-L+2(L+1)=E(T_k)+L+2$

$E(T_{k+1})-I(T_{k+1})= (E(T_k)+L+2)- (I(T_k)+L) = E(T_k)-I(T_k)+2$

$= 2k+2= 2(k+1)$

Εφόσον E(T) = I(T) + 2n αρκεί να μελετηθούν οι ιδιότητες μόνο του E(T) ή του I(T).

Ο μέσος αριθμός συγκρίσεων S(n) σε μια επιτυχημένη αναζήτηση είναι :

$$S(n) = 1 + \frac{I}{n}$$

(υπόθεση: οι κόμβοι έχουν την ίδια πιθανότητα αναζήτησης)

Ο μέσος αριθμός συγκρίσεων U(n) σε μια αποτυχημένη αναζήτηση είναι :

$$U(n) = \frac{E}{n+1}$$

Συνδιάζοντας τις ανωτέρω σχέσεις 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 στο δένδρο, 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 ;

Μέγιστη (όταν 1 στοιχείο ανά επίπεδο)

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

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

(στο i επίπεδο 2^{i-1} κόμβους)

$$0 + (1 + 1) + (2 + 2 + 2 + 2) + \ldots + (i-1)*2^{i-1} + \ldots + (h-1)* 2^{h-1}$$

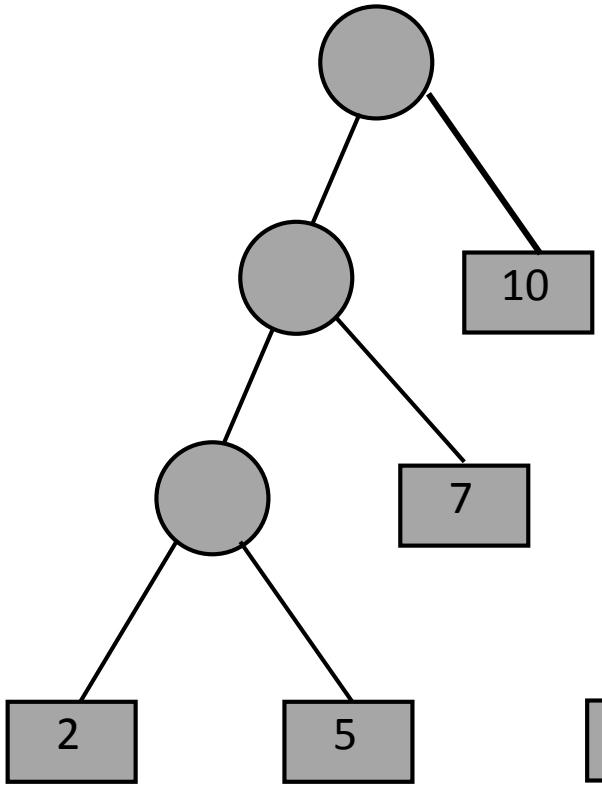
Όπου 1 <= i <= h, το επίπεδο του δένδρου. Αν ένας κόμβος βρίσκεται στο i επίπεδο, χρειαζόμαστε i βήματα και στην χειρότερη περίπτωση h βήματα, δηλαδή $\log_2 (n+1)$, εφόσον $n = 2^h-1$

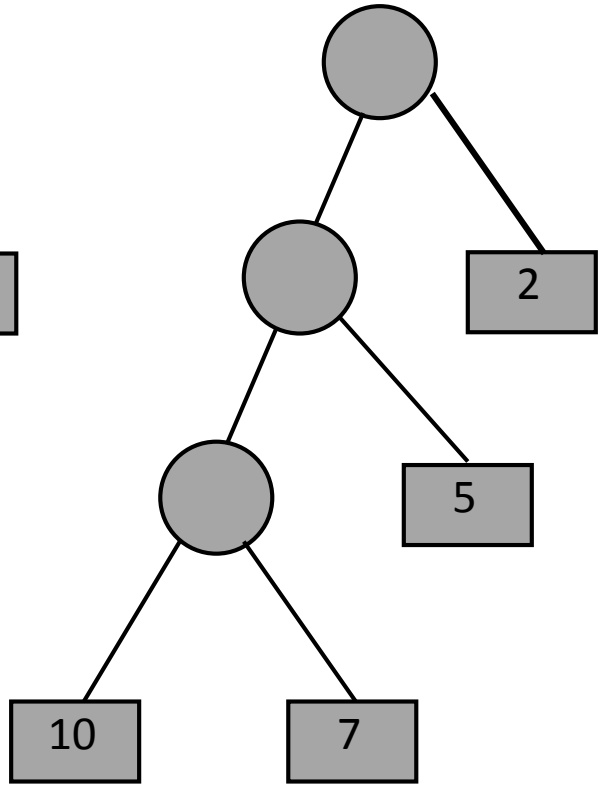
Επομένως το $I_{min} = O(n\log_2 n)$, εφόσον έχουμε n κόμβους

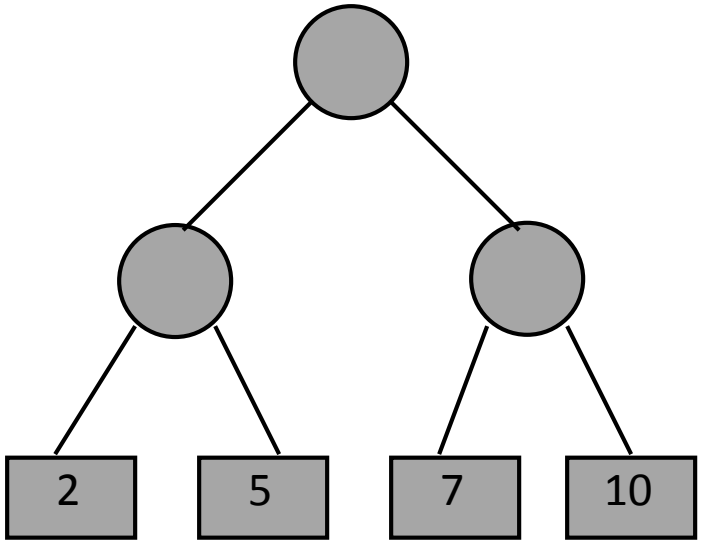
Γενίκευση:
Εξωτερικό Μονοπάτι Βάρους

- Έστω δένδρο με n εξωτερικούς κόμβους (n-1εσωτερικούς)
- Θετικοί αριθμοί (βάρη) $w_1 \ldots w_n$ αντιστοιχούνται στους εξωτερικούς κόμβους (συχνότητα πρόσβασης)
- Εξωτερικό Μονοπάτι Βάρους: $E = \sum_{1 \le i \le n} w_i I_i$
- Ποιο δένδρο με n εξωτερικούς θα μας δώσει το ελάχιστο Εξωτερικό Μονοπάτι βάρους;

Ποιός είναι ο καλύτερος κώδικας για το σύνολο των χαρακτήρων $\{C_1, C_2, \ldots, C_n\}$ με βάρη $w_1, w_2, \ldots, w_n$







E = 48

E = 3*2+3*5+2*7+1*10=

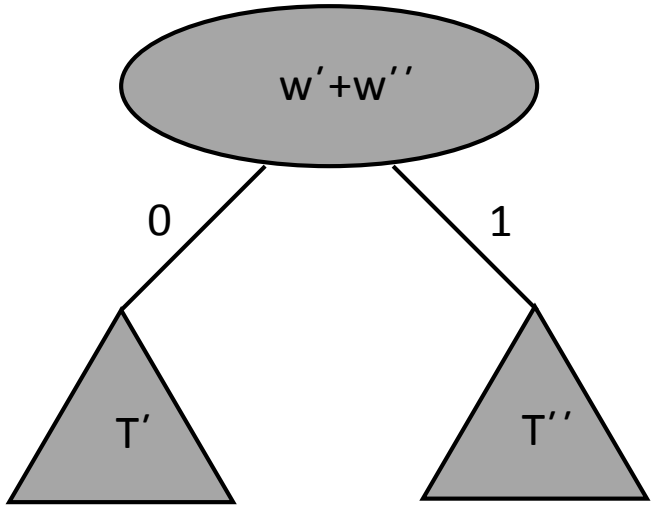
6+15+14+10 = 45

E = 63

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

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′′ και 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.

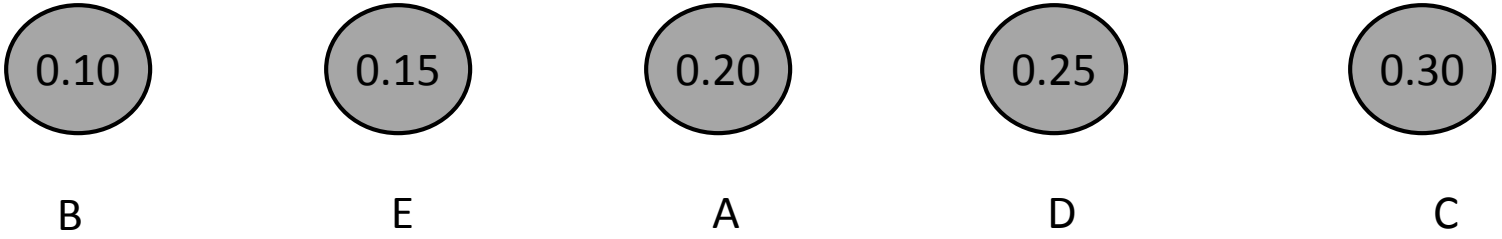


Στη συνέχεια ας υποθέσουμε ότι έχουμε τον παρακάτω πίνακα:

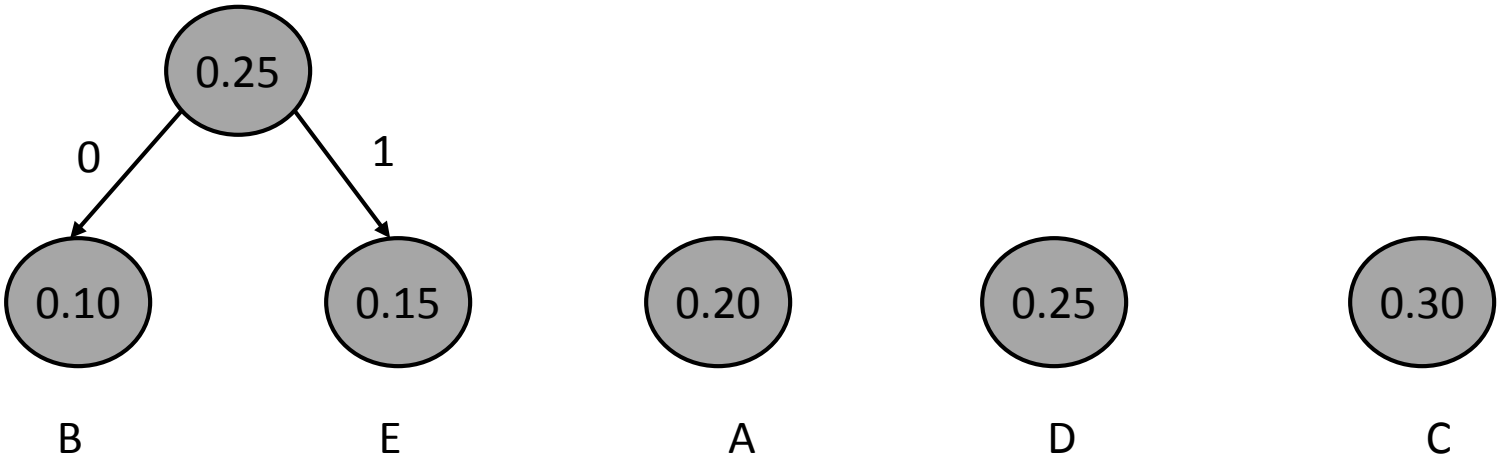
Χαρακτήρας	A	B	C	D	E
βάρος	0.2	0.1	0.3	0.25	0.15

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

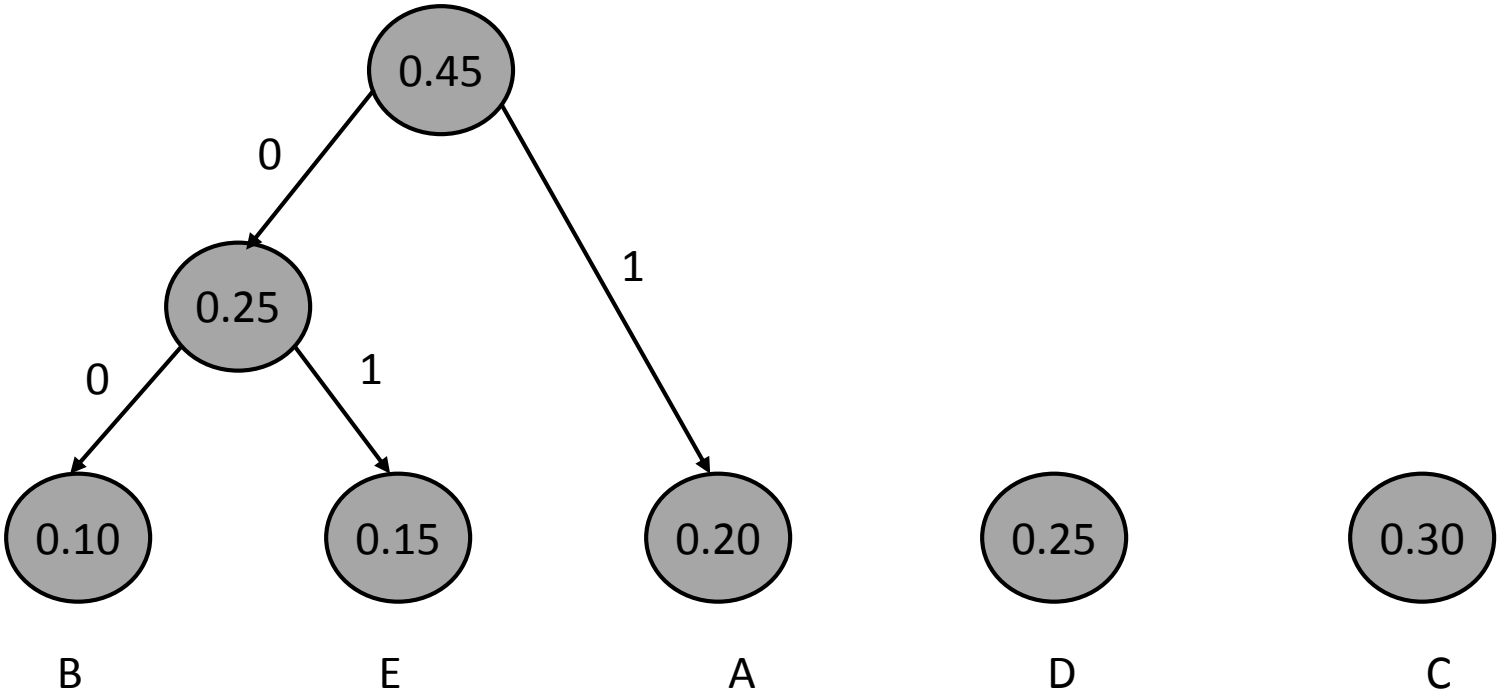
1)



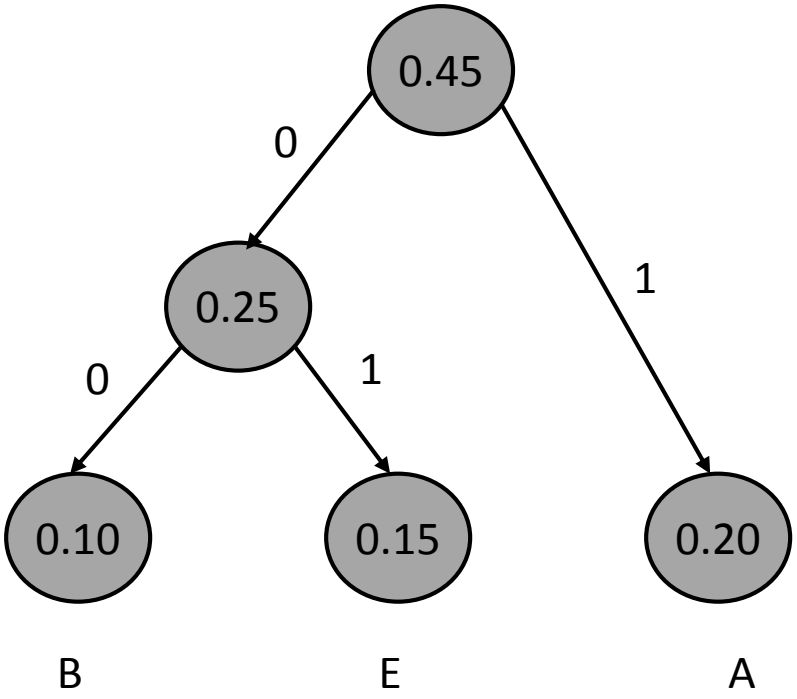
2)

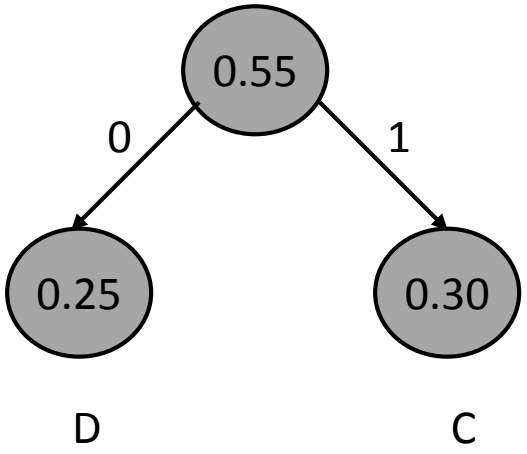


3)

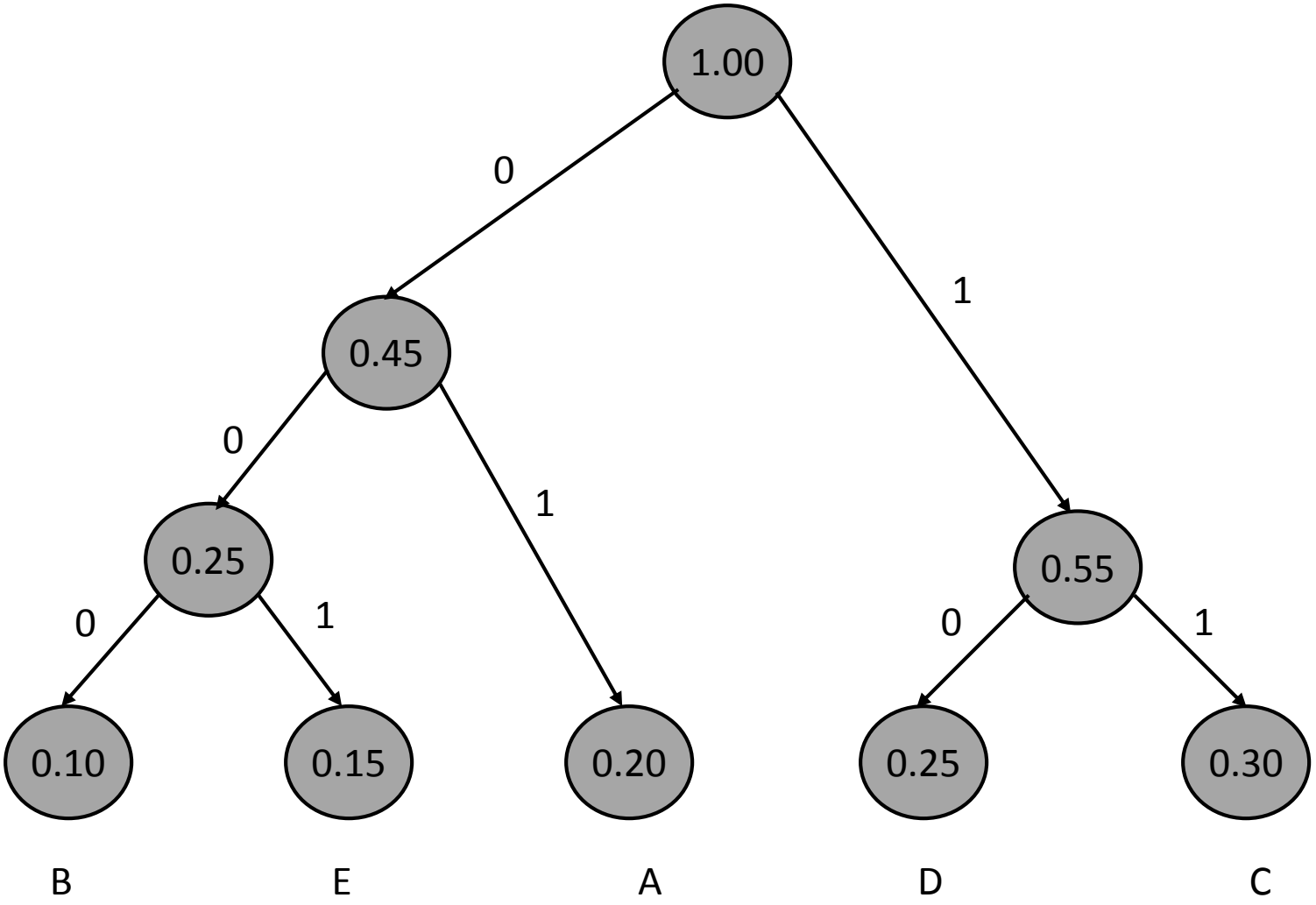


3)





5)



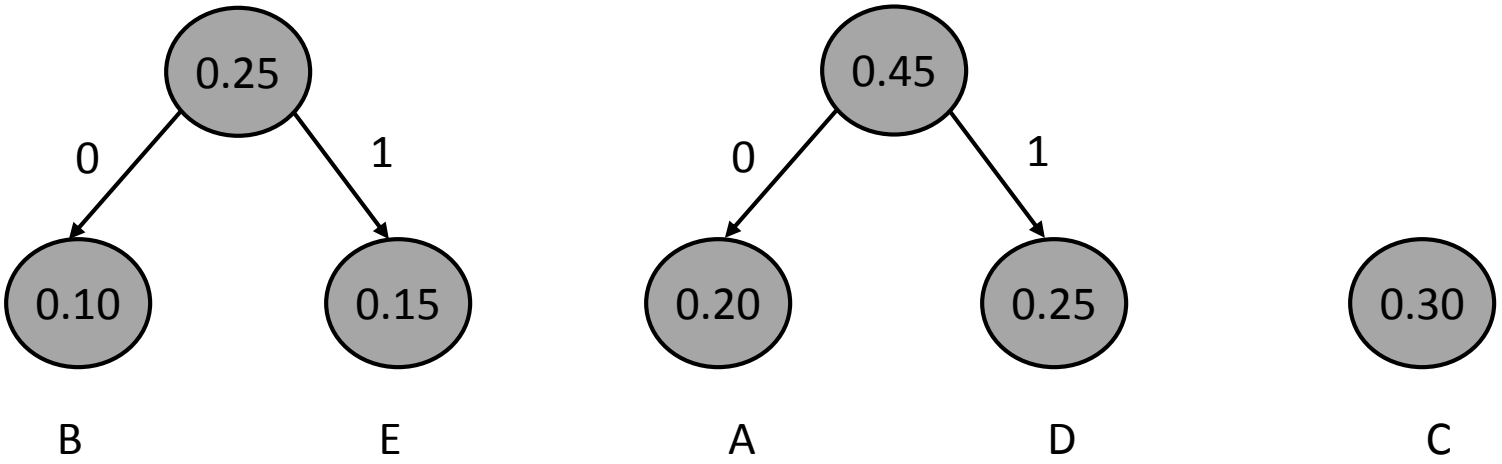
Οι κώδικες Huffman που λαμβάνονται από το παραπάνω δέντρο είναι:

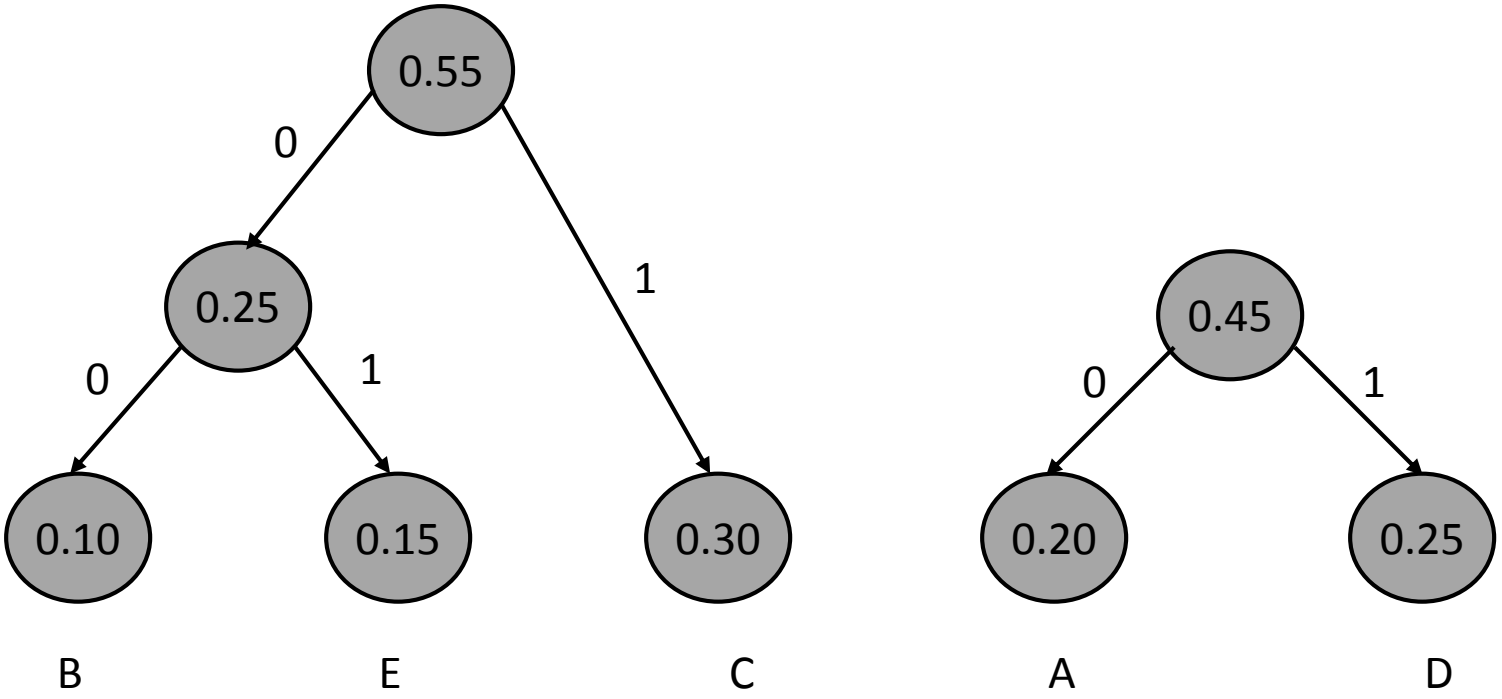
Χαρακτήρας	Κώδικας Huffmann
A	01
B	000
C	11
D	10
E	001

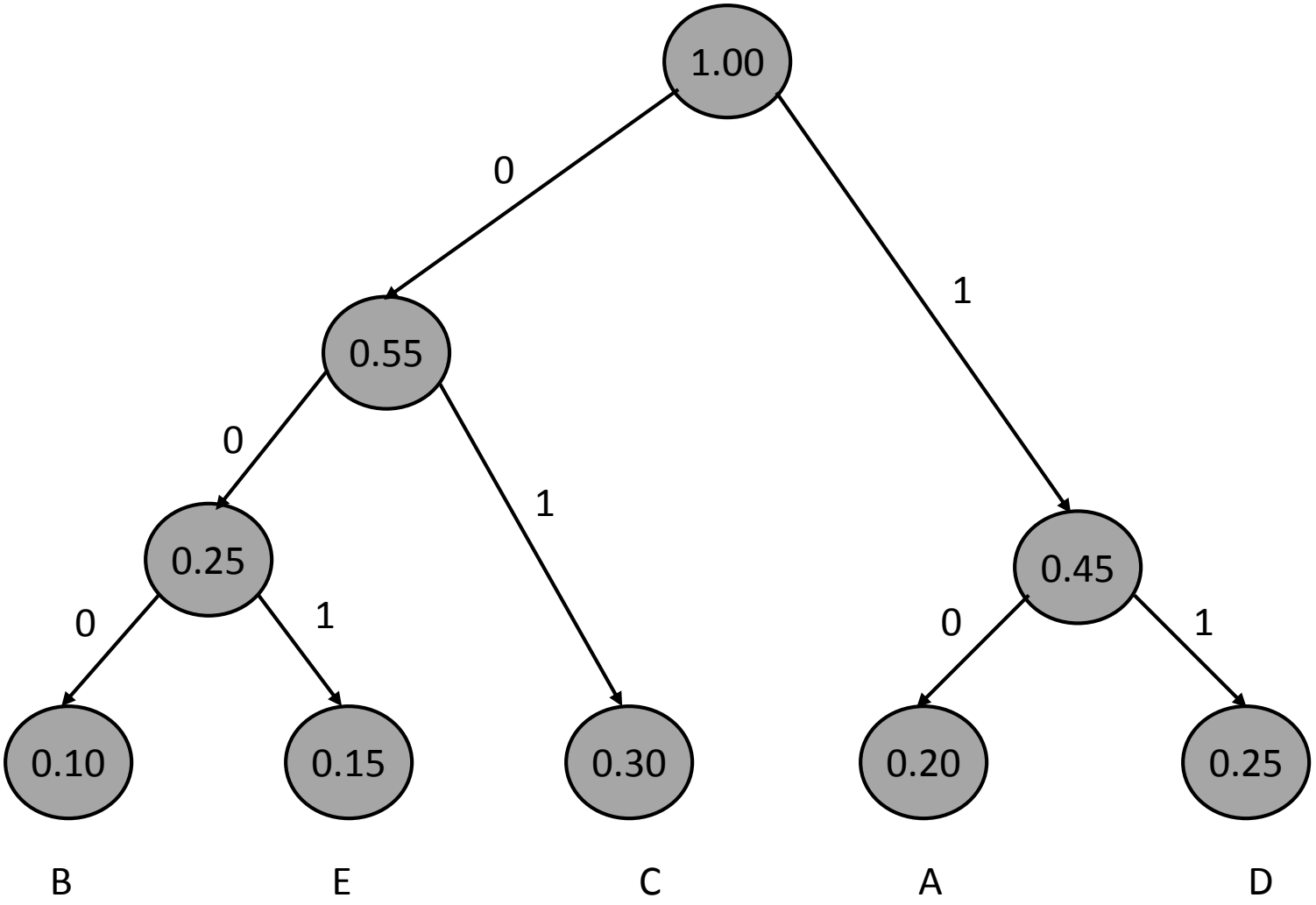
Επίσης η ποσότητα $E = \sum_{1 \le i \le n} w_i I_i$ είναι ίση με :

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
0.45
0
1
0
1
0.25
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 παρακάτω πίνακα :

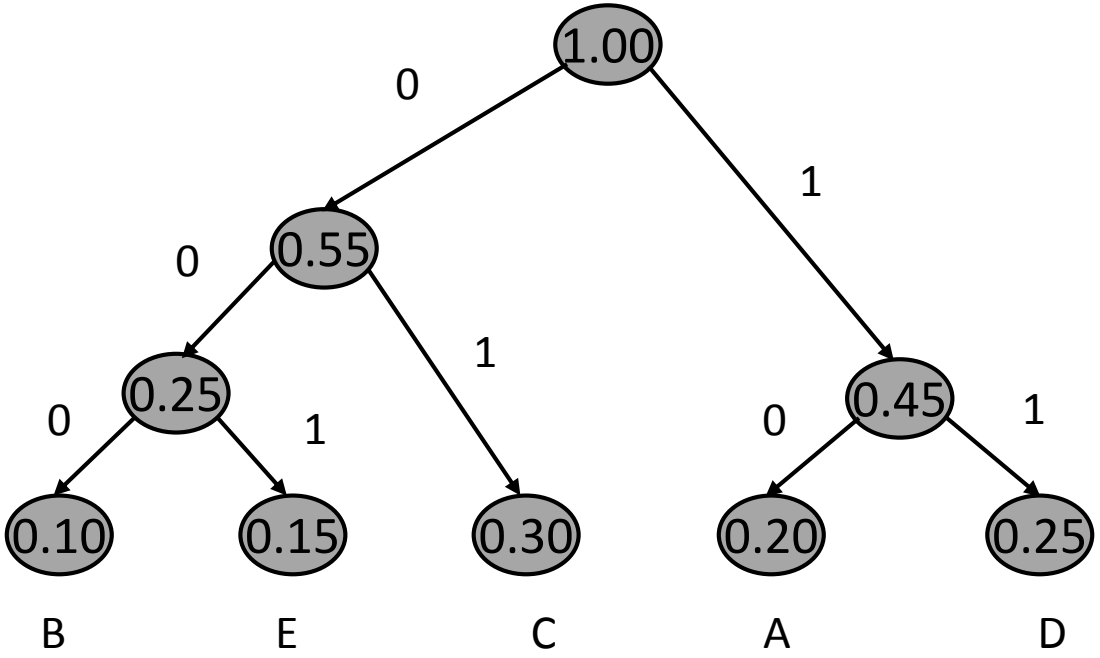
Χαρακτήρας	Κώδικας Huffmann
A	10
B	000
C	01
D	11
E	001

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. Έστω x είναι το επόμενο bit στη σειρά χαρακτήρων.
   β. Αν x == 0 τότε
   p = p .apaidi
   διαφορετικά
   p = p .dpaidi

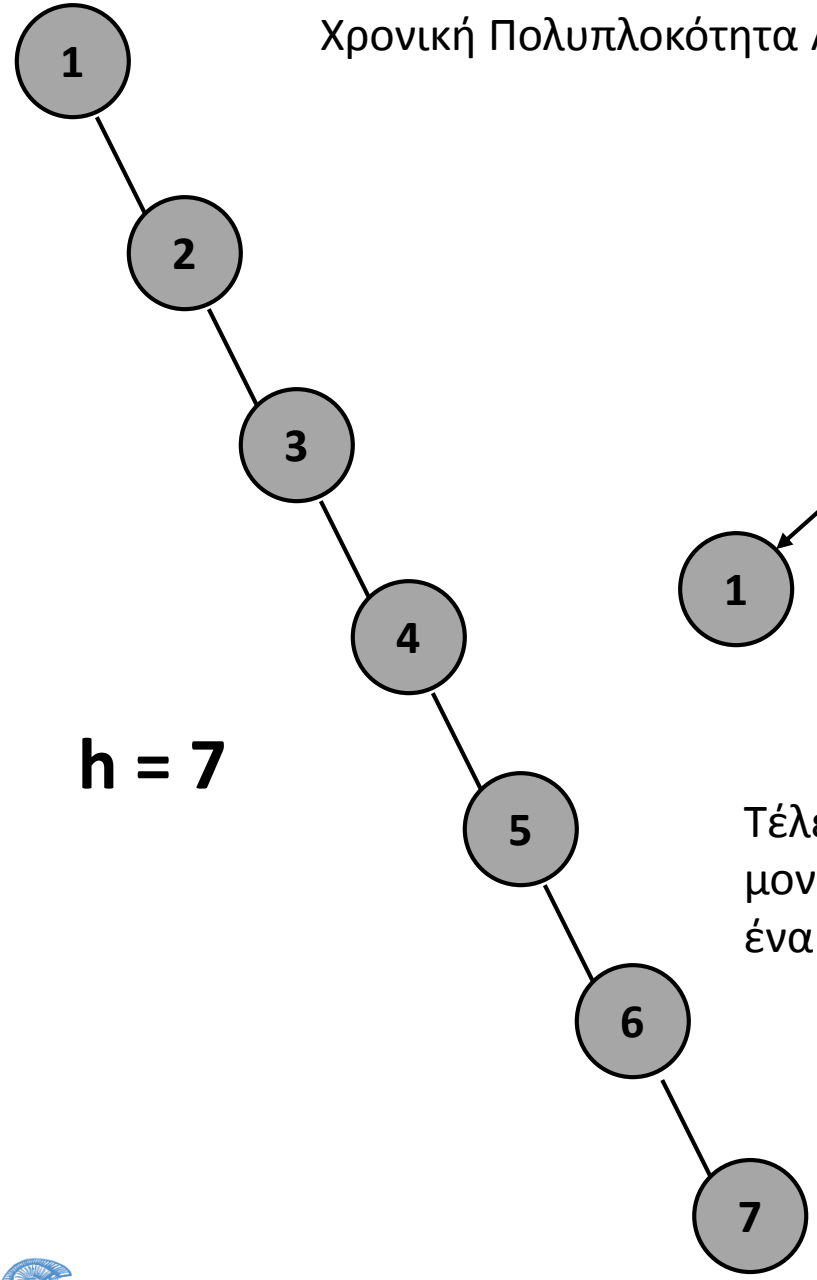
   γ. Αν ο p δείχνει σ' ένα φύλλο τότε:
   i. Να εκτυπωθεί ο χαρακτήρας αυτού του φύλλου
   ii. p = riza

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 αποκωδικοποίηση είναι: C C C A A

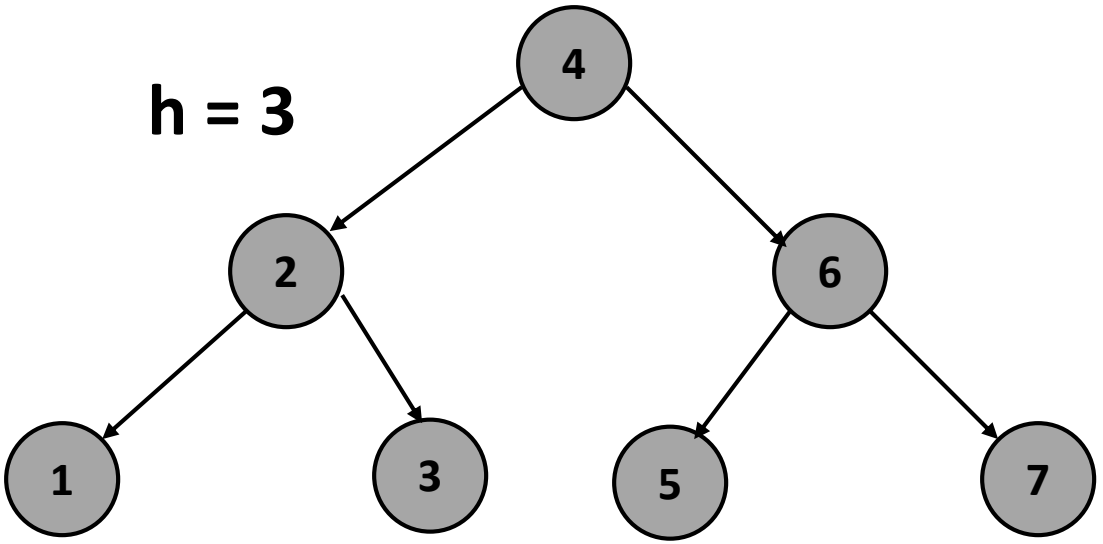


Ενώ με το πρώτο δένδρο θα ήταν: A A A D D

Χρονική Πολυπλοκότητα Αλγορίθμων για ΔΔΑ



h = 7



h = 3

Τέλεια ισοζυγισμένο δένδρο αν το μήκη των μονοπατιών προς τα φύλλα διαφέρουν το πολύ κατά ένα.

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;

n στοιχεία έχουν n! διατάξεις με ίση πιθανότητα εμφάνισης.

S συγκρίσεις για επιτυχημένη αναζήτηση, U για αποτυχημένη

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

Για την εισαγωγή

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

(1),(2)

$$(n+1)U(n) = 2n + U(0) + U(1) + \ldots + U(n-1) \quad \textbf{(3)}$$

n = n+1

$$n\,U(n-1) = 2(n-1) + U(0) + U(1) + \ldots + U(n-2) \quad \textbf{(4)}$$

(3) - (4)

$$U(n) = U(n-1) + \frac{2}{n+1} \quad \textbf{(5)}$$

$$U(n) = 0$$

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

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

- • •
- • •
- • •

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

ή $$U(n) = 2\,H_{n+1} - 2$$ **(6)**

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

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

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

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 $U(n) \cong 1.39 \log_2 n$

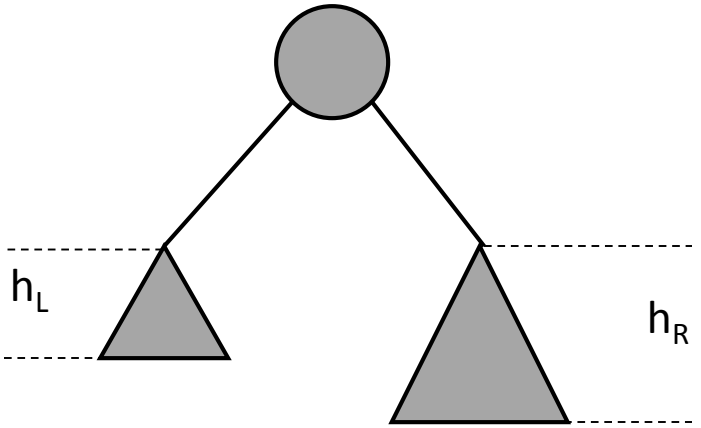
$$U(n) \cong 1.39 \log_2 n$$

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

AVL

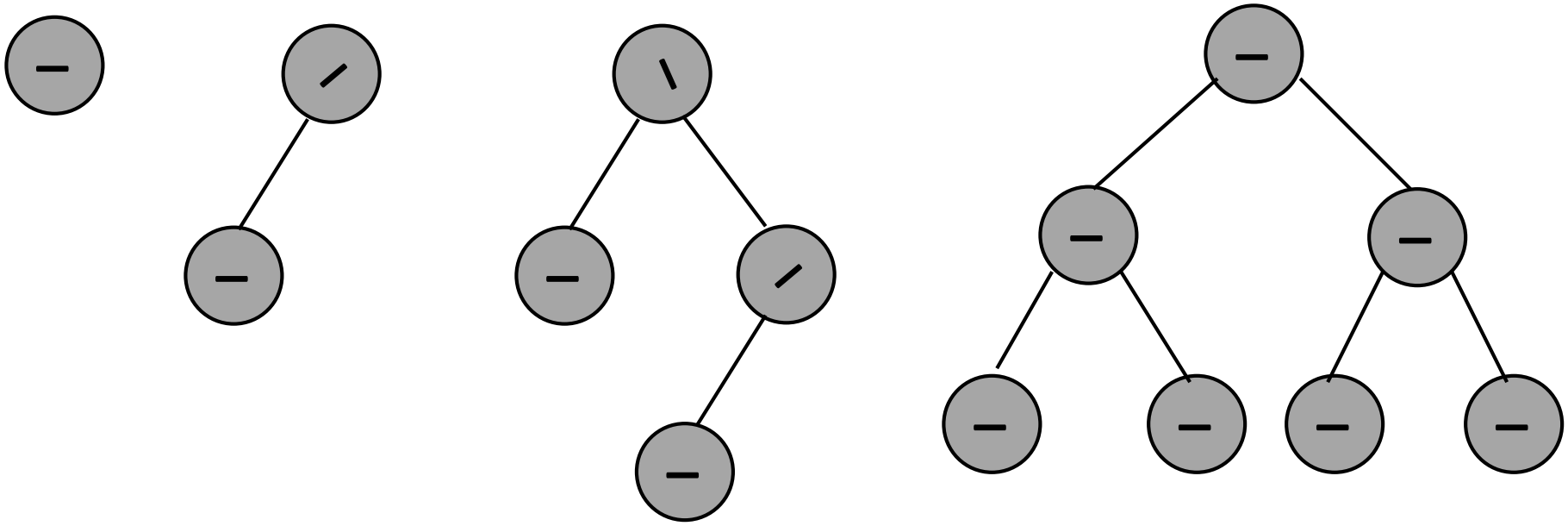
AVL δέντρα

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

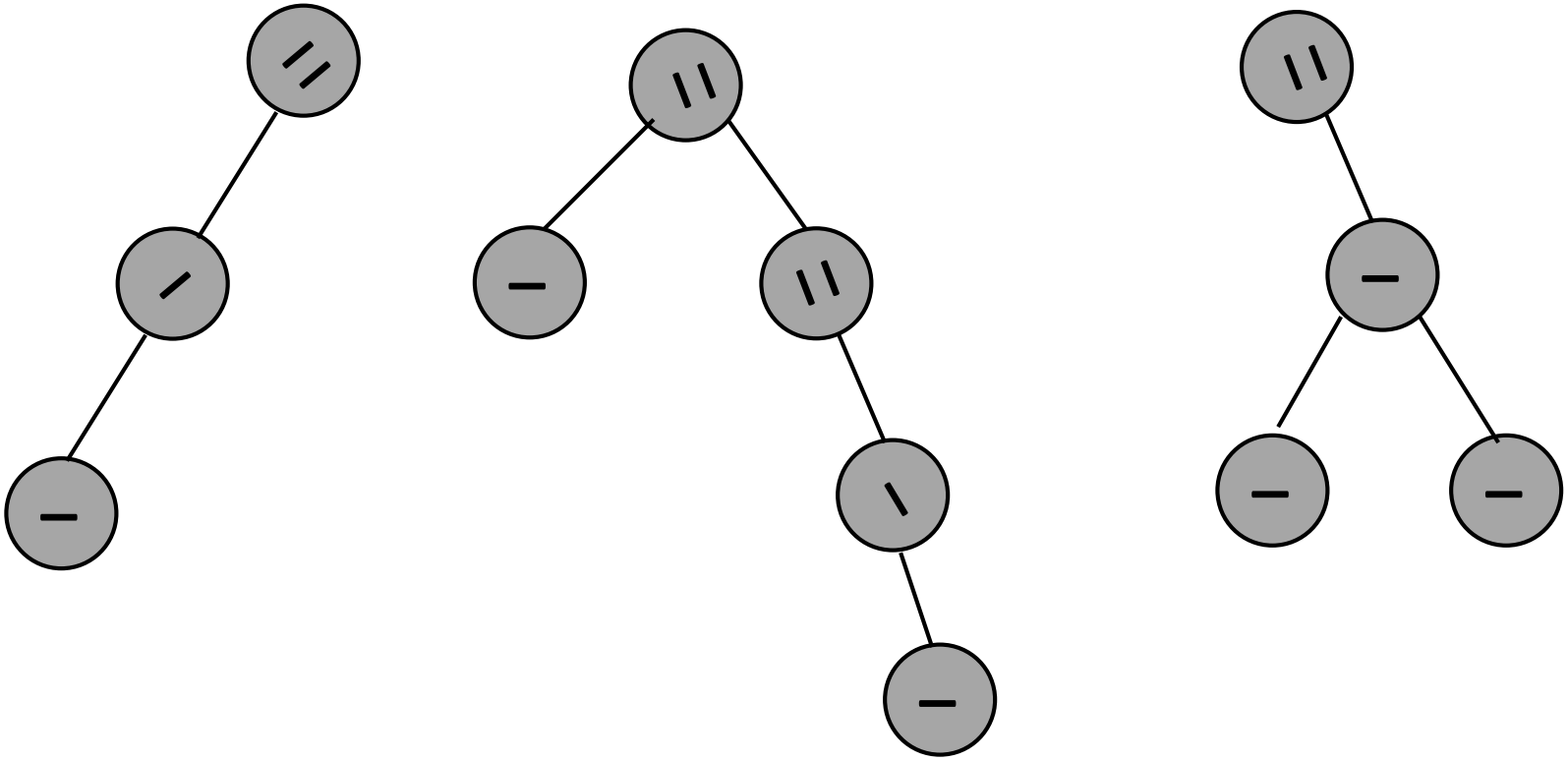


G.M. Adelson_Velkii και E.M. Landis 1962

AVL Δέντρα

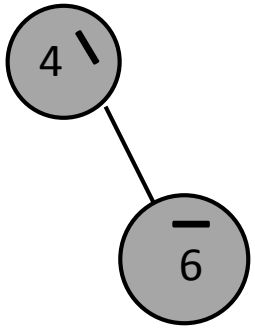


Μη AVL Δέντρα



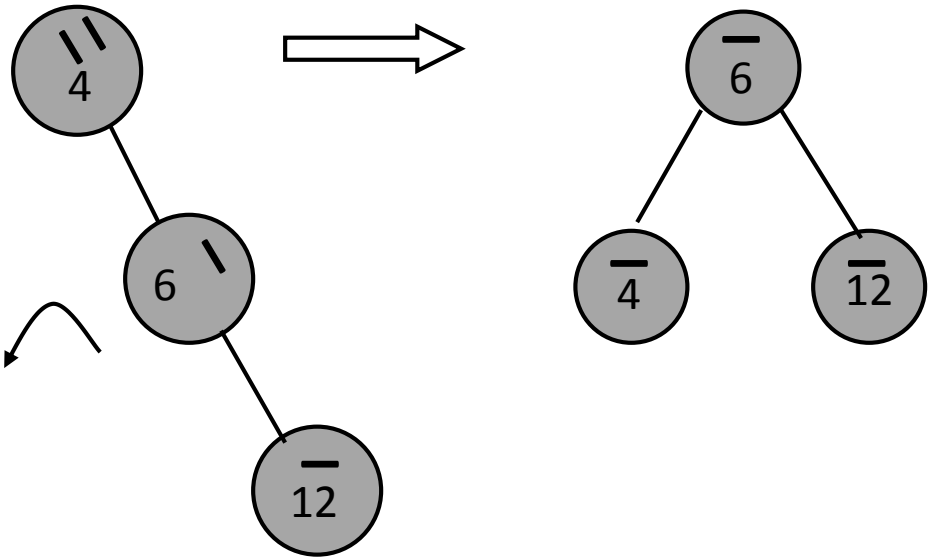
Εισαγωγή κόμβου

4, 6 :

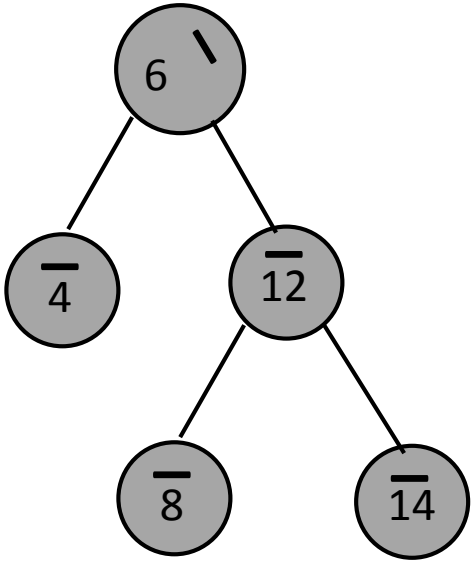


12 :

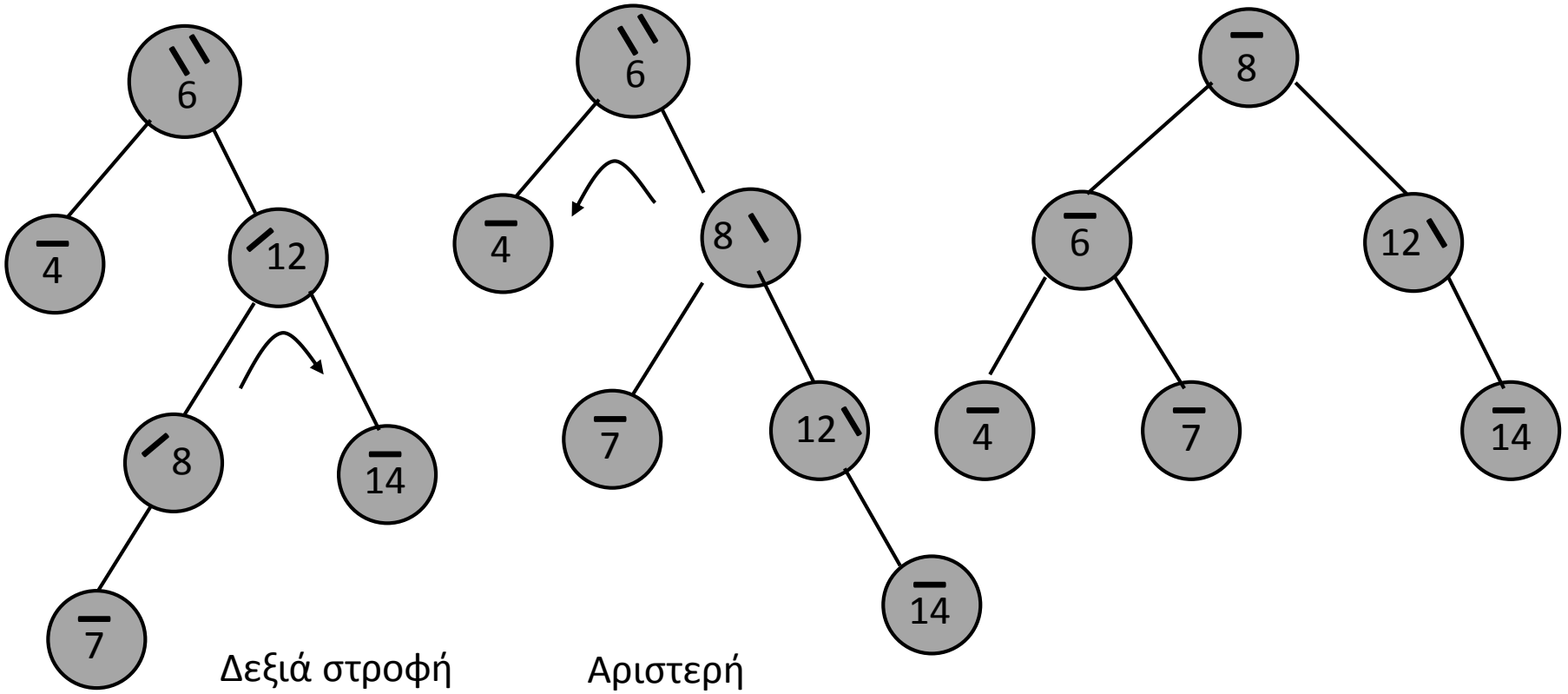
Αριστερή στροφή



8, 14 :



7 :



4 Περιπτώσεις

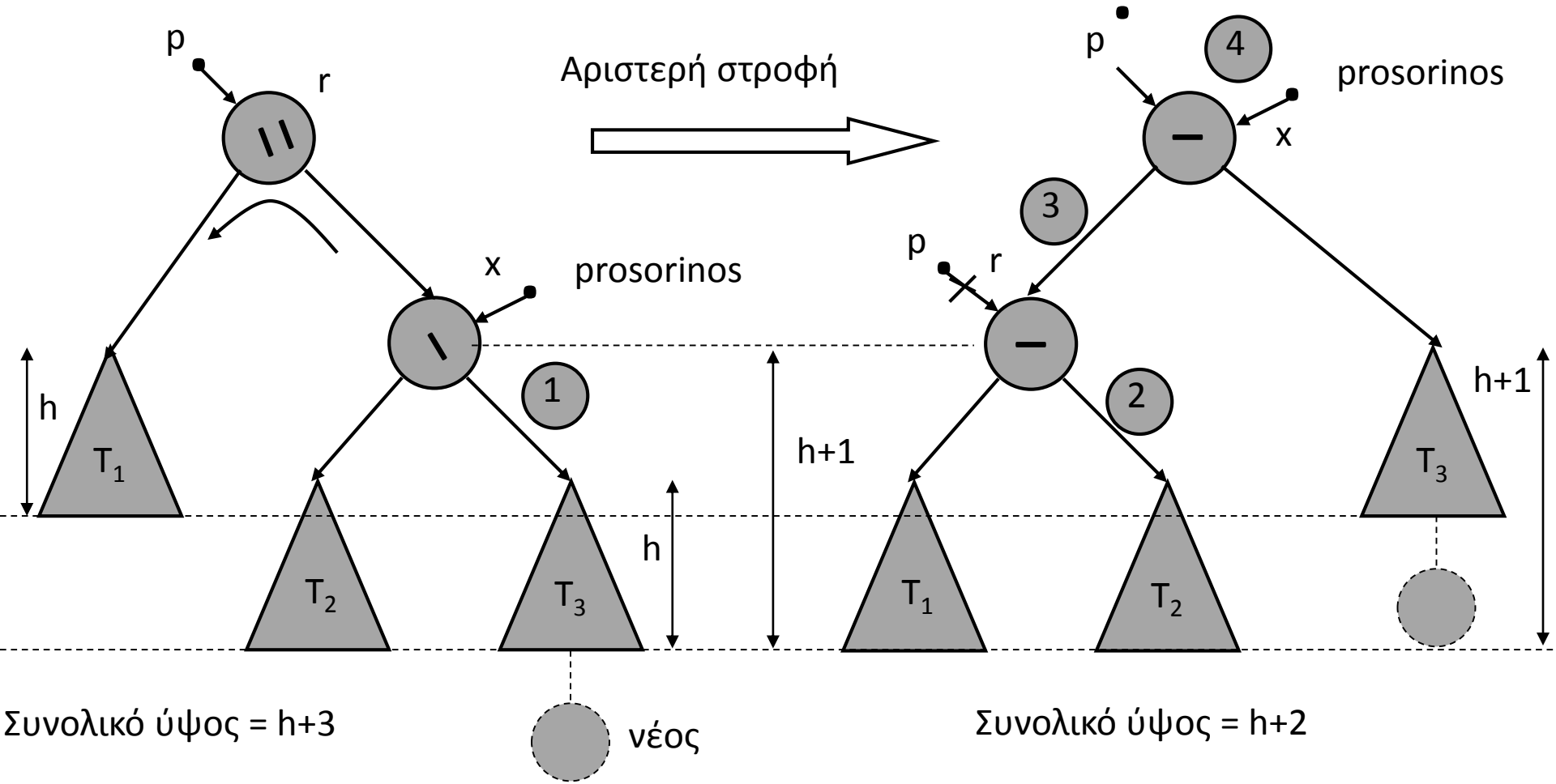
LL : ο νέος κόμβος Υ εισάγεται στο αριστερό υποδέντρο του αριστερού υποδέντρου του Α.

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.

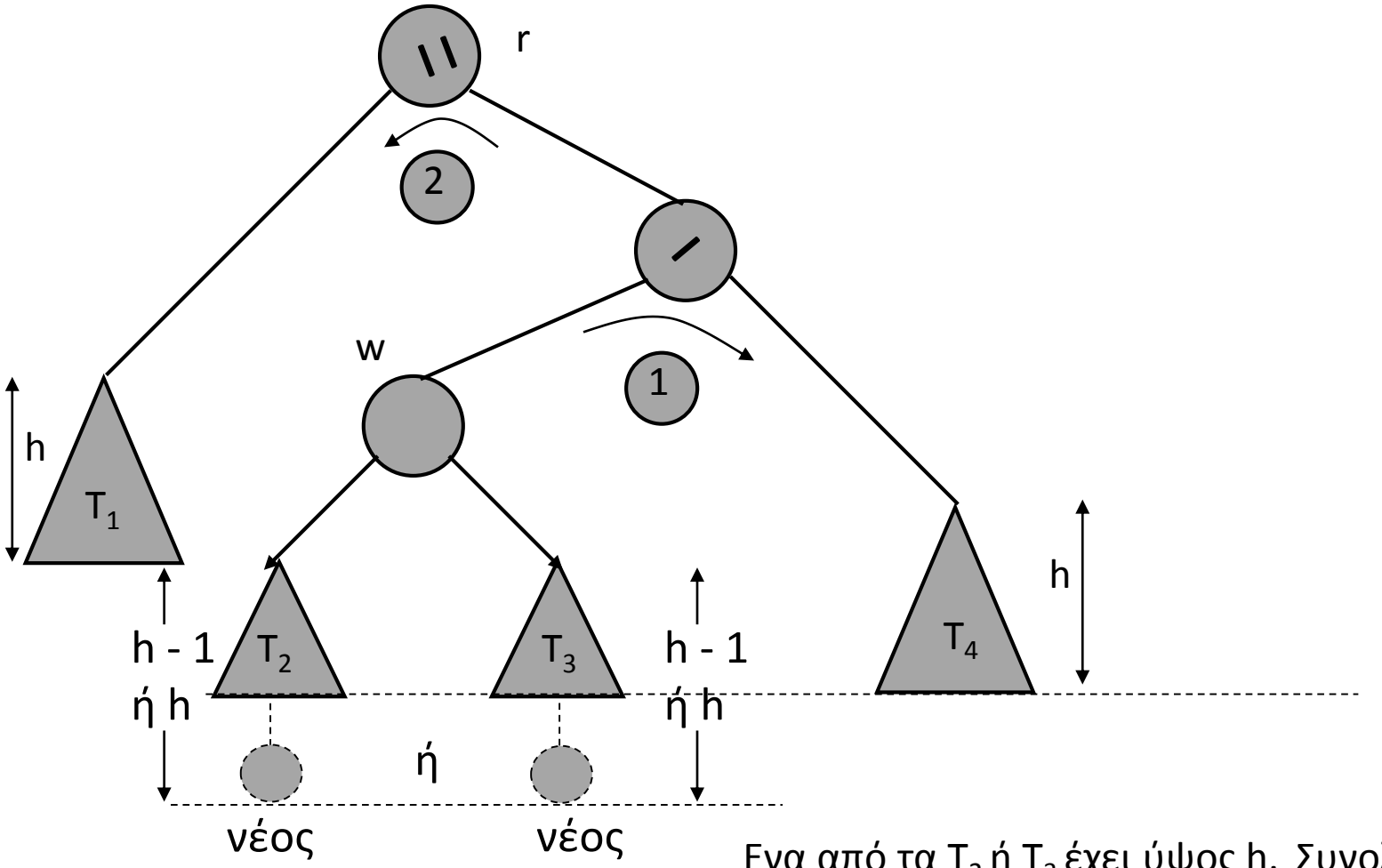
Περίπτωση 1 (RR) : Δεξιά Υψηλό



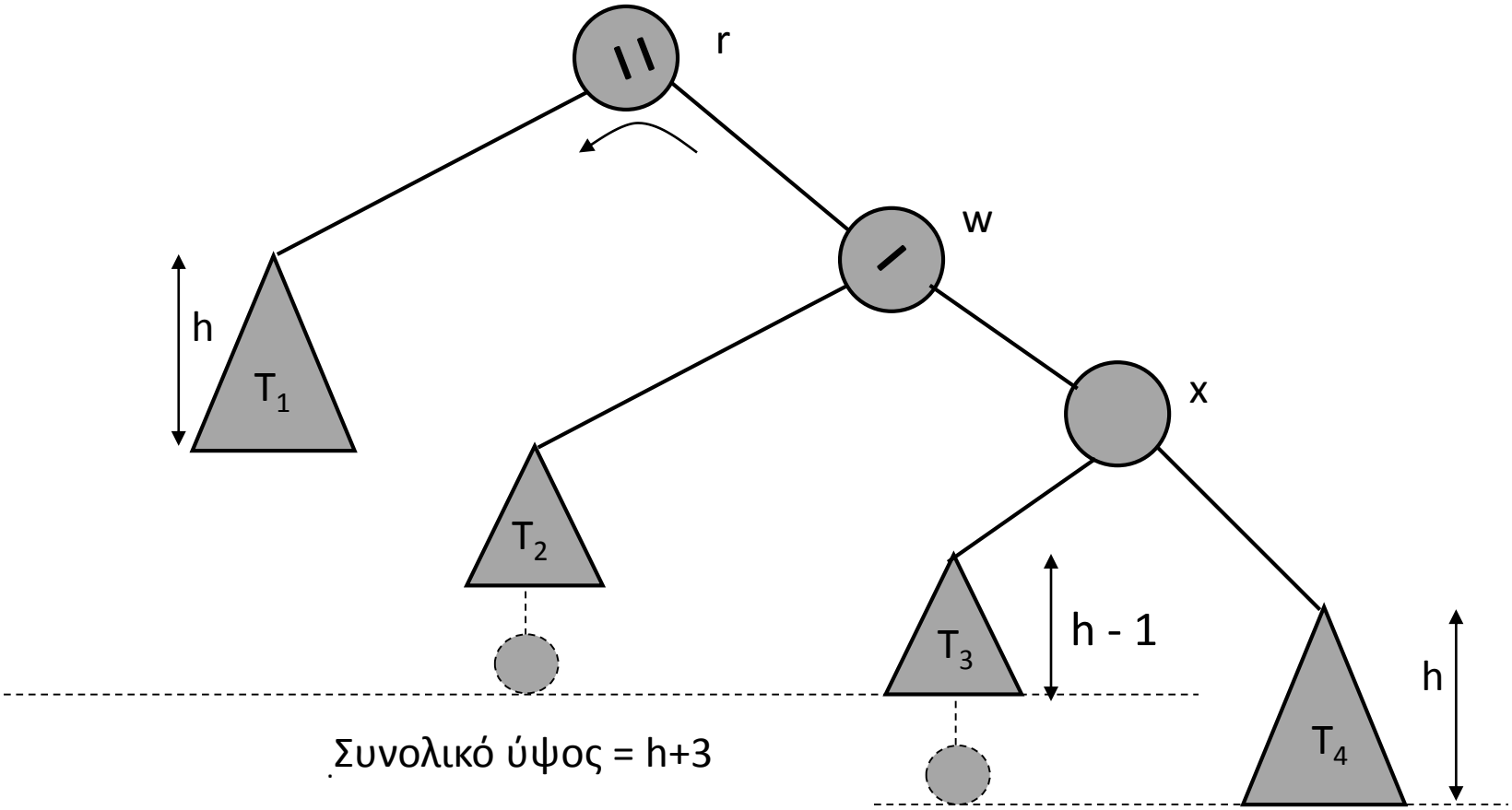
```
typedef  enum{TRUE, FALSE};
typedef  enum{AY, IY, DY}  paragon_isozygisis;
typedef struct typos_komvou  *typos_deikti;
typedef struct typos_komvou{
      typos_stoixeiou  dedomena;
      typos_deikti  apaidi, dpaidi;
      paragon_isozygisis  pi;
}  typos_komvou;
```

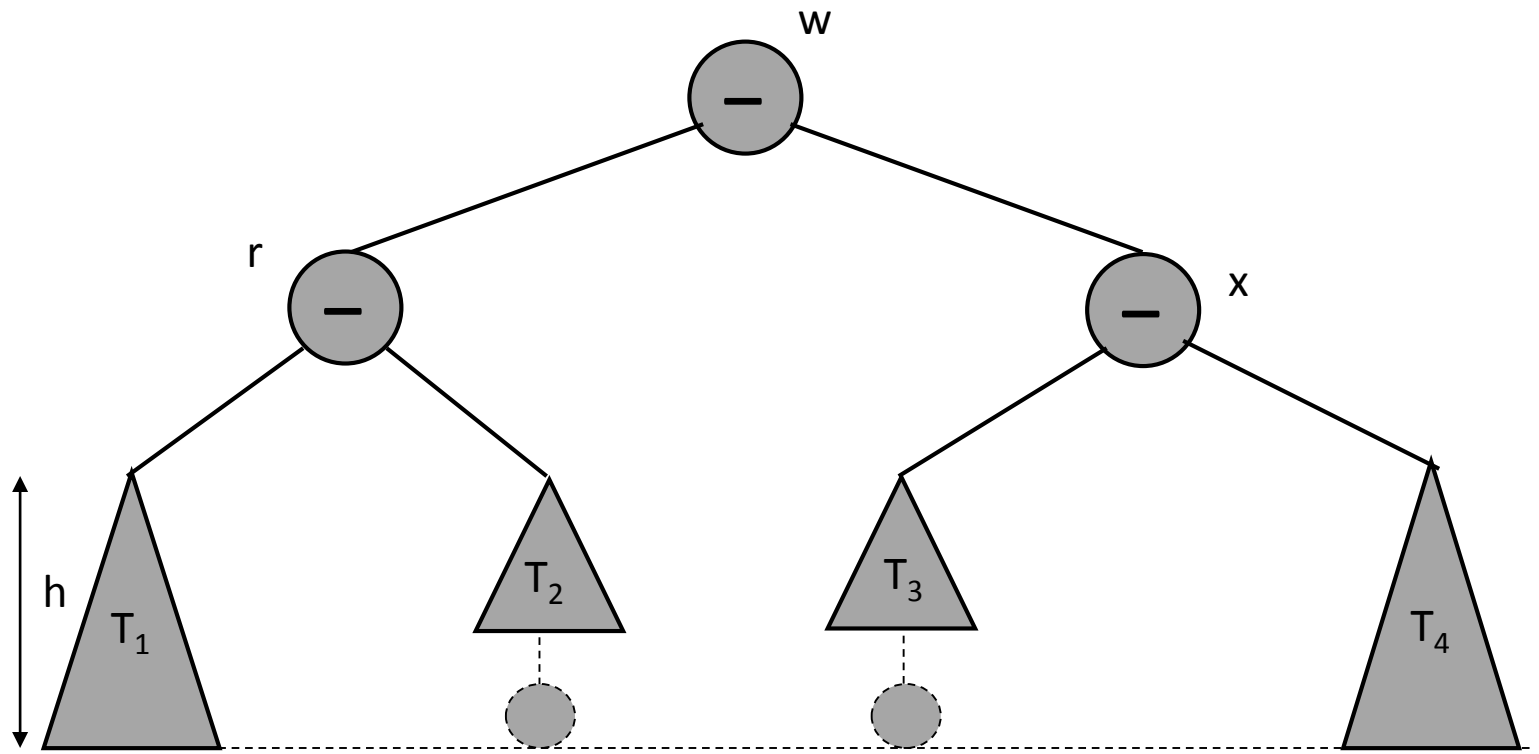
```
void aristeri_peristrofi (typos_deikti *p);{
/*Ο p δείχνει τη ρίζα του υποδέντρου που περιστρέφεται*/
      typos_deikti prosorinos;

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

Περίπτωση 2 (RL):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



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

παλιό w

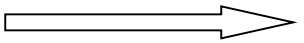
νέο r

νέο x

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

```
void dexi_varos(typos_deikti  *riza,boolean *ypsilotero){
  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;
    case IY: /*Δεν συμβαίνει η περίπτωση αυτή*/
      printf ( "Λάθος" );
      break;
```

συνέχεια



```
    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);
      /*ο x δείχνει τώρα τον w*/
      (*riza)->dpaidi = x;
      aristeri_peristrofi(riza);
      ypsilotero = FALSE;
    } // of switch
}/*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,
                   boolean *ypsilotero)
{
```

συνέχεια

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

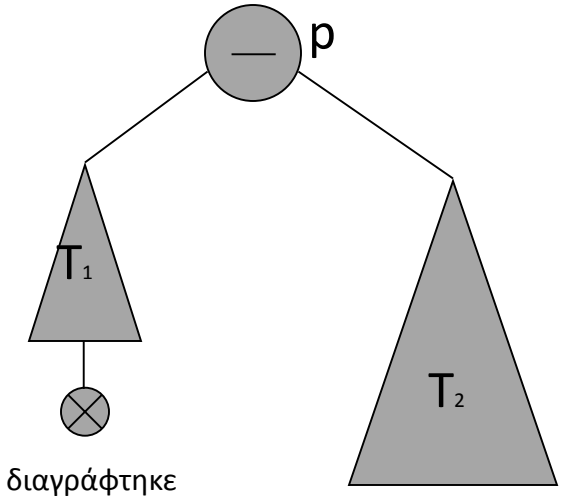
συνέχεια

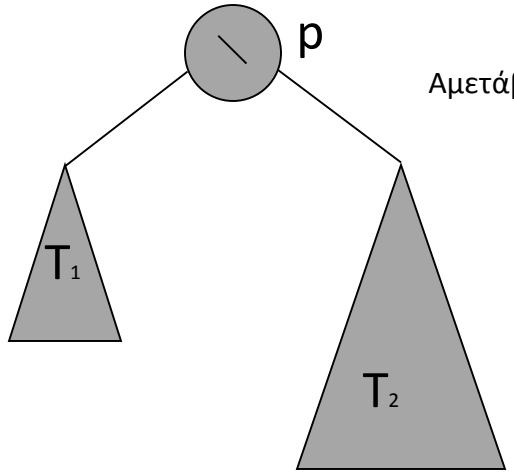
```
    if (*ypsilotero)
          switch ((*riza)->pi){
          case AY: aristero_varos(riza, ypsilotero);
                   break;
          case IY: (*riza)->pi=AY; *ypsilotero=TRUE;
                   break;
          case DY: (*riza)->pi=IY;  *ypsilotero=FALSE;
                           break;
          }
}
```

συνέχεια

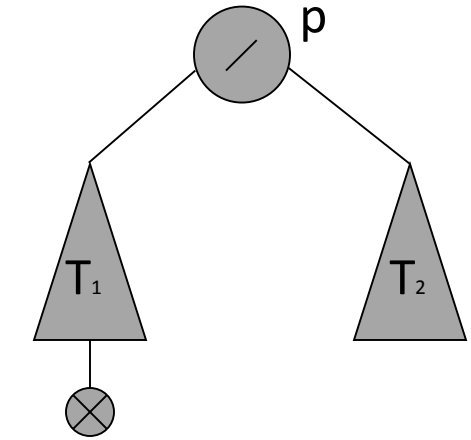
```
else{/*εισαγωγή στο δεξί υποδέντρο*/
      avl_eisagogi (&((*riza)->dpaidi),
                    prosorinos, stoixeio, ypsilotero);
      if (*ypsilotero)
            switch ((*riza)->pi){
            case AY: (*riza)->pi=IY; *ypsilotero=FALSE;
                        break;
            case IY:  (*riza)->pi=DY;  *ypsilotero=TRUE;
                        break;
            case DY: dexi_varos(riza, ypsilotero);
                        break;
            }
      }
   }
}/*avl_eisagogi*/
```

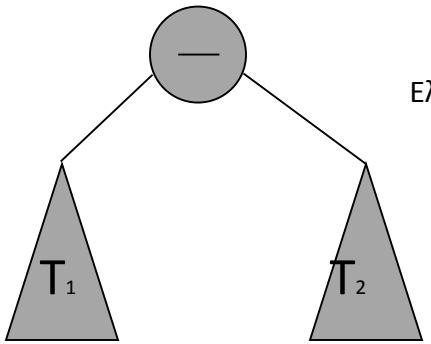
Διαγραφή



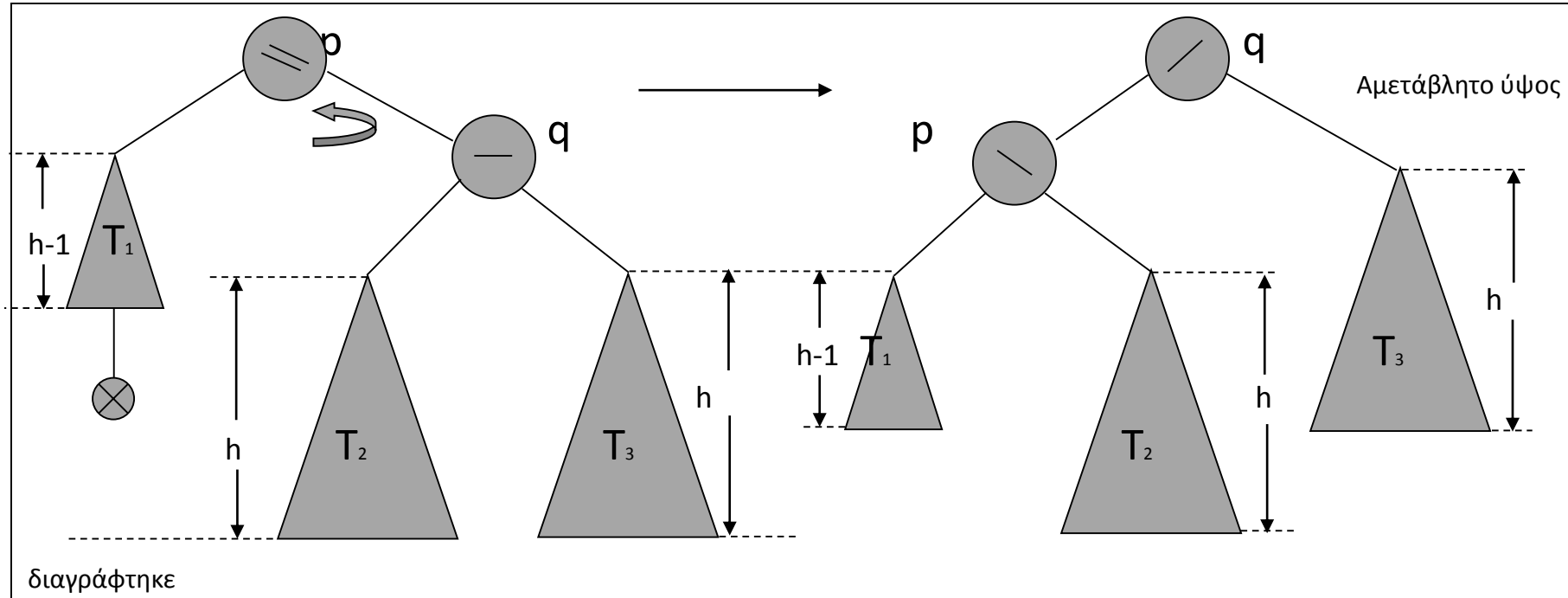


Περίπτωση 1

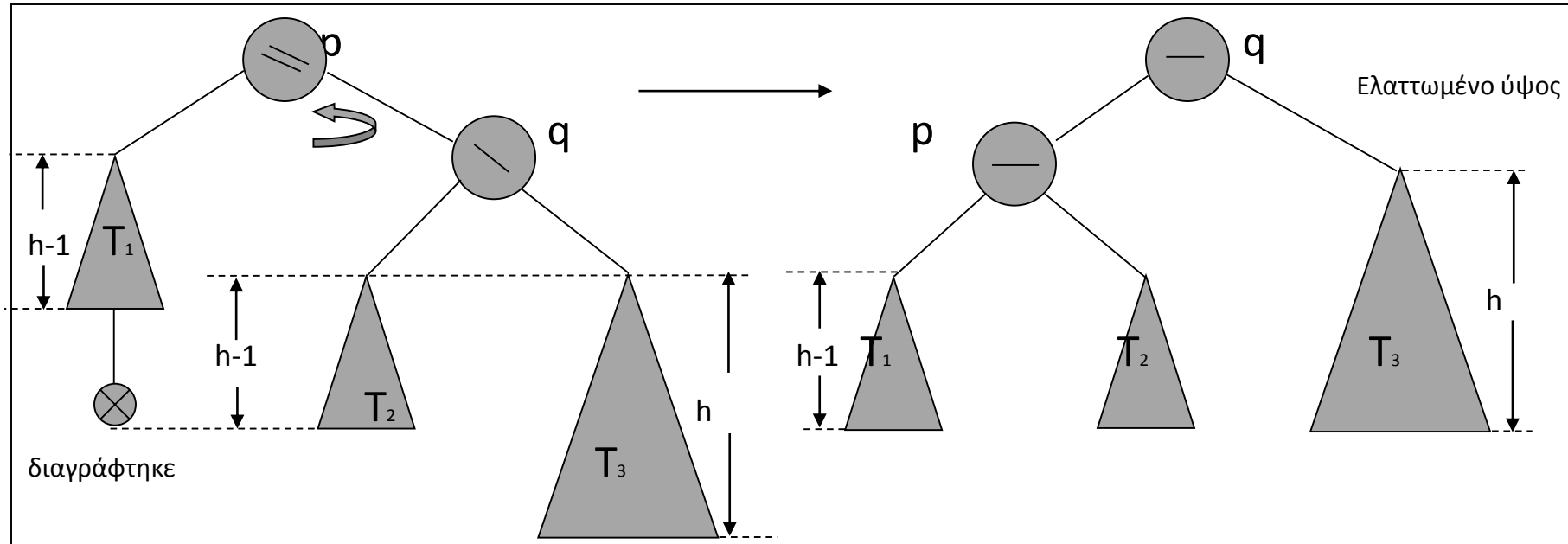




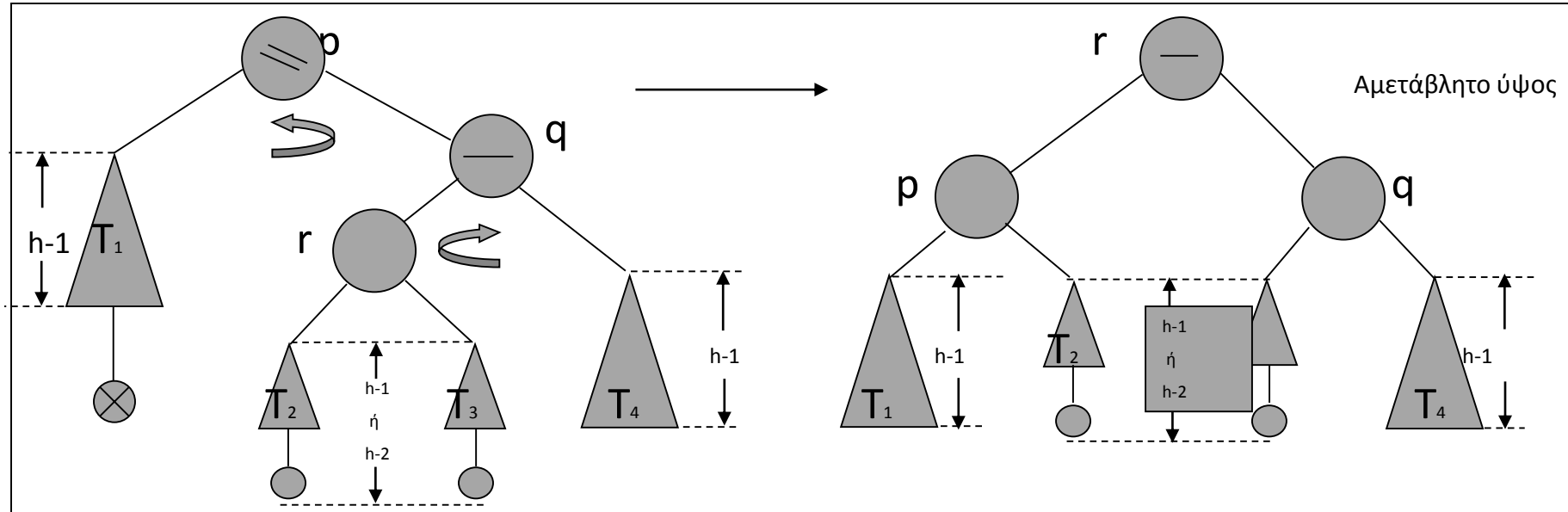
Περίπτωση 2



Περίπτωση 3α



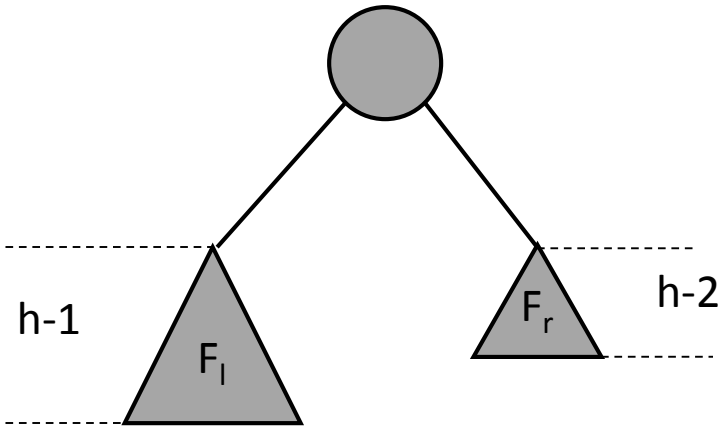
Περίπτωση 3b



Περίπτωση 3c

Σχήμα 8.22 Διάφορες περιπτώσεις διαγραφής κόμβου από ένα AVL δέντρο.

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.

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- 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.

Σημειώματα

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.