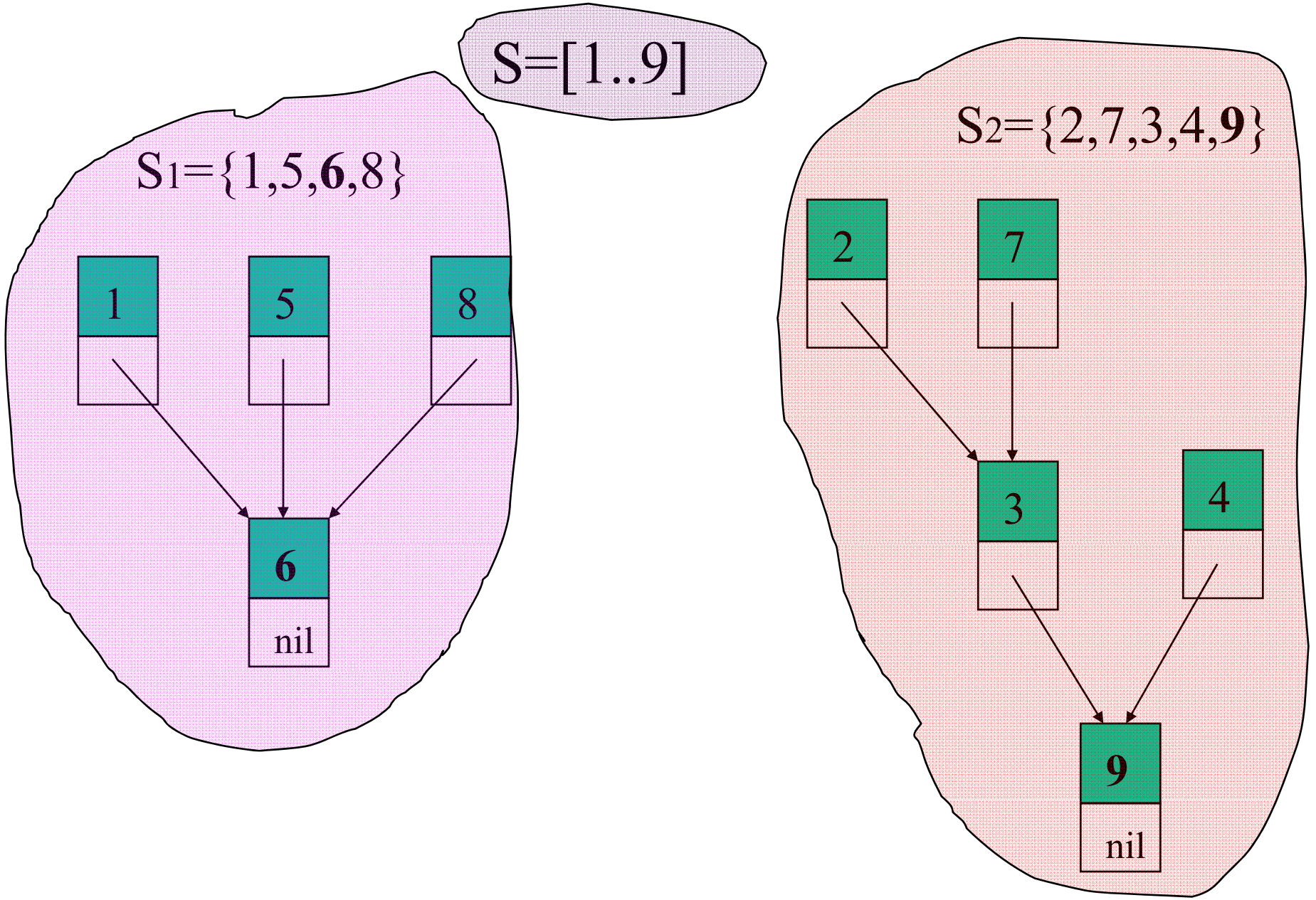
Βάση για γενικευμένες δομές δεδομένων

(όταν απαιτείται γρήγορη ανάκτηση πληροφορίας)

Αναπαράσταση: *αντι-κατευθυνόμενα δένδρα*

Μοναδική σύνδεση κάθε στοιχείου με το επόμενό του.


S=[1..9]
S1={1,5,6,8}
1
5
8
6
nil
S2={2,7,3,4,9}
2
7
3
4
9
nil

## `Find(x)`

Εντοπισμός του **x** και μετακίνηση προς τα κάτω μέχρι τη ρίζα για την εύρεση του ονόματος του συνόλου.

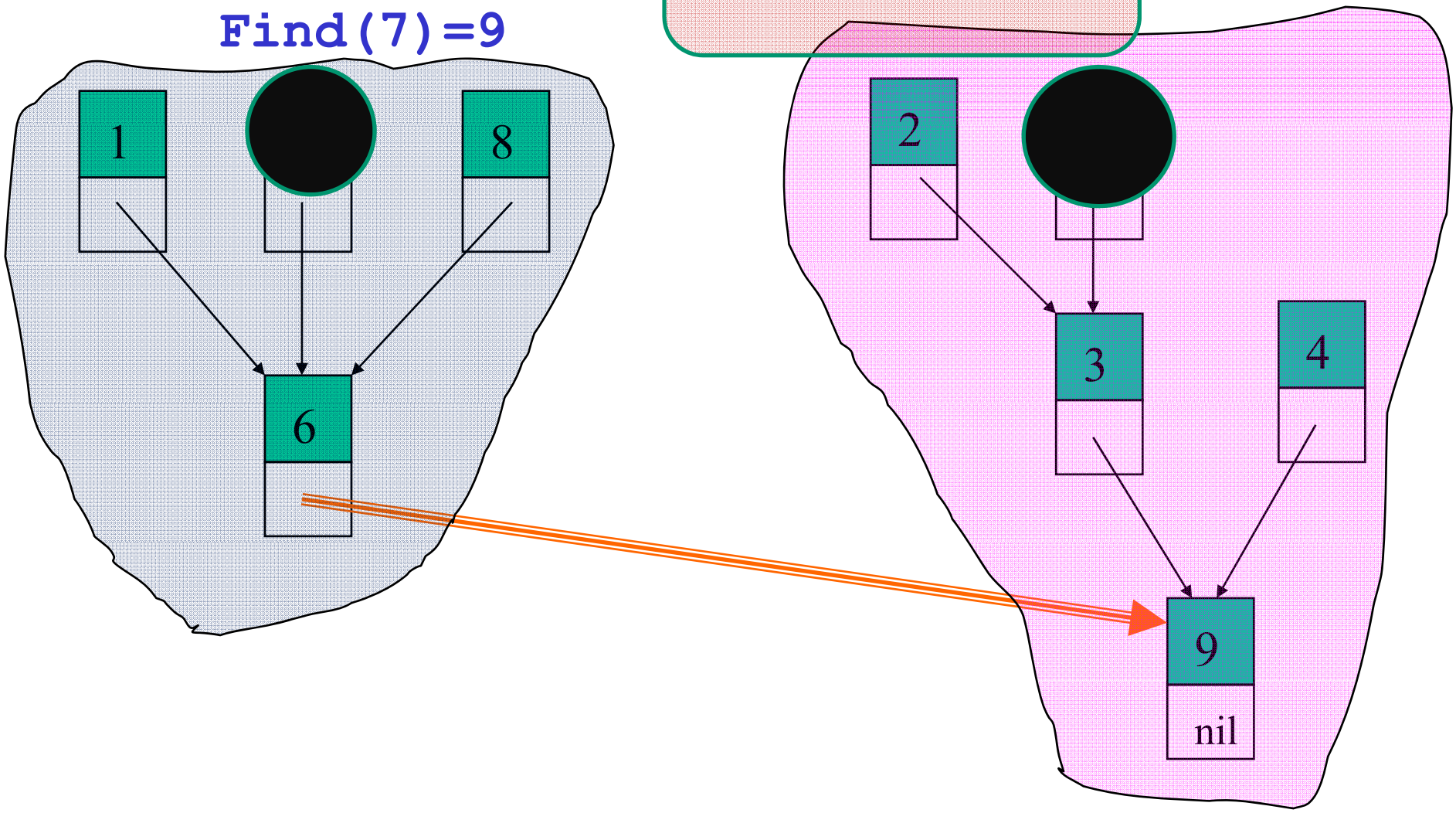
## `Union(u, v)`

Ενώνει δύο σύνολα με ρίζες **u** και **v**. Μία από τις δύο ρίζες ενώνεται με την άλλη.

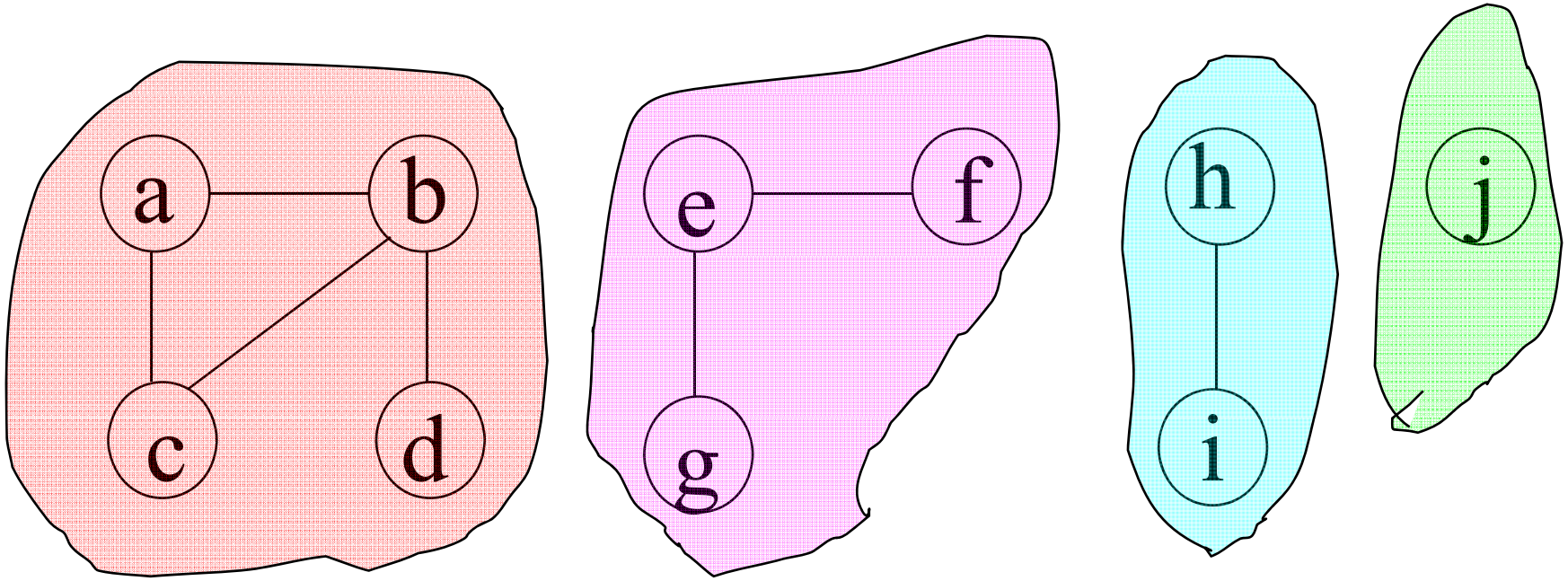
Find(5)=6

⇒ Union(6,9)=9

Find(7)=9

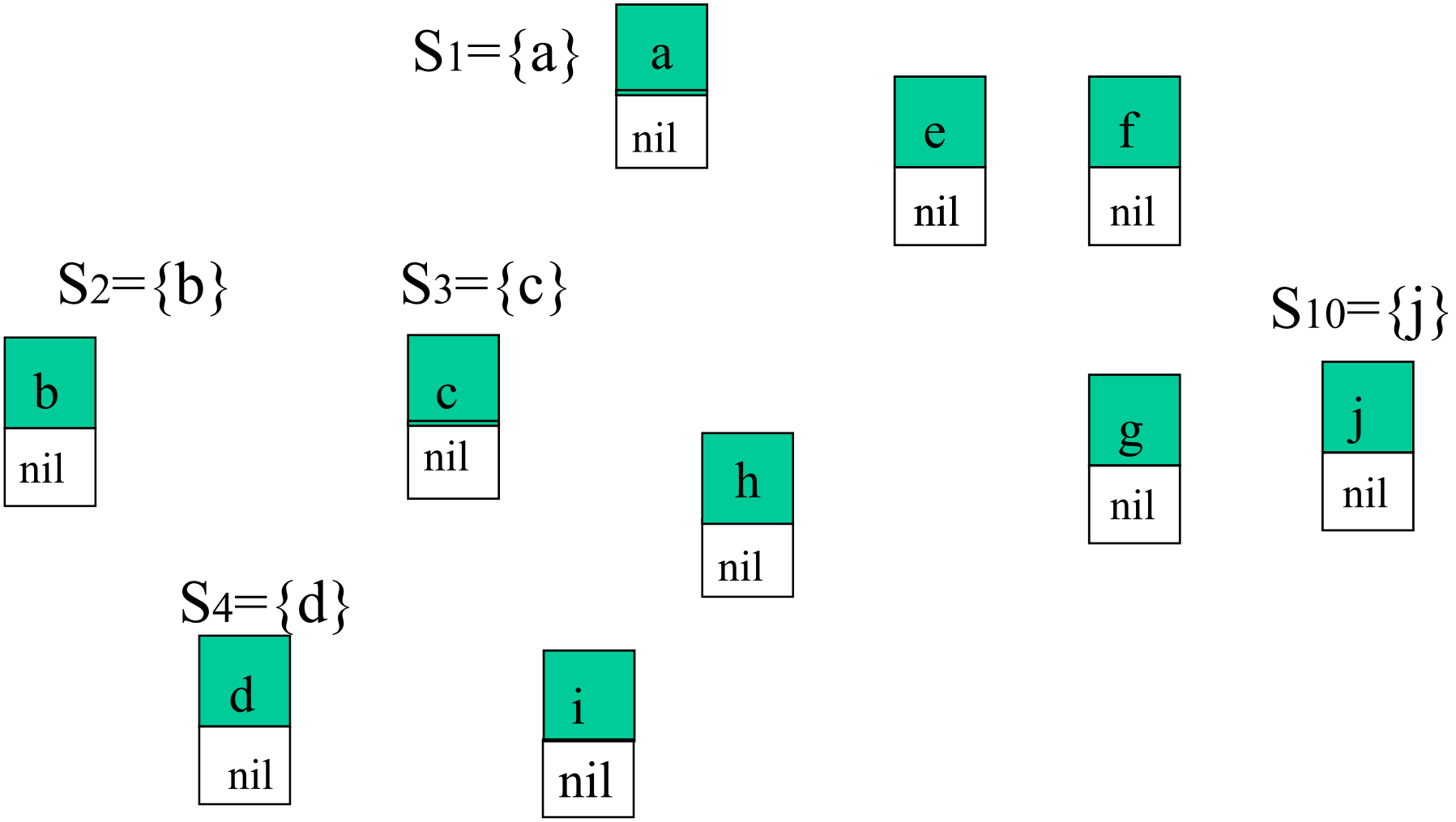


# Union and Find: παράδειγμα



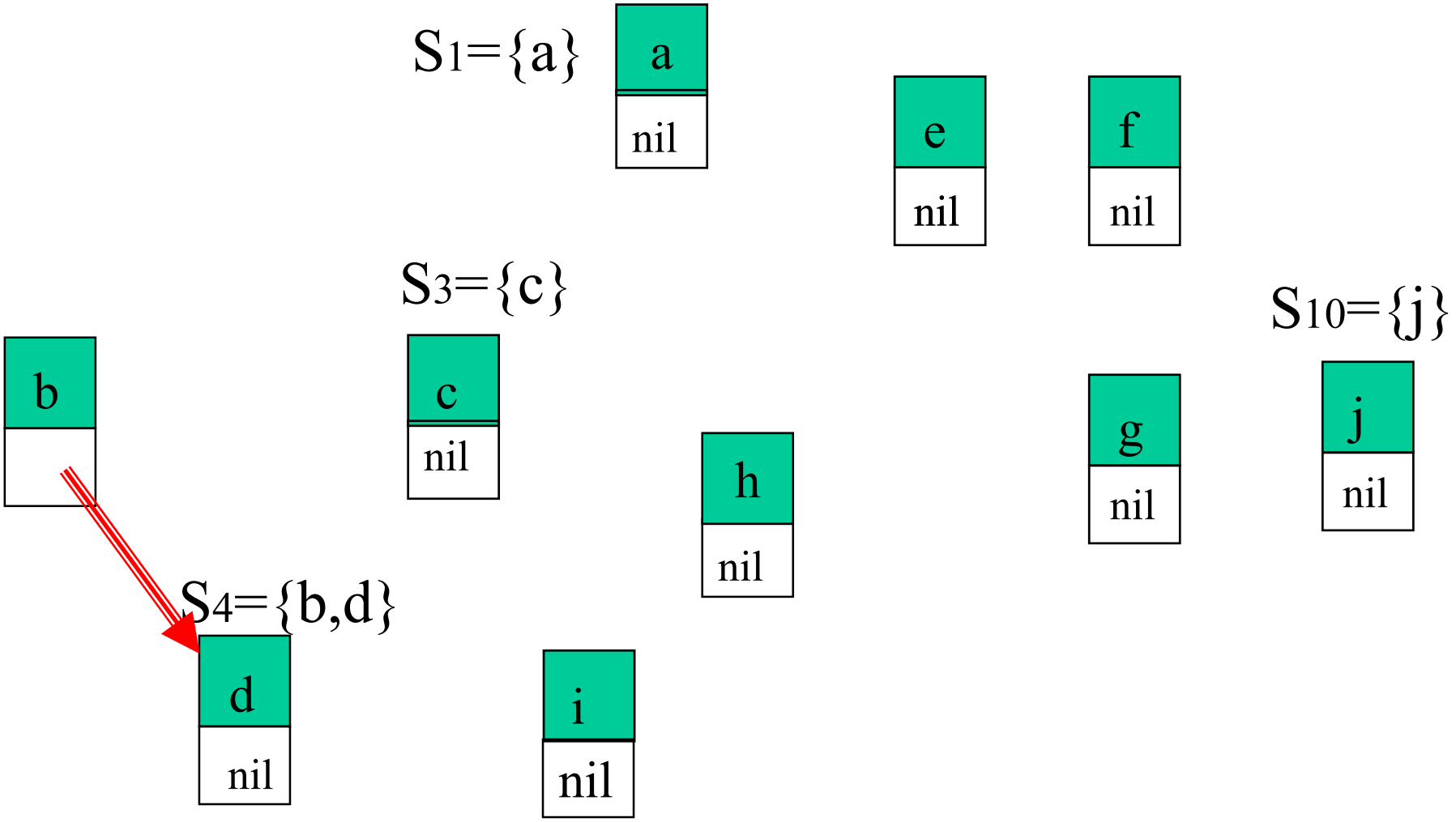

4 συνεκτικές συνιστώσες: {a,b,c,d}, {e,f,g}, {h,i}, {j}

S=[a,b,c,d,e,f,g,h,i,j]



Πλευρά [b, d]

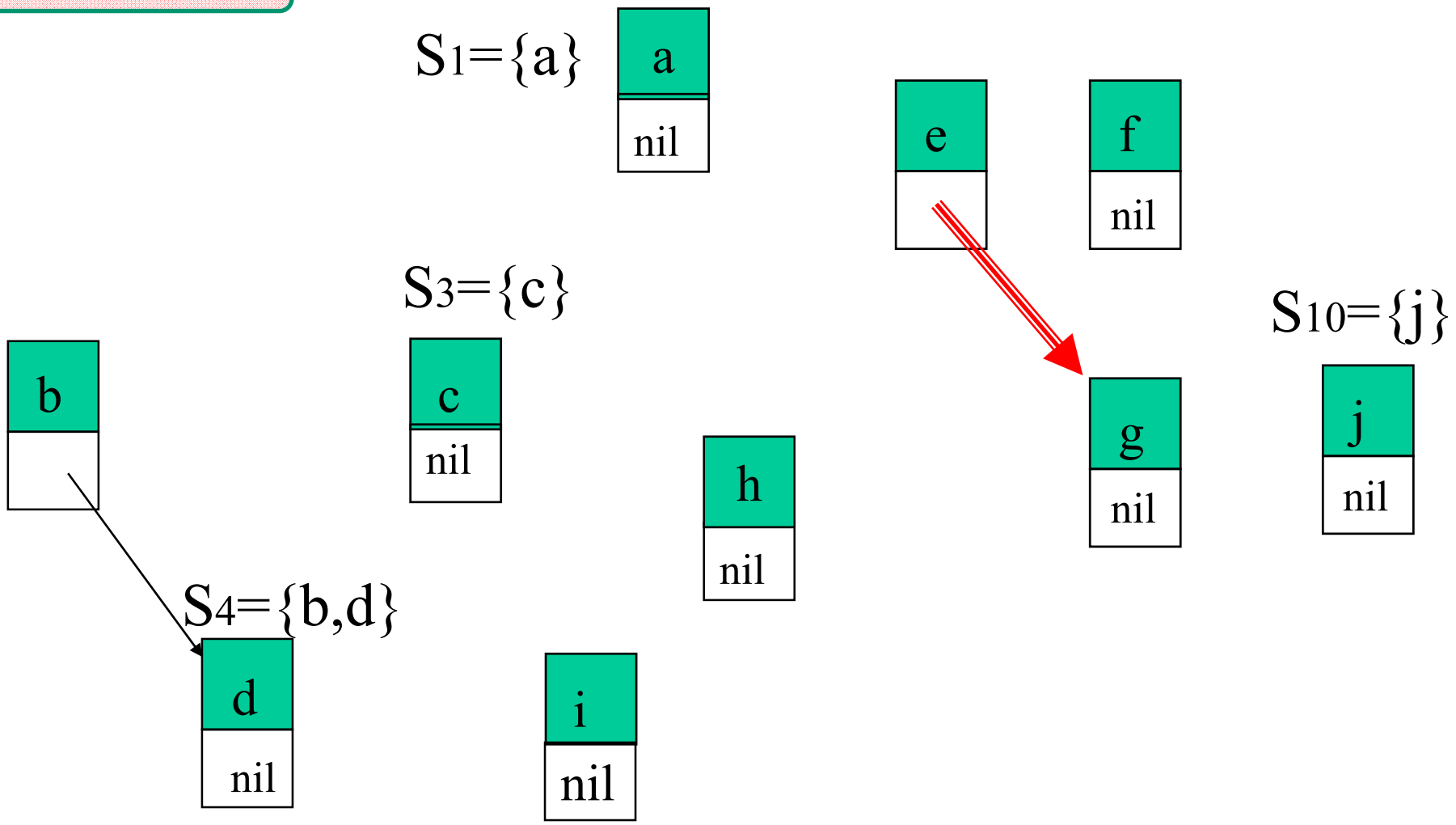
S=[a,b,c,d,e,f,g,h,i,j]



Πλευρά [b, d]

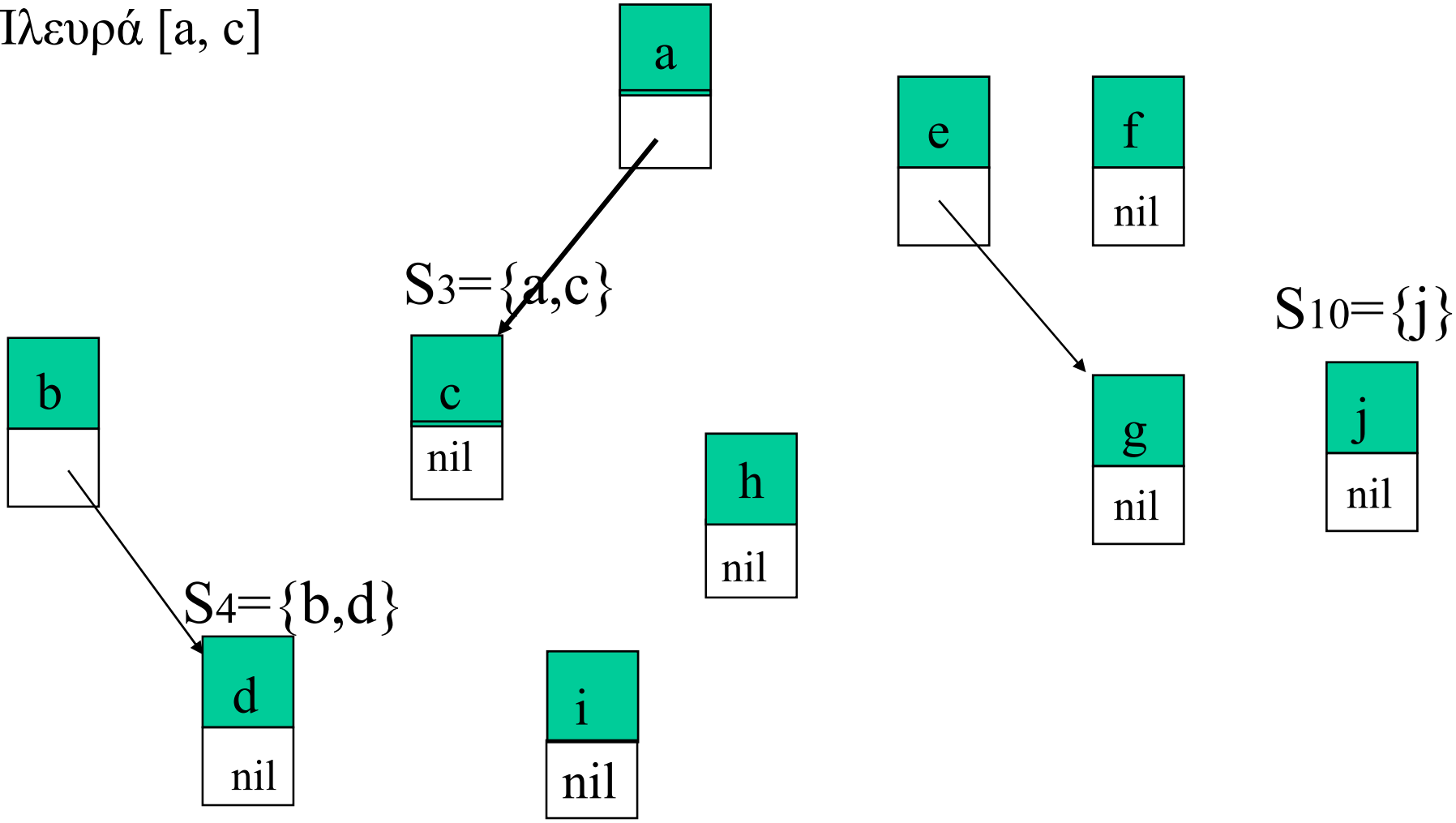
Πλευρά [e, g]

S=[a,b,c,d,e,f,g,h,i,j]



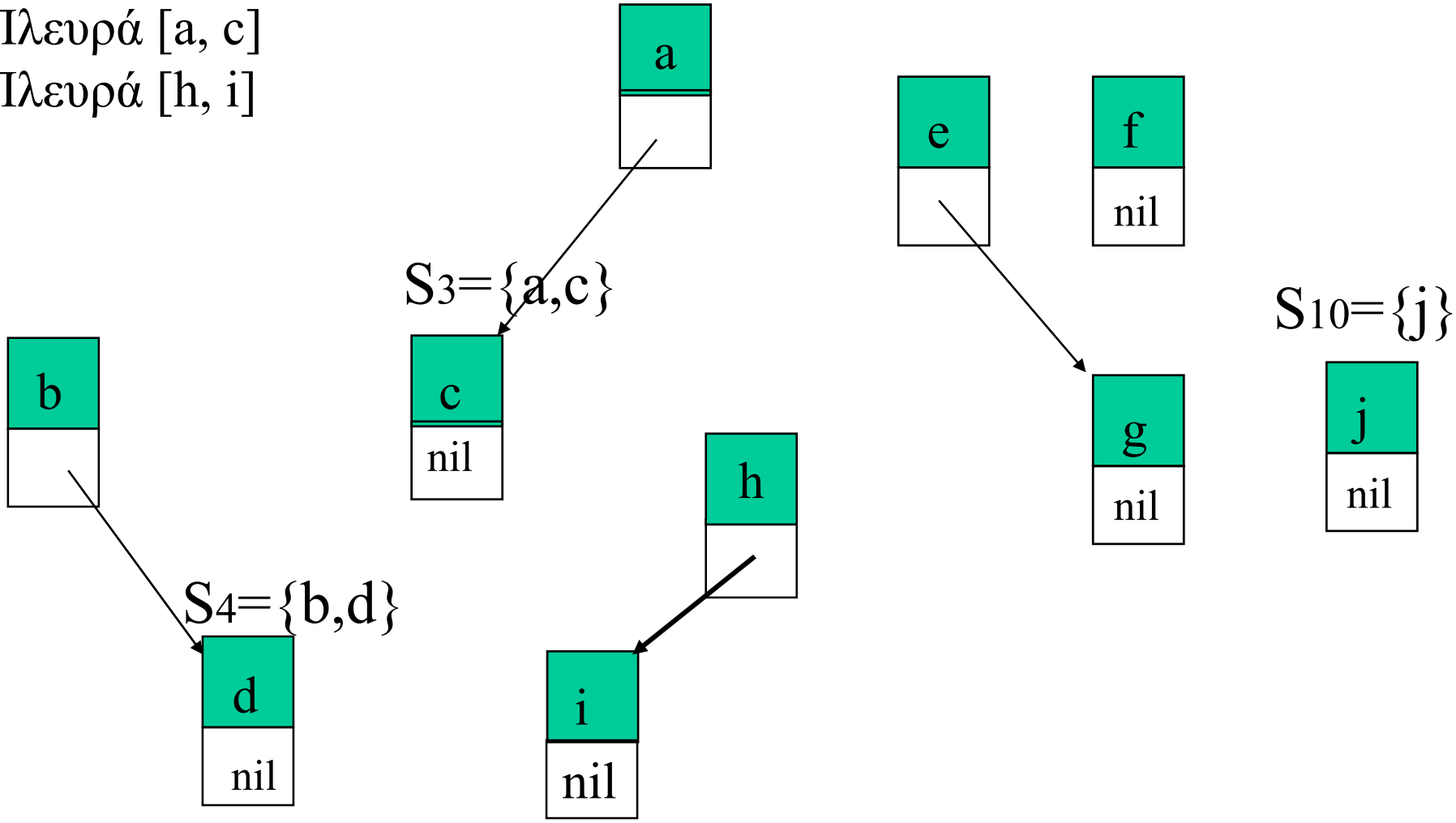
Πλευρά [b, d]
Πλευρά [e, g]
Πλευρά [a, c]

S=[a,b,c,d,e,f,g,h,i,j]



Πλευρά [b, d]
Πλευρά [e, g]
Πλευρά [a, c]
Πλευρά [h, i]

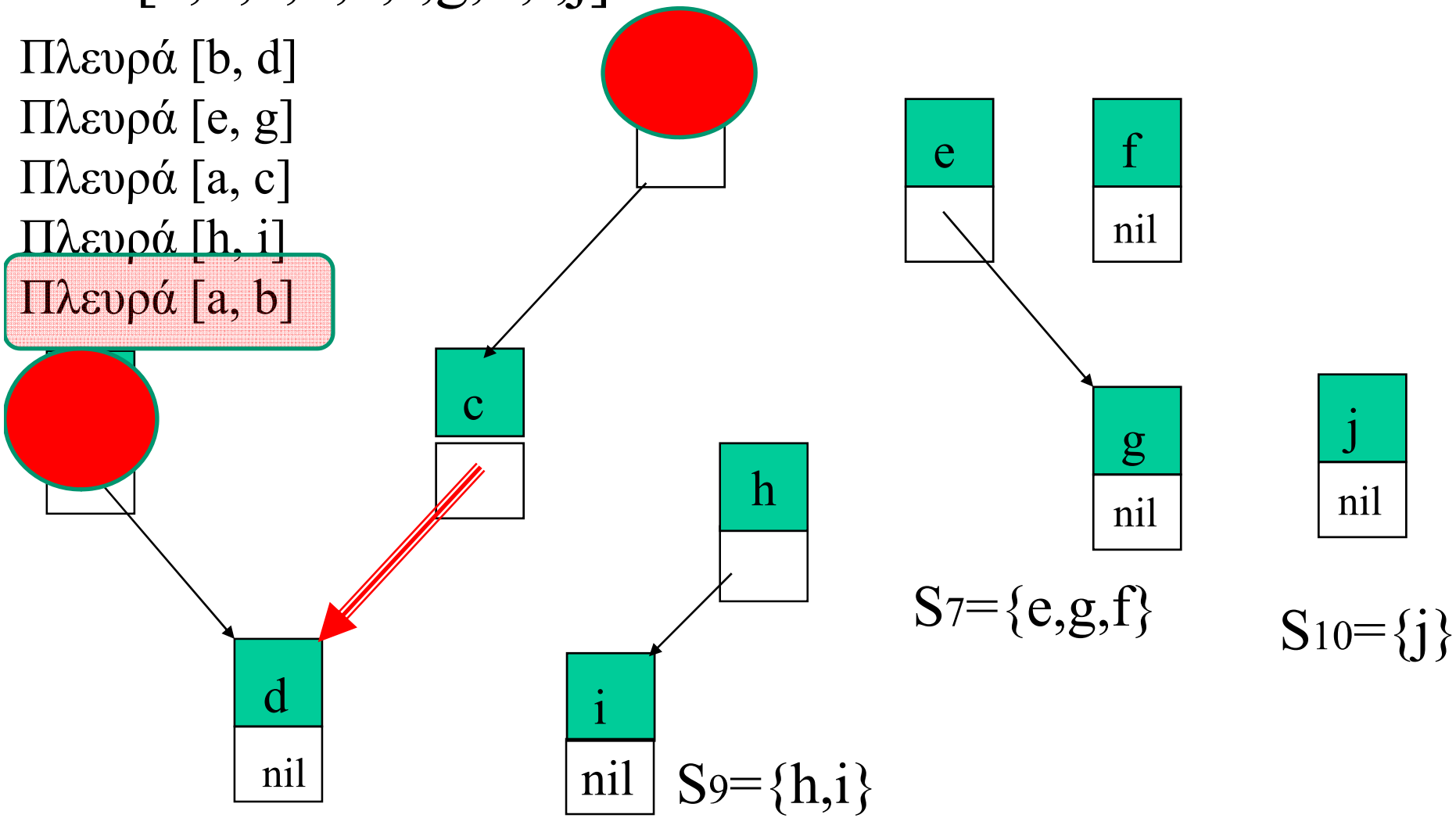
S=[a,b,c,d,e,f,g,h,i,j]



# ΣΥΝΟΛΑ: FIND(a), FIND(b), UNION(c,d)

S=[a,b,c,d,e,f,g,h,i,j]

Πλευρά [b, d]
Πλευρά [e, g]
Πλευρά [a, c]
Πλευρά [h, i]
Πλευρά [a, b]




$S_4=\{a,c,b,d\}$

$S_9=\{h,i\}$

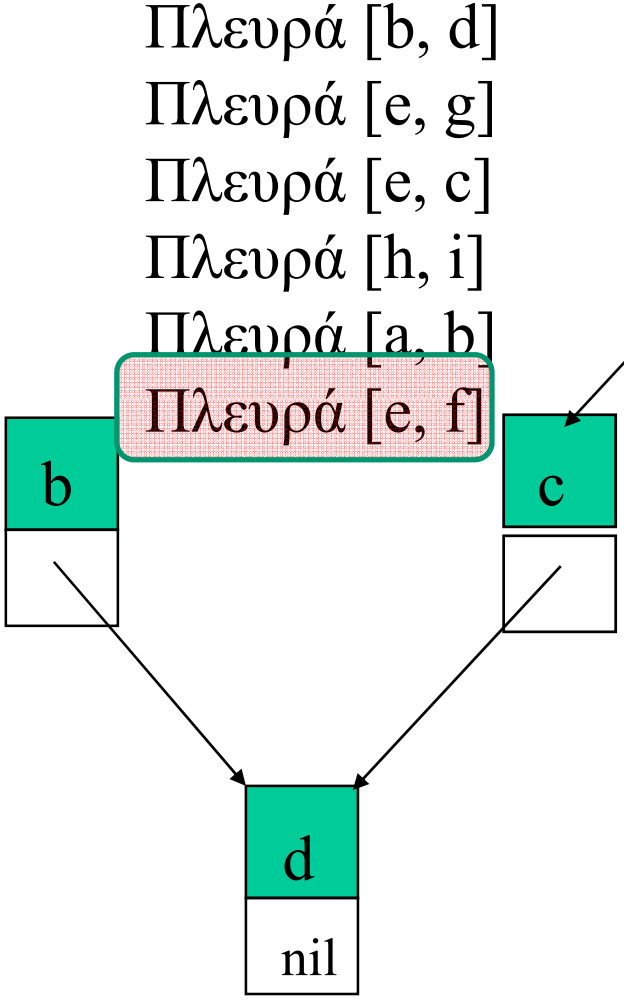
$S_7=\{e,g,f\}$

$S_{10}=\{j\}$

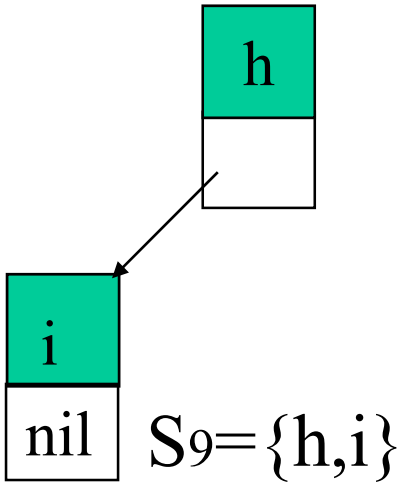
**ΣΥΝΟΛΑ:**
**FIND(e), FIND(f), UNION(g,f)**

S=[a,b,c,d,e,f,g,h,i,j]

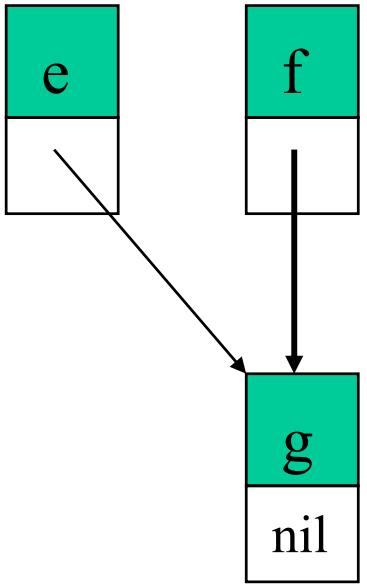
Πλευρά [b, d]
Πλευρά [e, g]
Πλευρά [e, c]
Πλευρά [h, i]
Πλευρά [a, b]
Πλευρά [e, f]



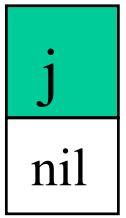

S4={a,c,b,d}




S9={h,i}




S7={e,g,f}

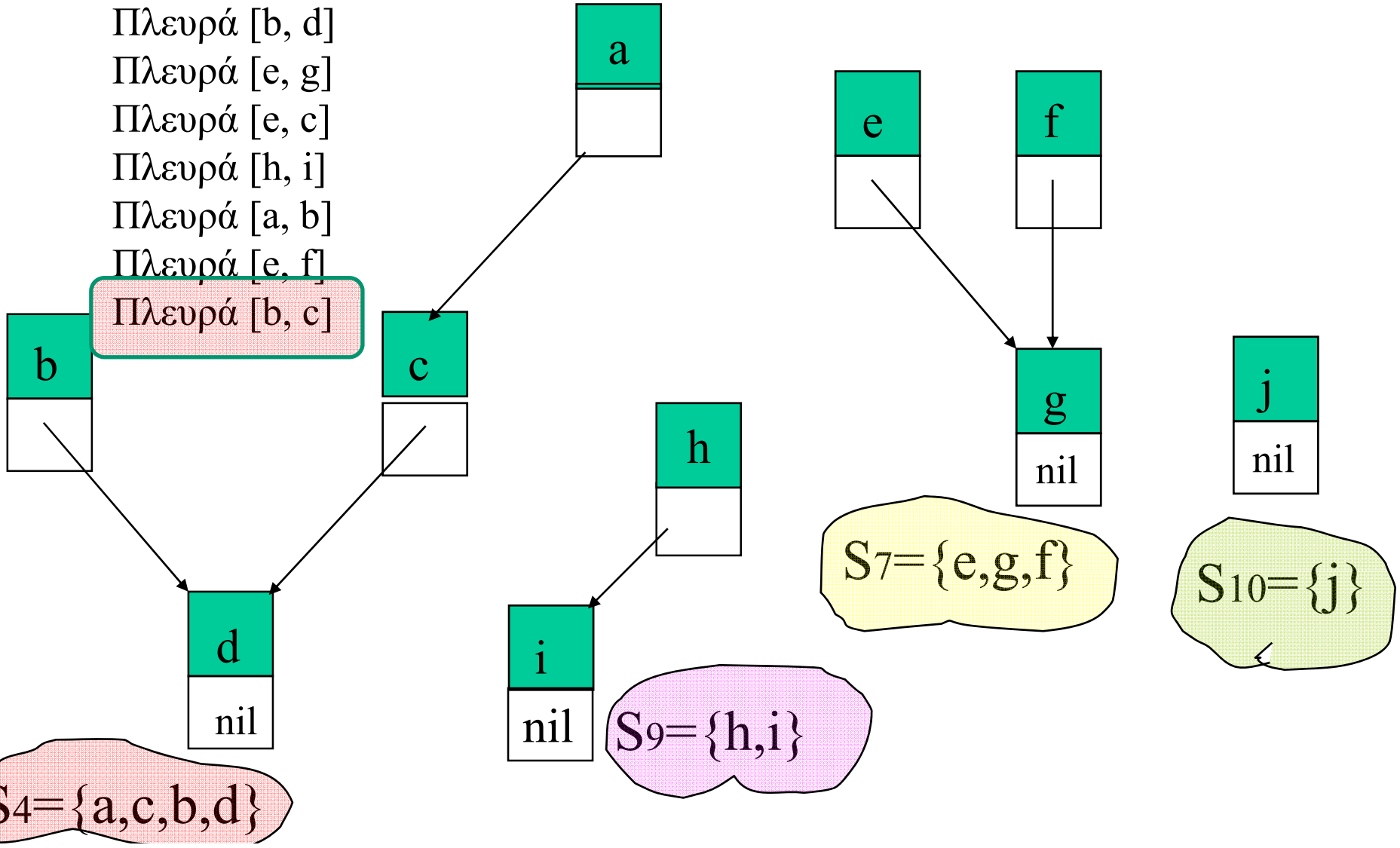



S10={j}

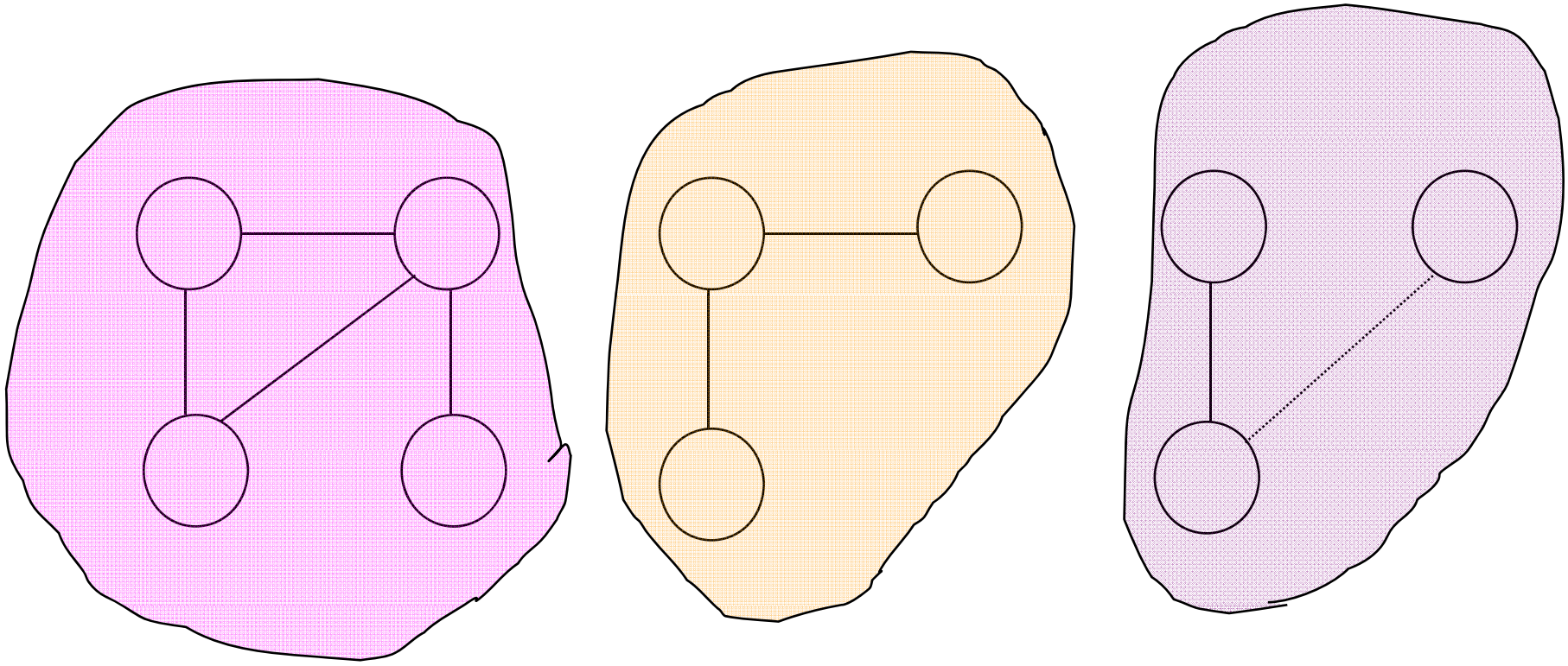
**ΣΥΝΟΛΑ:**
**FIND(b), FIND(c)**

S=[a,b,c,d,e,f,g,h,i,j]

Πλευρά [b, d]
Πλευρά [e, g]
Πλευρά [e, c]
Πλευρά [h, i]
Πλευρά [a, b]
Πλευρά [e, f]
Πλευρά [b, c]



# *Αλγόριθμος 2(Συνεκτικές Συνιστώσες με* **Union and Find)**



# Αλγόριθμος 2 για εύρεση Συνεκτικών Συνιστωσών

*Input*: $G = (V, E)$, $|V| = n$

CONNECTED_COMPONENTS (G)

Για κάθε κόμβο $v \in V$
    MAKE_SET (v)

Για κάθε πλευρά $[u, v] \in E$
    if FIND_SET (u) ≠ FIND_SET (v)
        then UNION (u, v)

E "στατικό" → ?

E "Δυναμικό"

# Επερωτήσεις για Συνεκτικές Συνιστώσες

$G = (V,E)$ , $|V| = n$

SAME_COMPONENT $(u,v)$

if FIND_SET$(u)$ = FIND_SET$(v)$

then return TRUE

else return FALSE

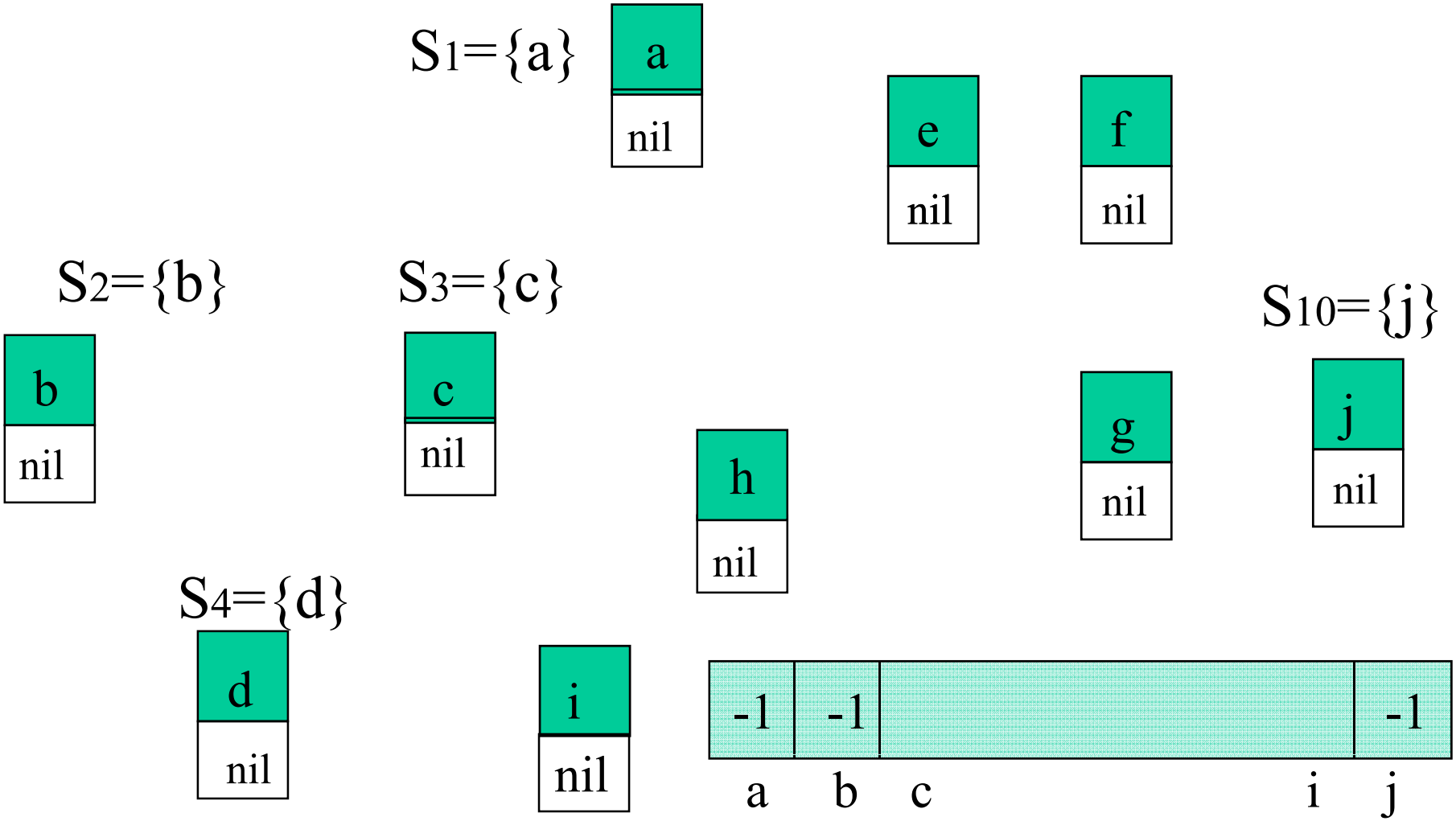
# Κωδικοποίηση των **n** στοιχείων

**`Next[x]`**

Δείχνει τον επόμενο του x στο σύνολο που βρίσκεται.

**`Next[ρίζας]`** = -(πληθικός αριθμός)

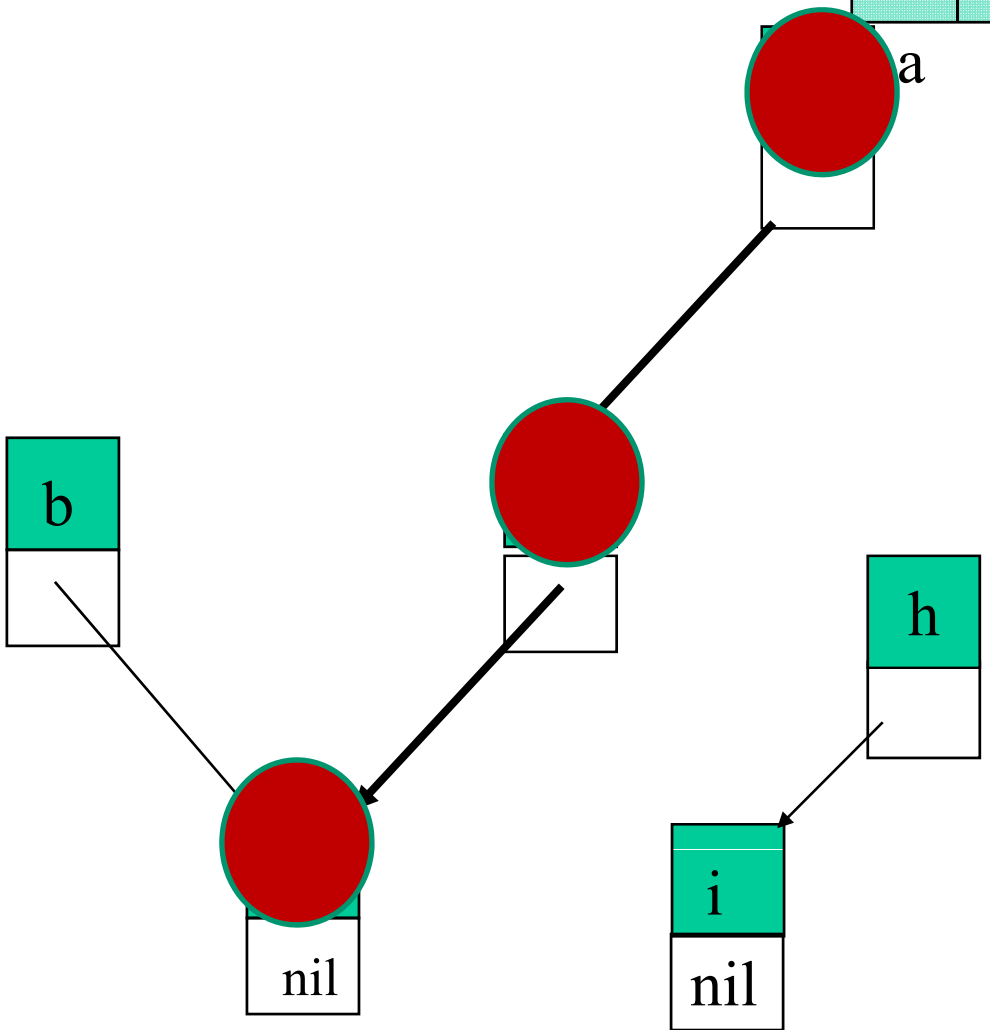
S=[a,b,c,d,e,f,g,h,i,j]



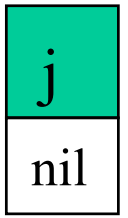
S=[a,b,c,d,e,f,g,h,i,j]

$S_4$={a,c,b,d}

| c | d | d | -4 | | -2 | -1 |
|---|---|---|---|---|---|---|
| a | b | c | d | | i | j |




$S_{10}$={j}



- Πρόσβαση στο x σε O(1)

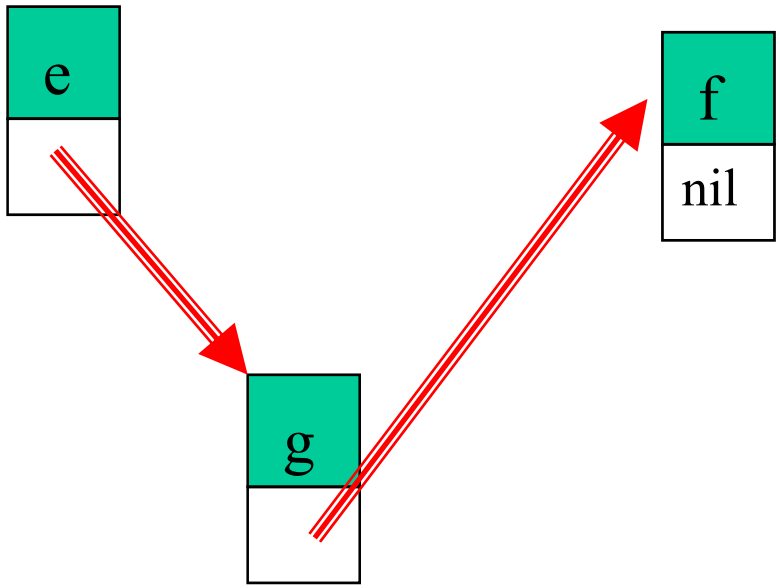
- **`Union(u, v)`** σε O(1)

- **`Find[x]`** → O(n) αν το αντικατευθυνόμενο δένδρο είναι εκφυλισμένο σε μία γραμμική λίστα

## ΣΥΝΟΛΑ: FIND(e),FIND(f) and UNION(g,f)

$S_7=\{e,g,f\}$

Find() σε O(n)




**Σύνδεση μεγαλύτερου στο μικρότερο μπορεί να δημιουργήσει λίστα**

## *Πολυπλοκότητα* `Find[x]`

Αν κατά την ένωση ενώνουμε το δένδρο με τους λιγότερους κόμβους σε αυτό με τους περισσότερους

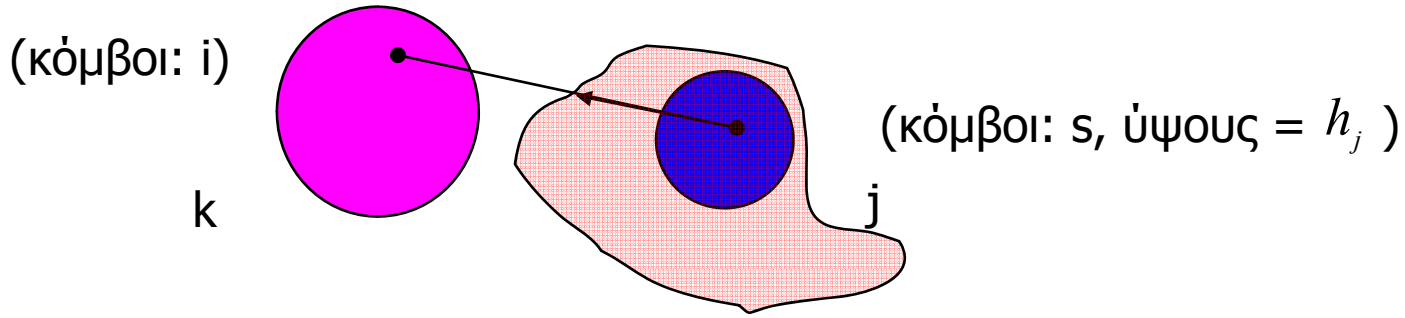
ΤΟΤΕ **Find[x]** σε **O(logn)**

Μία ακολουθία από **p** πράξεις **Union** και **Find** σε ένα σύνολο **n** στοιχείων έχει πολυπλοκότητα **O(p logn)**

# Find( ) → logn (επαγωγή)

Αποδειξη: Find() σε log(n)

Union(k,j) ⇒ j (μικροτερο) *στο* k (μεγαλυτερο)



```
Function Find(u)
    x := u;
    while Next[x] > 0 do
        x := Next[x]
    Find := x;
endwhile
```

```
Procedure Union(x,y)

  Newnodes := Next[x] + Next[y]

  if Next[x] > Next[y] then
     Next[x] := y;
     Next[y] := NewNodes;

  else
     Next[y] := x;
     Next[x] := NewNodes;
```