ΣΥΝΟΛΑ

Βάση για γενικευμένες δομές δεδομένων

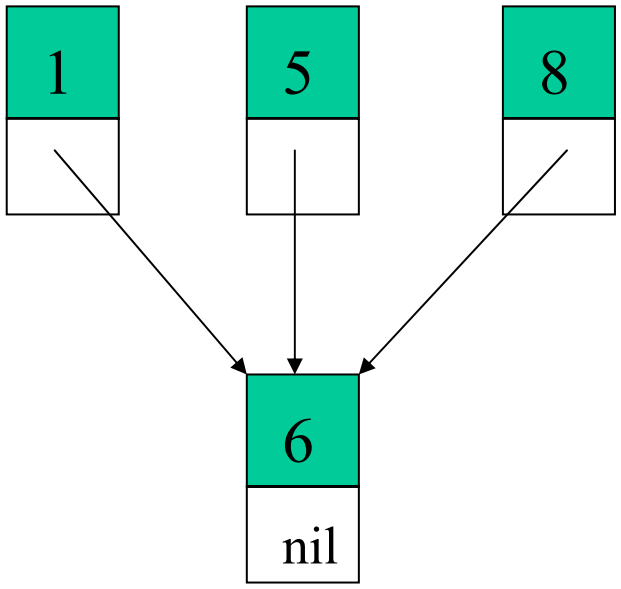
(όταν απαιτείται γρήγορη ανάκτηση πληροφορίας)

Αναπαράσταση: *αντι-κατευθυνόμενα δένδρα*

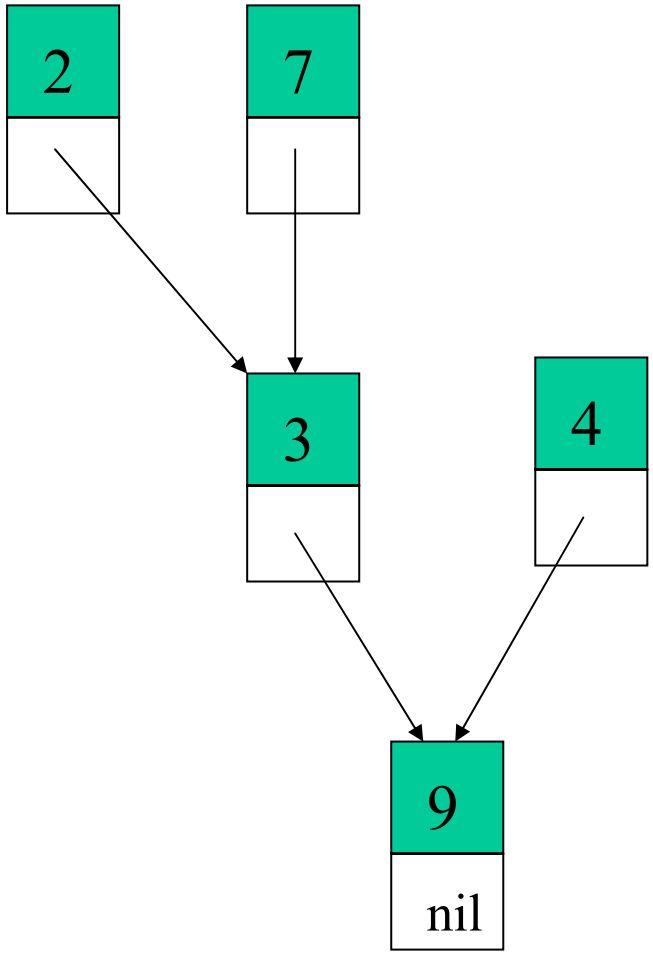
Μοναδική σύνδεση κάθε στοιχείου με το επόμενό του.

S=[1..9]

$S_1=\{1,5,6,8\}$



$S_2=\{2,7,3,4,9\}$



`Find(x)`

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

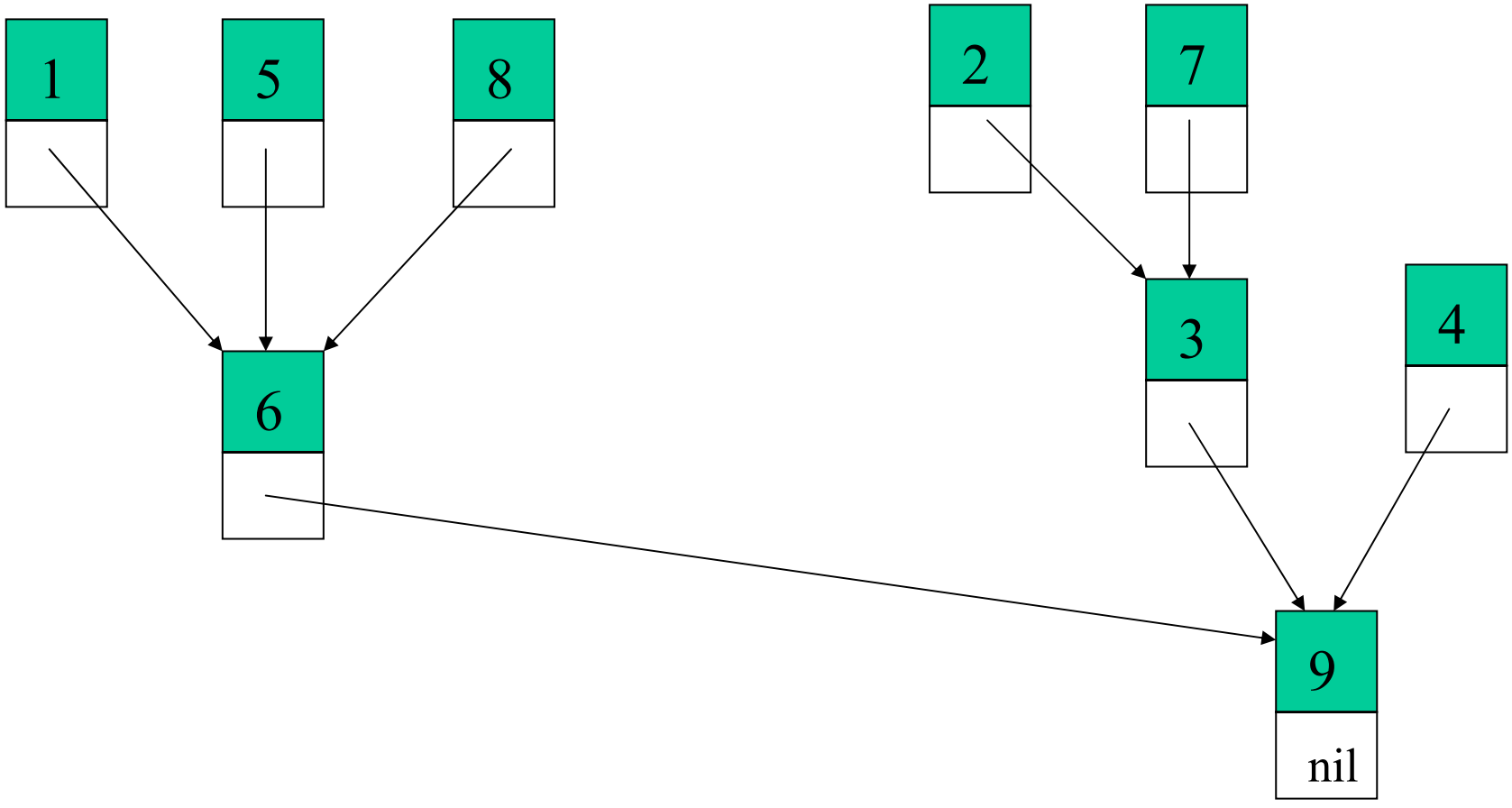
`Union(u, v)`

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

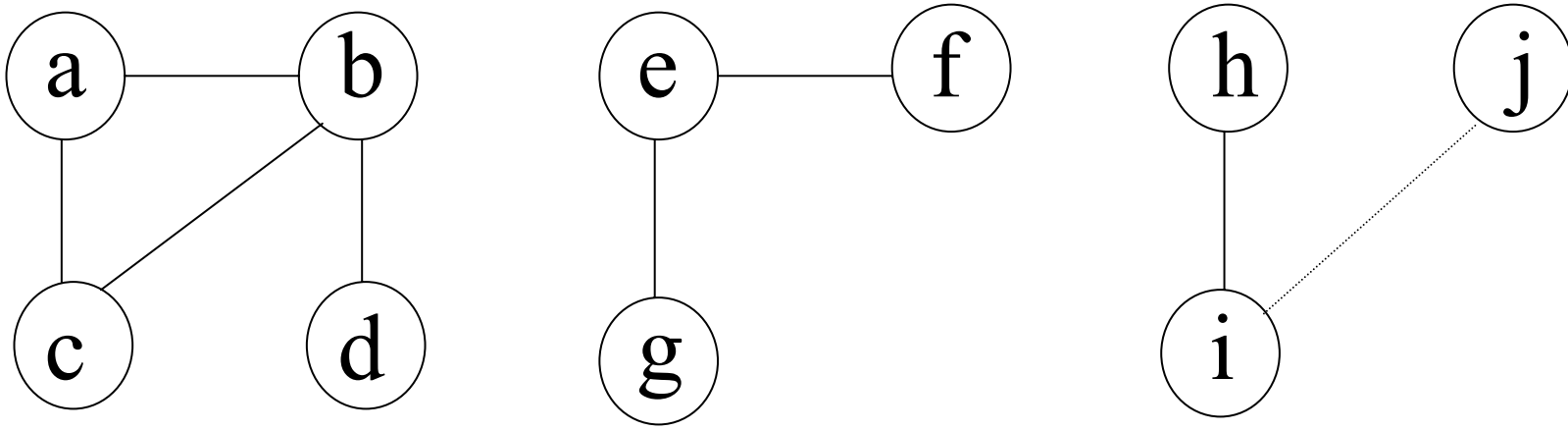
Find(5)=6

$\Rightarrow$ **Union(6,9)=9**

Find(7)=9

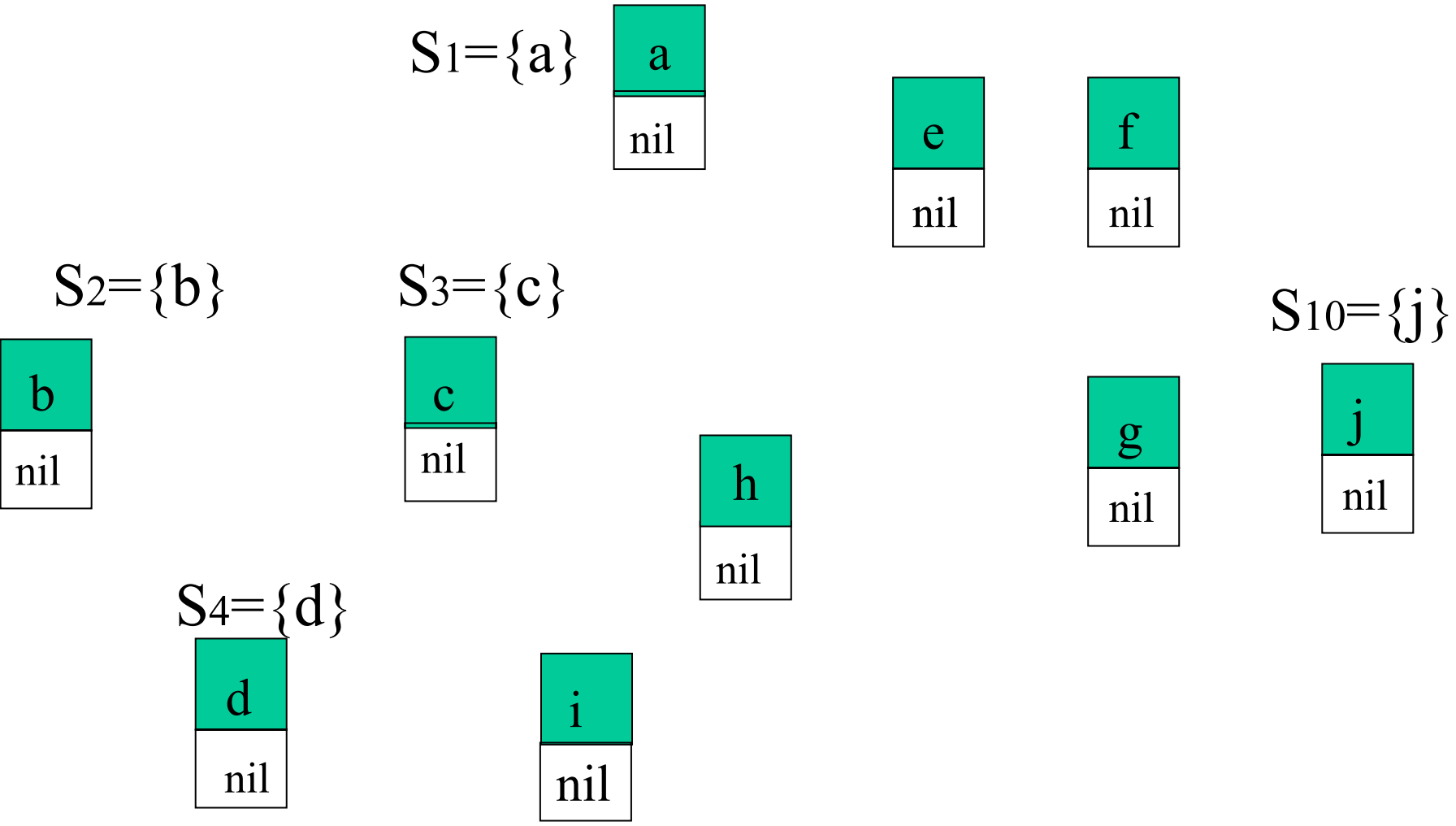


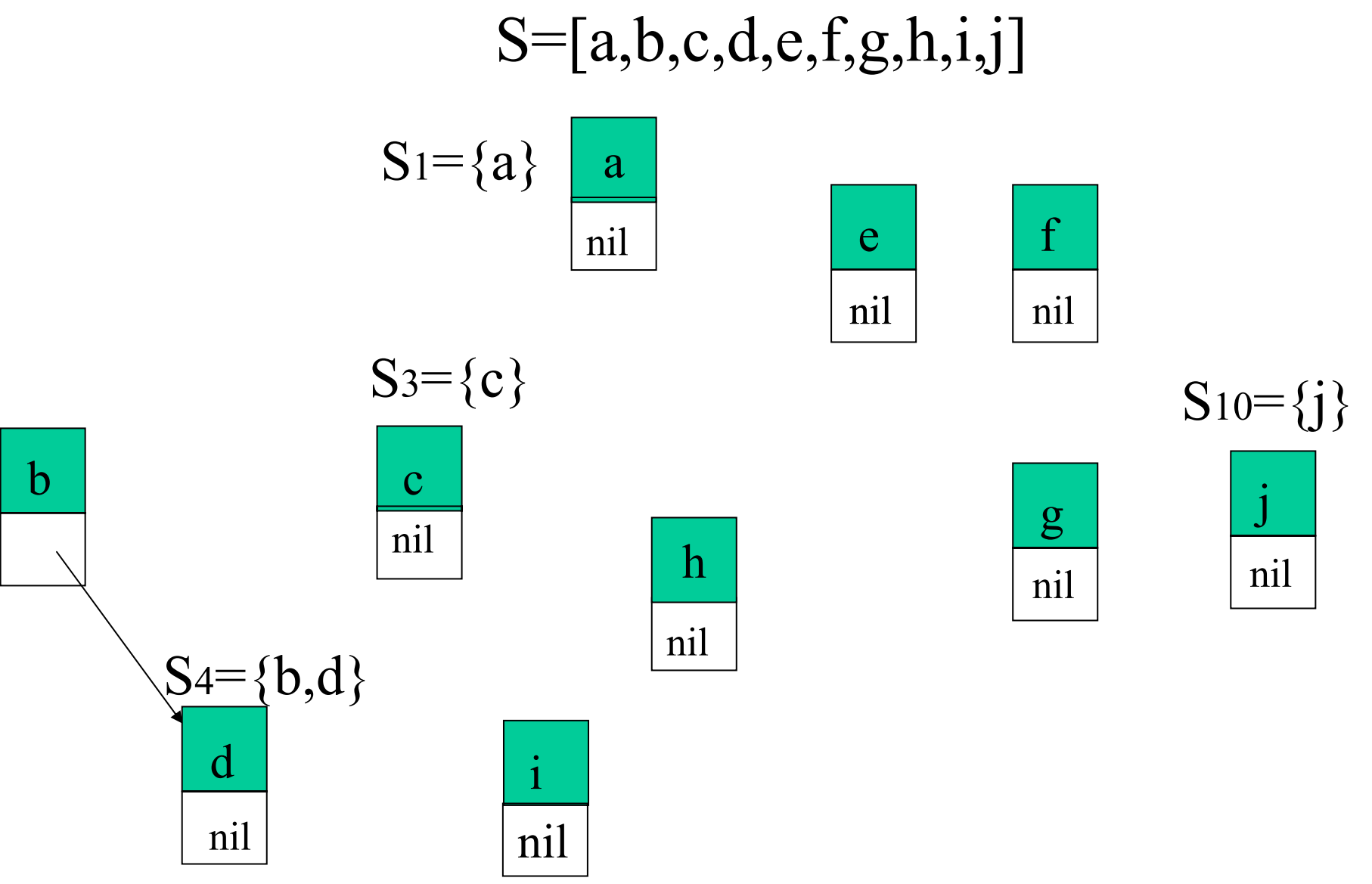
Union and Find: παράδειγμα

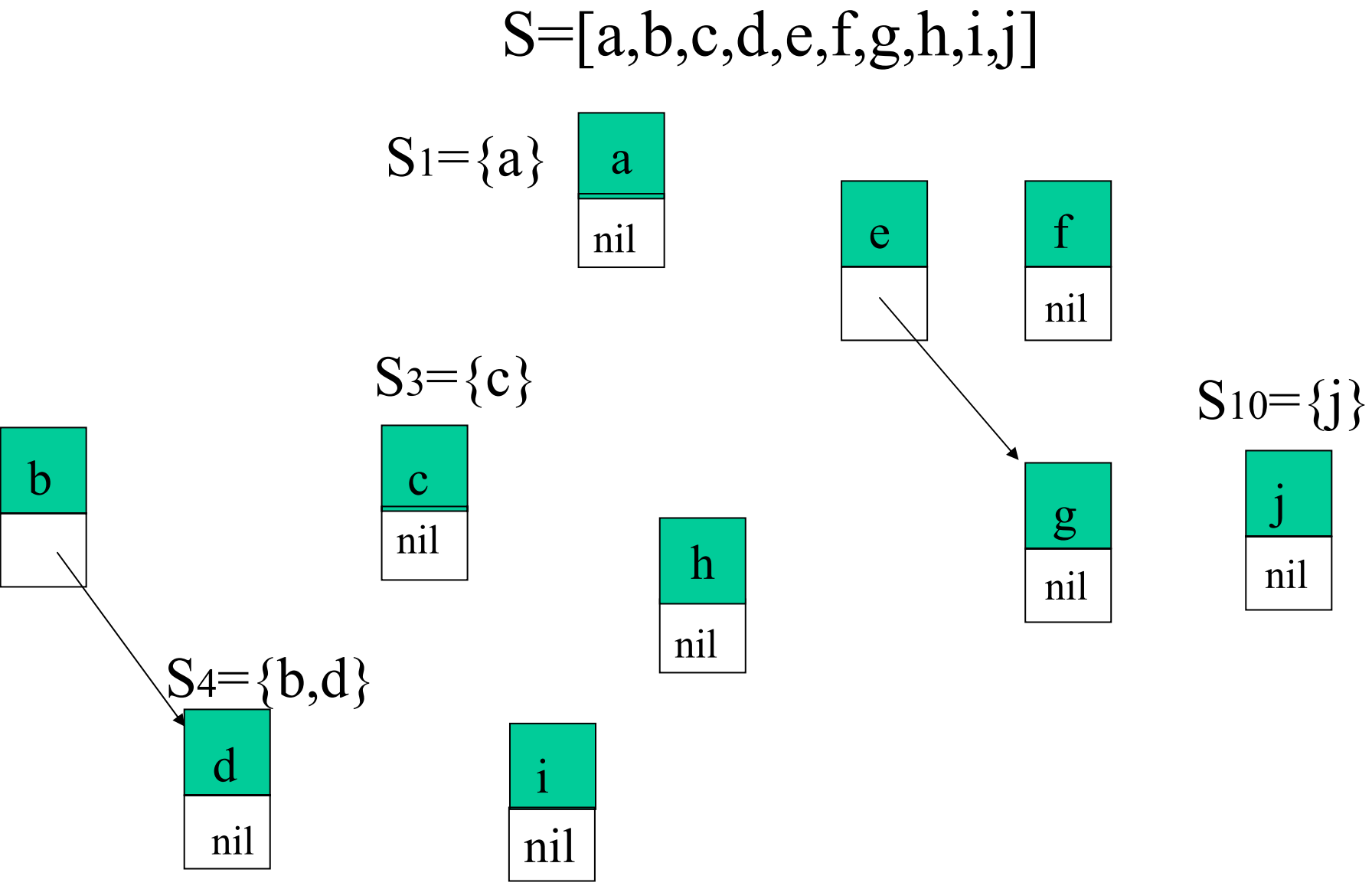


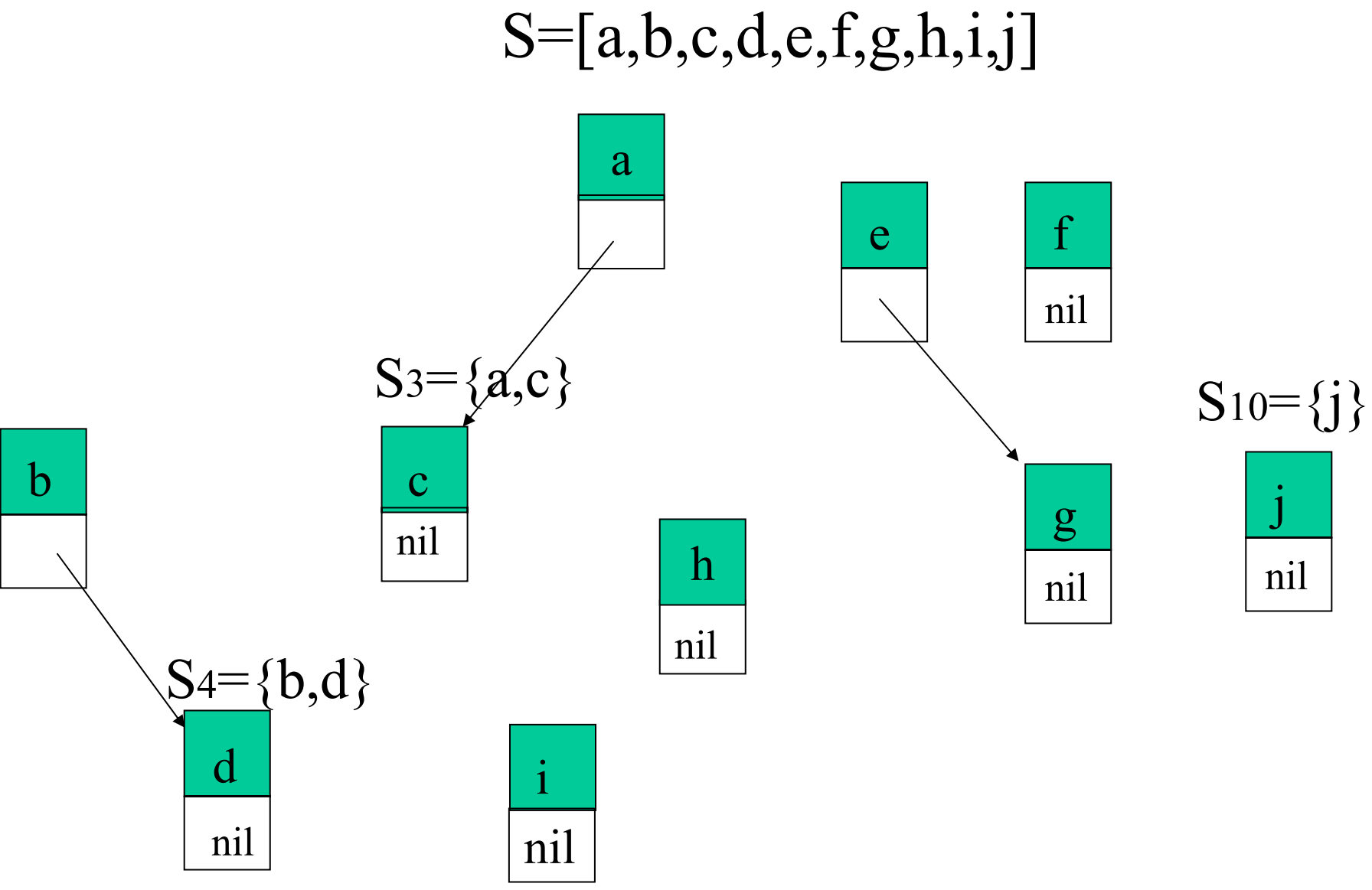
4 *συν. συνιστωσες* : $\{a, b, c, d\}, \{e, f, g\},$
$\{h, i\}, \{j\}$

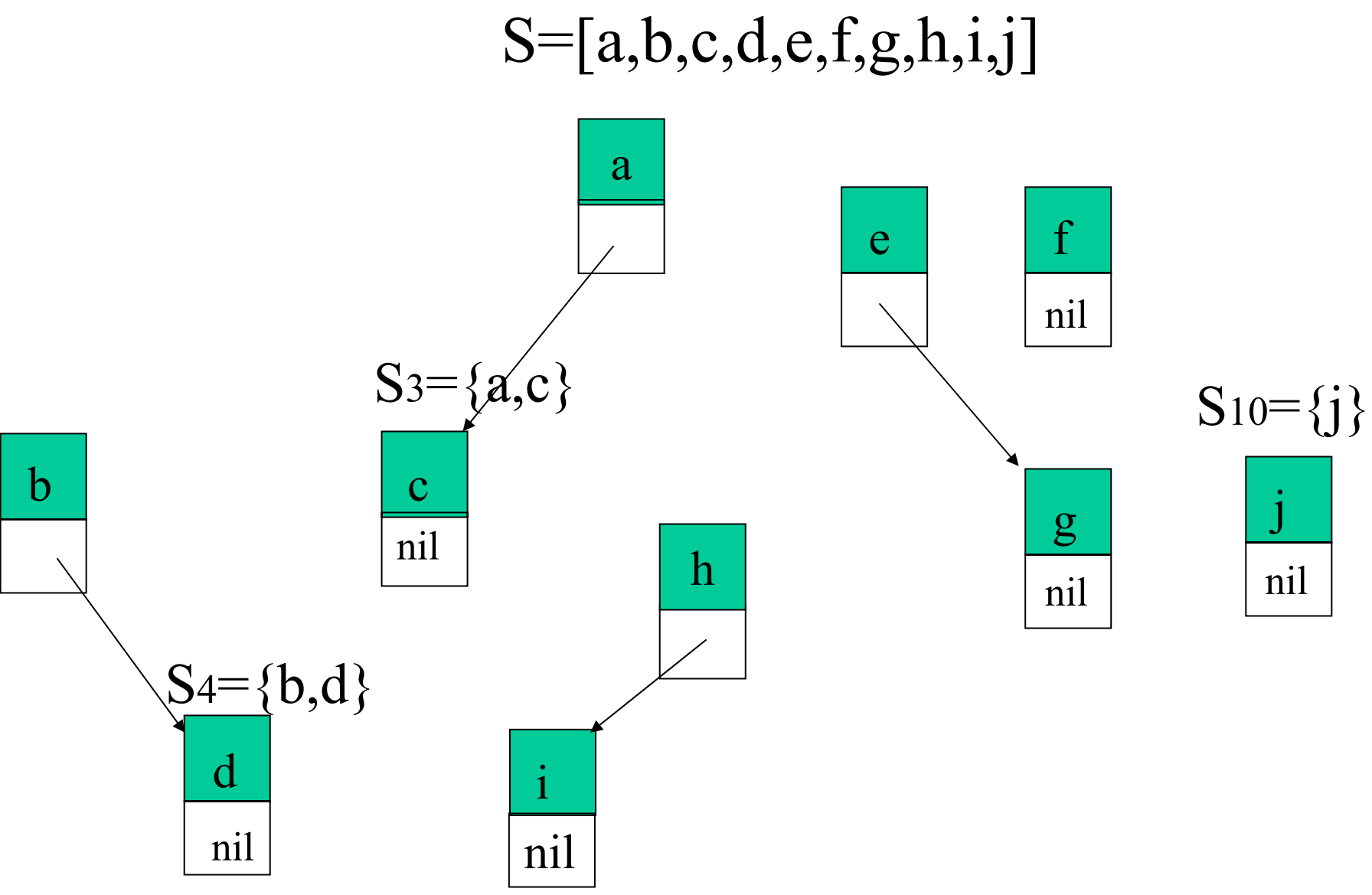
S=[a,b,c,d,e,f,g,h,i,j]



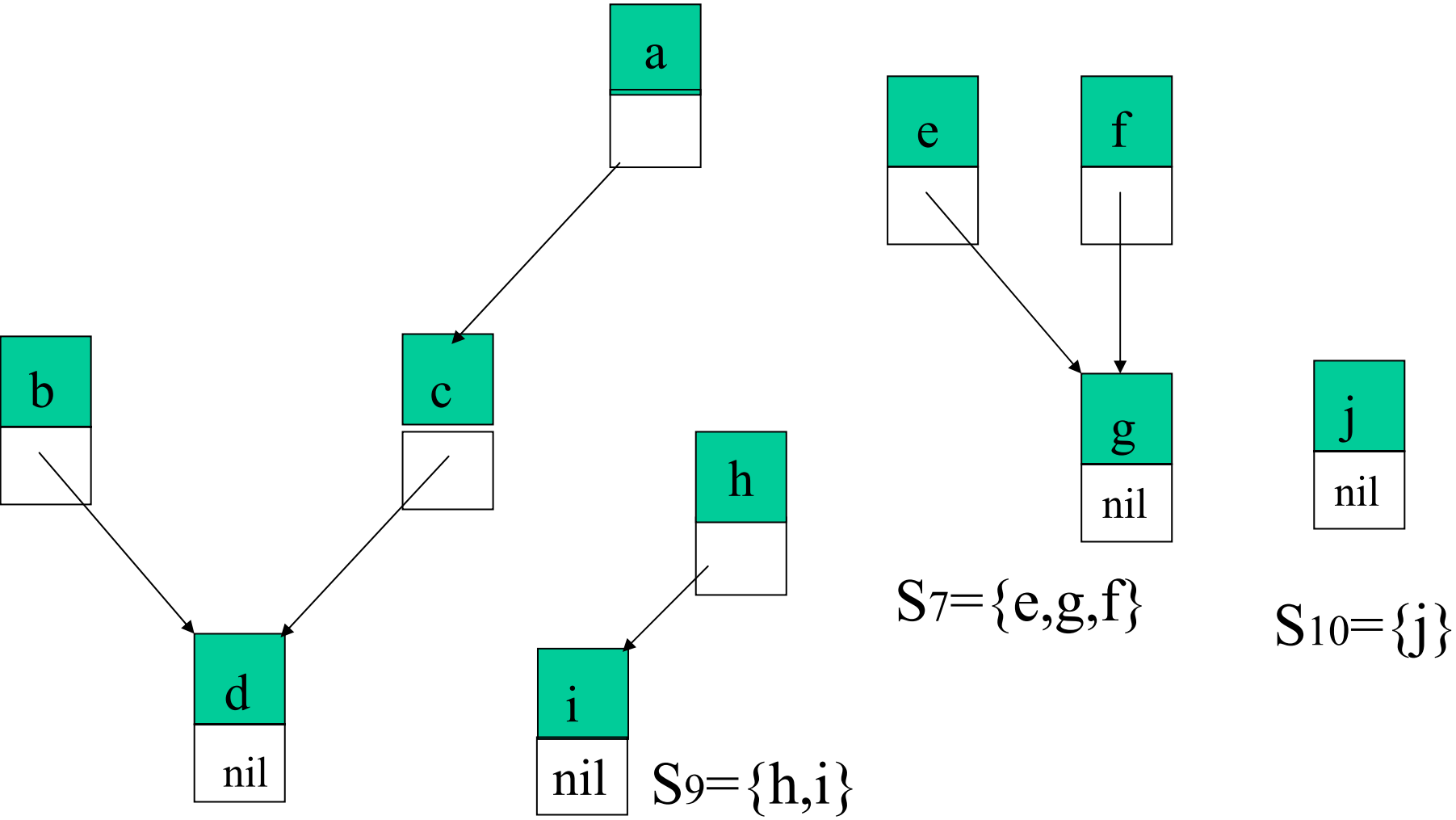

S=[a,b,c,d,e,f,g,h,i,j]
S1={a}
a
nil
e
nil
f
nil
S3={c}
S10={j}
b
c
nil
g
nil
j
nil
h
nil
S4={b,d}
d
nil
i
nil


S=[a,b,c,d,e,f,g,h,i,j]
S1={a}
a
nil
e
f
nil
S3={c}
S10={j}
b
c
nil
g
nil
j
nil
h
nil
S4={b,d}
d
nil
i
nil


S=[a,b,c,d,e,f,g,h,i,j]
a
e
f
nil
S3={a,c}
S10={j}
b
c
nil
g
nil
j
nil
h
nil
S4={b,d}
d
nil
i
nil


S=[a,b,c,d,e,f,g,h,i,j]
a
e
f
nil
S3={a,c}
S10={j}
b
c
nil
g
nil
j
nil
h
S4={b,d}
d
nil
i
nil

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



S9={h,i}

S7={e,g,f}

S10={j}

S4={a,c,b,d}

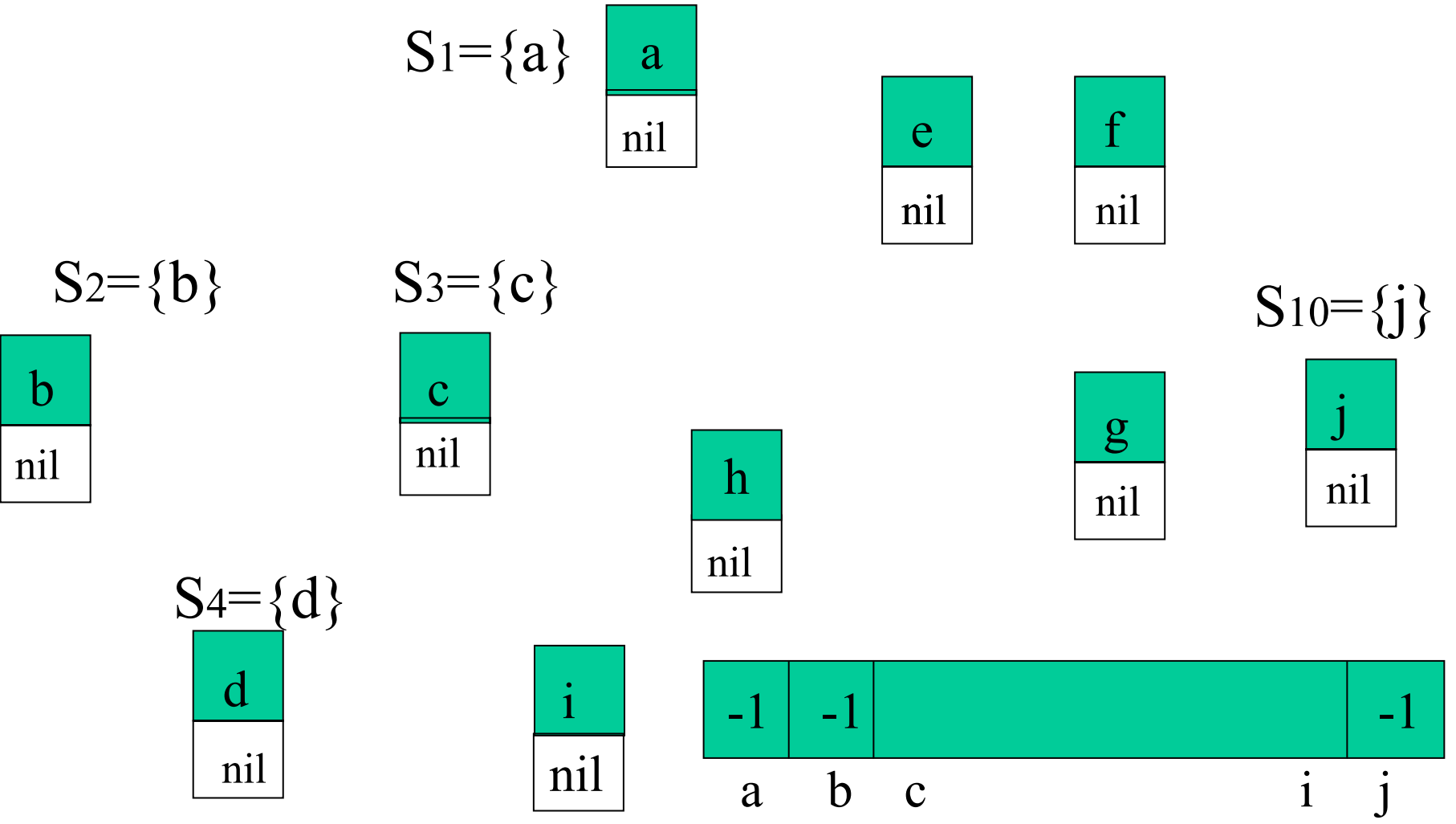
Κωδικοποίηση των **n** στοιχείων

`Next[x]`

Δείχνει τον επόμενο του x στο σύνολο που βρίσκεται.

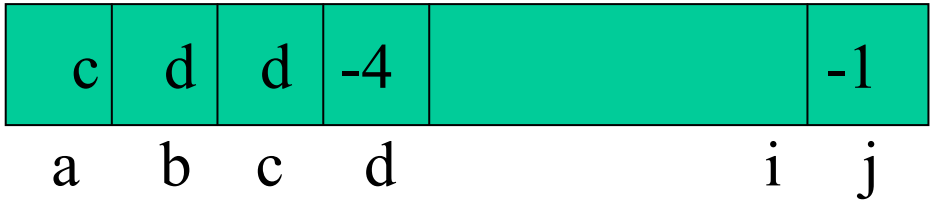
`Next[ρίζας]` = `-(πληθικός αριθμός)`

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}

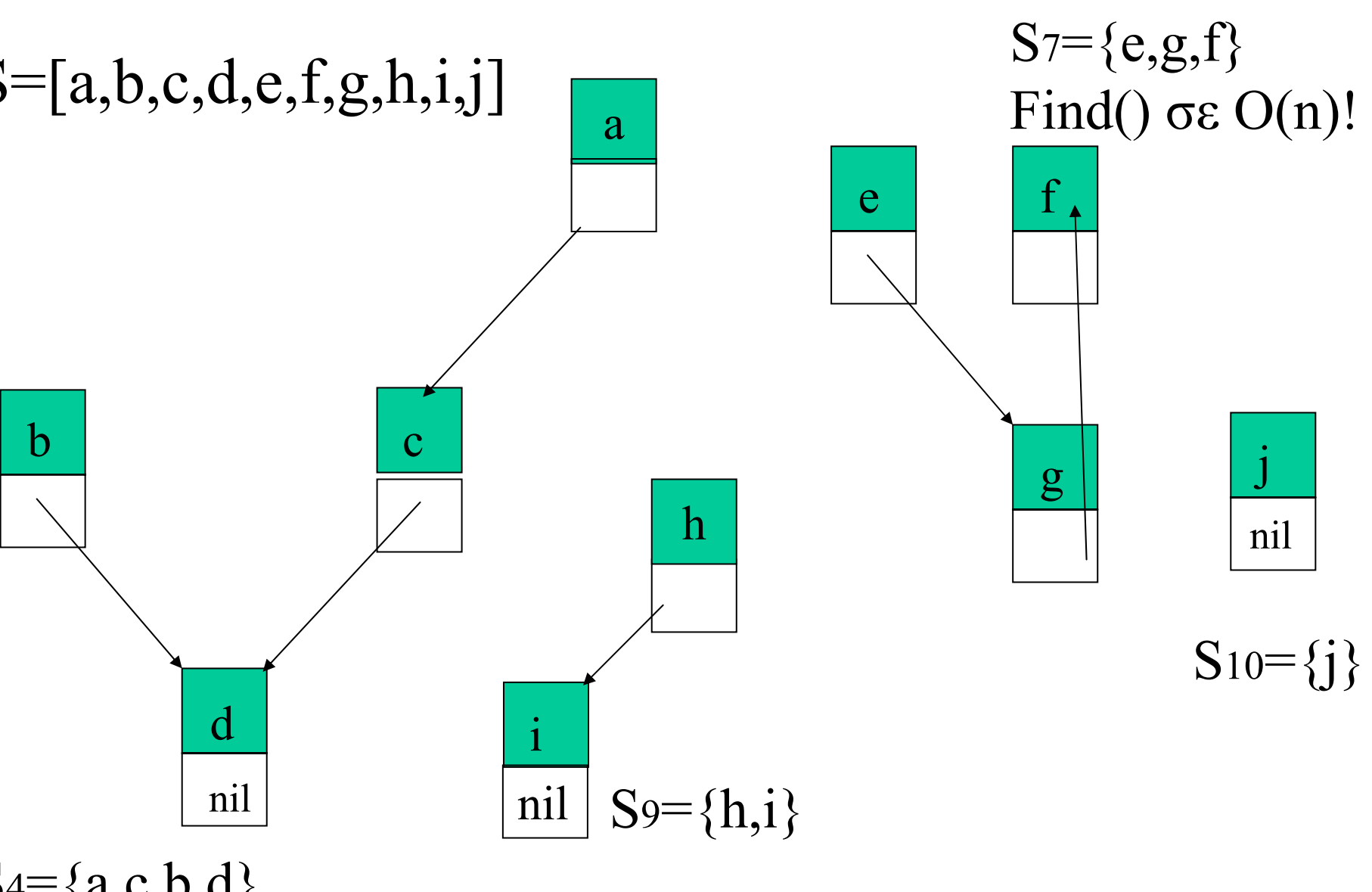


- Πρόσβαση στο x σε O(1)

- **`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,h)
Σύνδεση μεγαλύτερου στο μικρότερο μπορεί να δημιουργήσει λίστα



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

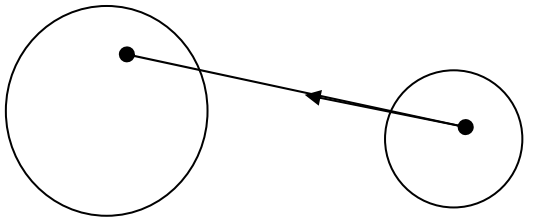
ΤΟΤΕ **`Find[x]`** σε O(logn)

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

Αποδειξη: Find() σε log(n)

$\text{Union}(\text{k,j}) \Rightarrow$ j (μικροτερο) $\sigma\tau o$ k (μεγαλυτερο)

(κόμβοι: l)

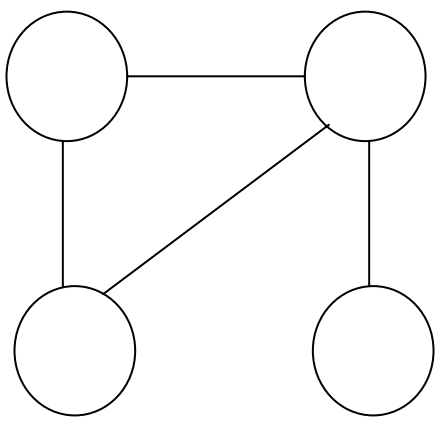


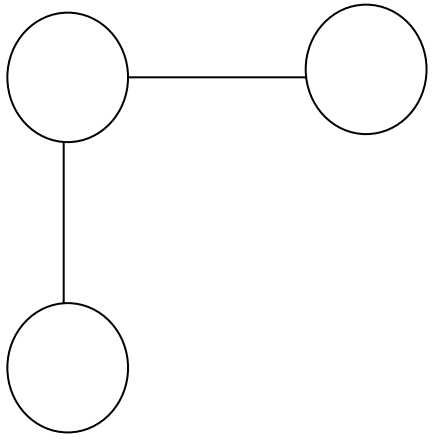
(κόμβοι: s, ύψους = h_j)

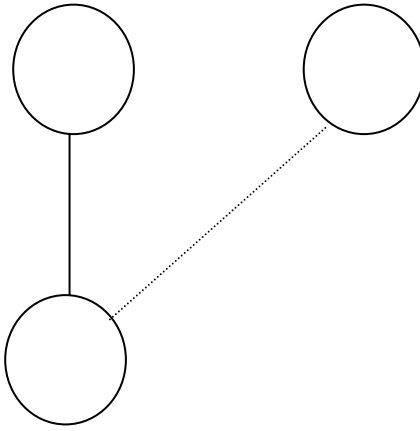
k

j

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 ∈ V
    MAKE_SET (v)

  Για κάθε πλευρά [u, v] ∈ 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

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

      Find := x;
```

```
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;
```