Disjoint Sets

Αδάμος Ττοφαρή Junior Washers Lessons

Περιεχόμενα

- Εισαγωγή
- Διεργασίες
- Πως είναι δομημένο
- Union
  - Visualization
  - Implementation
  - Optimization
- Find
  - Visualization
  - Implementation
  - Optimization

Εισαγωγή

Τα Disjoint Sets data structure ή αλλιώς union–find data structure είναι μια δομή δεδομένων για διαχείριση ασύνδετων συνόλων (Σύνολα τα οποία δεν έχουν κοινά στοιχειά μεταξύ τους).

Διεργασίες

Τα Disjoint Sets υποστηρίζουν 2 βασικές και χρήσιμες διεργασίες:

- Find: Διεργασία για εύρεση του συνόλου που ανήκει ένα συγκεκριμένο στοιχειό
- Union: Διεργασία για ένωση 2 Συνόλων

Πως είναι δομημένο

Τα Disjoint Sets αρχικά αποτελούνται από Ν σύνολα που το κάθε ένα περιέχει ένα στοιχειό. Με πιο άπλα λόγια το στοιχειό 42 ανήκει μονό του στο σύνολο 42.

Μετά συνήθως τα ερωτήματα που ακολουθούν είναι της μορφής «Το σύνολο που ανήκει το στοιχειό Α να ενωθεί με το σύνολο του στοιχειού Β» για αυτό καλώντας την Find βρίσκουμε το σύνολο που βρίσκετε το Α και το εντάσσουμε στο Β.

Αυτό μπορεί εύκολα να υλοποιηθεί με ένα μονοδιάστατο πίνακα που η τιμή που φιλάει το κάθε κελί είναι το "σύνολο" που ανήκει το κάθε στοιχειό.

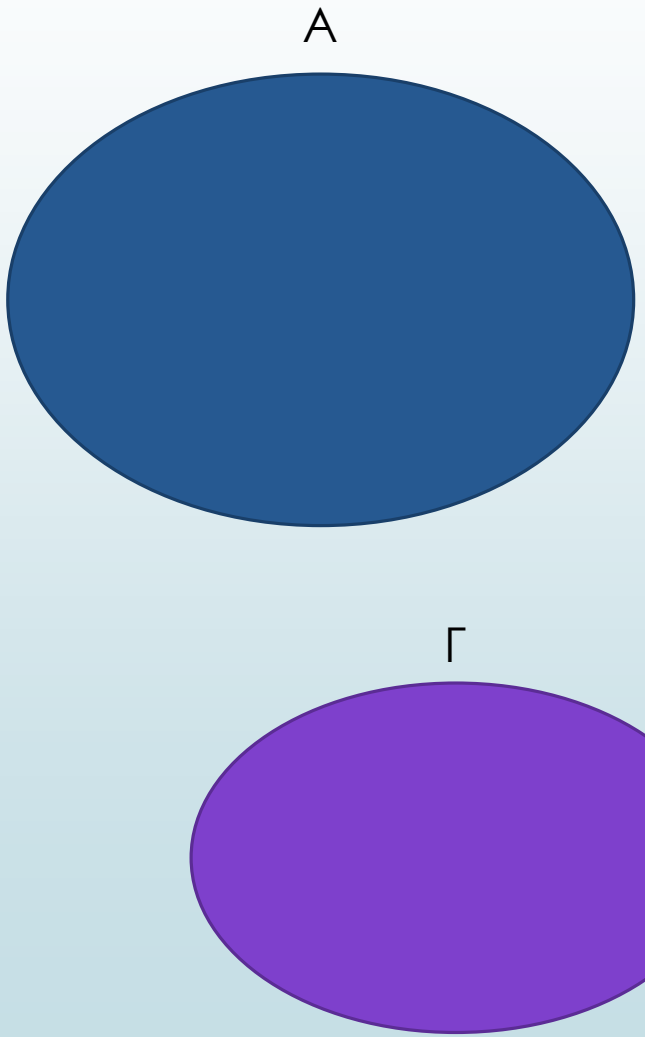
Union

Λόγο του ότι κάθε στοιχειό αρχικά βρίσκετε στο σύνολο που έχει το ίδιο όνομα με αυτό. Επιπλέων κάθε στοιχειό στον πινάκα δείχνει σε ποιο "σύνολο" βρίσκετε.

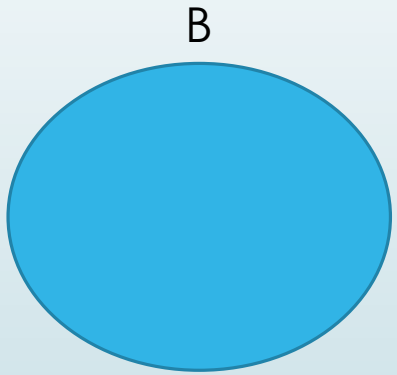
Όταν καλείτε το ερώτημα: «Το σύνολο που ανήκει το στοιχειό Α να ενωθεί με το σύνολο του στοιχειού Β»

Χρησιμοποιώντας την Find από το Α μπορούμε να **βρούμε το στοιχειό D που ανήκει στο σύνολο του με το ίδιο όνομα (Σύνολο D)** και να αλλάξουμε το την τιμή στην θέση D και να την κάνουμε ίσο με Β έτσι την επομένη φορά που θα καλέσουμε την Find από το Α, όταν φτάσει στο στοιχειό D θα συνεχίσει την πορεία του στην εύρεση του στοιχειού με το ίδιο όνομα με το σύνολο.

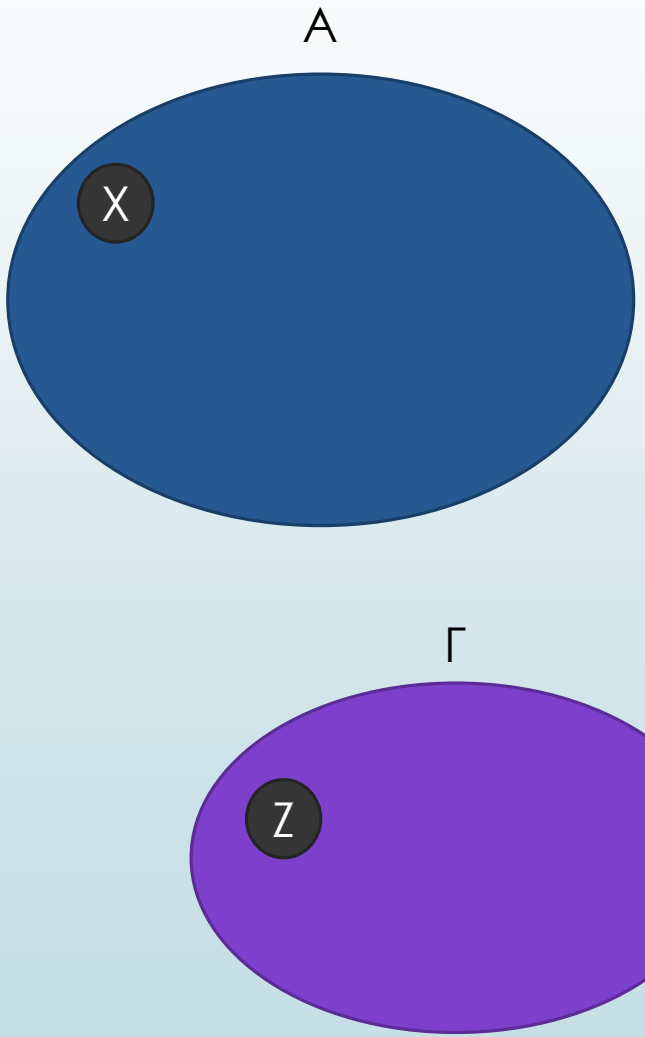
Visualization



Έχουμε 3 σύνολα (Α,Β,Γ) που ήδη μετά από κάποιες διεργασίες ήδη αποτελούνται από κάποια στοιχεία

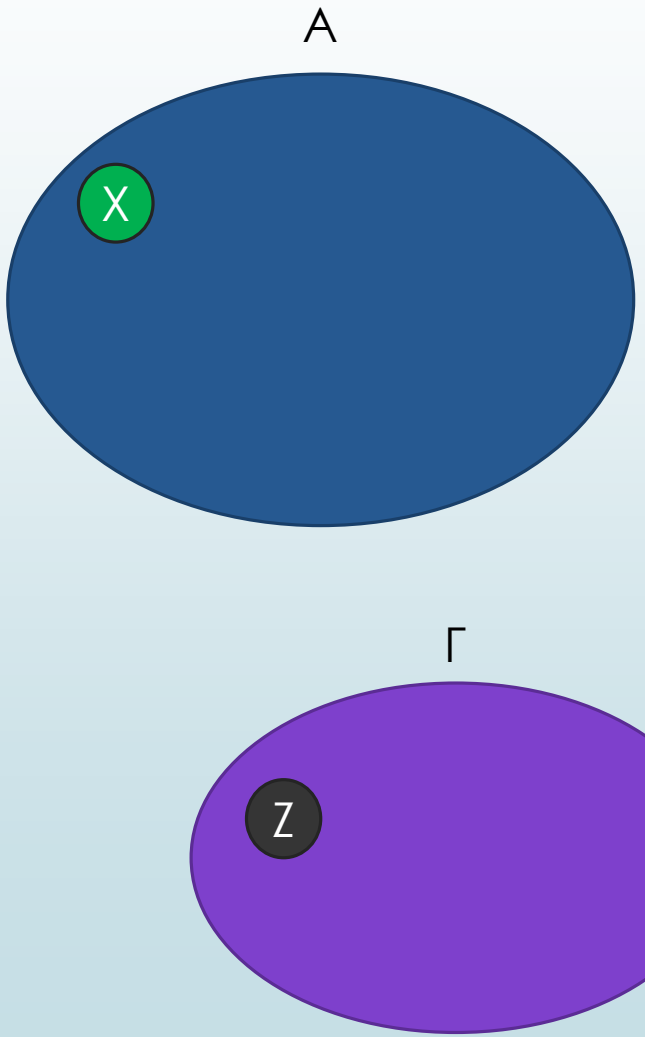


Visualization



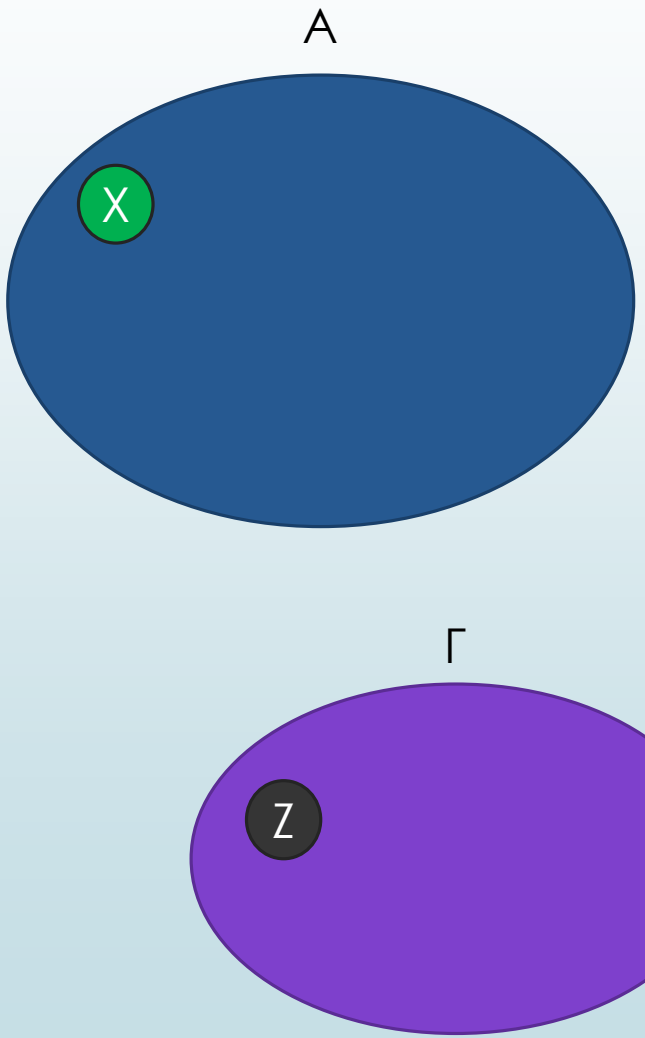
Έστω ότι το στοιχείο Χ ανήκει στο σύνολο Α
το στοιχείο Υ ανήκει στο σύνολο Β
και το στοιχείο Ζ στο σύνολο Γ

Visualization



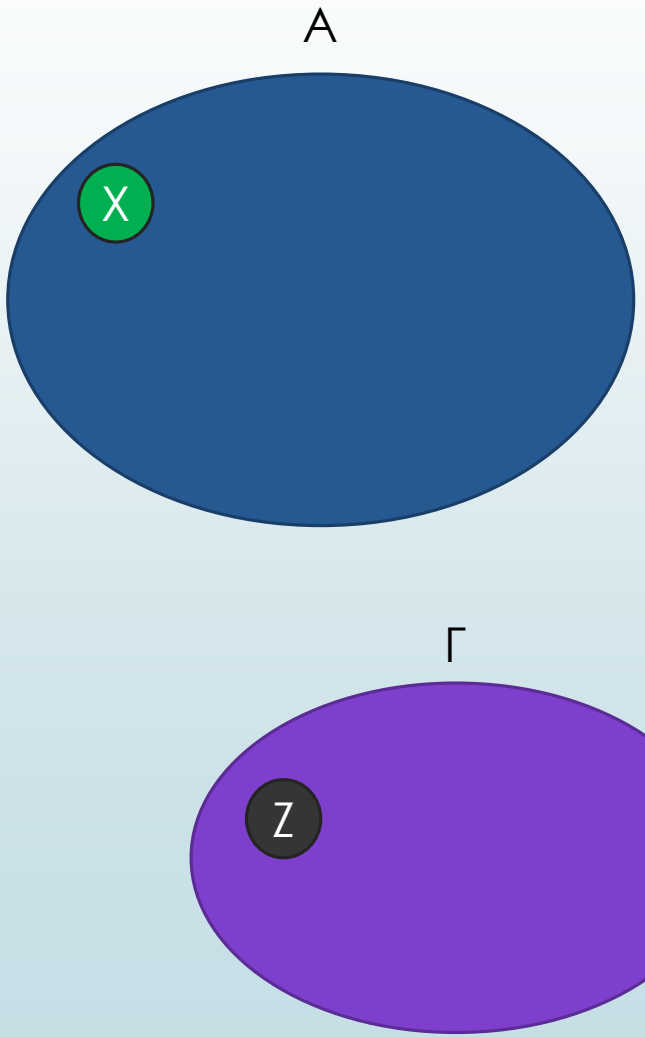
Και θέλω να ενώσω τα σύνολα που βρίσκονται τα στοιχεία X και Y

Visualization

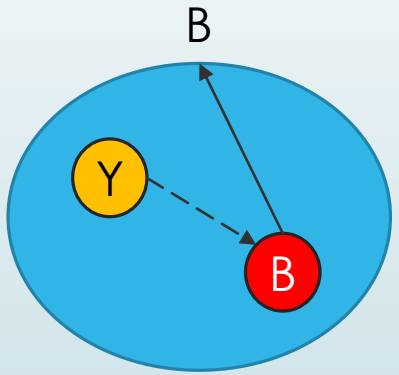


Καλώ την συνάρτηση Find για το στοιχείο Y

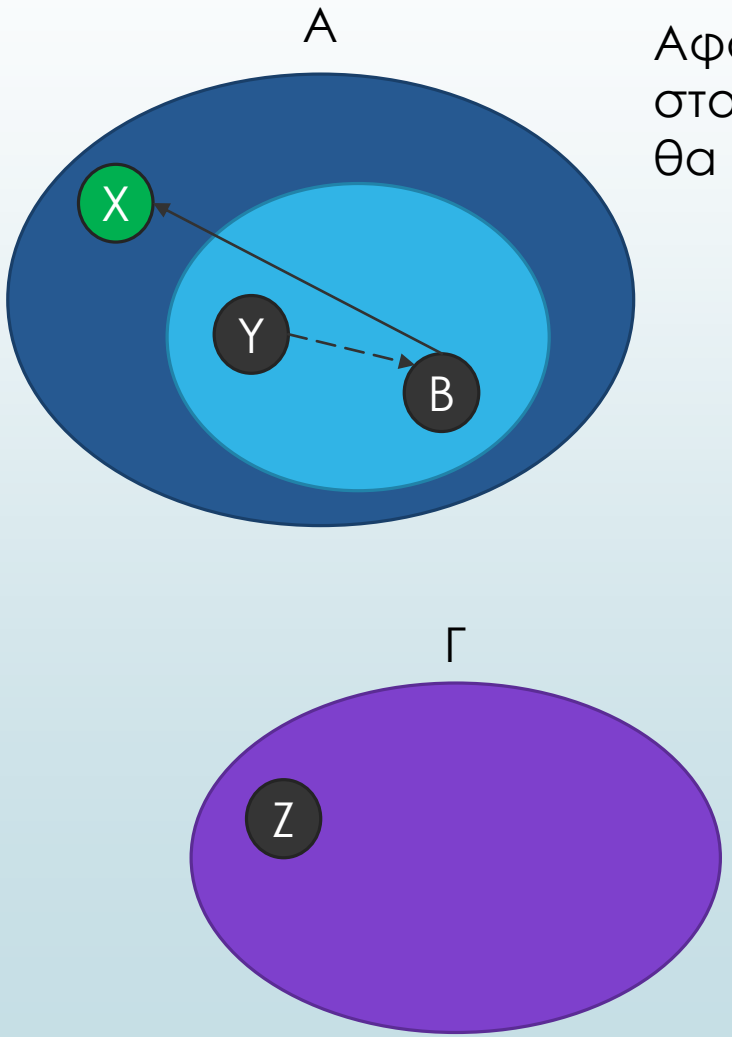
Visualization



Με την βοήθεια της Find βρίσκω το στοιχείο το οποίο έχει το ίδιο όνομα με το σύνολο



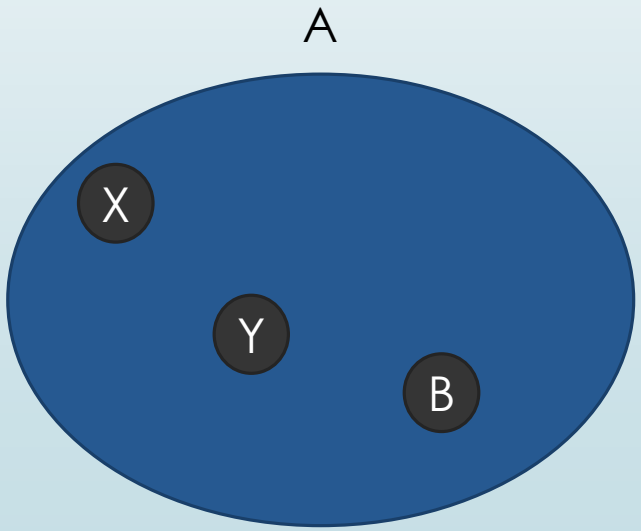
Visualization

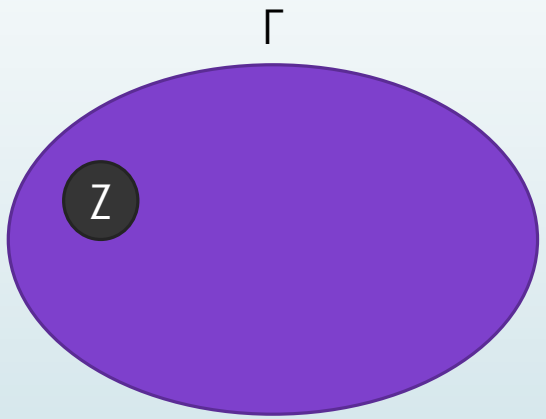


Αφού βρίσκω το στοιχείο το ενώνω με το στοιχείο στο άλλο σύνολο έτσι ώστε επόμενη φορά που θα καλέσω την Find να μου δήξει το άλλο σύνολο

Visualization

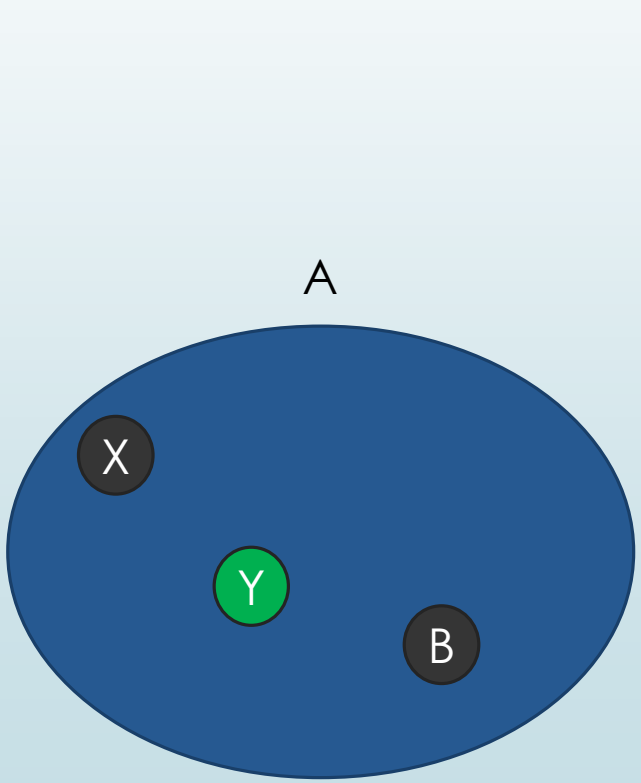
Έτσι χάνετε το σύνολο Β και παραμένει το σύνολο Α

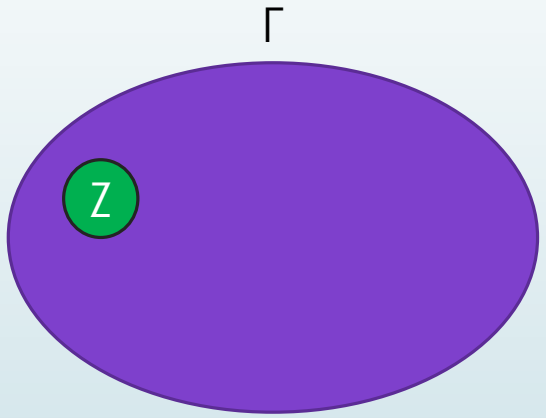




Visualization

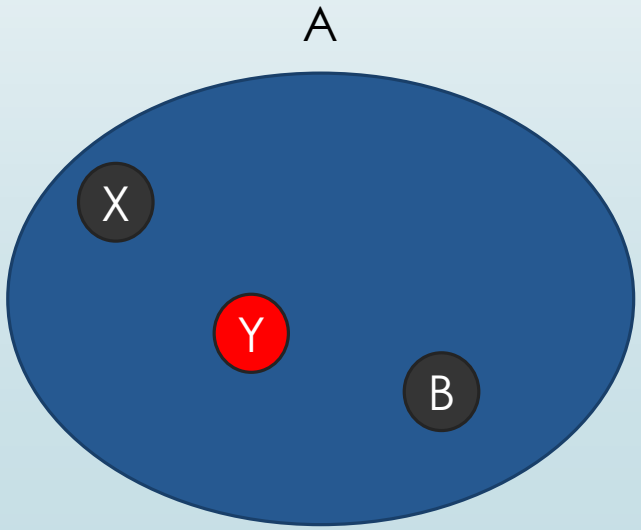
Και θέλω να ενώσω τα σύνολα που βρίσκονται τα στοιχεία Z και Y

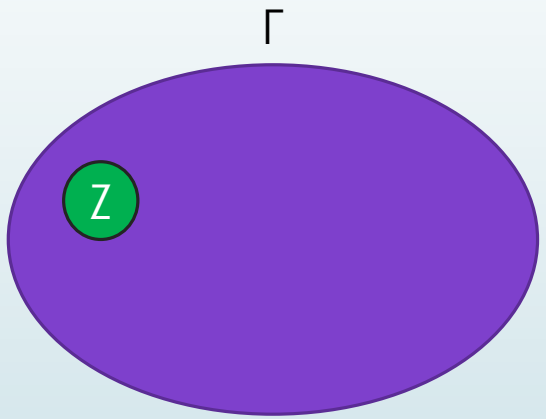




Visualization

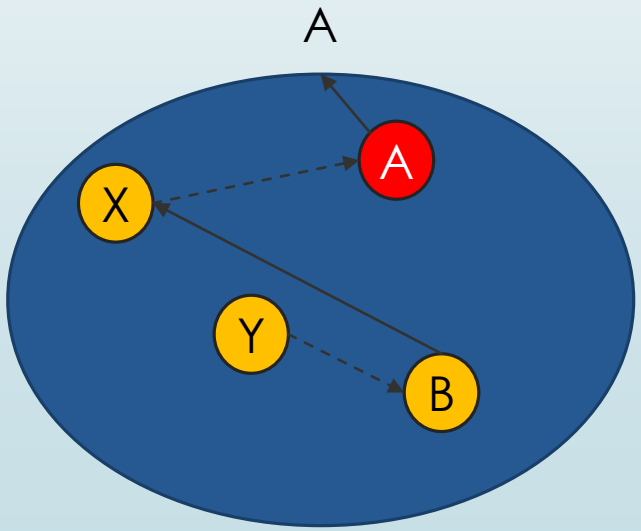
Καλώ την συνάρτηση Find για το στοιχείο Y

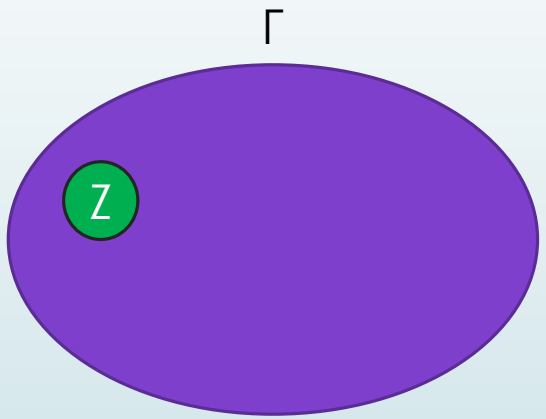




Visualization

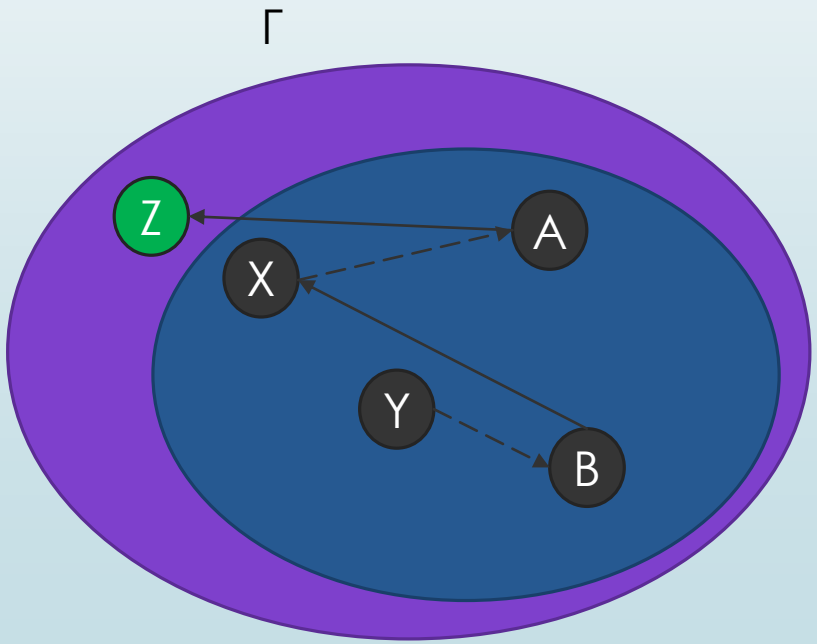
Με την βοήθεια της Find βρίσκω το στοιχείο το οποίο έχει το ίδιο όνομα με το σύνολο





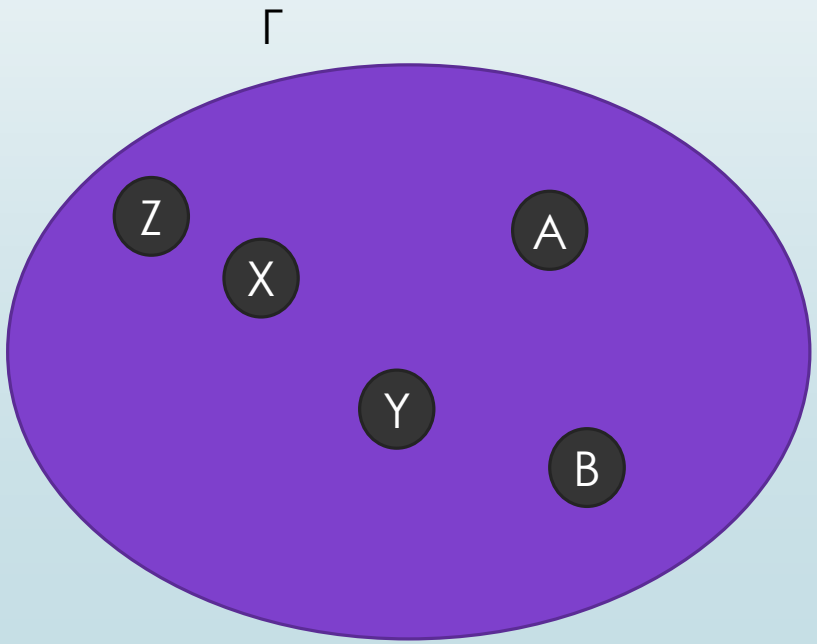
Visualization

Αφού βρίσκω το στοιχείο το ενώνω με το στοιχείο στο άλλο σύνολο έτσι ώστε επόμενη φορά που θα καλέσω την Find να μου δήξει το άλλο σύνολο



Visualization

Έτσι χάνετε το σύνολο Α και παραμένει το σύνολο Γ



Implementation

```
void onion(int x, int y){
	parent[find(y)]=x;
	return;
}
```

Optimization

Όπως κάποιος μπορεί να προσέξει αυτή η μέθοδος είναι χρονοβόρα και στο τέλος της ημέρας μπορεί να έχει γραμμική πολυπλοκότητα (O(N)) που δεν είναι επιθυμητή. Για αυτό θα ήταν καλό όταν ενώνουμε τα σύνολα να μην τα ενώνουμε σε ένα τυχαίο στοιχείο αλλά στο στοιχείο που ορίζει το σύνολο(το στοιχείο που έχει ίδιο όνομα με το σύνολο).

Implementation

```
void onion(int x, int y){
	parent[find(y)]=find(x);
	return;
}
```

Find

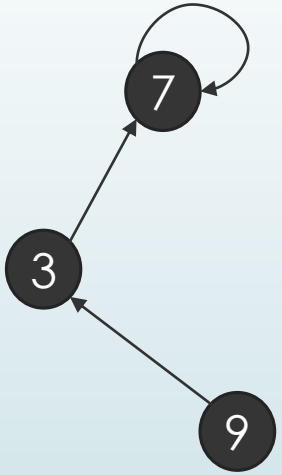
Λόγο του ότι κάθε στοιχειό αρχικά βρίσκετε στο σύνολο που έχει το ίδιο όνομα με αυτό. Επιπλέων κάθε στοιχειό στον πίνακα δείχνει σε ποιο "σύνολο" βρίσκετε.

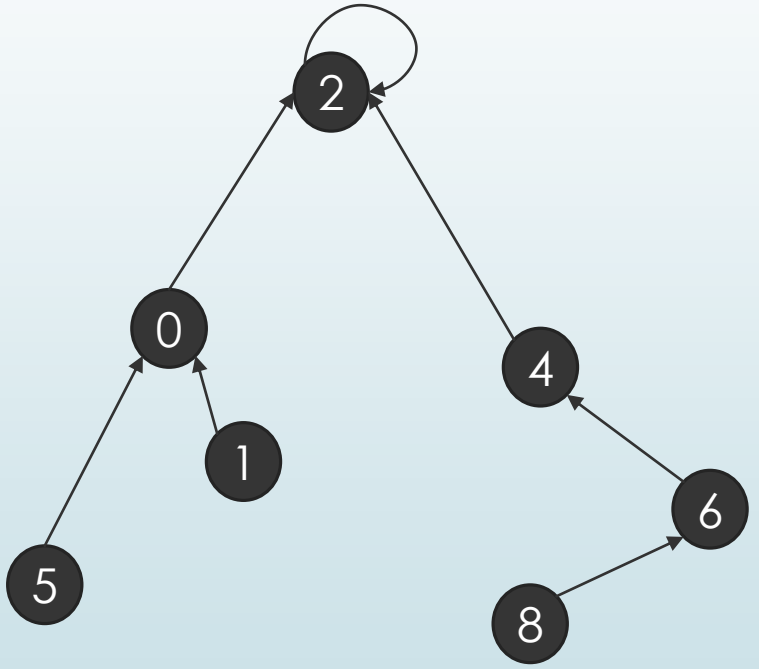
Όταν καλείτε το ερώτημα: «Σε ποιο σύνολο βρίσκετε το στοιχείο Α;»

Χρησιμοποιώντας αναδρομή και σε κάθε βήμα να πηγαίνουμε στο "σύνολο" που βρίσκετε (πατέρας) το κάθε στοιχείο μέχρι να βρούμε το στοιχείο που έχει ίδιο όνομα με το σύνολο που είναι αυτό που ψάχνουμε.

Visualization

Έστω ότι έχουμε τα εξής στοιχεία (0-9)
και τα αντίστοιχα σύνολα που δημιουργούνται
από τις σχέσεις τους(7 και 2)

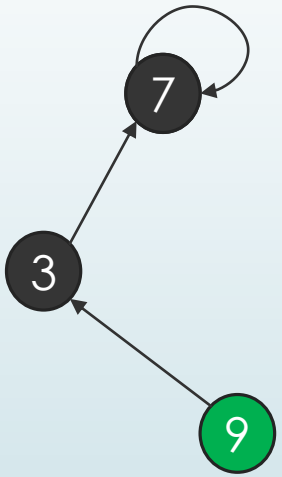


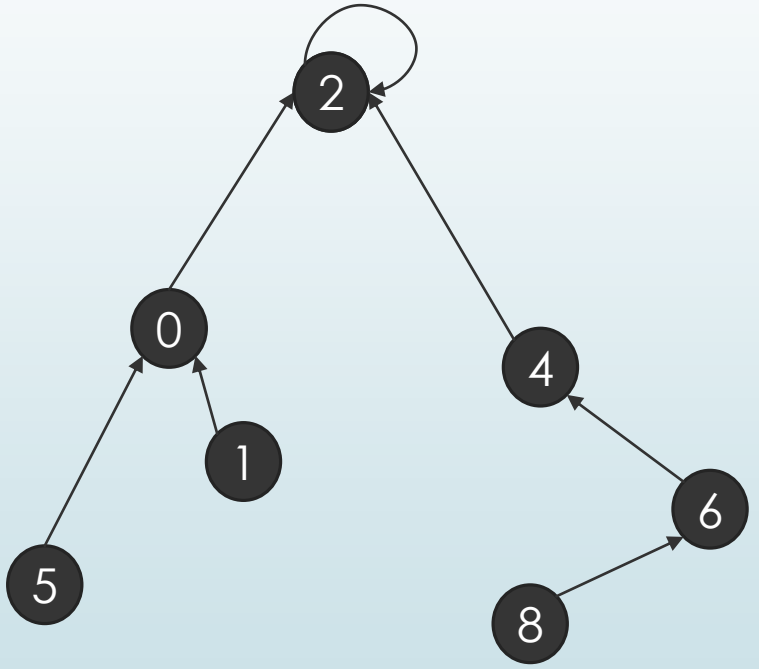


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Σε ποιο σύνολο βρίσκετε το στοιχείο 9;

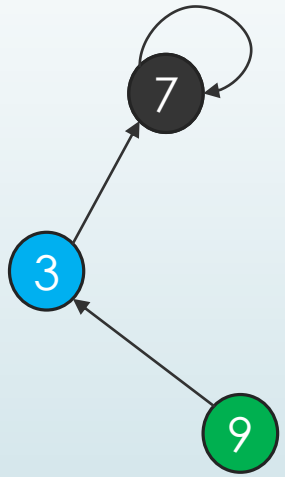


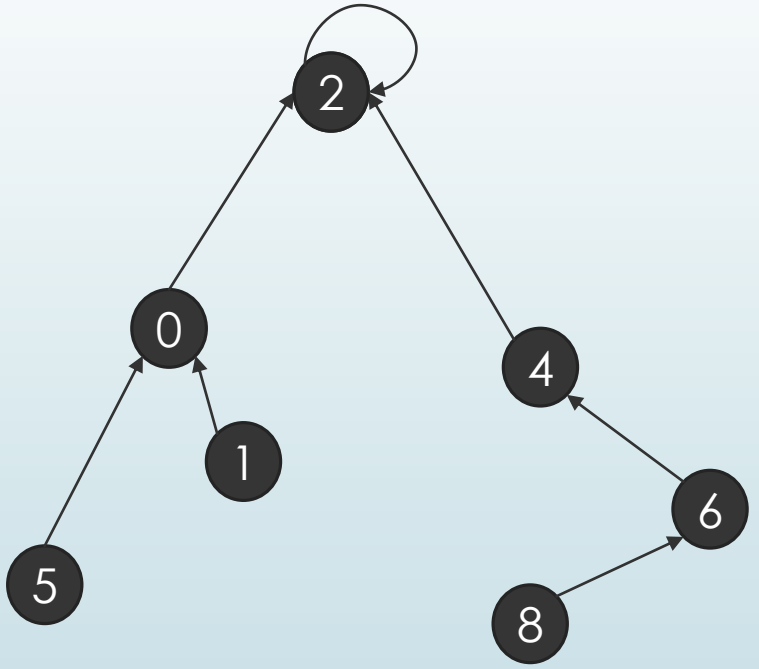


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Παρατηρούμε ότι είναι στο "σύνολο" 3

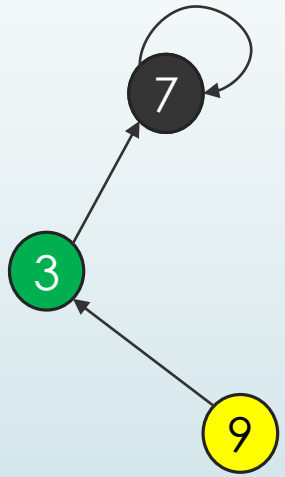


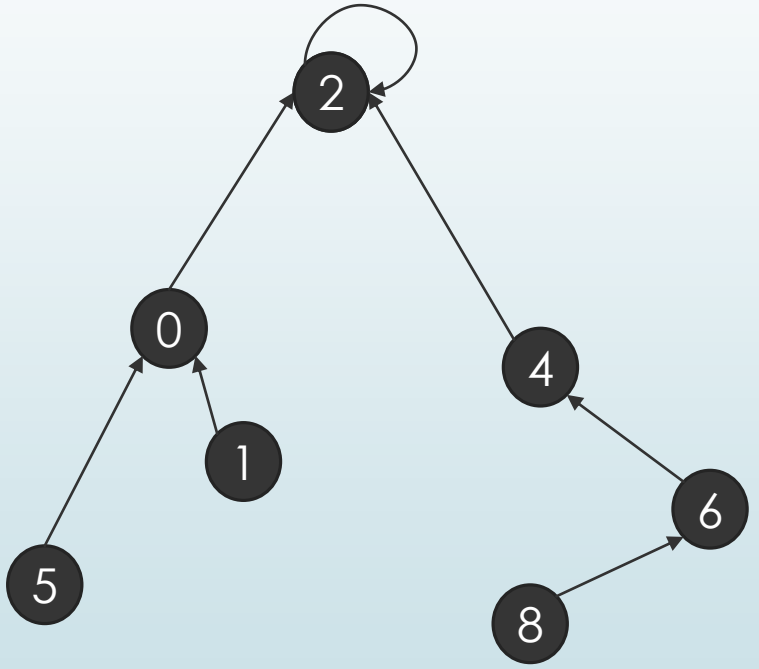


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Μετατοπιζόμαστε στο 3

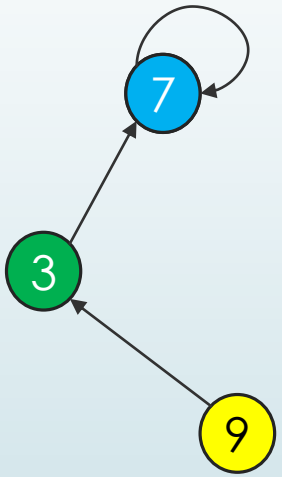


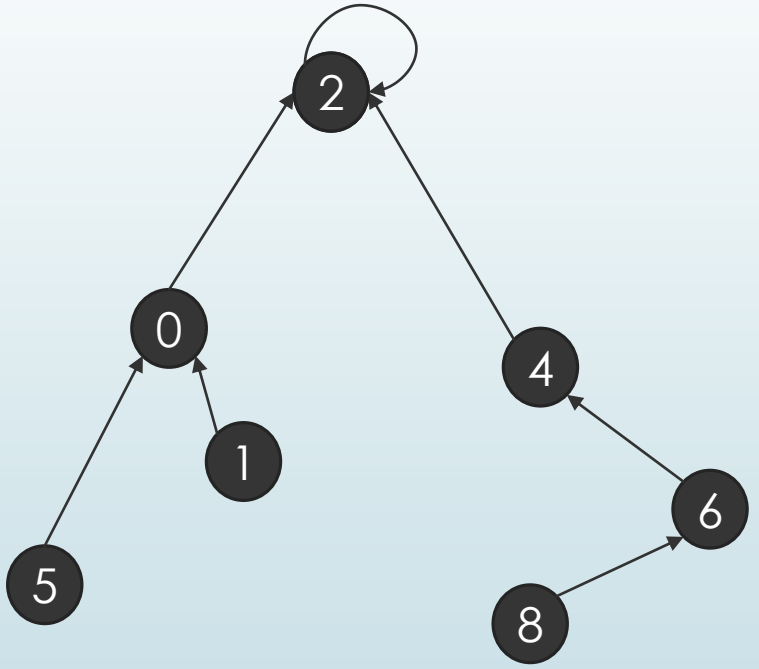


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Παρατηρούμε ότι είναι στο "σύνολο" 7

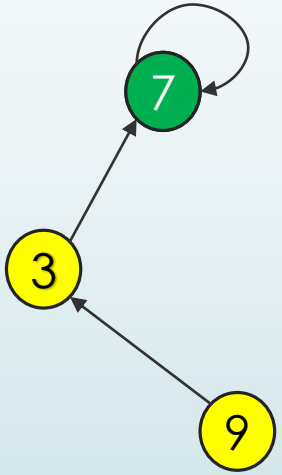


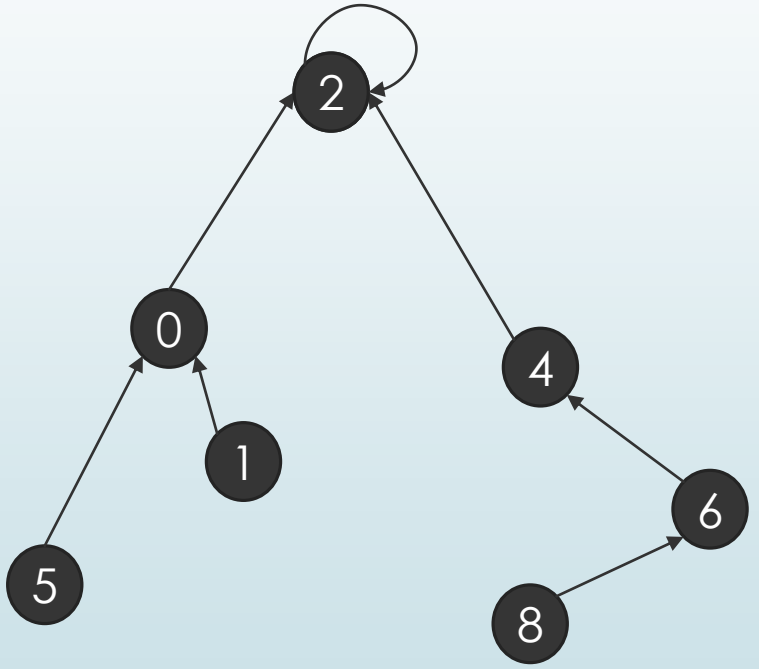


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Μετατοπιζόμαστε στο 7

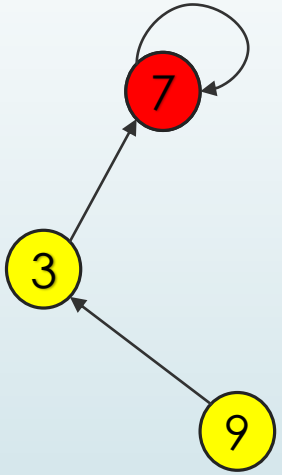


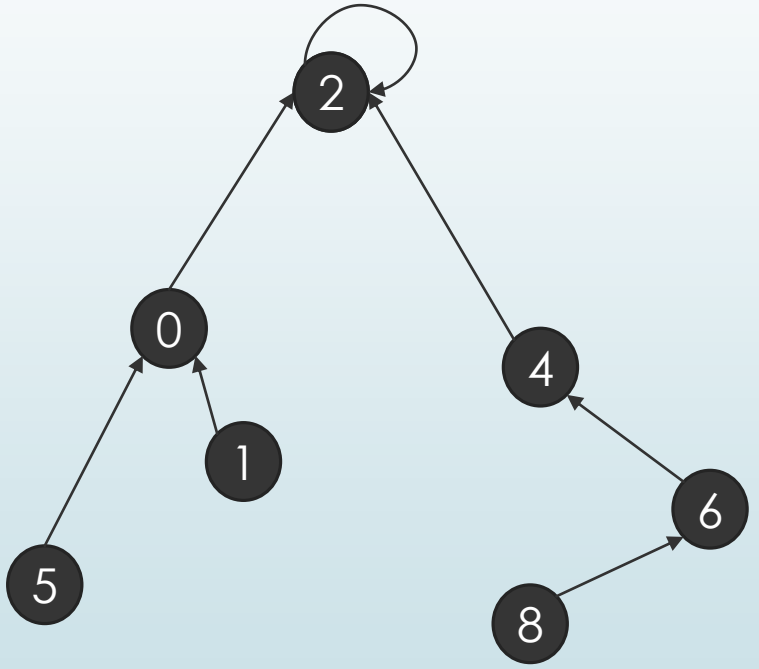


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Παρατηρούμε ότι είναι στο "σύνολο" 7 που έχει το **ίδιο όνομα** με το στοιχειό άρα το σύνολο που ψάχναμε είναι το 7

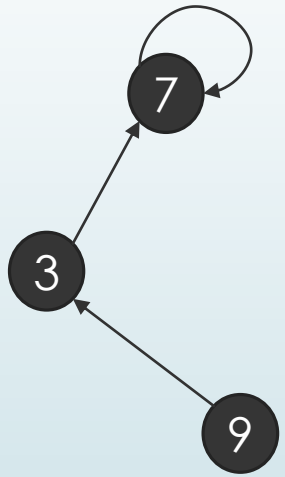


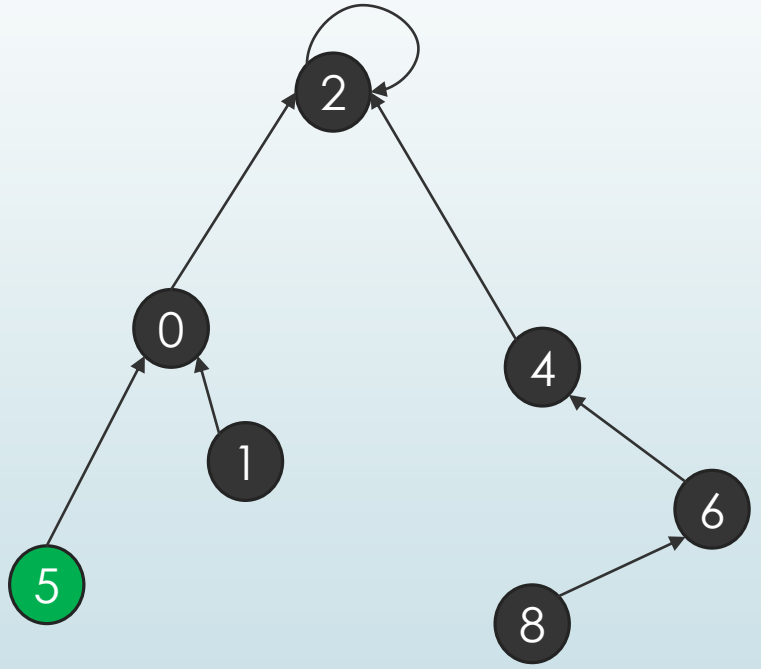


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Σε ποιο σύνολο βρίσκετε το στοιχείο 5;

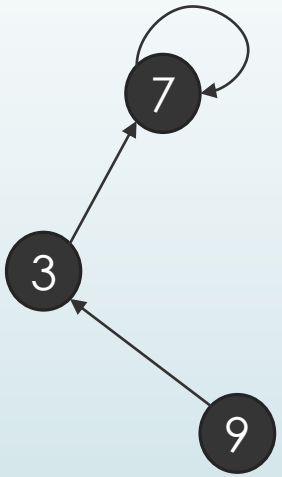


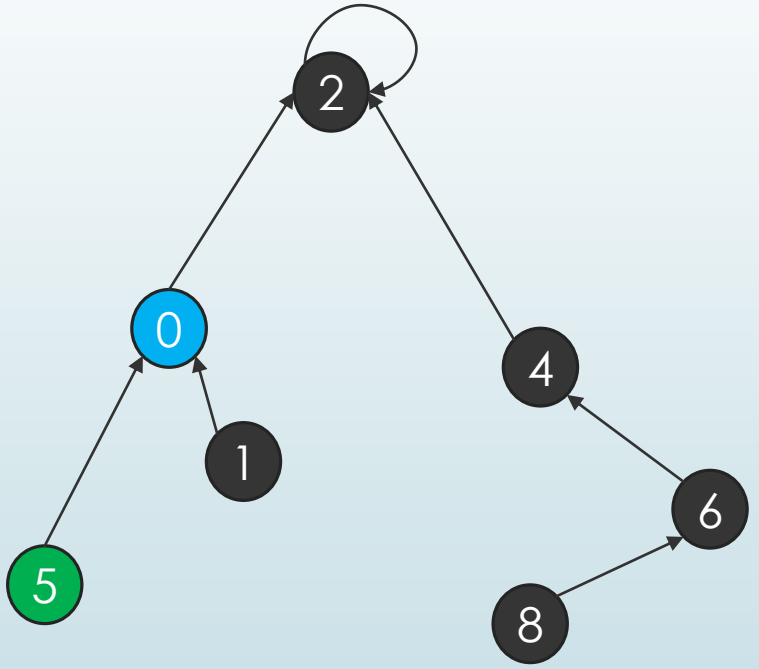


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Παρατηρούμε ότι είναι στο "σύνολο" 0

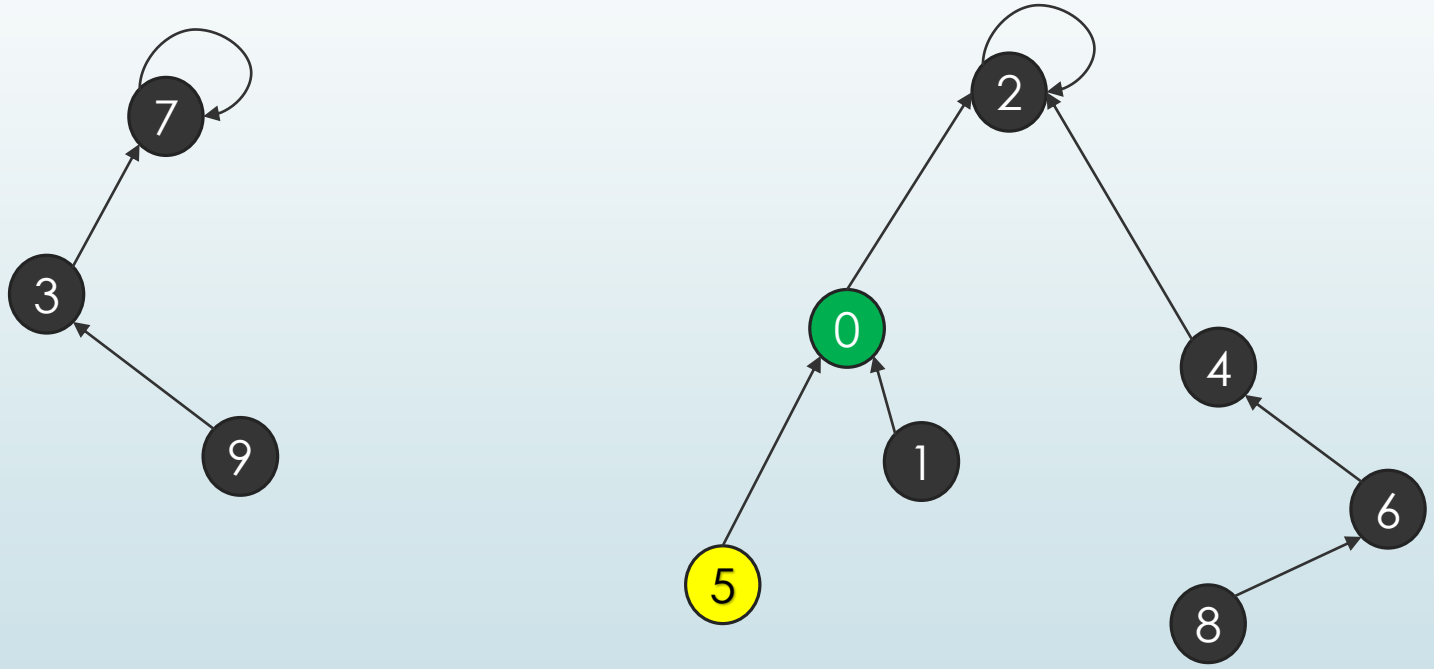




2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

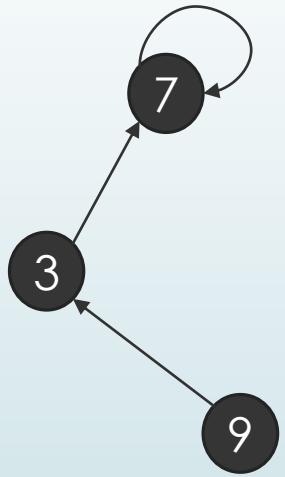
Μετατοπιζόμαστε στο 0

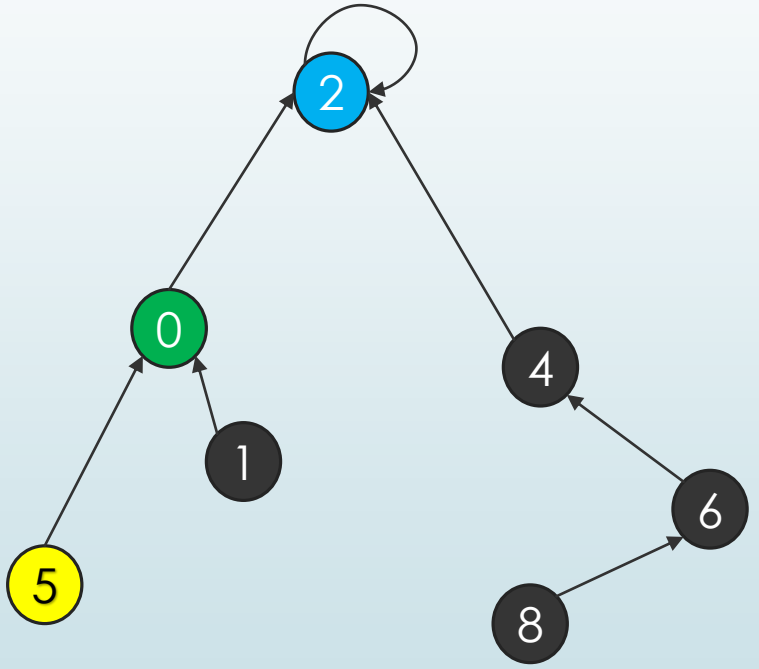


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Παρατηρούμε ότι είναι στο “σύνολο” 2

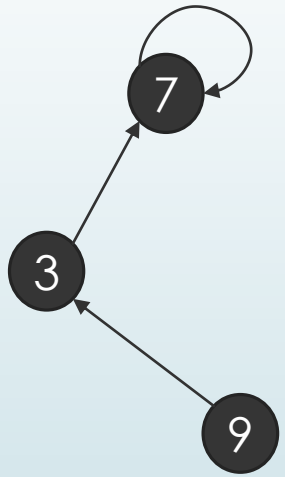


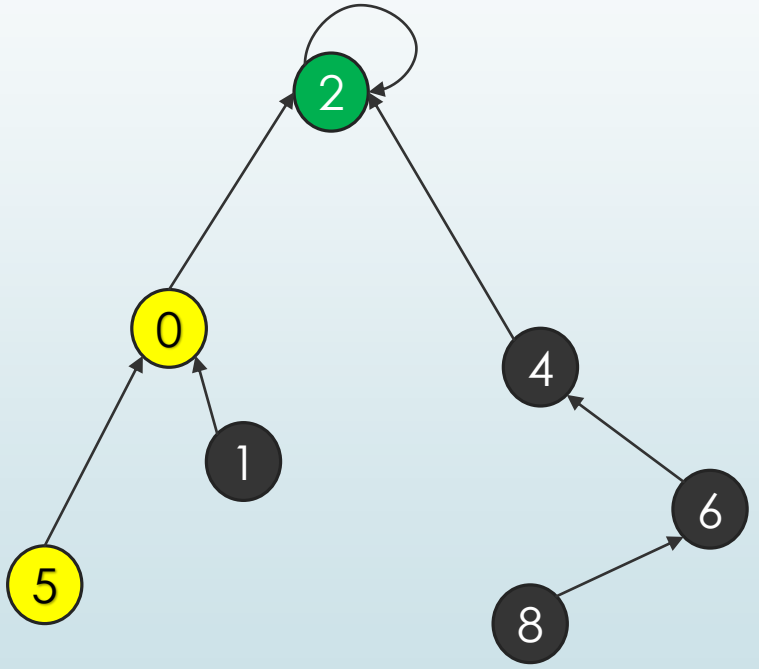


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Μετατοπιζόμαστε στο 2

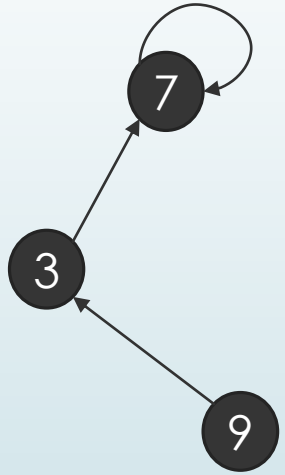


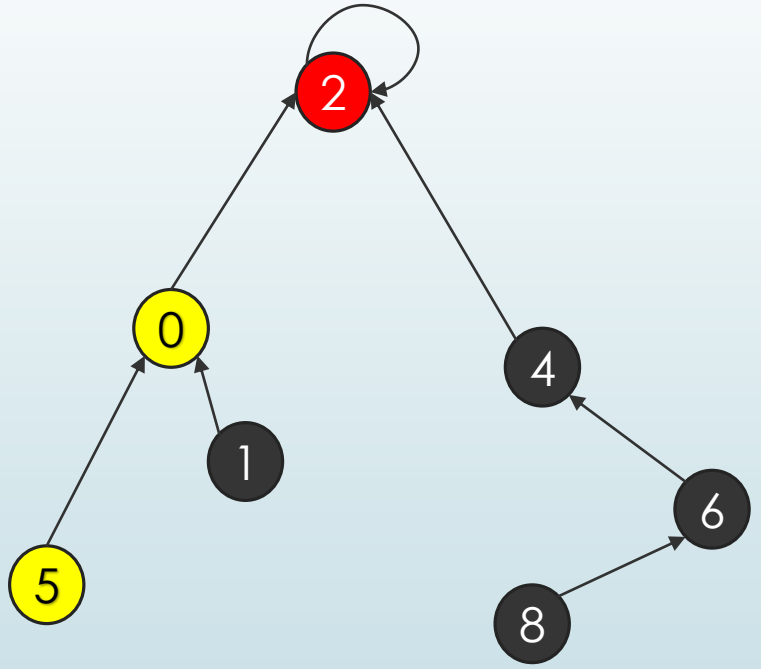


2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Visualization

Παρατηρούμε ότι είναι στο "σύνολο" 2 που έχει το **ίδιο όνομα** με το στοιχειό άρα το σύνολο που ψάχναμε είναι το 2





2	0	2	7	2	0	4	7	6	3
0	1	2	3	4	5	6	7	8	9

Implementation

```
int find(int x){
        if(parent[x]==x)
                return x;
        return find(parent[x]);
}
```

Optimization

Στη χειρότερη περίπτωση μπορούν να μας δοθούν τα δεδομένα με τέτοιο τρόπο έτσι ώστε στο τέλος της ημέρας να χρειάζεται γραμμικός χρόνος στην εύρεση του Συνόλου. Έτσι για να κάνουμε γρηγορότερη την διαδικασία όταν καλούμε την find στην επιστροφή να αποθηκεύουμε σε κάθε στοιχειό που περάσαμε το σύνολο που βρίσκετε.

Implementation

```
int find(int x){
	if(parent[x]==x)
		return x;
	parent[x]=find(parent[x]);
	return parent[x];
}
```

It's not at all important to get it right the first time. It's vitally important to get it right the last time.

adamos2468@gmail.com