

# Προχωρημένες Δομές Δεδομένων

Απρίλιος 2011

Γιάννης Χατζημύχος  
[feedward@gmail.com](mailto:feedward@gmail.com)

Προκατασκευαστικό Camp 23ου ΠΔΠ

# Προχωρημένες Δομές Δεδομένων

1) **Union-Find (Disjoint sets)**

2) **Hash tables** -- Βαλκανιάδες

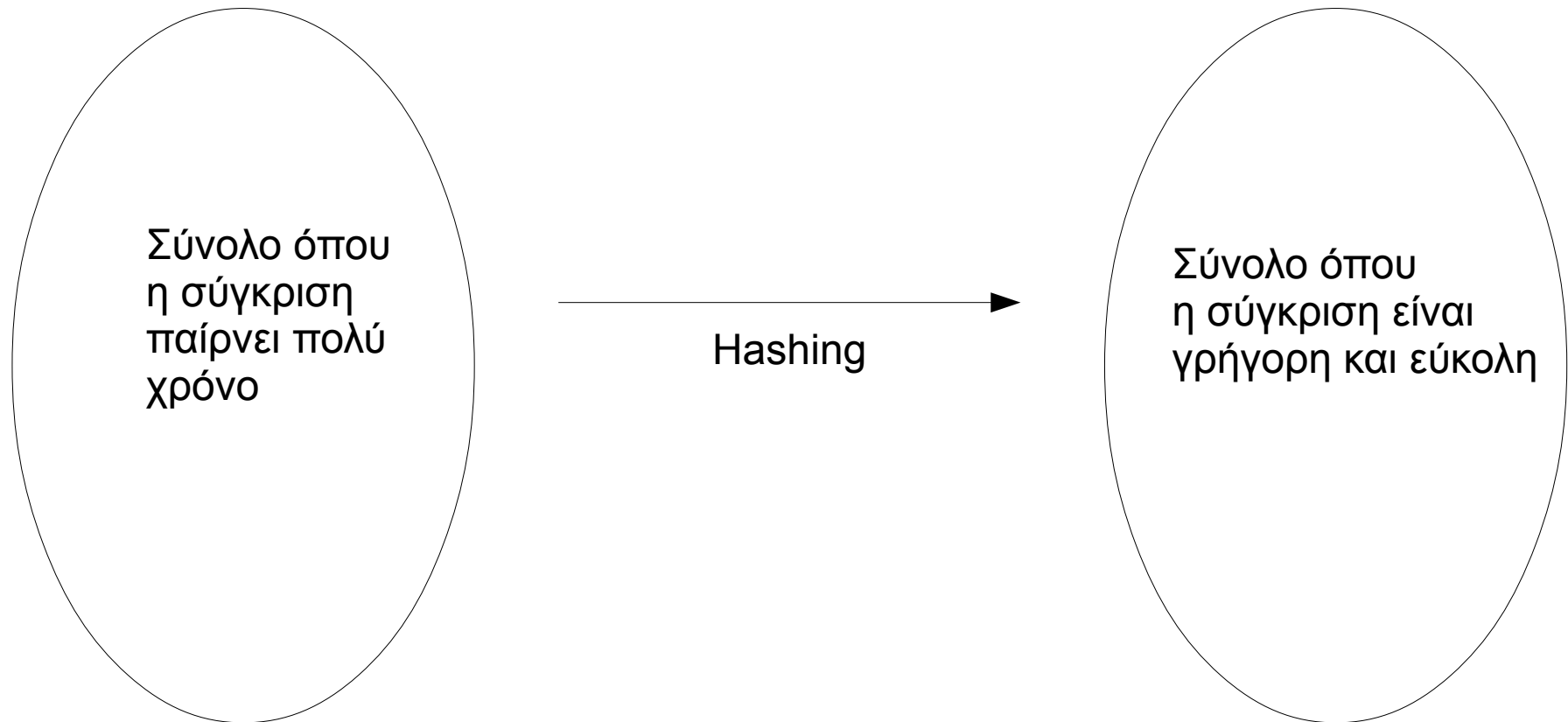
3) **Tries**

4) Δέντρα

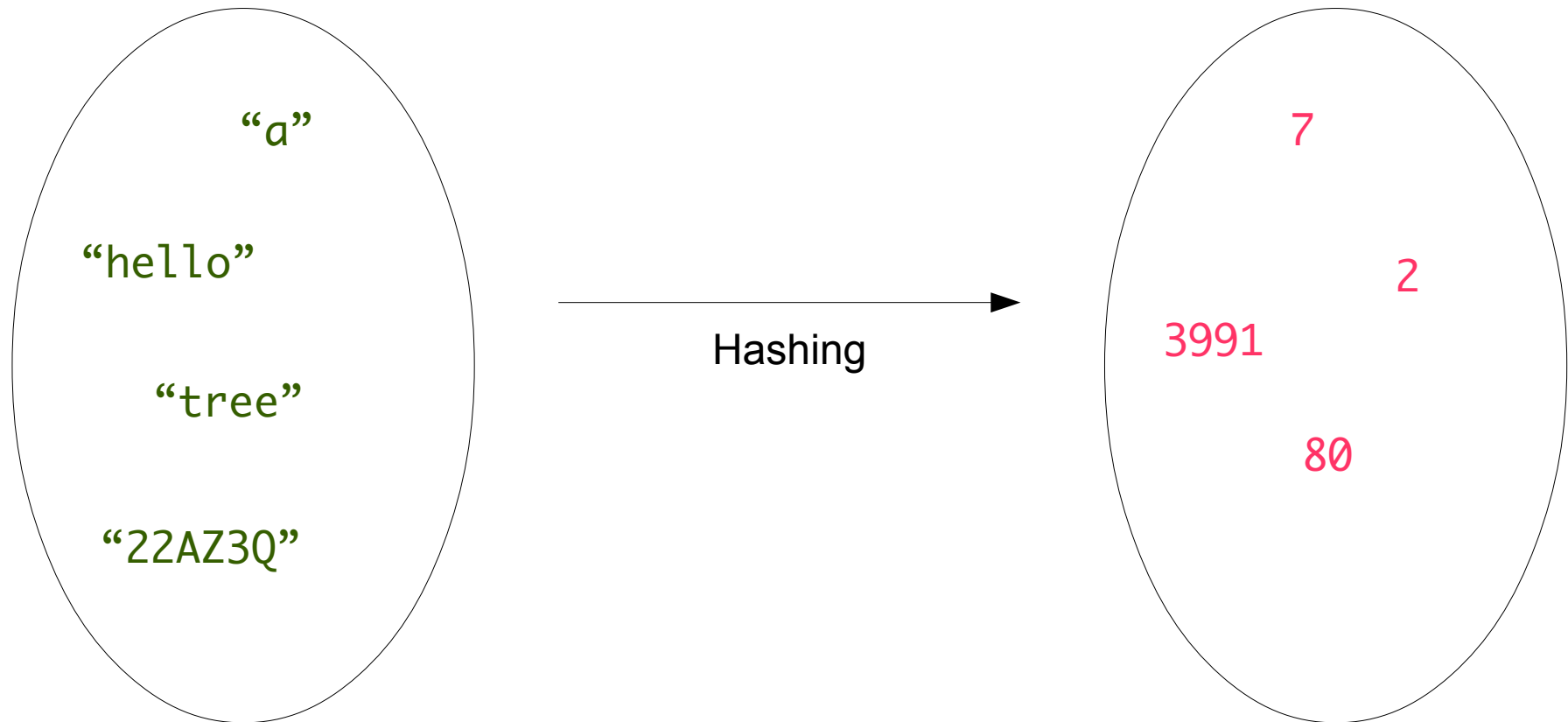
- Interval Trees
- **Binary Indexed Trees**
- Quad Trees -- Βαλκανιάδες

5) **Suffix Arrays** -- Βαλκανιάδες

# Hashing



# Hashing



# Hash function

- Μετατρέπει ένα στοιχείο του πρώτου συνόλου σε ένα στοιχείο του δεύτερου.
- Λειτουργεί πάντα με τον ίδιο τρόπο (όσες φορές και να δώσουμε ένα στοιχείο του πρώτου συνόλου θα μας επιστρέψει το ίδιο στοιχείο του δεύτερου)
- π.χ.

$H(\text{"hello"}) = 33$

$H(\text{"test"}) = 87$

$H([1, 9, 4, 3]) = 11$

# Collisions

- Δύο στοιχεία του πρώτου συνόλου δείχνουν στο ίδιο δεύτερο σύνολο!
- π.χ.
  - $H(\text{"hello"}) = 33$
  - $H(\text{"bye"}) = 33$
- Θέλουμε να έχουμε όσο το δυνατόν λιγότερα collisions. Αν είναι δυνατόν, να μην έχουμε κανένα.
- Πρέπει να επιλέξουμε ένα καλό hash function.

# Perfect Hashing

- Κάθε στοιχείο του δεύτερου συνόλου αντιστοιχεί σε μοναδικό στοιχείο του πρώτου συνόλου.
- Είναι 1-1.
- Έχει χρόνο εκτέλεσης  $O(1)$ .

# Παράδειγμα

- Χαζό hash function που μετατρέπει μια συμβολοσειρά σε έναν αριθμό:

```
int hash(char *word, int length) {  
    int i, H = 0;  
    for (i = 0; i < length; i++) {  
        H += word[i];  
    }  
    return H;  
}
```


- Έχει πολλά collisions:
  - $\text{hash}(\text{"AB"}) = 65 + 66 = 131$
  - $\text{hash}(\text{"BA"}) = 66 + 65 = 131$



# Παράδειγμα (linear hashing)

- Γραμμικό hash function: θεωρεί μια συμβολοσειρά ως αριθμό εκφρασμένο στο  $|\Sigma|$ -ικο σύστημα και τον μετατρέπει στο δεκαδικό

```
int hash(char *word, int length) {  
    int i, H = 0, base = 1;  
    for (i = 0; i < H; i++) {  
        H = H[i] + s[i]*base;  
        base *= 256;  
    }  
    return H;  
}
```



Το πλήθος των διαφορετικών  
συμβόλων

# Παράδειγμα (linear hashing)

- **Πλεονέκτημα:**

Δεν έχει καθόλου collisions

- **Μειονεκτήματα:**

Μπορεί να δώσει πολύ μεγάλα αποτελέσματα που δεν χωράνε ούτε σε ακεραίους 64bit. Έτσι πρέπει να χρησιμοποιήσουμε αριθμητική υπολοίπων για να περιορίσουμε το αποτέλεσμα. Αυτό οδηγεί σε:

- Μερικά collisions
- Αύξηση του χρόνου εκτέλεσης

# Double hashing

- Για να μειώσουμε ακόμα περισσότερο τα collisions μπορούμε να συνδυάσουμε δύο μεθόδους hashing:

$$h_1(\text{"hello"}) = 32$$

$$h_2(\text{"hello"}) = 991$$

$$H(\text{"hello"}) = \langle h_1(\text{"hello"}), h_2(\text{"hello"}) \rangle = \langle 32, 991 \rangle$$