Cuckoo Hashing

- Αλγόριθμοι και Πολυπλοκότητα
- Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
- Εθνικό Μετσόβιο Πολυτεχνείο

Πρόβλημα (ADT) Λεξικού

- Δυναμικά μεταβαλλόμενη συλλογή αντικειμένων που αναγνωρίζονται με «κλειδί» (π.χ. κατάλογοι, πίνακες ΒΔ)

- Λεξικό: συλλογή αντικειμένων με μοναδικό «κλειδί»
  - «Κλειδί»: αριθμός ή τύπος δεδομένων με ολική διάταξη
  - Γενίκευση και για μη-μοναδικά κλειδιά

- ADT λεξικού υποστηρίζει ακολουθίες λειτουργιών:
  - Αναζήτηση στοιχείου με κλειδί x
    - member(x): ελέγχει ύπαρξη στοιχείου με κλειδί x
    - search(x): επιστρέφει δείκτη σε θέσεις x
  - Εισαγωγή στοιχείου με κλειδί x
  - Διαγραφή στοιχείου με κλειδί x

Υλοποιήσεις Λεξικού

- Μη-ταξινομημένη διασυνδεδεμένη λίστα:
  - Εισαγωγή: O(1)
  - Αναζήτηση / τυχαία διαγραφή: O(n)
  - Κατάλληλη όταν συχνές εισαγωγές, σπάνιες αναζητήσεις / διαγραφές μεμονωμένες ή στο τέλος (π.χ. log file)

- Ταξινομημένος πίνακας:
  - (Δυαδική) αναζήτηση: O(log n)
  - Στατική συλλογή: «εισαγωγή» O(log n) / στοιχείο
    - Χρόνος ταξινόμησης: O(n log n)
  - Δυναμική συλλογή: εισαγωγή / διαγραφή O(n)
  - Κατάλληλη για συχνές αναζητήσεις όταν τα δεδομένα μεταβάλλονται σπάνια (π.χ. αγγλο-ελληνικό λεξικό)

Υλοποιήσεις Λεξικού

- Ζυγισμένο (δυαδικό) δέντρο αναζήτησης:
  - Αναζήτηση / εισαγωγή / διαγραφή: O(log n)
  - Μέγιστο / ελάχιστο / προηγούμενο / επόμενο / k-οστό: O(log n)
  - Range queries σε γραμμικό χρόνο
  - Πλήρως δυναμική – επιπλέον χώρος για δείκτες
- Πίνακας κατακερματισμού (hashing)
  - Υπάρχουν πολλοί αλγόριθμοι, όμως θέλουμε κάτι πολύ γρήγορο
  - Ιδανικά όλα σε O(1)

ΠΩΣ ;;

Αλγόριθμοι Hashing

- Hashing with Chaining
  - Lists: εισαγωγή σε O(1), αναζήτηση και διαγραφή: O(n)
  - Balanced Trees: εισαγωγή, αναζήτηση και διαγραφή σε O(logn)
- Perfect Hash Function
  - Hash Function χωρίς conflicts
  - Πολύ περίπλοκη υλοποίηση
- Cuckoo Hashing
  - Αναζήτηση και διαγραφή σε O(1)
  - Εισαγωγή σε amortized, expected O(1)

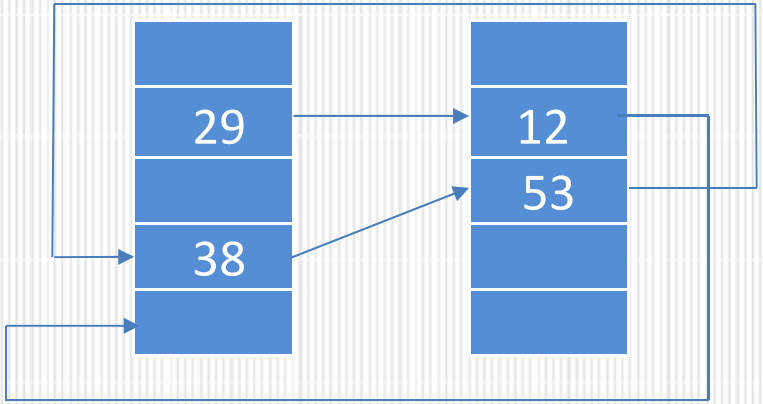
Cuckoo Hashing

- Δύο πίνακες μεγέθους r και δύο hash functions $h_1(x)$, $h_2(x)$
  - Επιστέφουν «random» hash values στο {1, …, r}
  - Έχουμε δύο πιθανές θέσεις για κάθε στοιχείο
  - Αποθηκεύουμε το πολύ ένα στοιχείο σε κάθε θέση
- Διαγραφή / Αναζήτηση: O(1)
  - Έχουμε να κοιτάξουμε μόνο σε δύο θέσεις
- Εισαγωγή σε amortized O(1)
  - Θέλουμε να τοποθετήσουμε στη δομή το στοιχείο x
  - Ελέγχουμε τη θέση $h_1(x)$
  - Αν είναι άδεια, βάζουμε το x και επιστρέφουμε
  - Αλλιώς βάζουμε το x και «ξεσπιτώνουμε» το στοιχείο y που υπήρχε εκεί
  - Αυτό προορίζεται τώρα για τη θέση $h_1(y)$ ή $h_2(y)$, ανάλογα με το που βρισκόταν πριν κοκ

Cuckoo Hashing

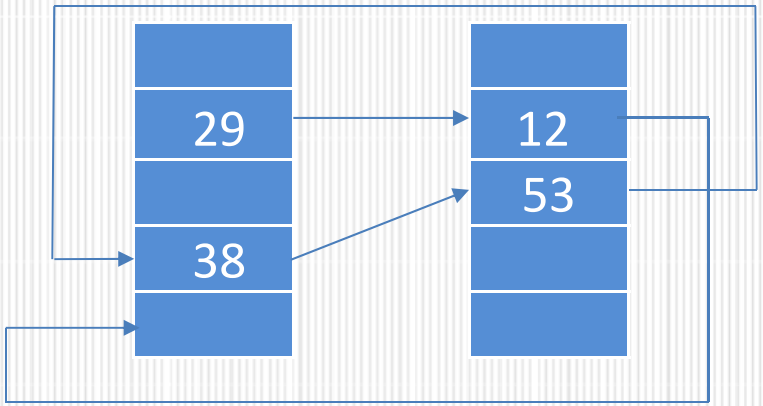
- Εισαγωγή
  - Οι αλλαγές θέσεων συνεχίζονται μέχρι να βρεθεί κενή θέση ή μέχρι να γίνουν n προσπάθειες
  - Αν η εισαγωγή αποτύχει, τότε κάνουμε rehashing
  - Επιλέγονται νέες hash functions και δομείται ξανά όλος ο πίνακας
  - Hash functions: πολυωνυμικές συναρτήσεις με τυχαίους συντελεστές

- Η εισαγωγή αποτυγχάνει αν προκύψει κύκλος δηλαδή αν επισκεφτούμε ξανά την ίδια θέση με το ίδιο στοιχείο π.χ. 38, 53
  => rehashing



Εισαγωγή

```
void insert (x) {
    if (T[h1(x)] == x or T[h2(x)] == x)
        return;
    pos = h1(x);
    for (i=0; i<n; i++) {
        if (T[pos] == NULL) {T[pos] = x; return;}
        swap(x, T[pos]);
        if (pos = h1(x)) pos=h2(x); else pos=h1(x);}
   rehash();
   insert(x);
}
```



Παράδειγμα

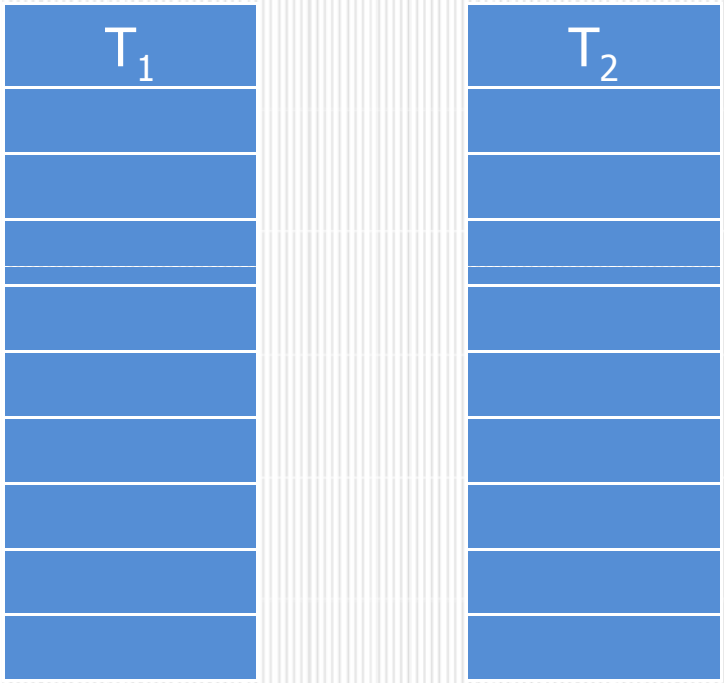
- Hash tables 8 θέσεων T_1, T_2
- $h_1(x) = x \bmod 8$, $h_2(x) = x \bmod 7$
- Θέλουμε να εισάγουμε τα στοιχεία: {11, 50, 47, 75, 39, 51, 106, 162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

Παράδειγμα

- {11, 50, 47, 75, 39, 51, 106, 162}

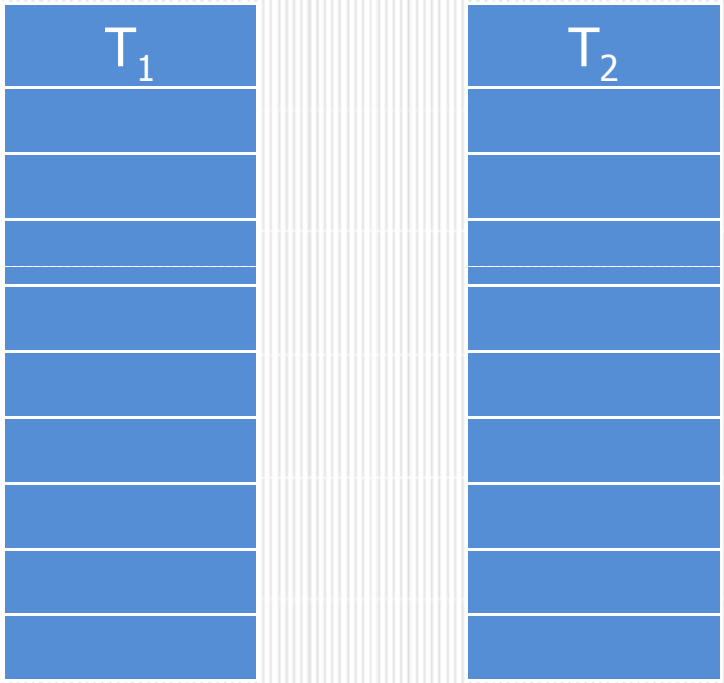
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {11, 50, 47, 75, 39, 51, 106, 162}

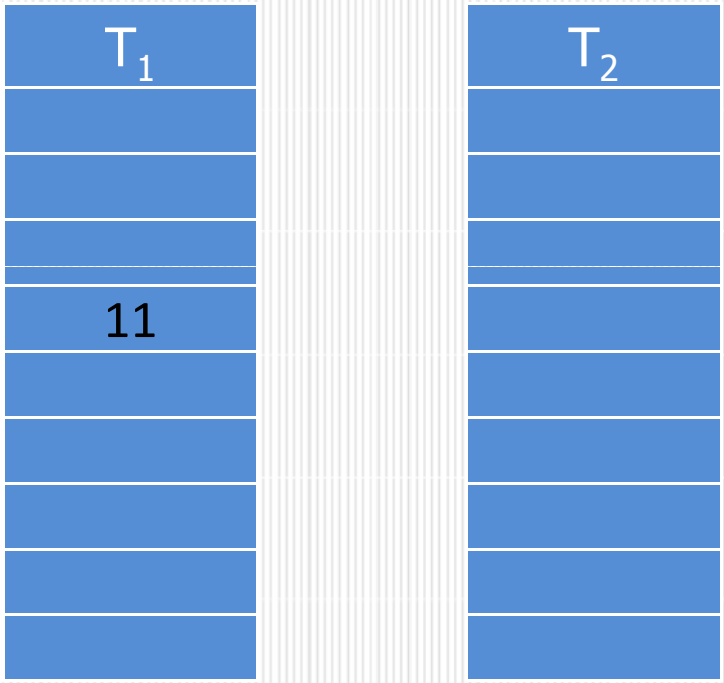
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {50, 47, 75, 39, 51, 106, 162}

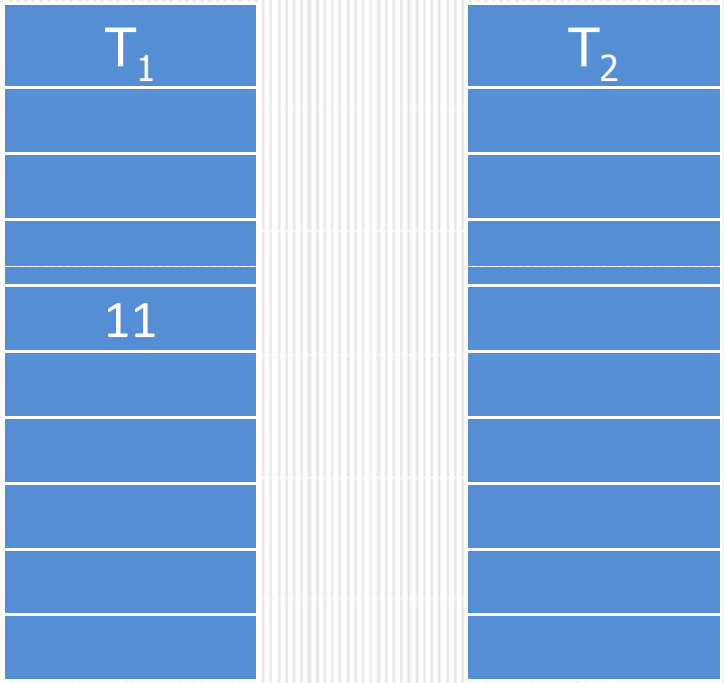
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {50, 47, 75, 39, 51, 106, 162}

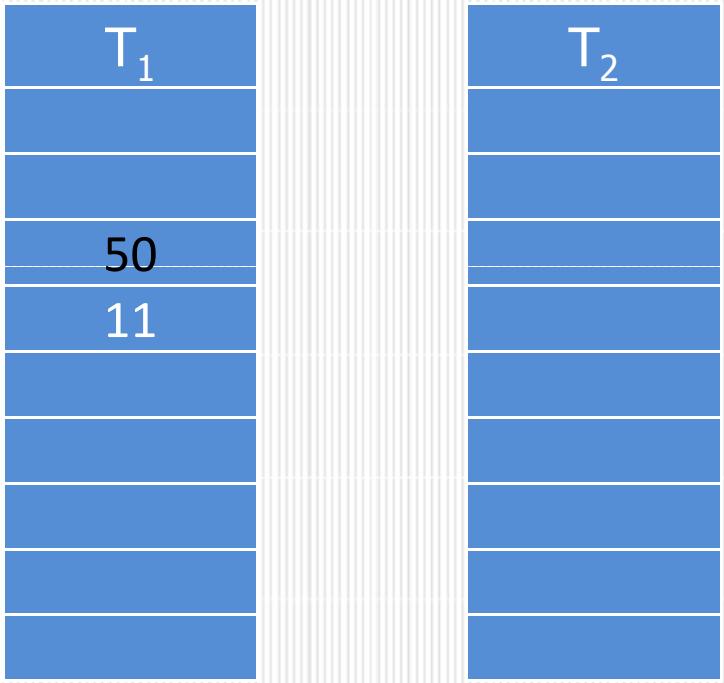
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {47, 75, 39, 51, 106, 162}

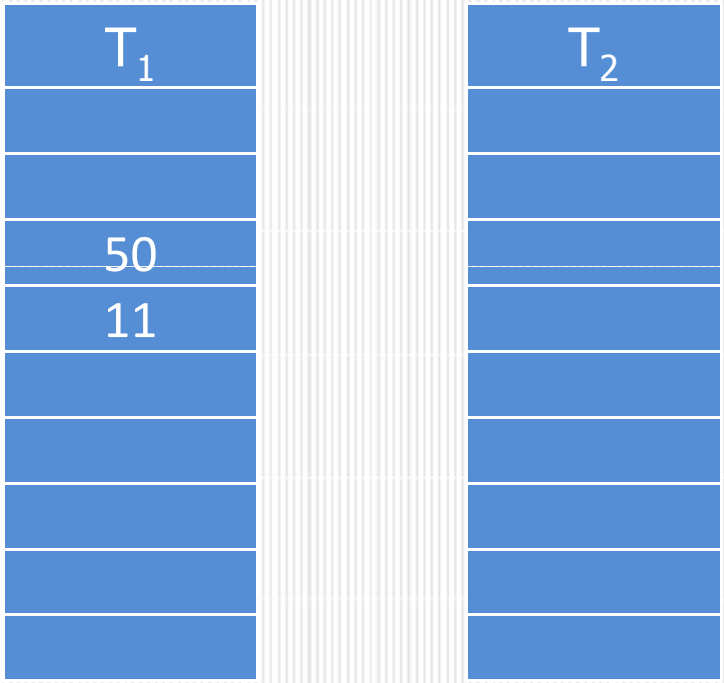
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {47, 75, 39, 51, 106, 162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {75, 39, 51, 106, 162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

T_1
50
11
47

T_2

Παράδειγμα

- {75, 39, 51, 106, 162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

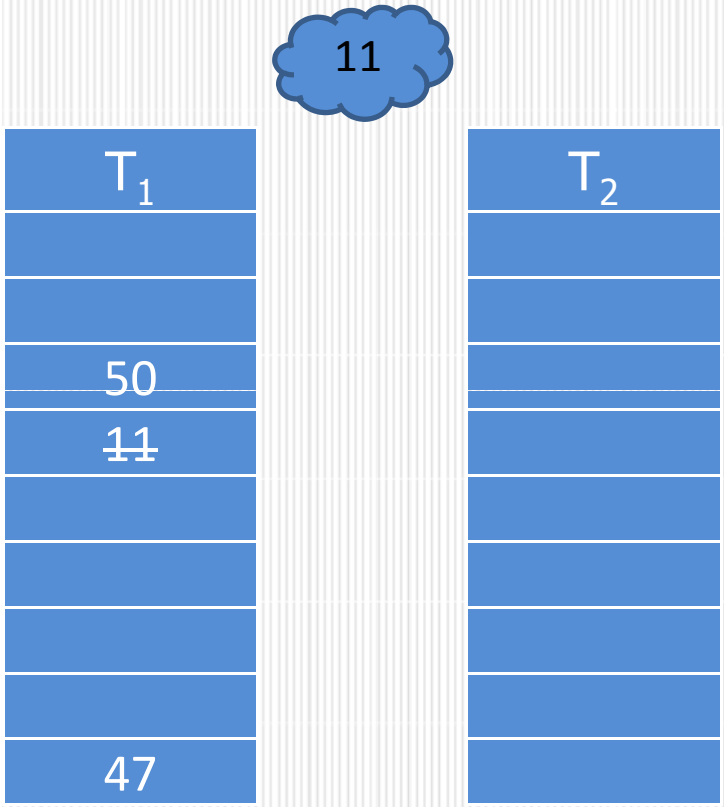
T_1
50
11
47

T_2

Παράδειγμα

- {75, 39, 51, 106, 162}

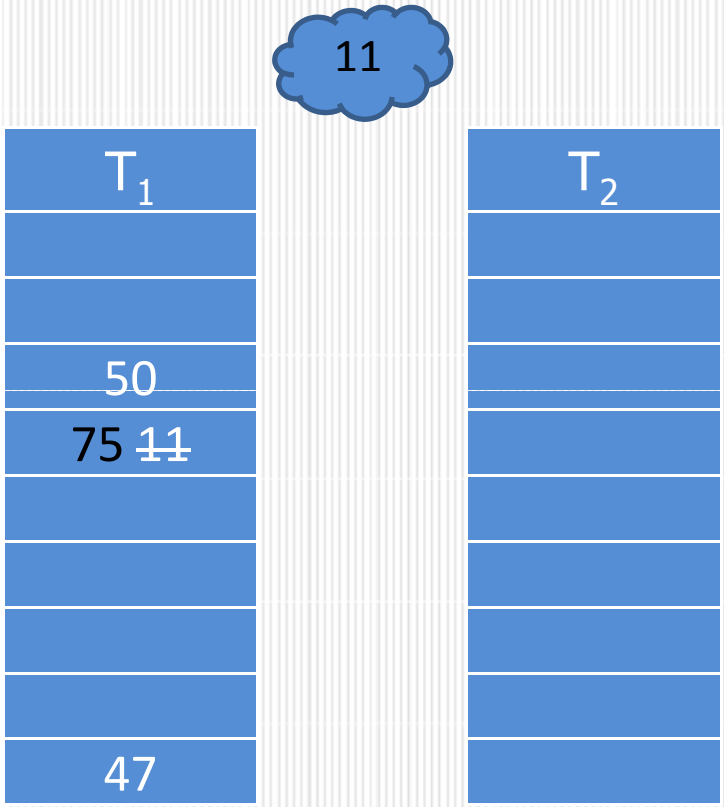
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {39, 51, 106, 162}

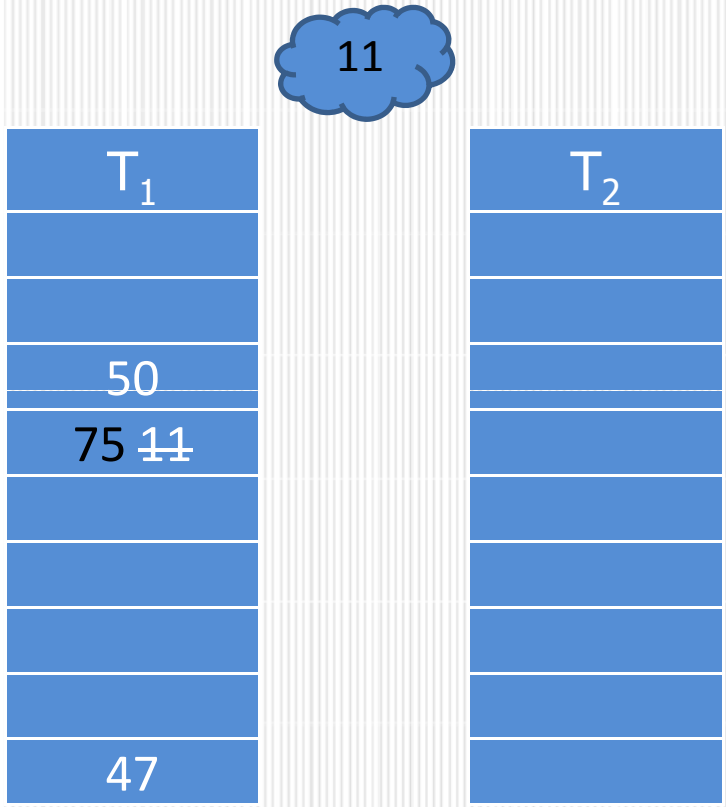
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {39, 51, 106, 162}

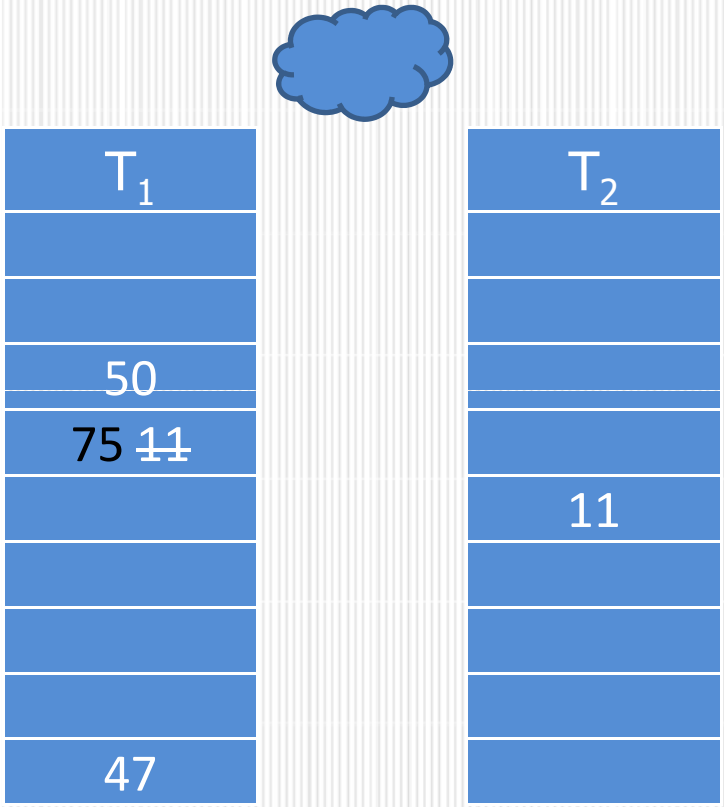
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {39, 51, 106, 162}

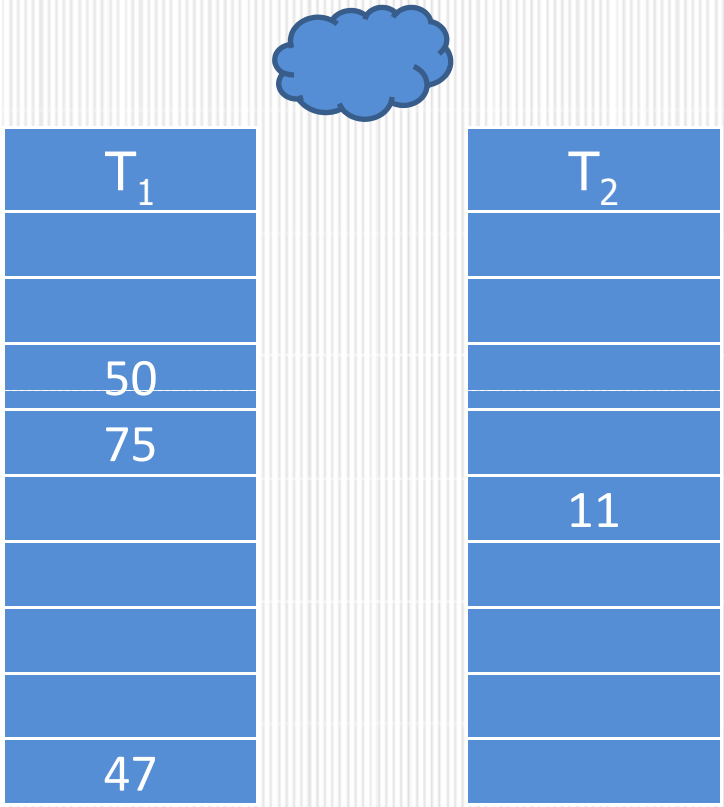
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {39, 51, 106, 162}

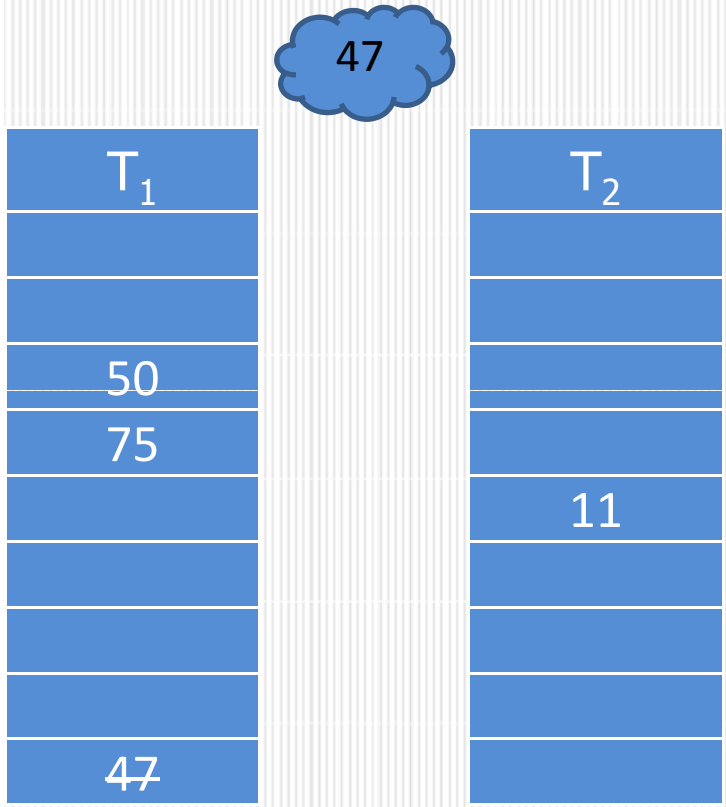
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {39, 51, 106, 162}

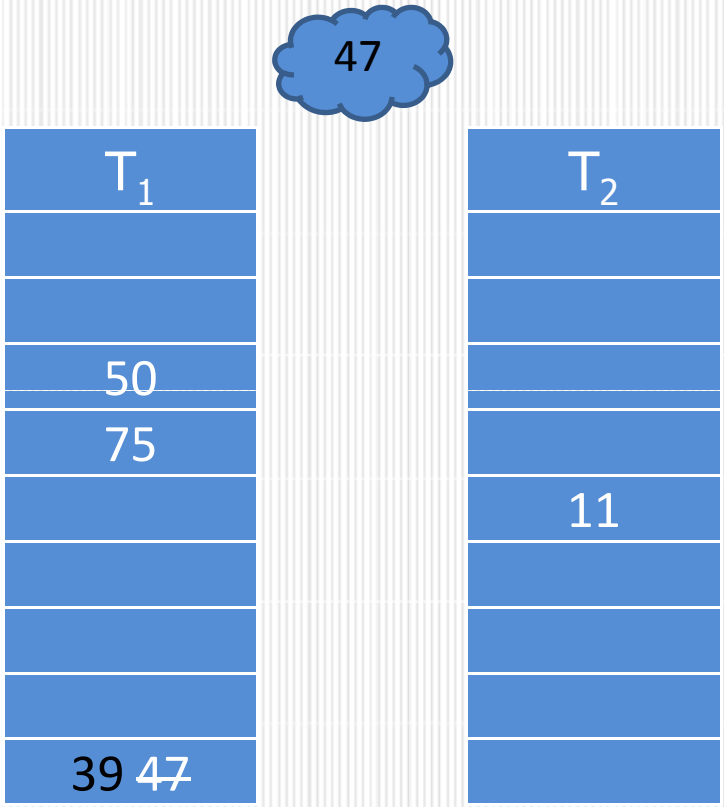
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {51, 106, 162}

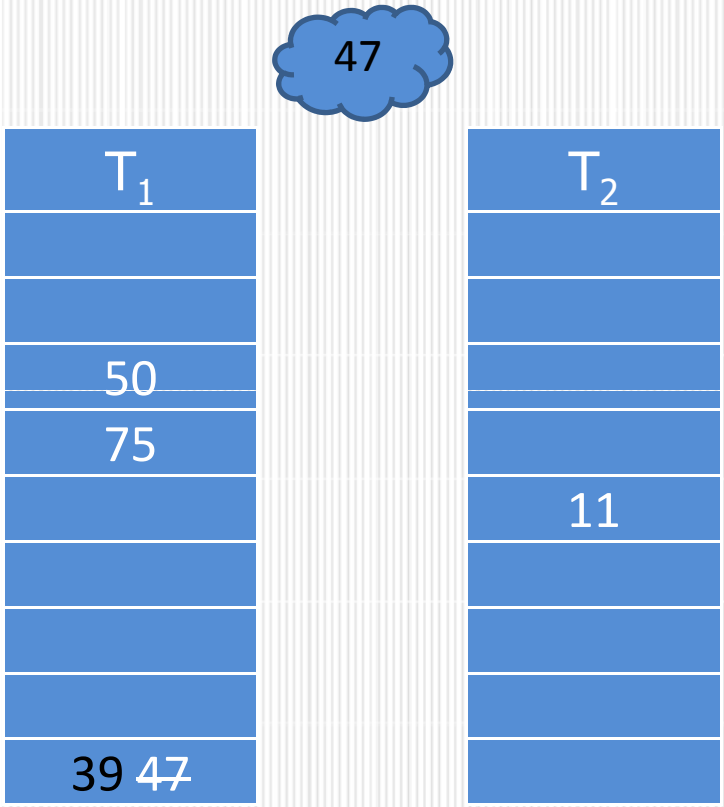
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {51, 106, 162}

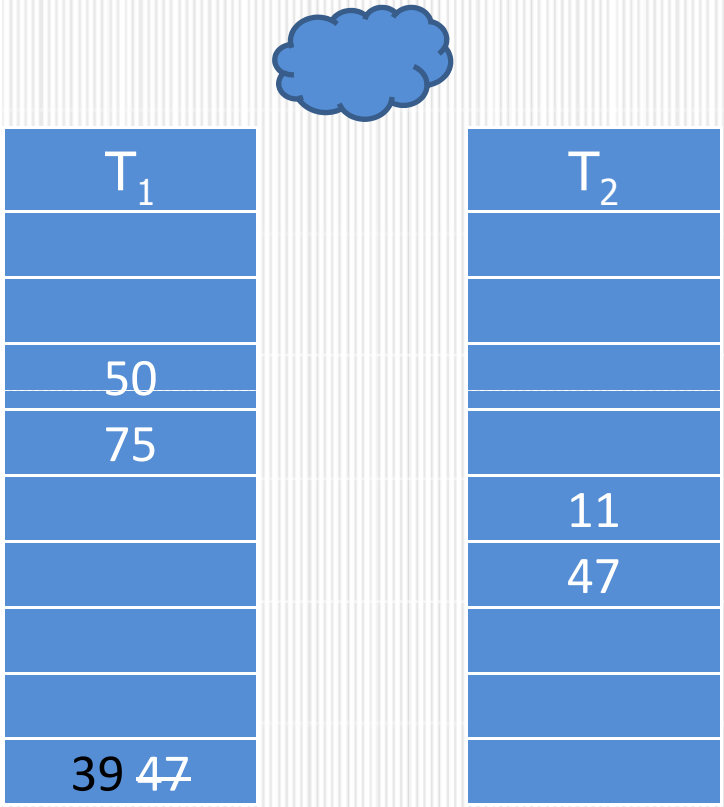
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {51, 106, 162}

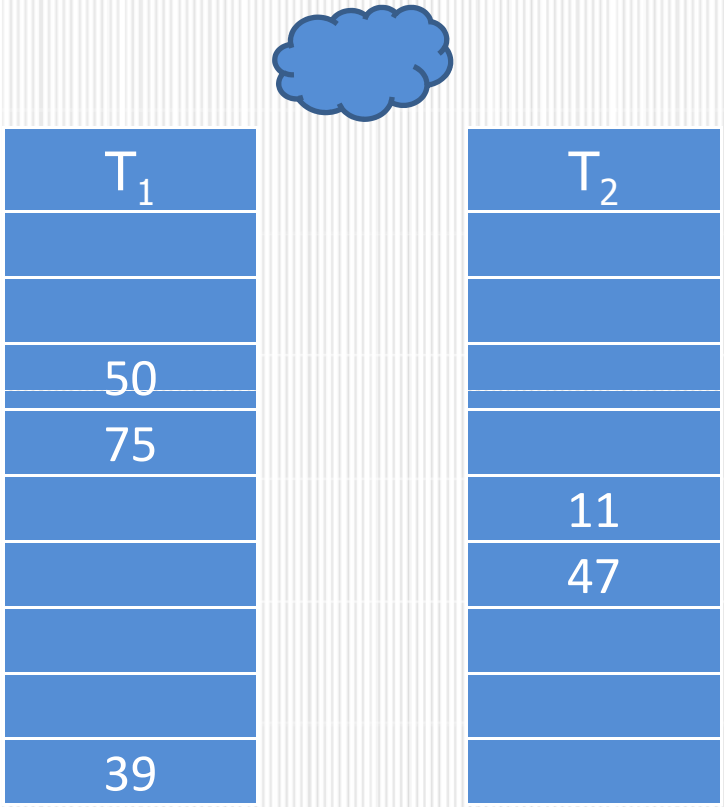
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {51, 106, 162}

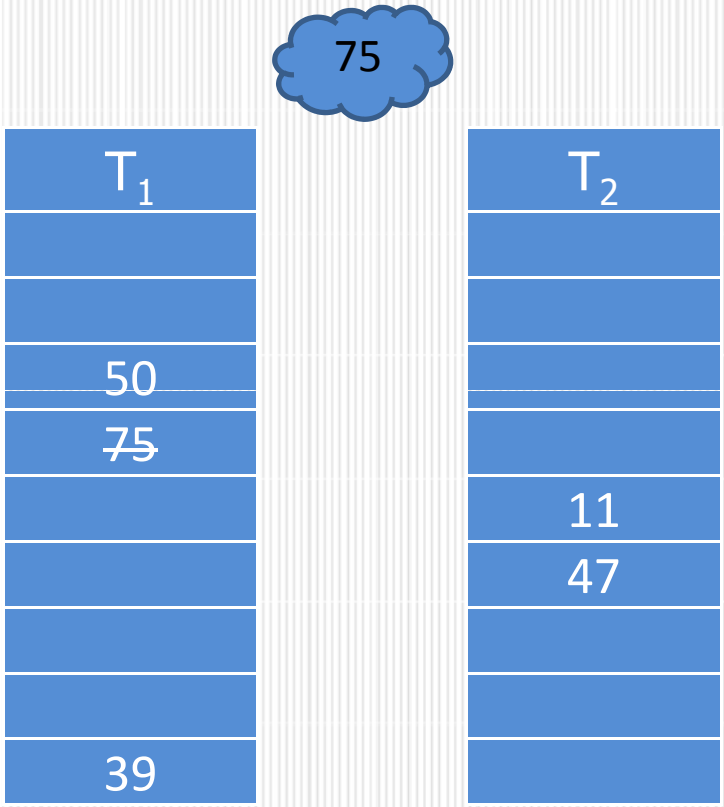
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {51, 106, 162}

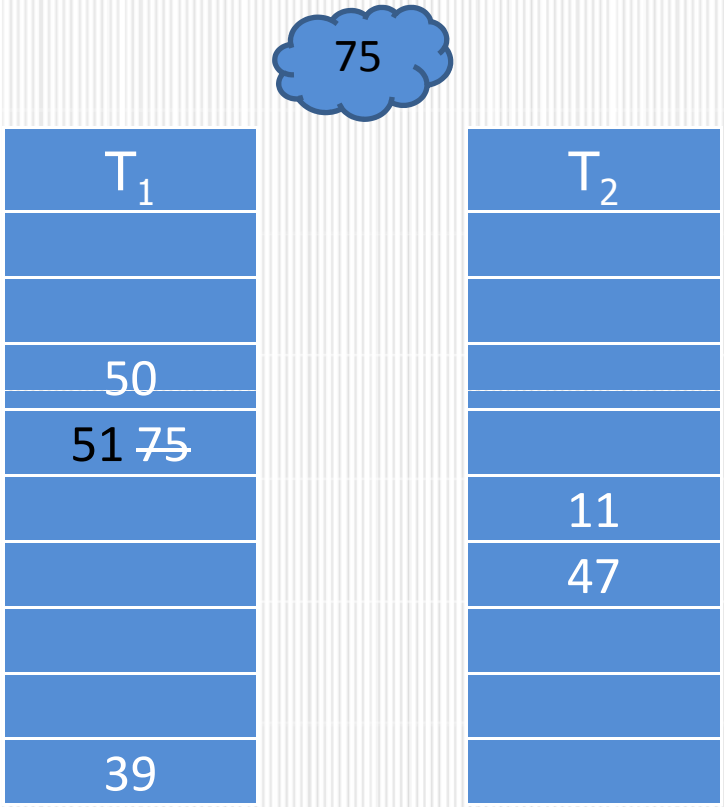
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

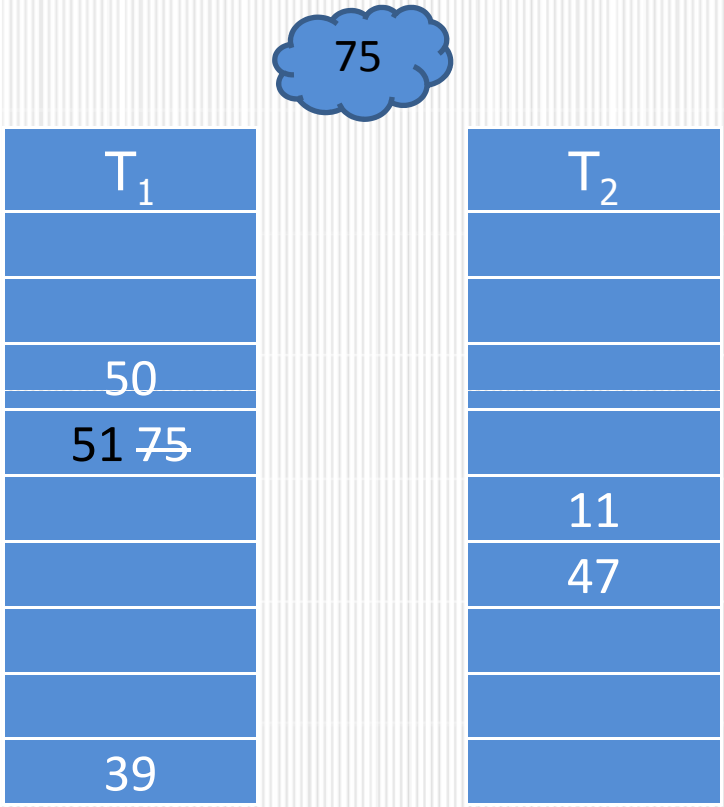
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

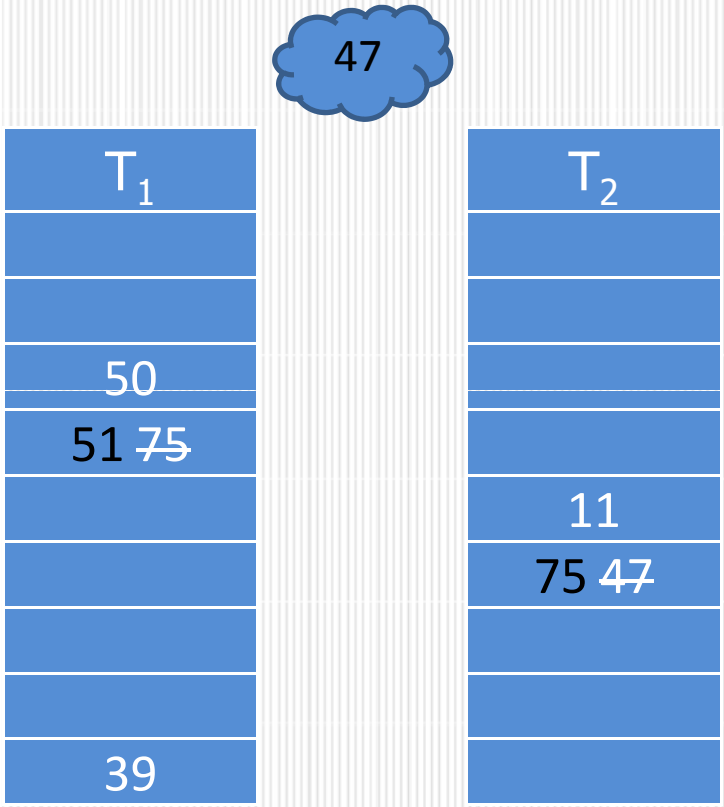
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

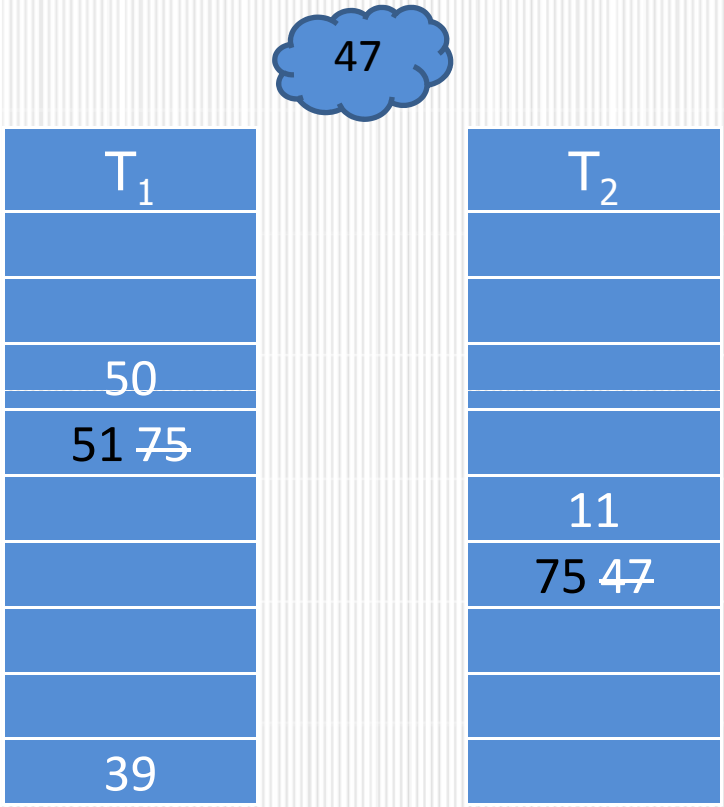
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

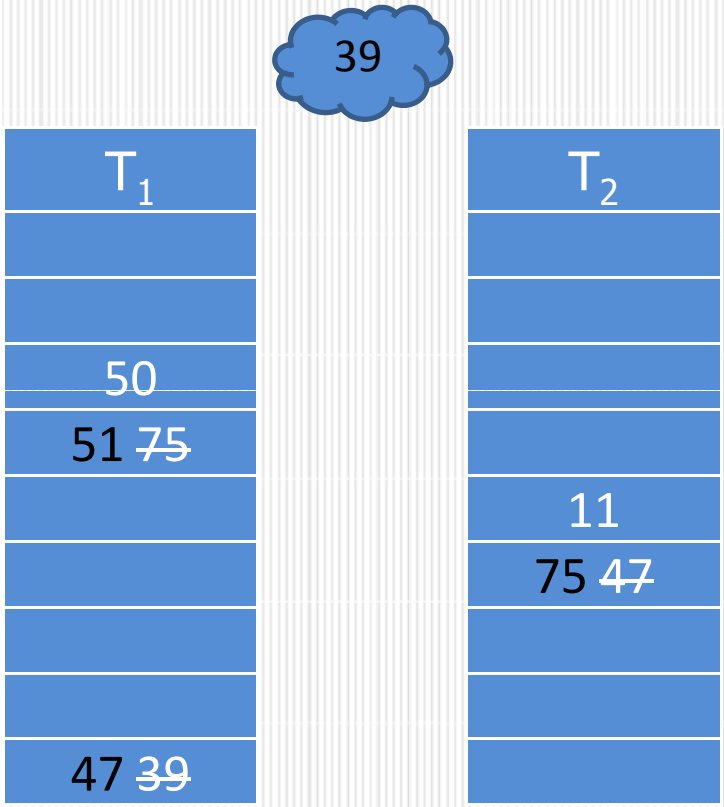
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

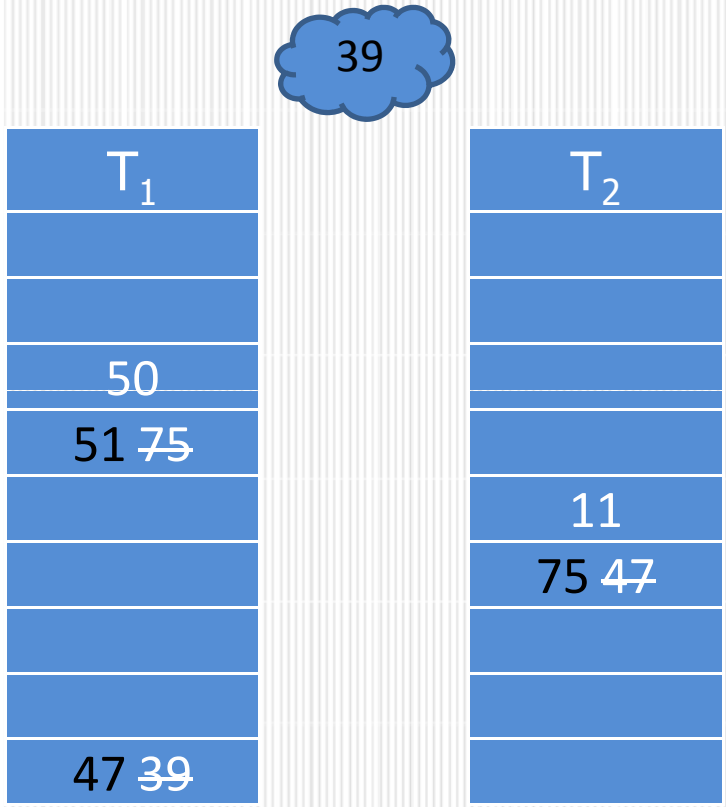
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

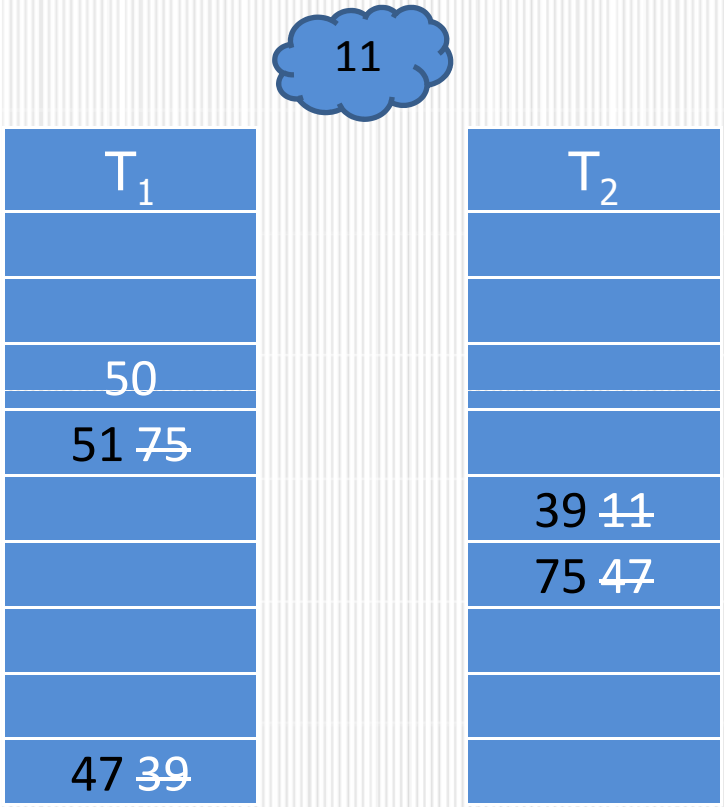
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

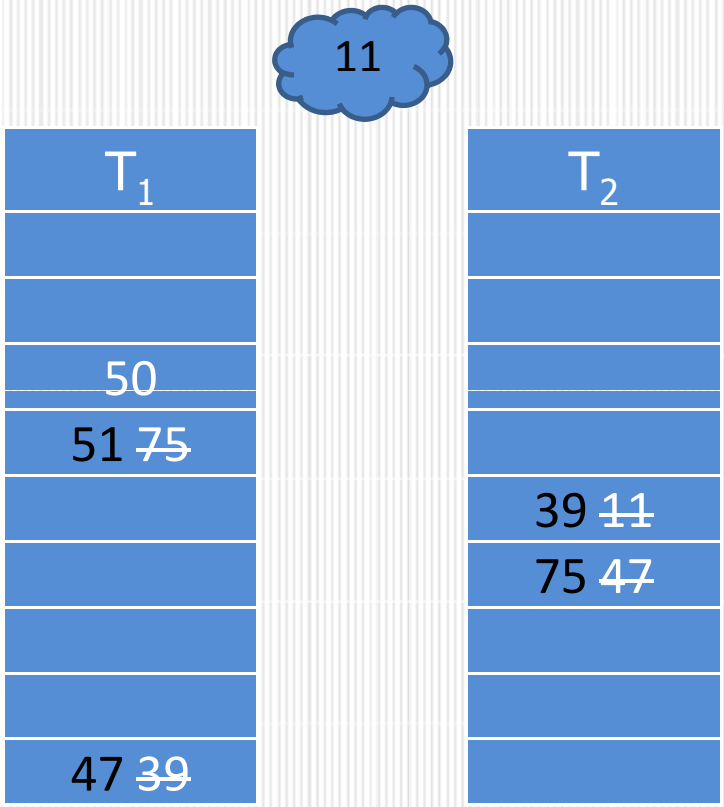
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

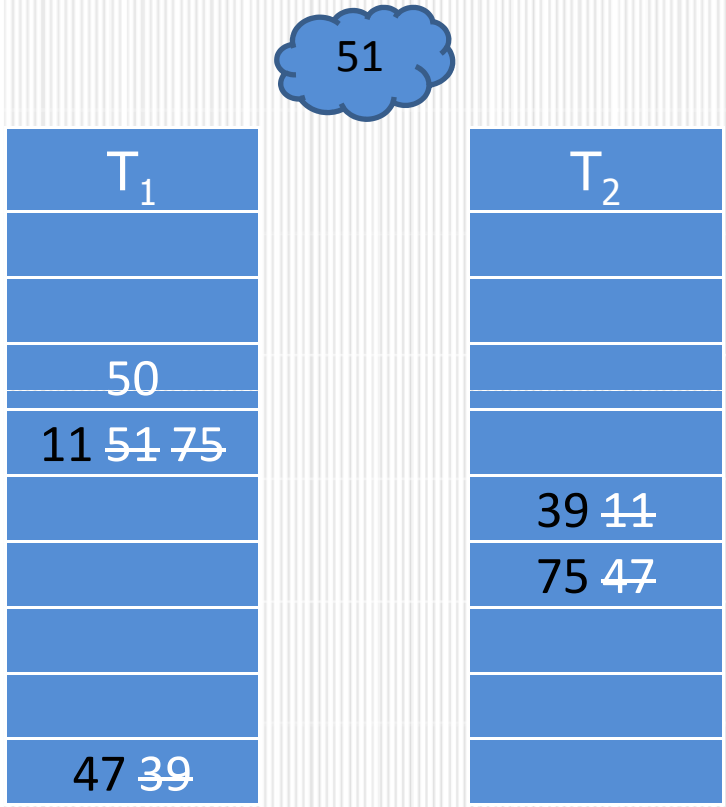
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

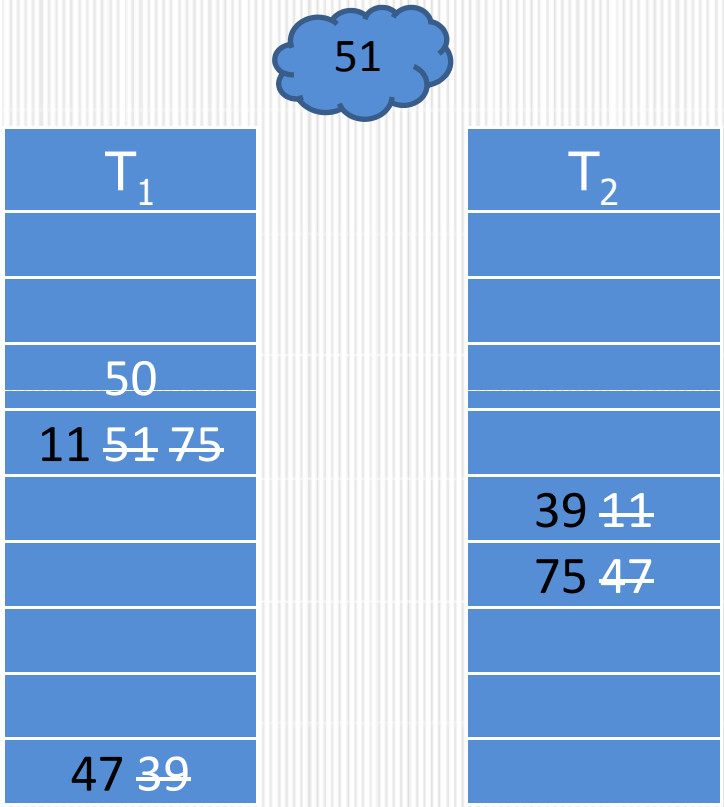
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

T_1	T_2
50	51
11 ~~51~~ ~~75~~	
	39 ~~11~~
	75 ~~47~~
47 ~~39~~	

Παράδειγμα

- {106, 162}

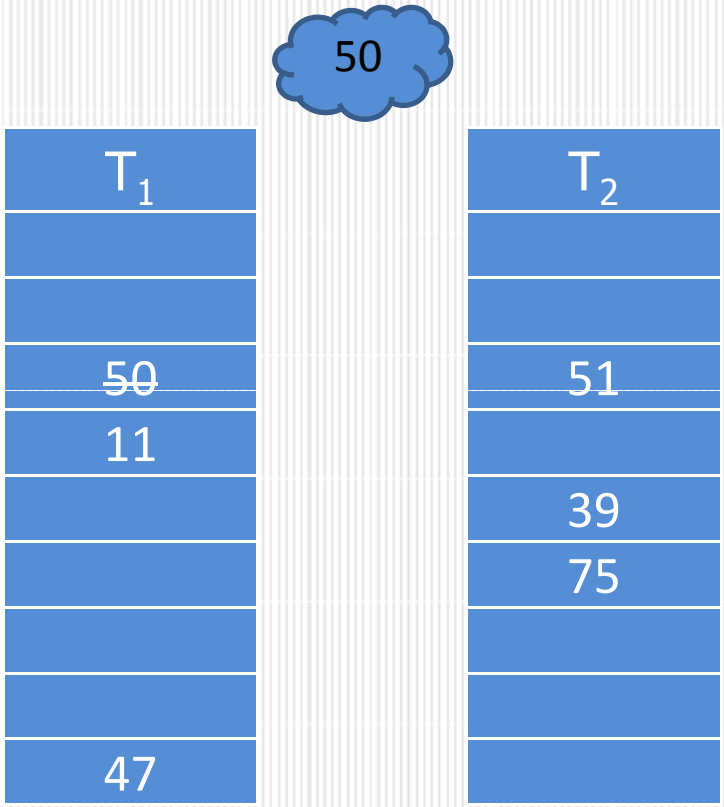
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

T_1	T_2
50	51
11	
	39
	75
47	

Παράδειγμα

- {106, 162}

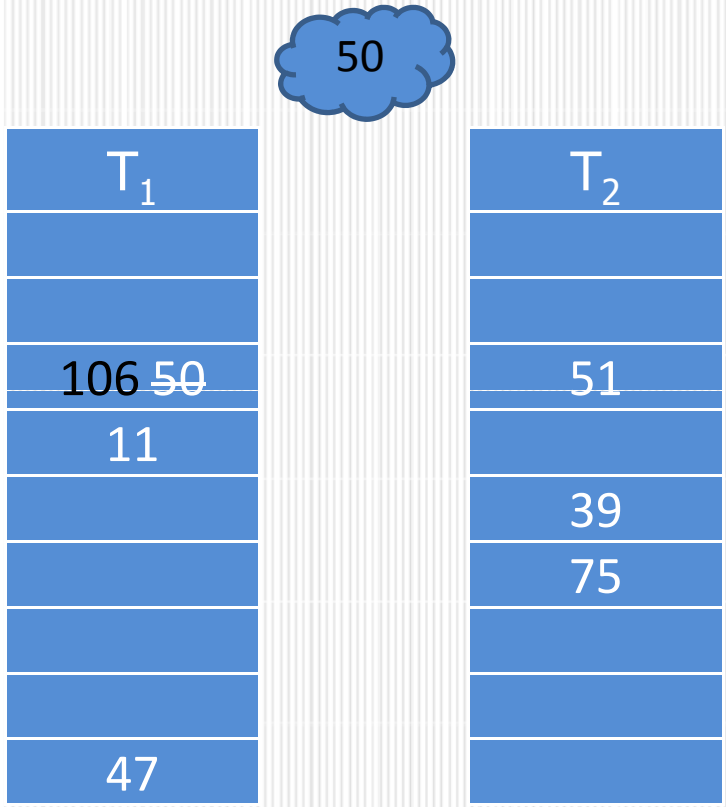
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {162}

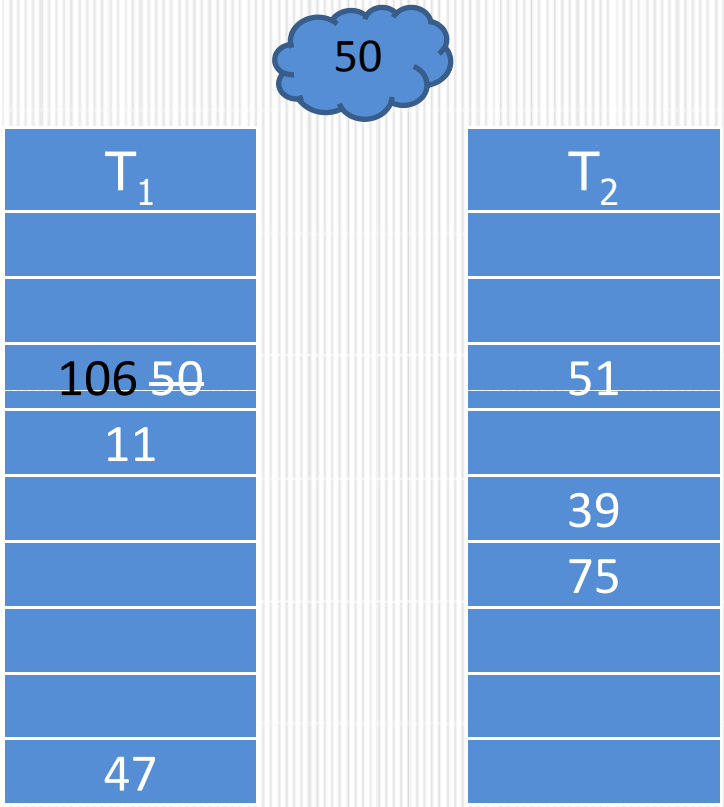
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

T_1	T_2
	50
106 ~~50~~	51
11	
	39
	75
47	

Παράδειγμα

- {162}

T_1
106
11
47

T_2
50
51
39
75

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

Παράδειγμα

- {162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

106

T_1	T_2
	50
~~106~~	51
11	
	39
	75
47	

Παράδειγμα

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

- {}

106

T_1	T_2
	50
162 ~~106~~	51
11	
	39
	75
47	

Παράδειγμα

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

- {}

106

T_1	T_2
	50
162 ~~106~~	51
11	
	39
	75
47	

Παράδειγμα

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

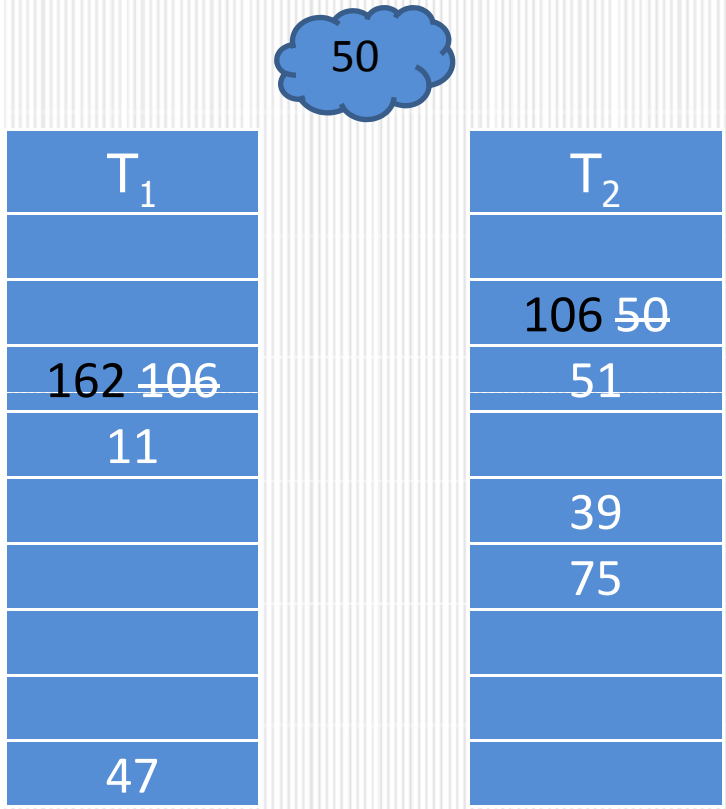
- {}

50

T_1	T_2
	106 ~~50~~
162 ~~106~~	51
11	
	39
	75
47	

Παράδειγμα

- {}



x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

Παράδειγμα

- {}

T_1	T_2
	106 ~~50~~
50 ~~162~~ ~~106~~	51
11	
	39
	75
47	

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

Παράδειγμα

- {}

T_1	T_2
	106 ~~50~~
50 ~~162~~ ~~106~~	51
11	
	39
	75
47	

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

Παράδειγμα

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

- {}

106

T_1
50 ~~162~~ ~~106~~
11
47

T_2
162 ~~106~~ ~~50~~
51
39
75

Παράδειγμα

- {}

106

T_1
50 ~~162~~ ~~106~~
11
47

T_2
162 ~~106~~ ~~50~~
51
39
75

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

Παράδειγμα

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

- {}

50

T_1
106 ~~50~~ ~~162~~
~~106~~
11
47

T_2
162 ~~106~~ ~~50~~
51
39
75

Παράδειγμα

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1

- {}

50

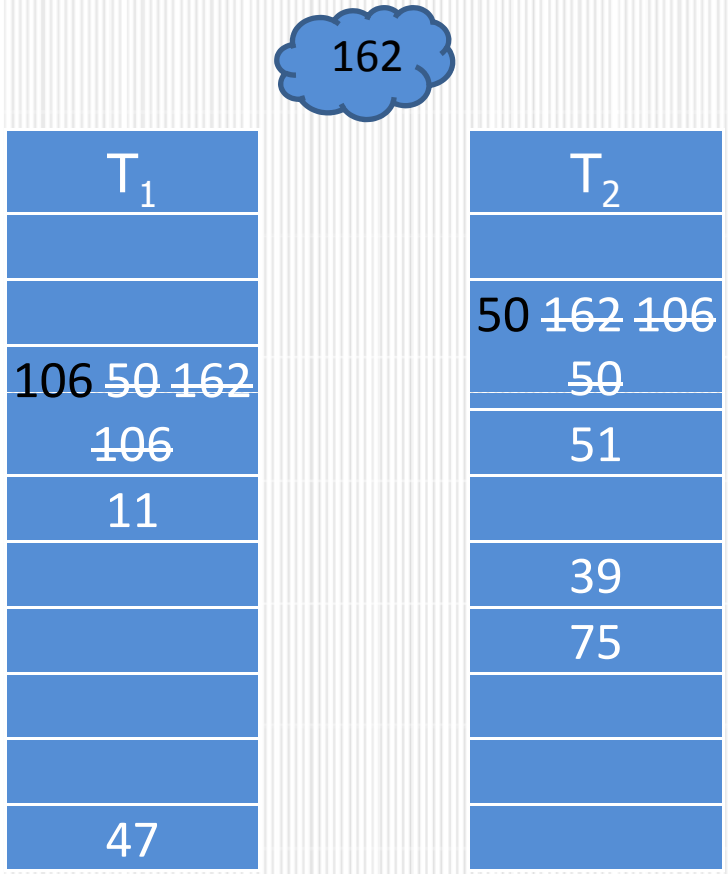
T_1
106 ~~50~~ ~~162~~ ~~106~~
11
47

T_2
162 ~~106~~ ~~50~~
51
39
75

Παράδειγμα

- {}

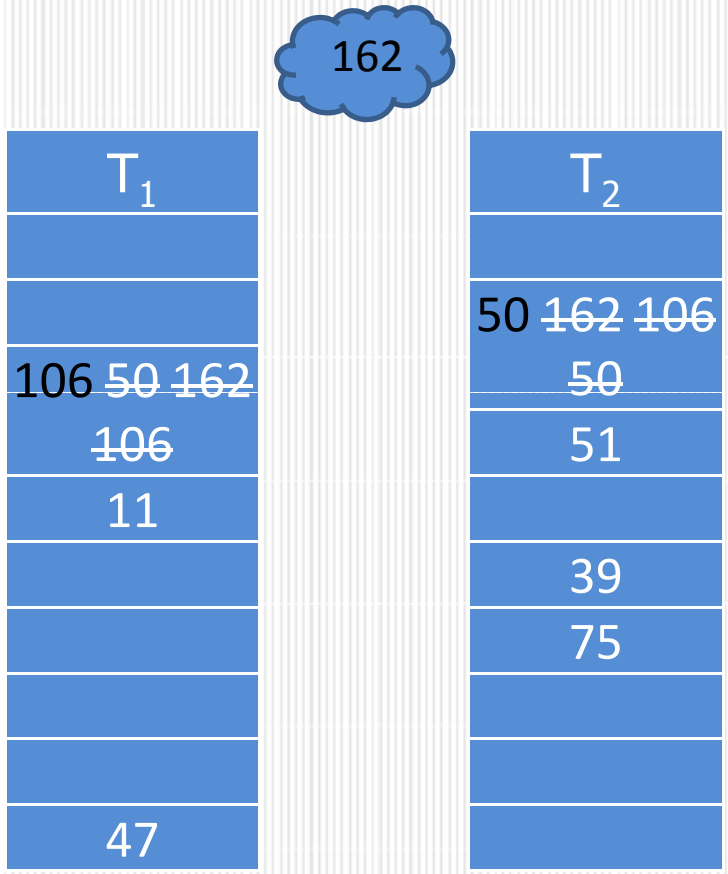
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {}

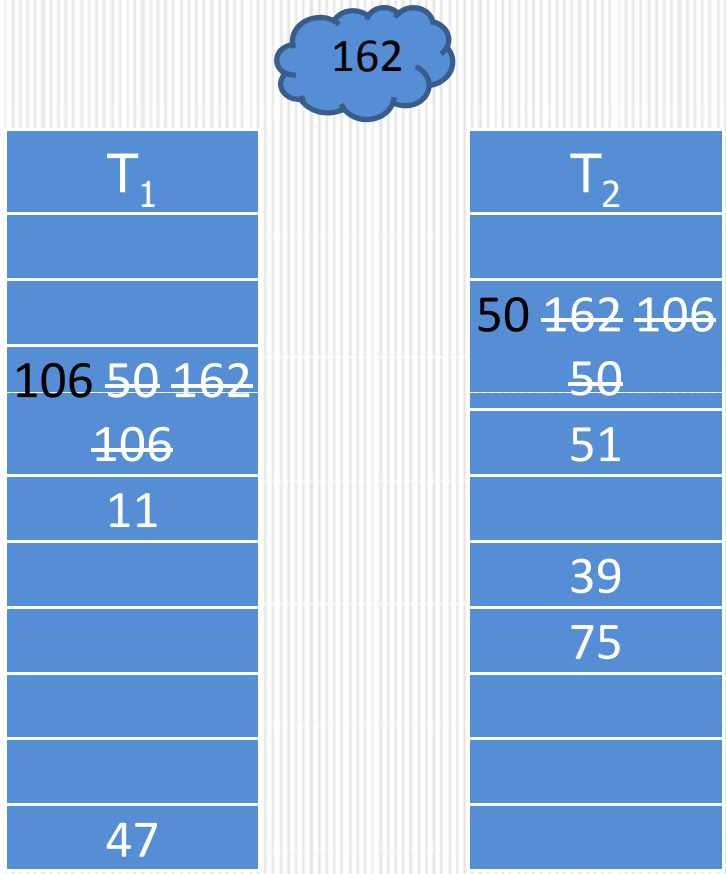
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Rehashing

Ανάλυση Cuckoo Hashing

- Υπάρχουν δύο λεπτομέρειες που περιπλέκουν την ανάλυση
  - Κάθε στοιχείο μπορεί να εντοπιστεί σε δύο διαφορετικές θέσεις
  - Η ακολουθία των αντικαταστάσεων μπορεί να γίνει αρκετά εκτενής
- Κομψή Λύση το Cuckoo graph

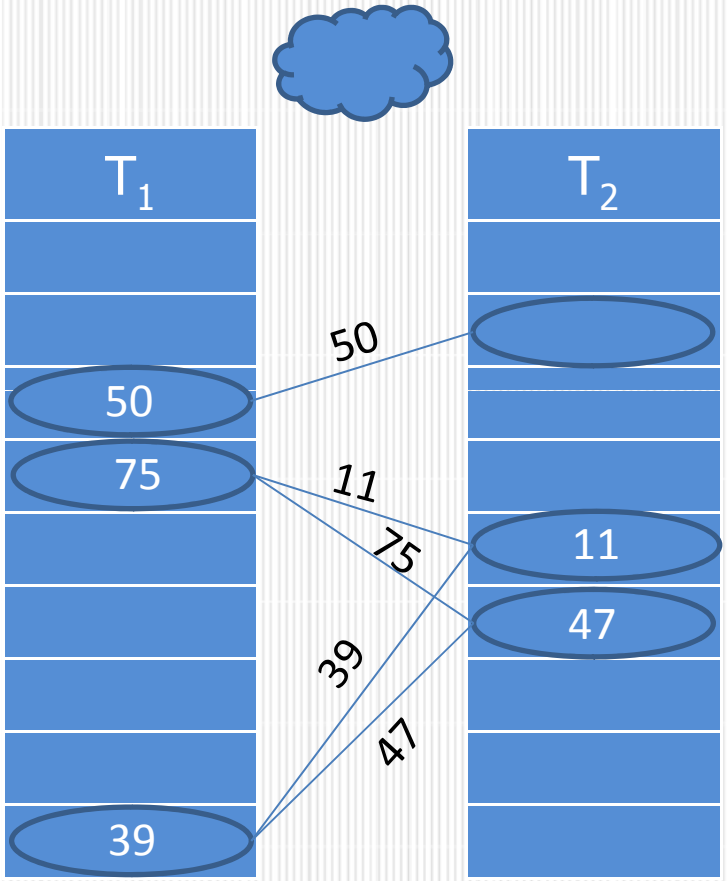
Cuckoo graph

- Περιγραφή
  - Κάθε θέση του πίνακα γίνεται κόμβος
  - Κάθε στοιχείο του πίνακα γίνεται ακμή
  - Διμερές γράφημα που προκύπτει από το hash table
- Κάθε νέα εισαγωγή προσθέτει μια ακμή στο γράφημα

Παράδειγμα

- {51, 106, 162}

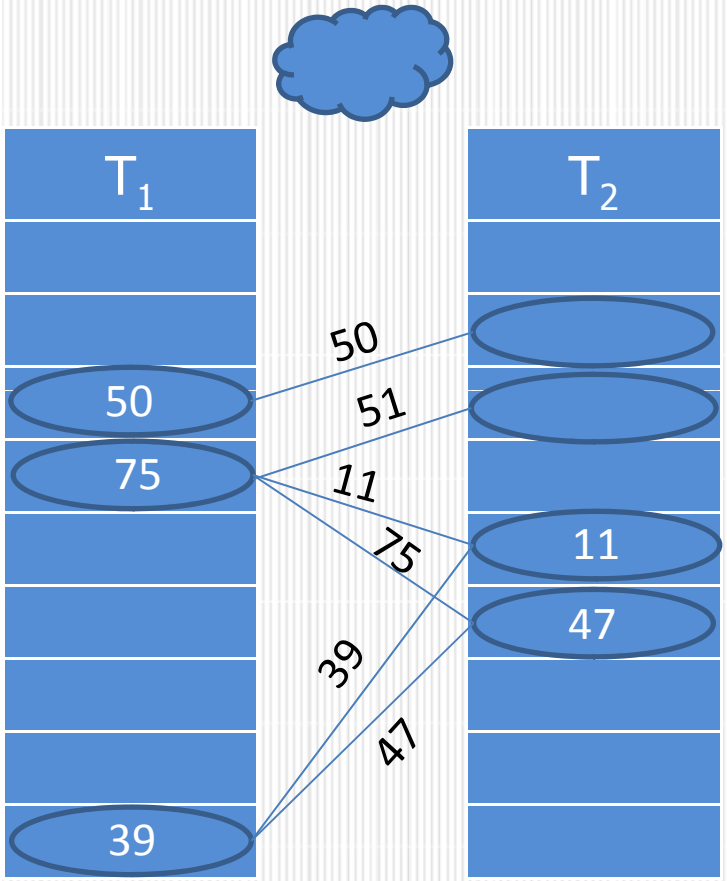
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {51, 106, 162}

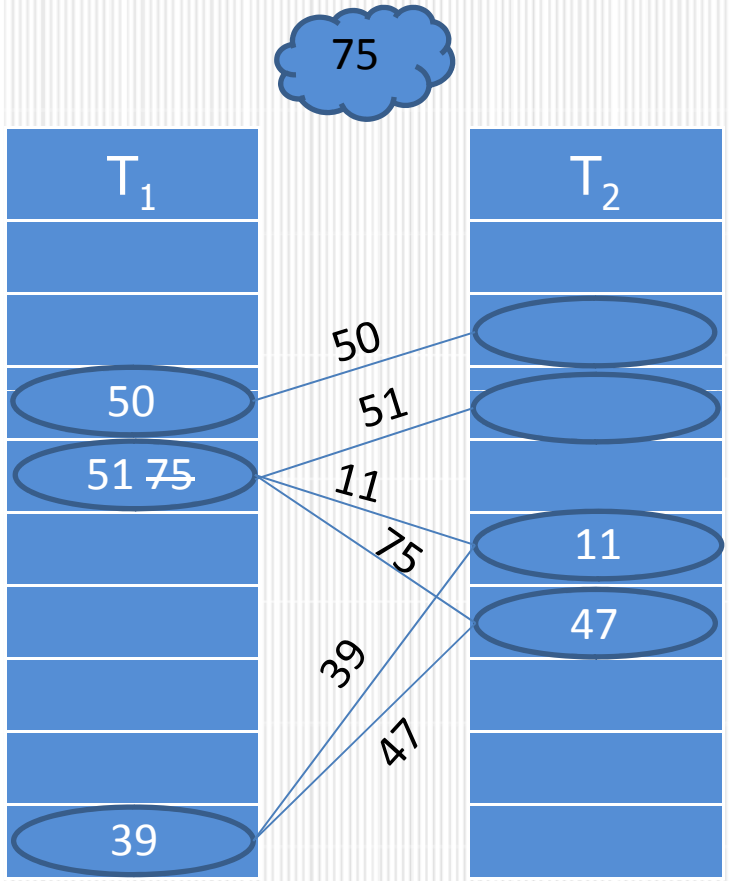
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

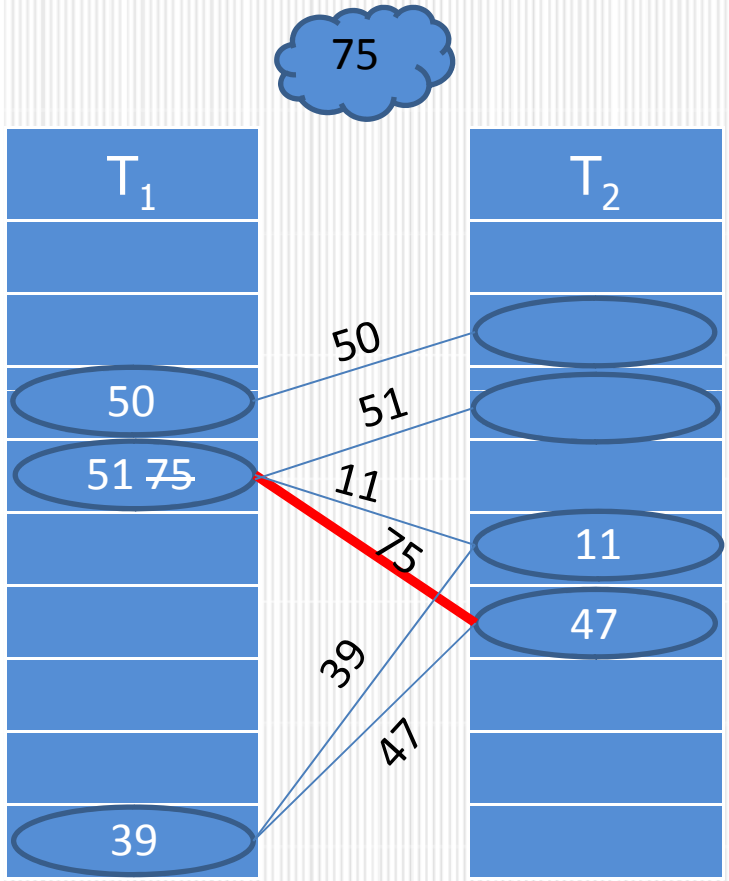
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

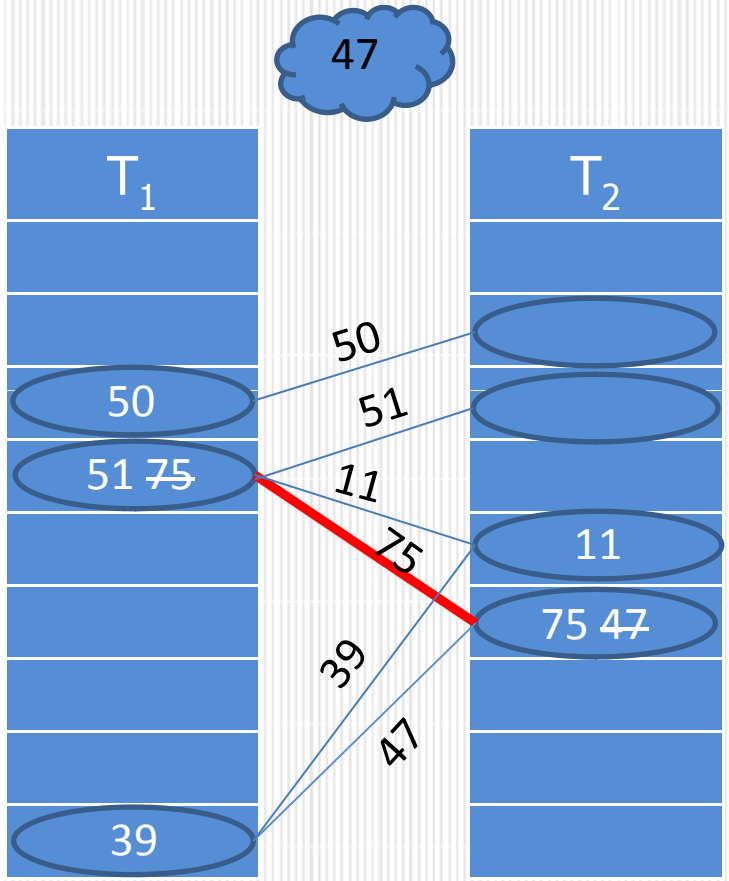
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

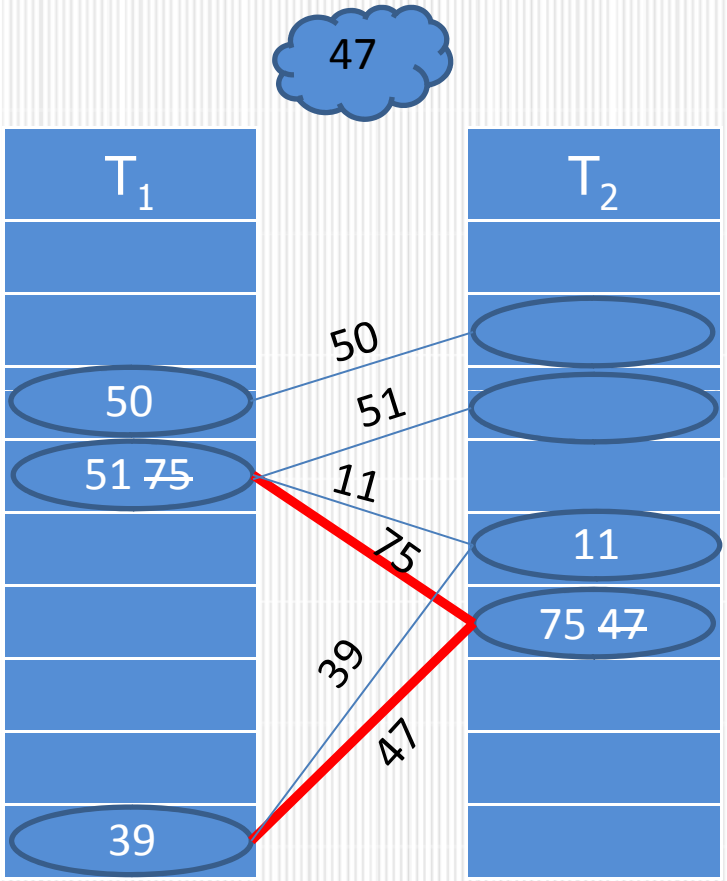
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

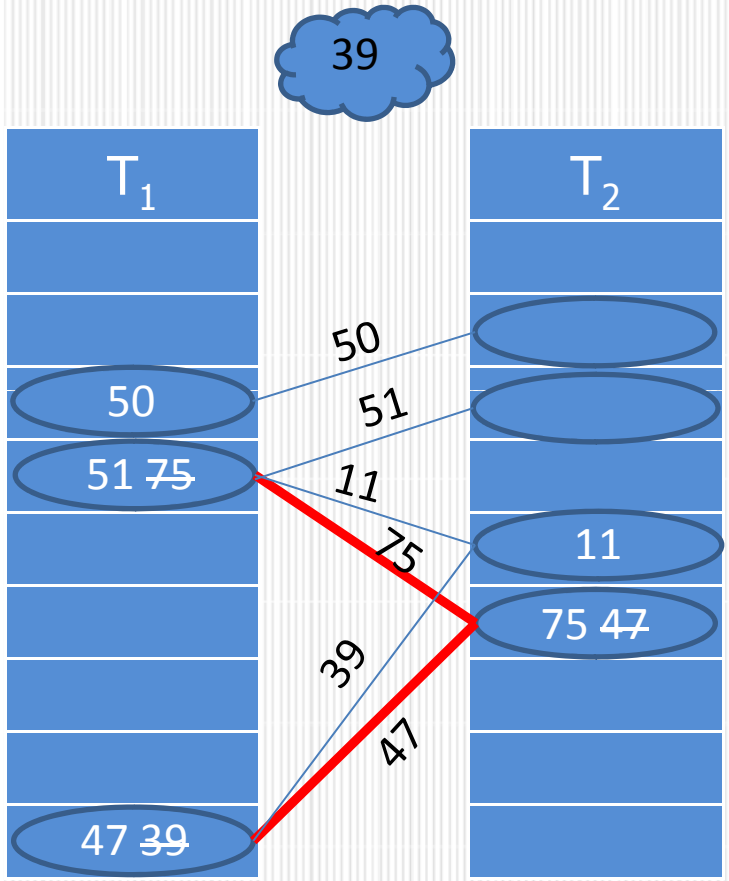
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

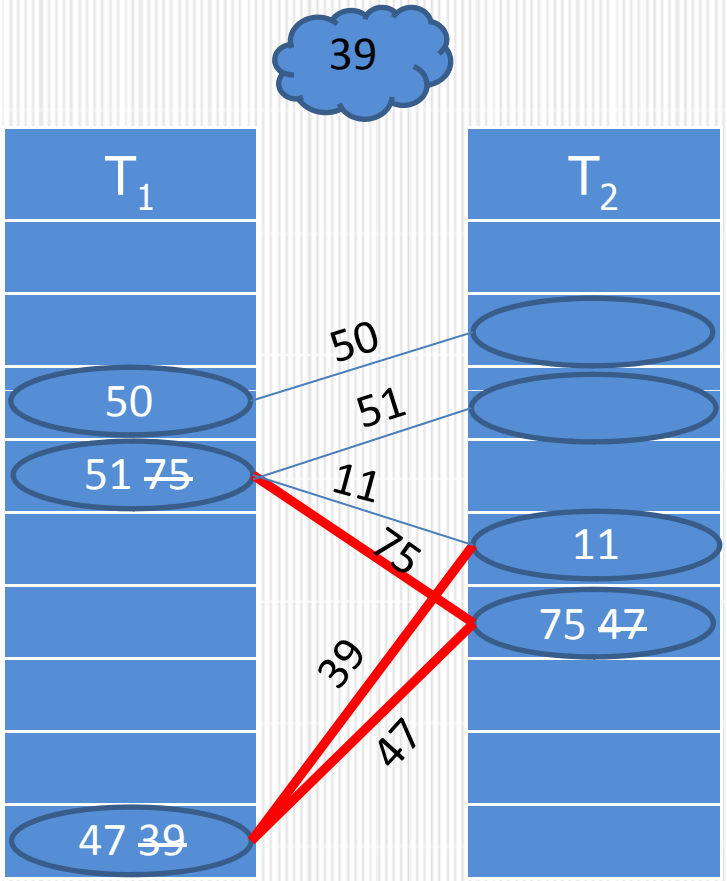
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

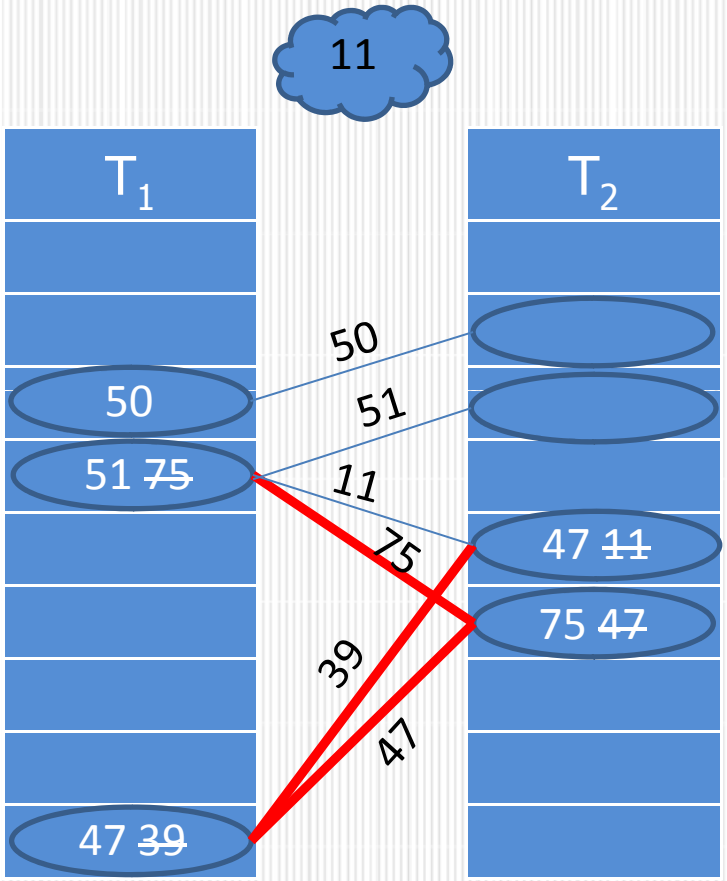
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

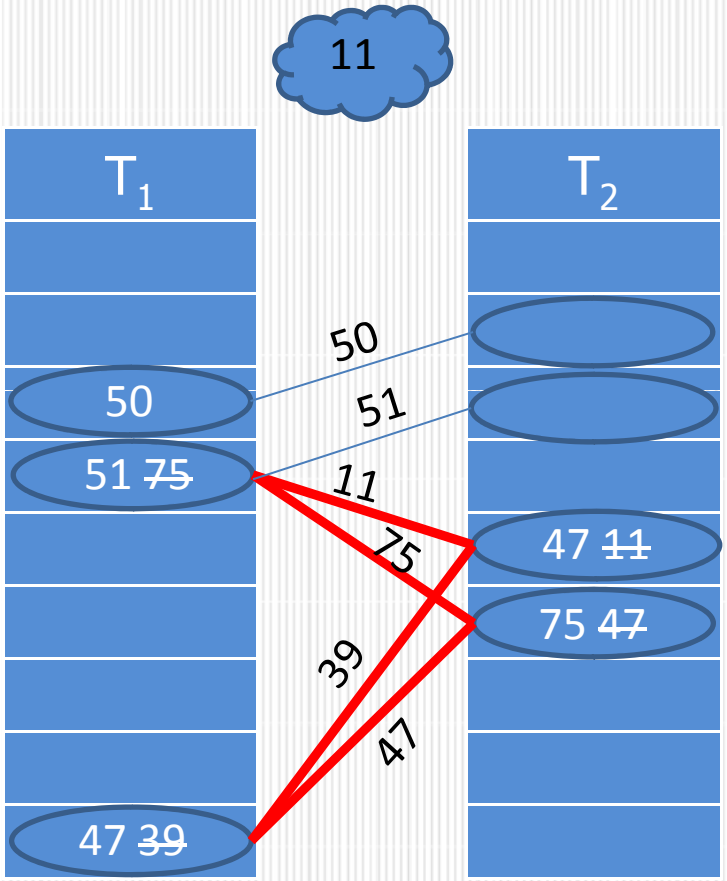
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

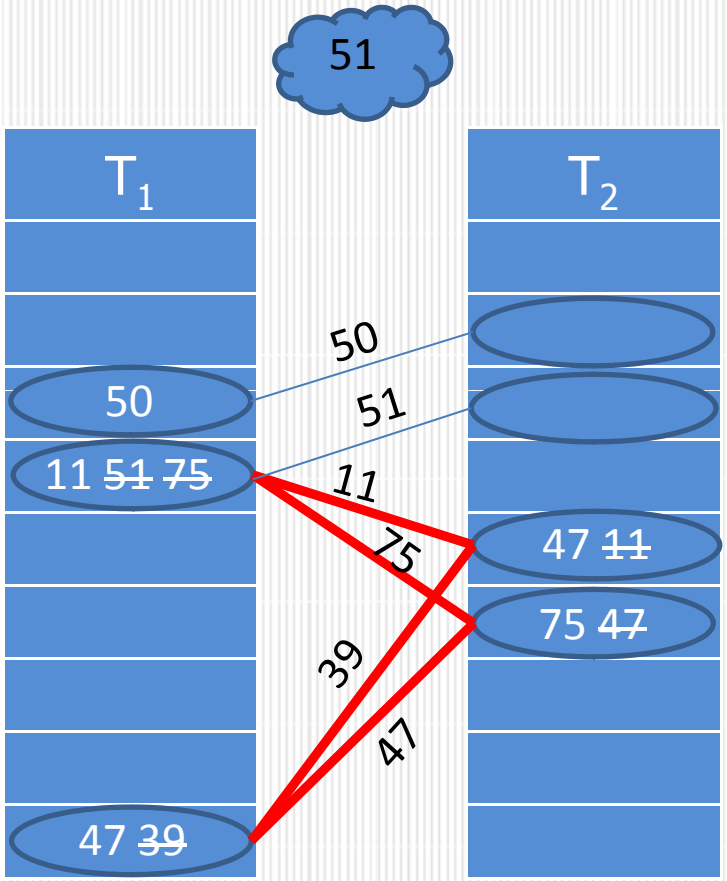
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

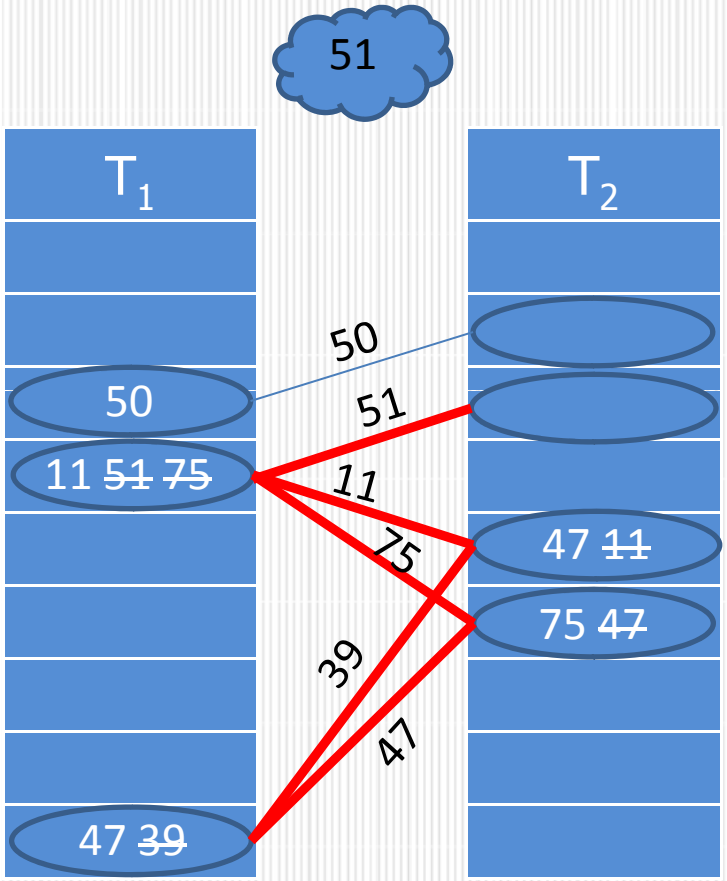
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

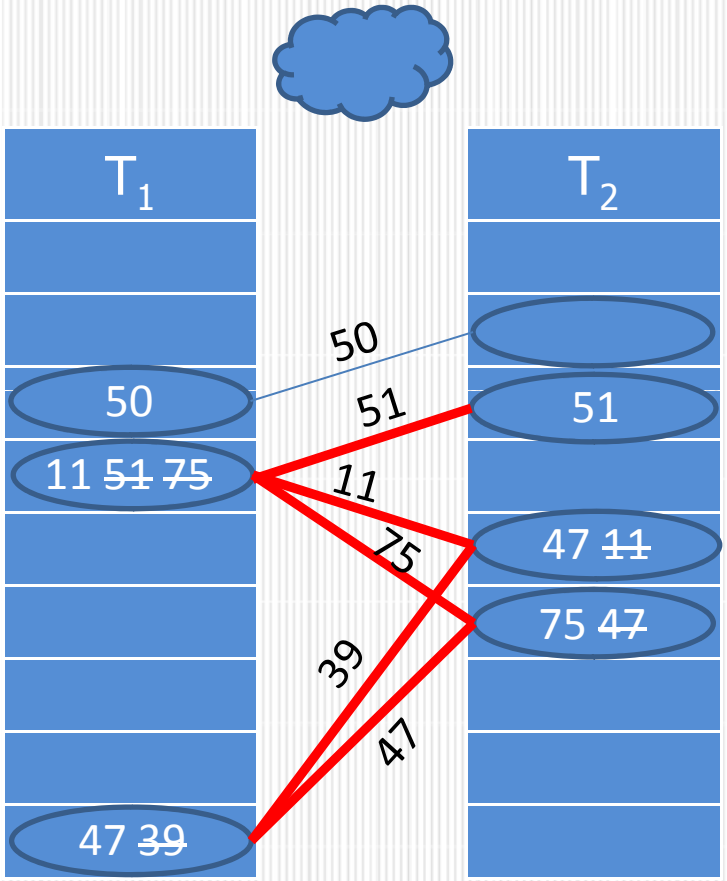
x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Παράδειγμα

- {106, 162}

x	h1(x)	h2(x)
11	3	4
50	2	1
47	7	5
75	3	5
39	7	4
51	3	2
106	2	1
162	2	1



Cuckoo graph

- Αν το x εισαχθεί σε ένα πίνακα, η εισαγωγή αποτυγχάνει αν η συνεκτική συνιστώσα που περιέχει το x, έχει 2 ή περισσότερους κύκλους

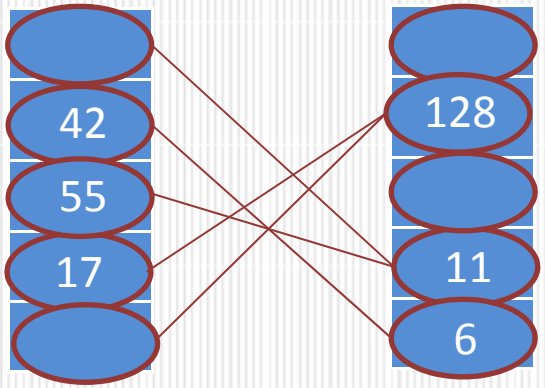
- Για να έχουμε ≥ 2 κύκλων σε k κόμβους θέλουμε τουλάχιστον k+1 ακμές (στοιχεία)
  => παραπάνω στοιχεία από όσα χωράνε
  => rehashing

- Αν το x εισαχθεί σε ένα πίνακα, η εισαγωγή επιτυγχάνει αν η συνεκτική συνιστώσα που περιέχει το x, έχει το πολύ ένα κύκλο

- Αν το x, εισαχθεί σε μία συνεκτική συνιστώσα με k κόμβους, τότε η εισαγωγή θα πραγματοποιήσει το πολύ 2k «εξώσεις»

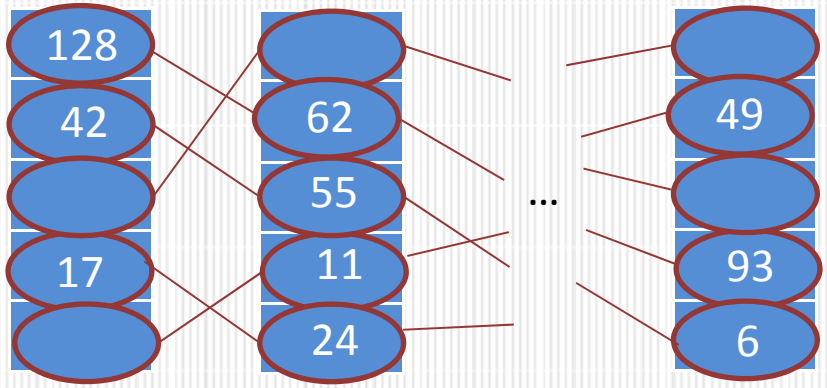
Space Efficiency

- Space utilization ≈ 50%
- Όταν έχουν εισαχθεί n/2 στοιχεία-ακμές είναι πλέον πολύ πιθανόν να υπάρξει κύκλος σε κάποια επόμενη εισαγωγή => rehashing
- Όμως μπορεί και πολύ καλύτερα…



d-ary Cuckoo Hashing

- Χρησιμοποιούμε d αντί για δύο hash tables / functions
- Κατά την εισαγωγή έχουμε ένα δέντρο υποψήφιων εισαγωγών-εξώσεων
- Χρησιμοποιούμε BFS για να επιλέξουμε τη θέση εισαγωγής
- Όχι και τόσο εύκολη υλοποίηση
- Όμως, έχει πολύ υψηλή απόδοση
  - Lookup / Delete: O(1)
  - Amortized insert: O(1)
  - Space efficiency: ≥ 97% (d=4)



Συμπεράσματα

- Cuckoo Hashing
  - \+ Εξαιρετική δομή για την επίλυση του προβλήματος του λεξικού υποστηρίζοντας εισαγωγή, διαγραφή και αναζήτηση
  - \+ Απλή υλοποίηση
  - \+ Αποδοτικό στη μέση περίπτωση
  - \- Δεν υπάρχει κάποιο «καλό» σύνολο hash functions
- Περαιτέρω μελέτη
  - Εύρεση «καλού» συνόλου hash functions
  - Ακριβέστερη ανάλυση εισαγωγών

ΕΡΩΤΗΣΕΙΣ;;

Πηγές

- http://www.corelab.ntua.gr/courses/algorithms/slides/05_Union-Find.pdf
- http://web.stanford.edu/class/cs166/lectures/13/Small13.pdf
- http://www.it-c.dk/people/pagh/papers/cuckoo-undergrad.pdf
- www.cs.utexas.edu/~whunt/files/cuckoo.pp