Προγραμματισμός ΙΙ (ECE116)

#7 πίνακες κατακερματισμού (hash tables)

Σπύρος Λάλης
lalis@uth.gr

Δομές αναζήτησης … μέχρι στιγμής

Δομή	Πολυπλοκότητα αναζήτησης
Table	$O(\mathrm{N})$
Sorted Table	$O(\log_2 \mathrm{N})$
List	$O(\mathrm{N})$
Sorted List	$O(\mathrm{N})$
Balanced Tree*	$O(\log_2 \mathrm{N})$

*θα γίνει αργότερα, αν προλάβουμε

Υπάρχει κάτι καλύτερο;

- Ένα ιδανικό ευρετήριο θα υποστήριζε την λειτουργία αναζήτησης (ιδανικά και προσθήκης / απομάκρυνσης) με **σταθερό** κόστος
- **Ανεξάρτητα** από το πλήθος των στοιχείων που περιέχει το ευρετήριο

- Μια τέτοια ιδιότητα φαίνεται αδύνατη

- Με κατάλληλα «μαγικά», είναι εφικτή

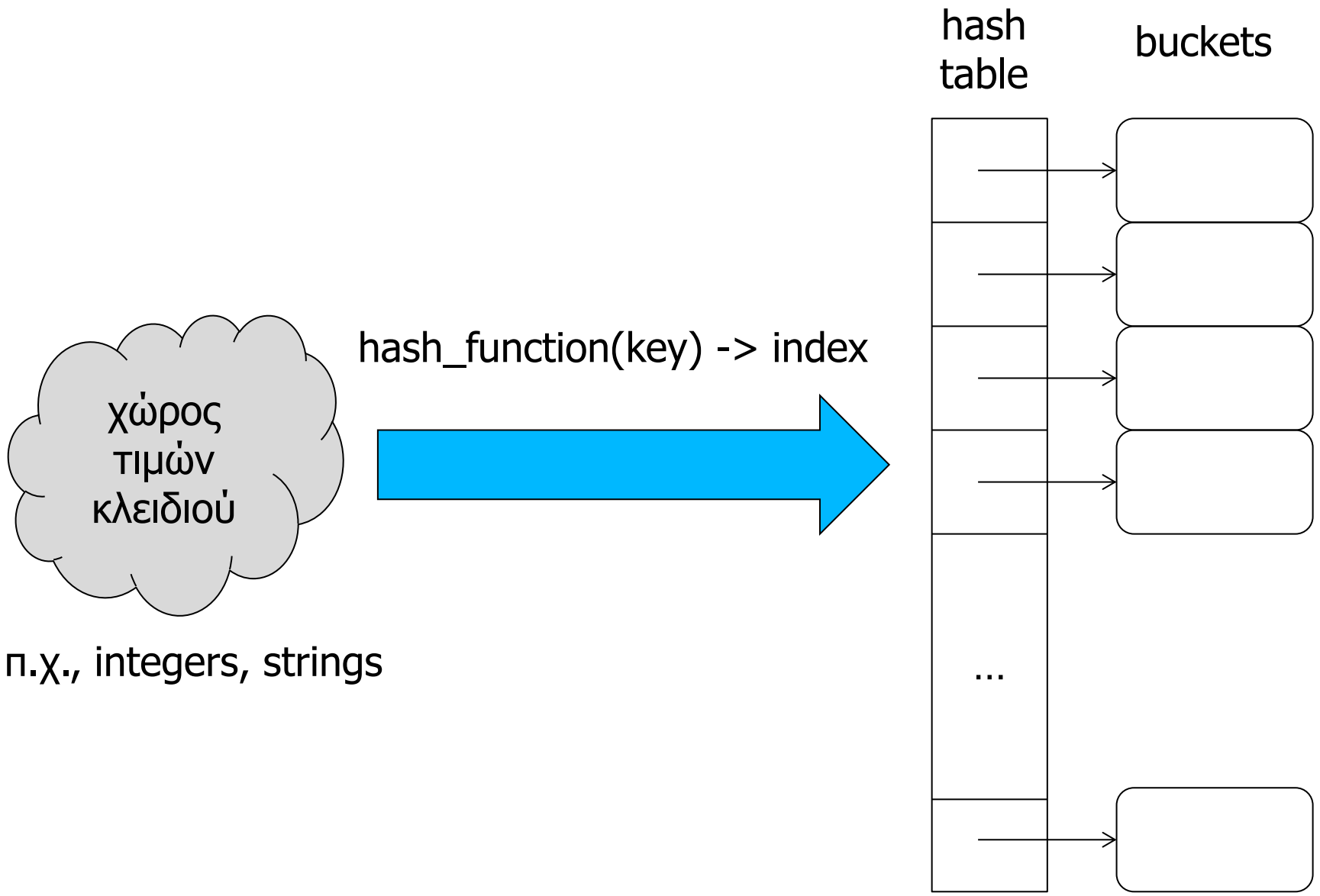
- **Πίνακες κατακερματισμού** (hash tables)

Δομές αναζήτησης

Δομή	Πολυπλοκότητα αναζήτησης
Table	$O(\mathrm{N})$
Sorted Table	$O(\log_2 \mathrm{N})$
List	$O(\mathrm{N})$
Sorted List	$O(\mathrm{N})$
Balanced Tree	$O(\log_2 \mathrm{N})$
Hashtable	$O(1)$

Πίνακας κατακερματισμού

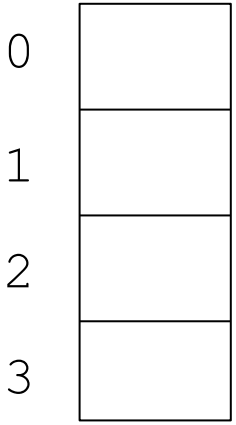
- Ειδικός δυναμικός πίνακας από **δοχεία** (buckets) που περιέχουν τις εγγραφές (δομές με δεδομένα)
  - πίνακας Ν θέσεων: Ν δοχεία
- Η αντιστοίχιση των εγγραφών στα δοχεία του πίνακα γίνεται χρησιμοποιώντας ως «κλειδί» την τιμή από ένα ή περισσότερα πεδία της εγγραφής
- Το κλειδί **μετασχηματίζεται** μέσα από μια **συνάρτηση κατακερματισμού** (hash function)
- Η συνάρτηση επιστρέφει την τιμή (hash value) που **υποδεικνύει την θέση** (το δοχείο) όπου βρίσκεται (ή πρέπει να εισαχθεί) η εγγραφή
  - η τιμή περιορίζεται στο μέγεθος του πίνακα (με modulo)



π.χ., integers, strings

hash_function:

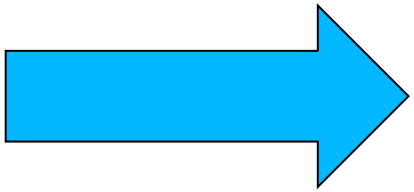
```
str[0] % 4
```

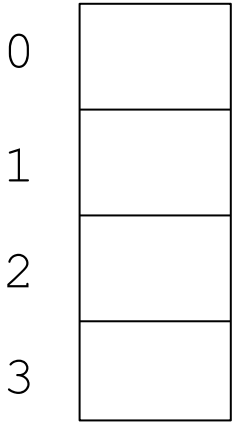


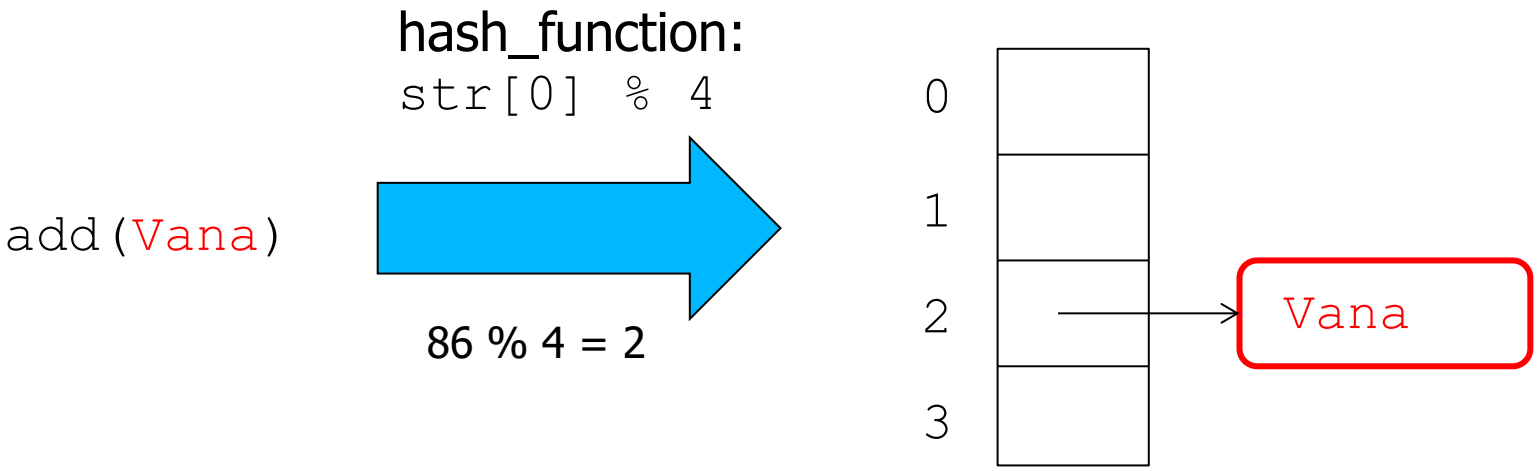
hash_function:

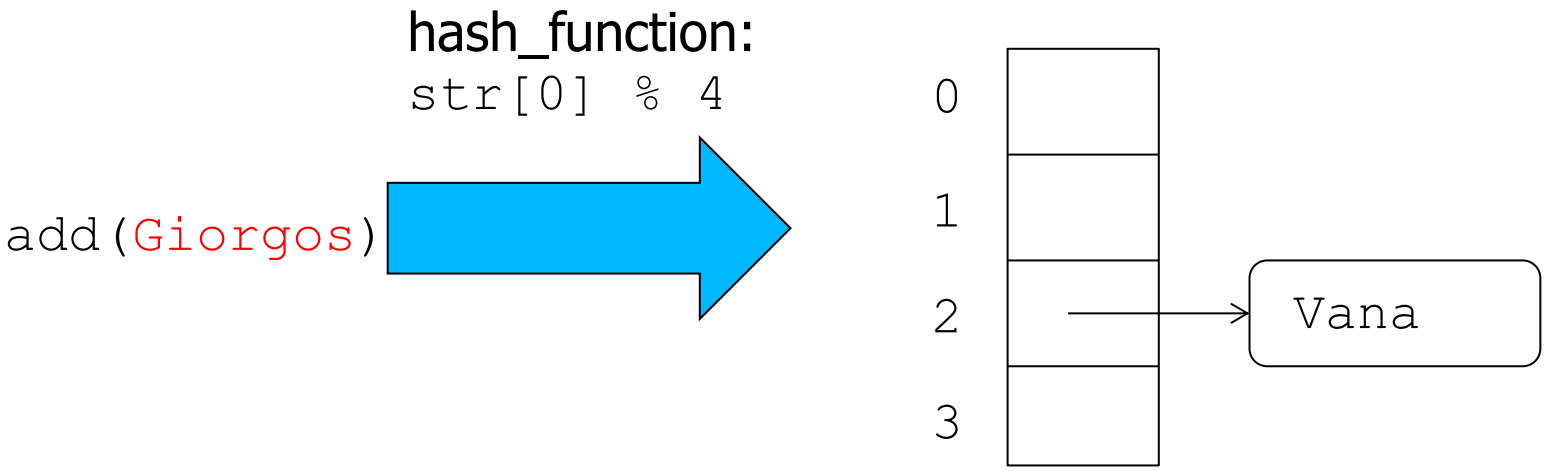
```
str[0] % 4
```

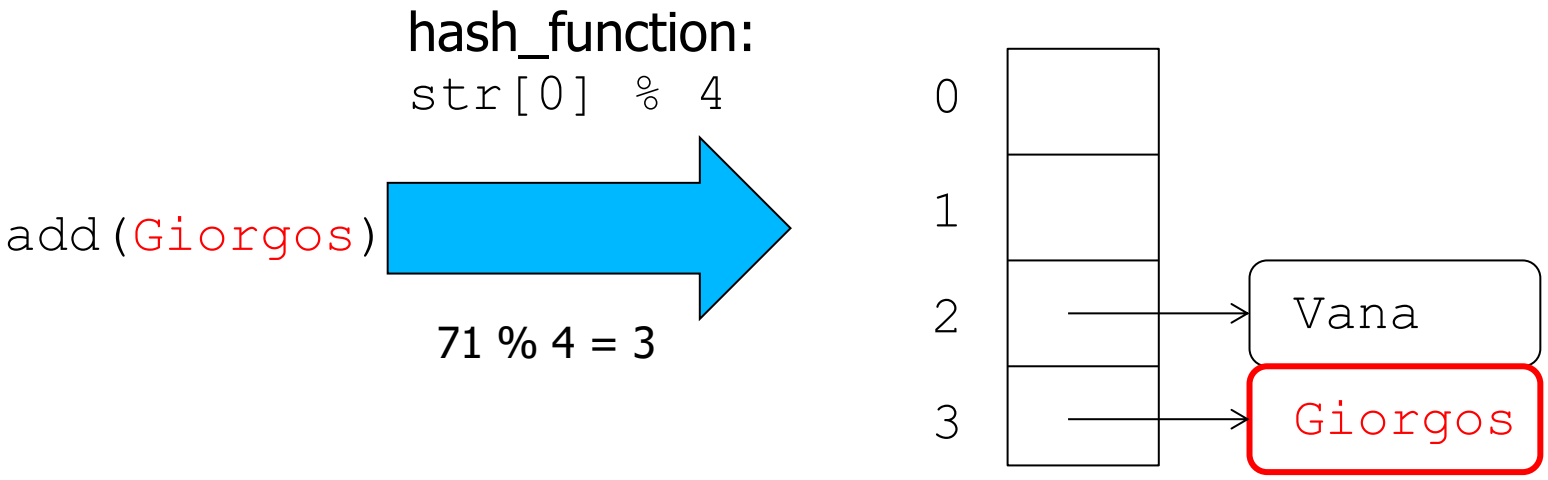
```
add(Vana)
```

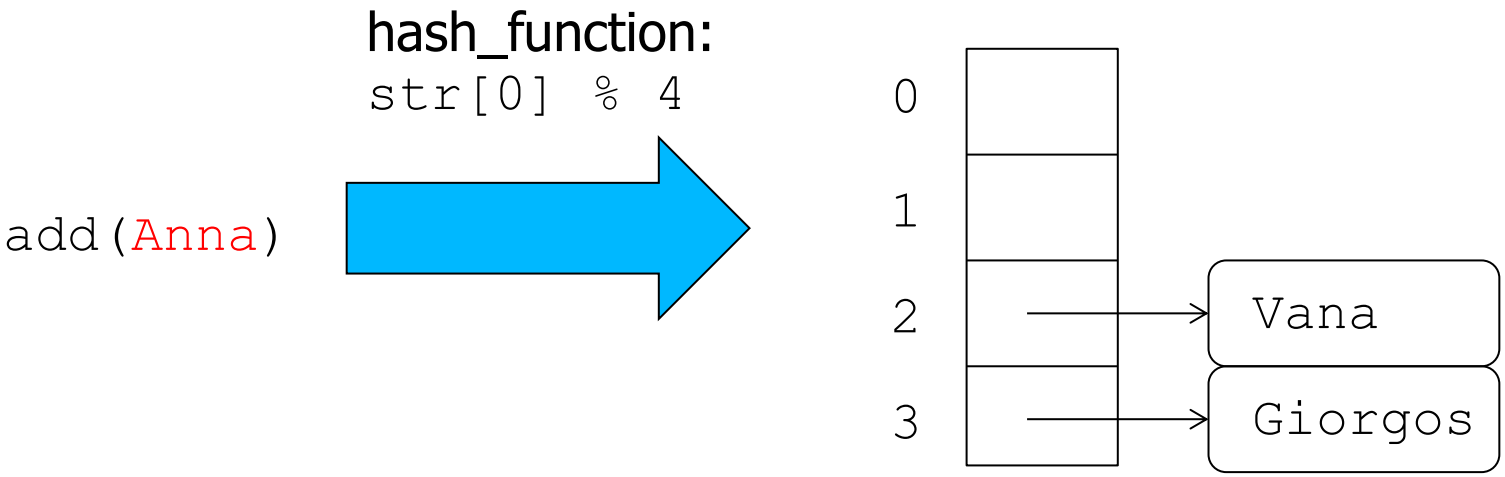


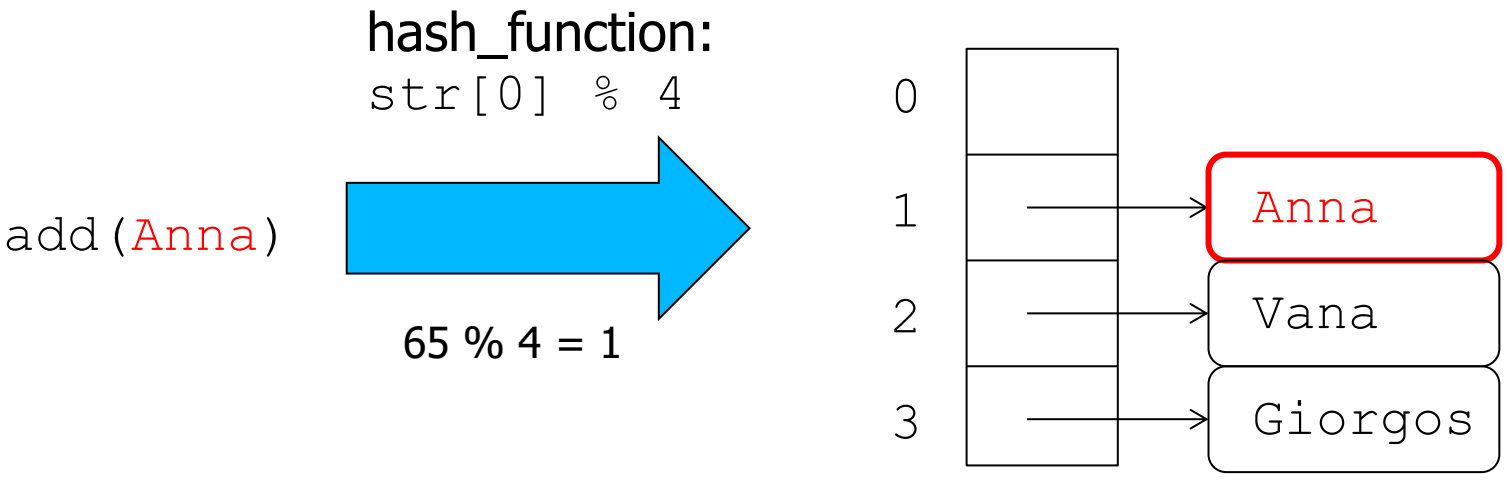


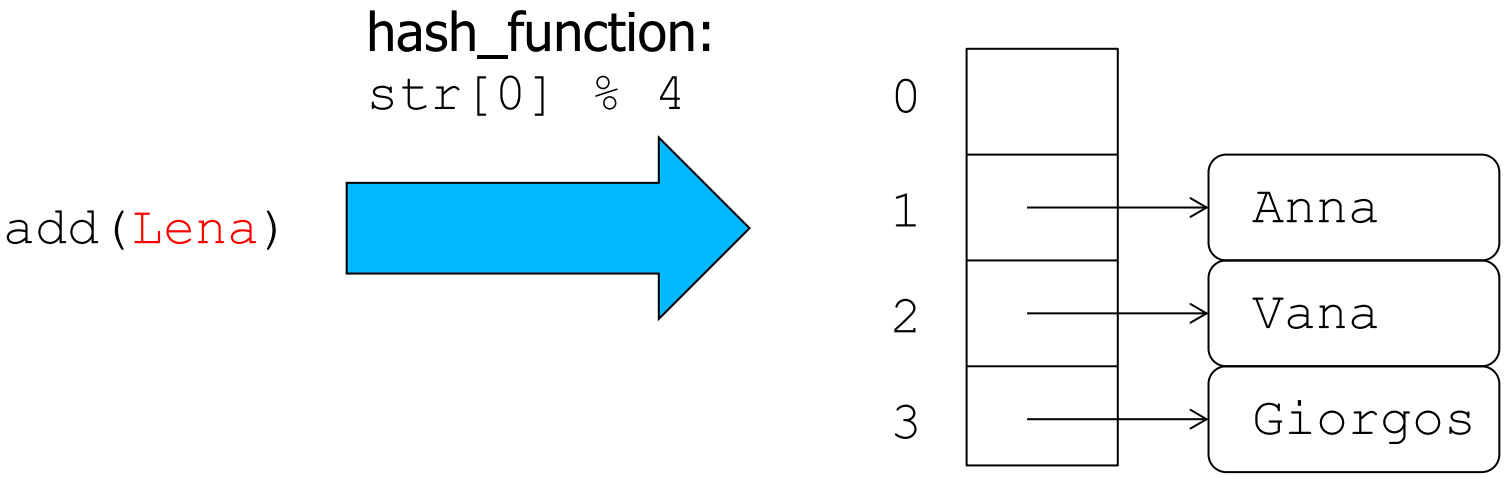

hash_function:
str[0] % 4
add(Vana)
86 % 4 = 2
0
1
2
3
Vana

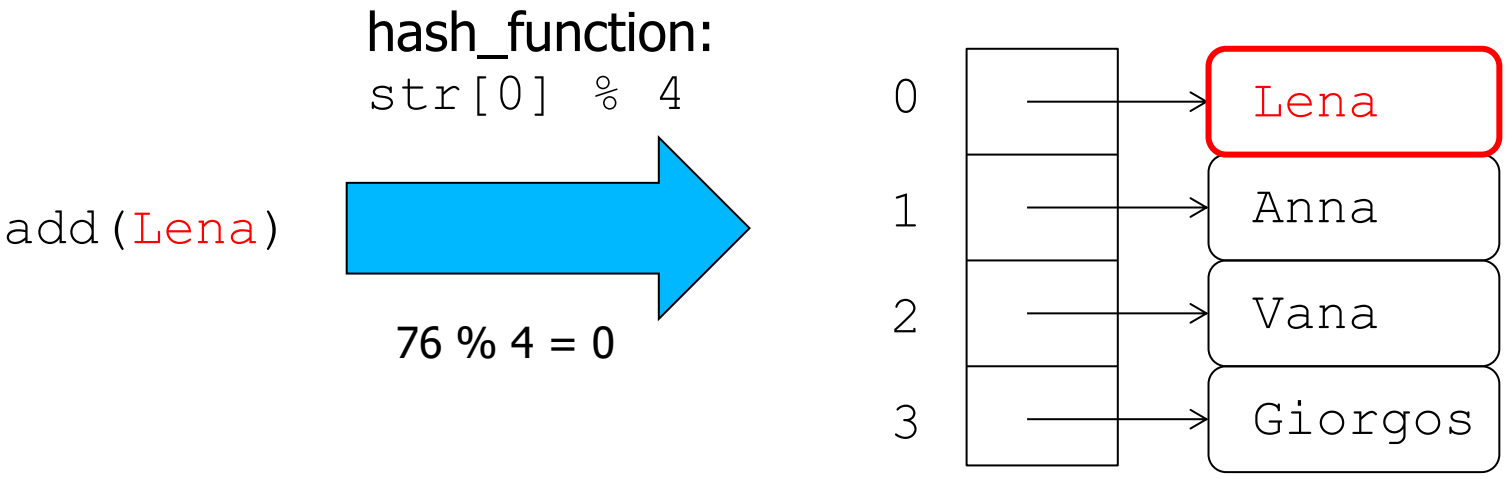

hash_function:
str[0] % 4
add(Giorgos)
0
1
2
3
Vana


hash_function:
str[0] % 4
add(Giorgos)
71 % 4 = 3
0
1
2
3
Vana
Giorgos


hash_function:
str[0] % 4
add(Anna)
0
1
2
3
Vana
Giorgos


hash_function:
str[0] % 4
add(Anna)
65 % 4 = 1
0
1
2
3
Anna
Vana
Giorgos


hash_function:
str[0] % 4
add(Lena)
0
1
2
3
Anna
Vana
Giorgos

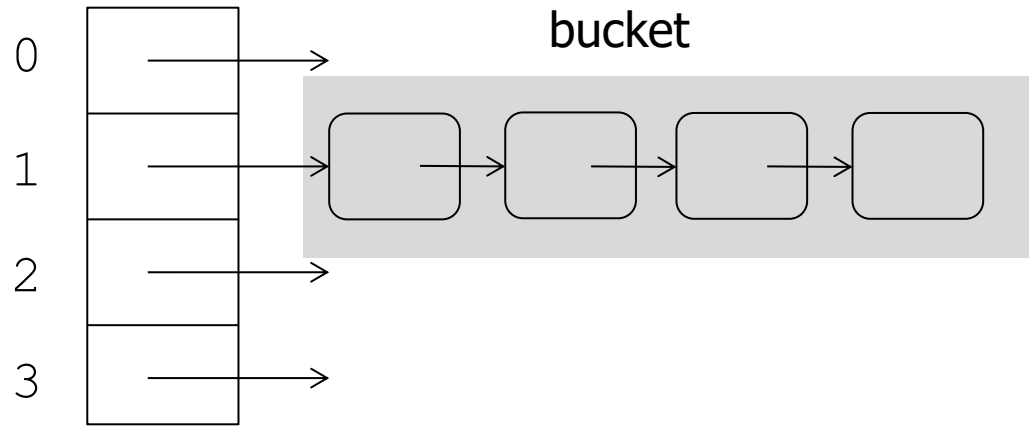

hash_function:
str[0] % 4
add(Lena)
76 % 4 = 0
0
1
2
3
Lena
Anna
Vana
Giorgos

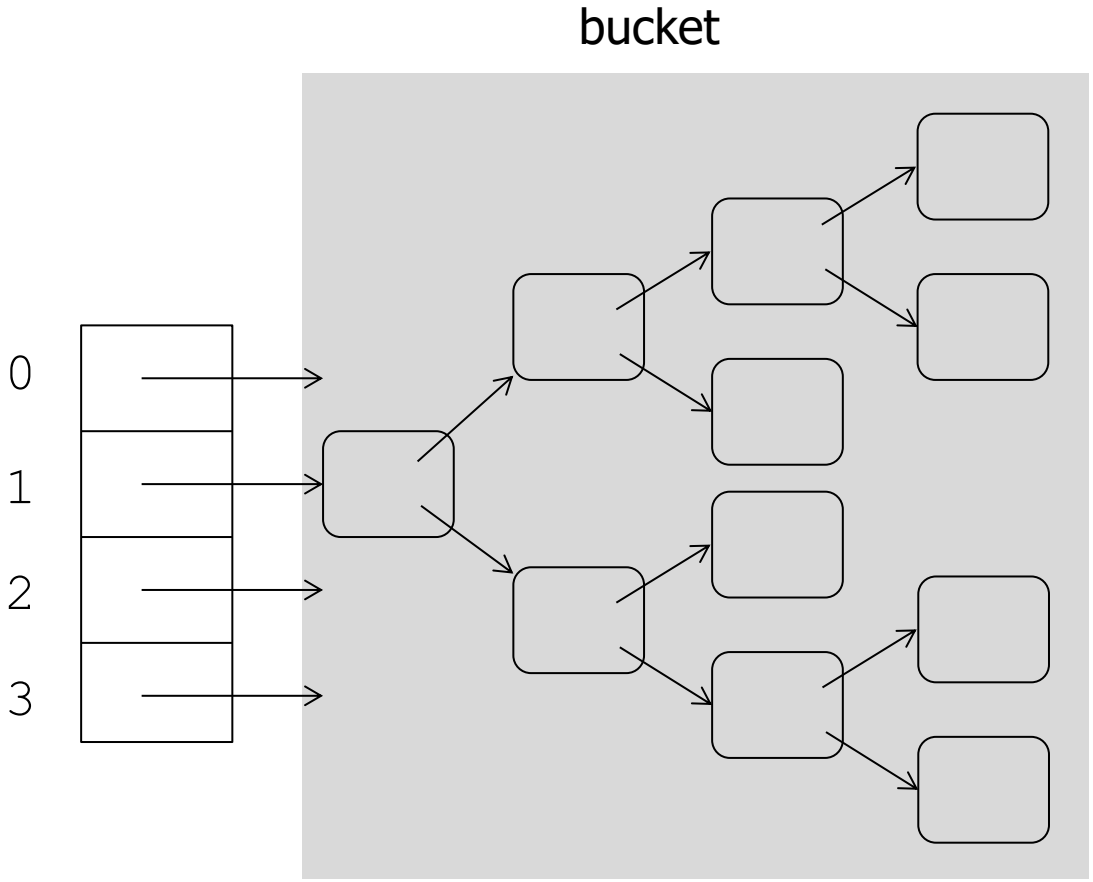
Συγκρούσεις – Δοχεία

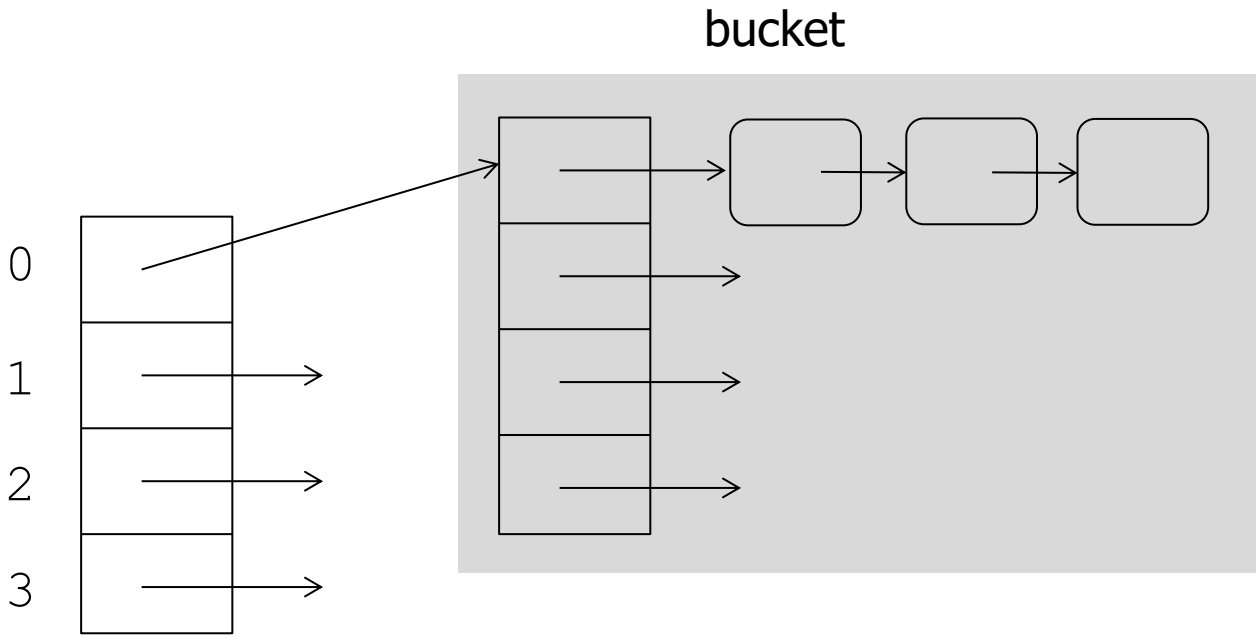
- Η συνάρτηση κατακερματισμού μπορεί να επιστρέψει την **ίδια τιμή** για **διαφορετικές** τιμές κλειδιών
- Αυτό ονομάζεται **σύγκρουση** (collision)

- Οι εγγραφές με ίδια hash values αντιστοιχίζονται στο **ίδιο** δοχείο
- Πρέπει να υποστηριχθεί προσθήκη, αναζήτηση και απομάκρυνση εγγραφών μέσα **στα δοχεία**
  - με βάση την τιμή του κλειδιού (αφού το hash value είναι το ίδιο)

- Τα δοχεία υλοποιούνται με **δομές αναζήτησης**
- Δυναμικός πίνακας, λίστα, δέντρο ή άλλο ένα hash table!


bucket
0
1
2
3

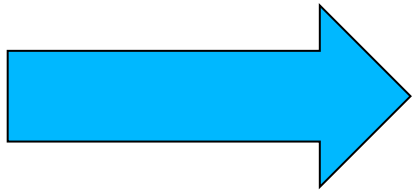

bucket
0
1
2
3

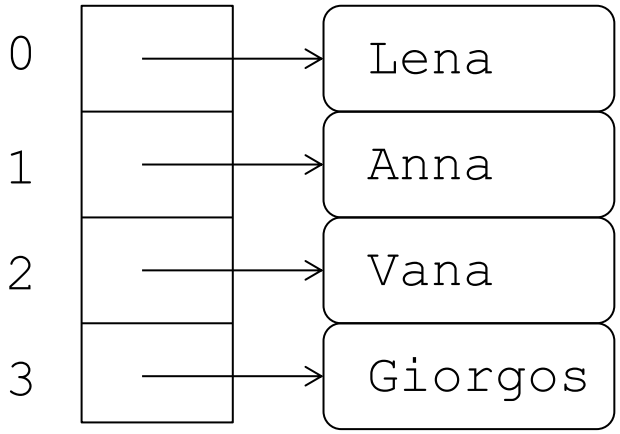

bucket
0
1
2
3

hash_function:

```
str[0] % 4
```

```
add(Manos)
```

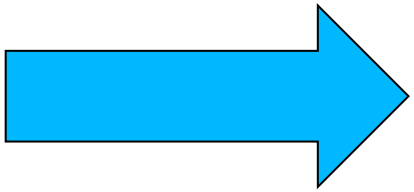




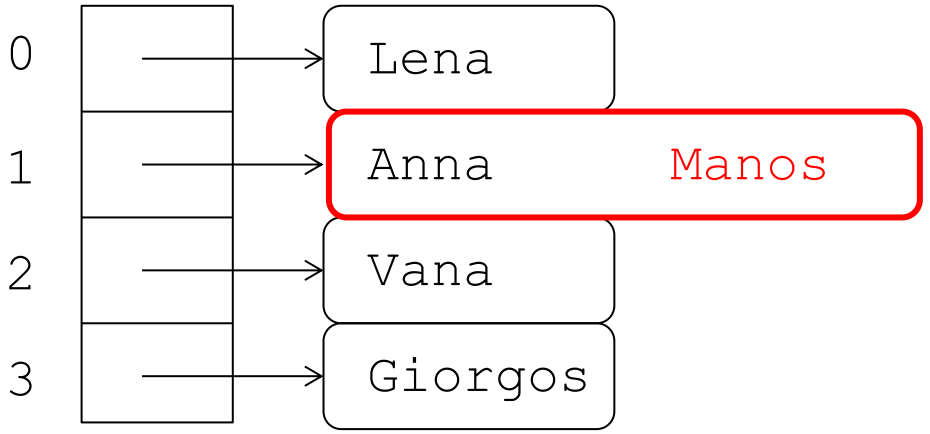
hash_function:

```
str[0] % 4
```

add(Manos)



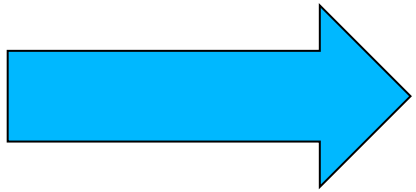
77 % 4 = 1

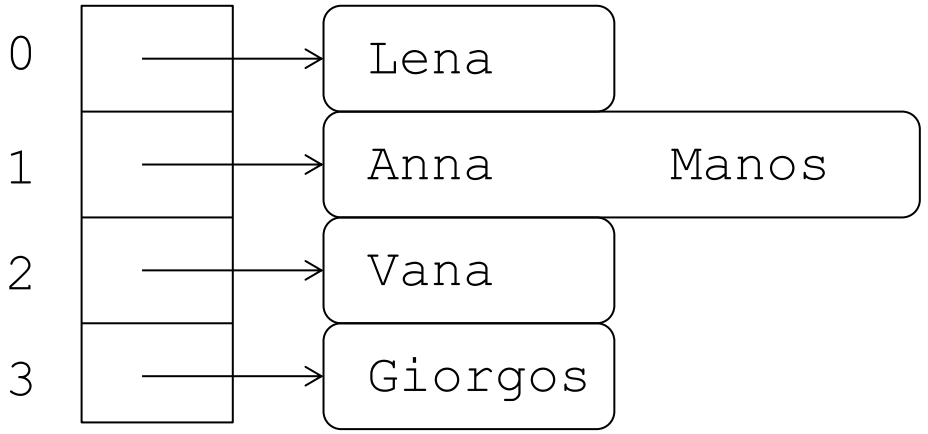


hash_function:

```
str[0] % 4
```

```
add(Katia)
```

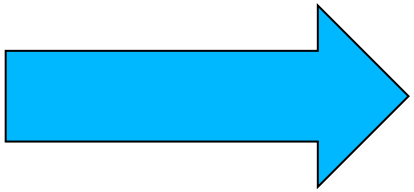




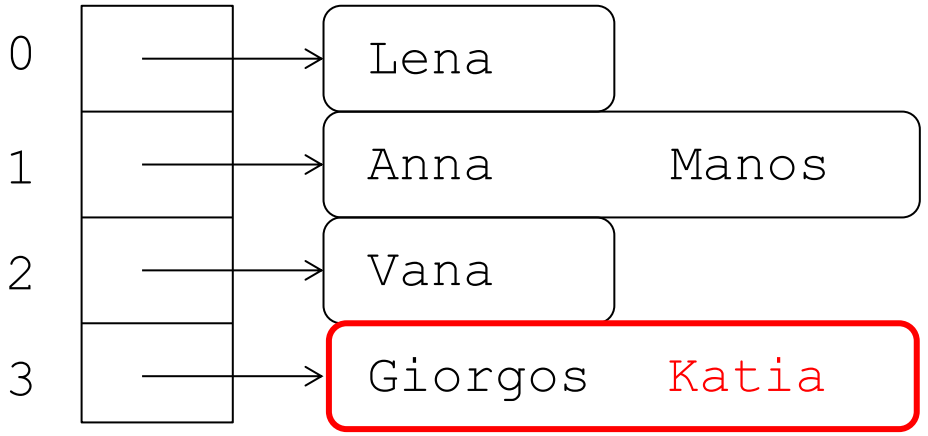
hash_function:

```
str[0] % 4
```

```
add(Katia)
```



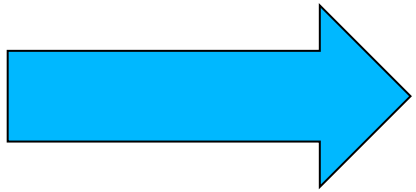
75 % 4 = 3

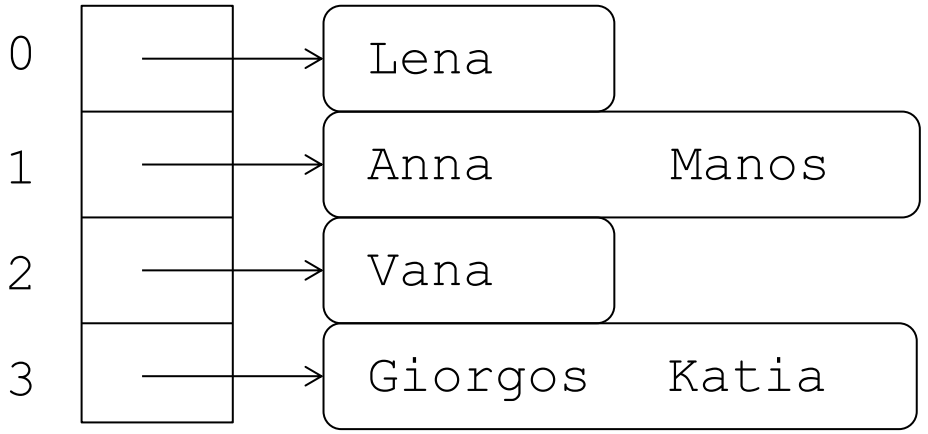


hash_function:

```
str[0] % 4
```

```
add(Babis)
```

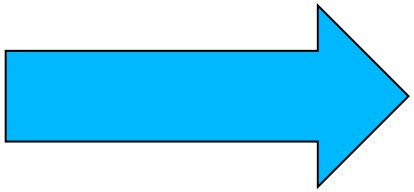




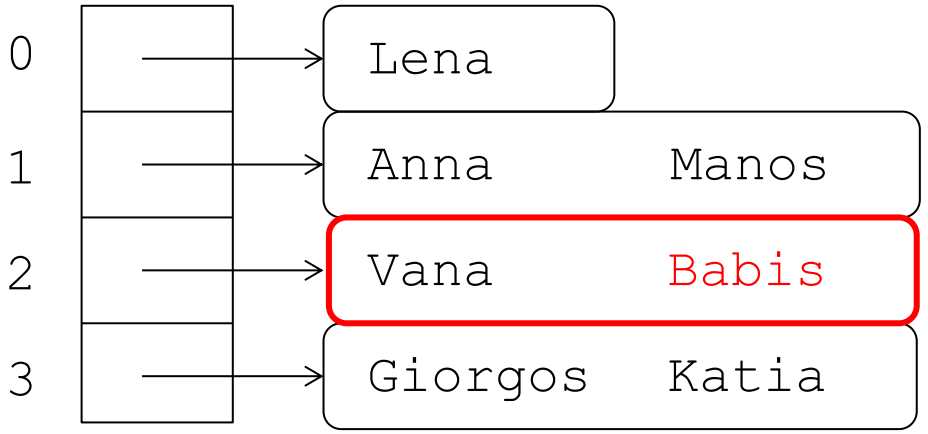
hash_function:

```
str[0] % 4
```

```
add(Babis)
```



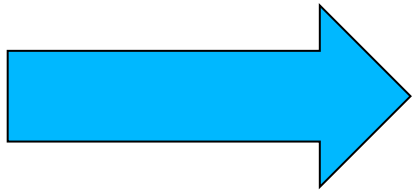
66 % 4 = 2

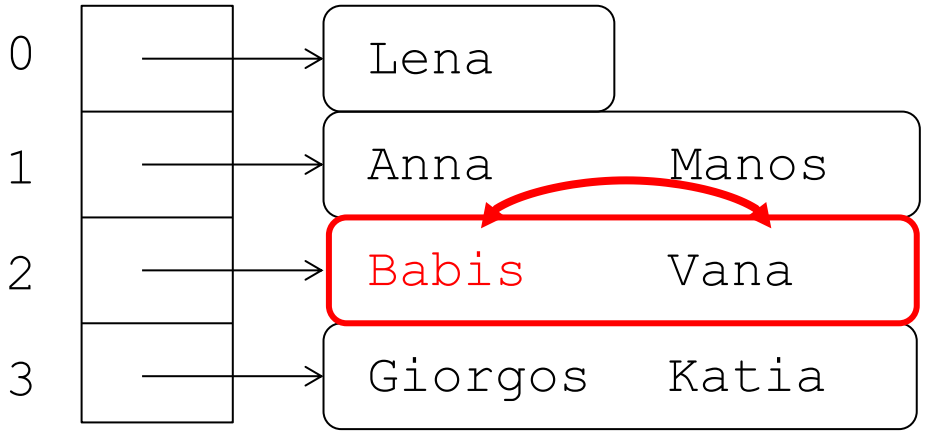


hash_function:

```
str[0] % 4
```

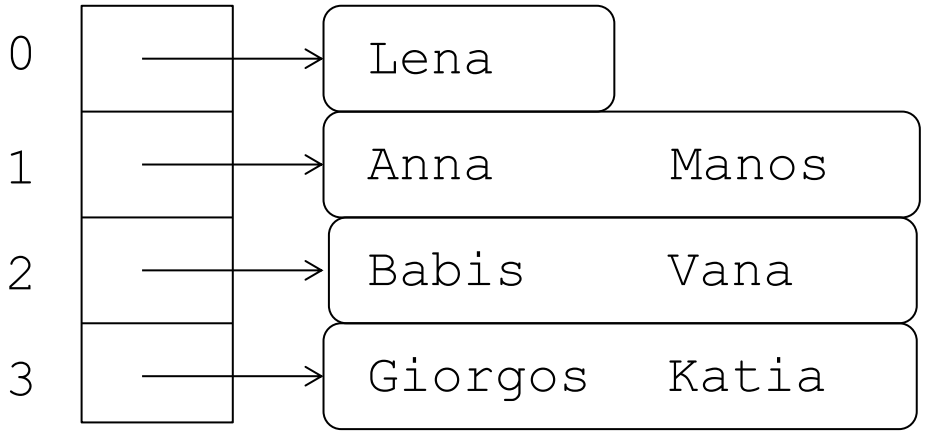
add(Babis)





hash_function:

```
str[0] % 4
```



Απόδοση / κόστος αναζήτησης

Καλή απόδοση

- Ο πίνακας έχει **αρκετά μεγάλο** μέγεθος
- Χρησιμοποιείται **καλή** συνάρτηση κατακερματισμού
  - υπάρχει **ομοιόμορφη** κατανομή των εγγραφών στα δοχεία
- Κάθε δοχείο έχει πολύ **λίγες** εγγραφές

=> κόστος αναζήτησης: $O(1)$

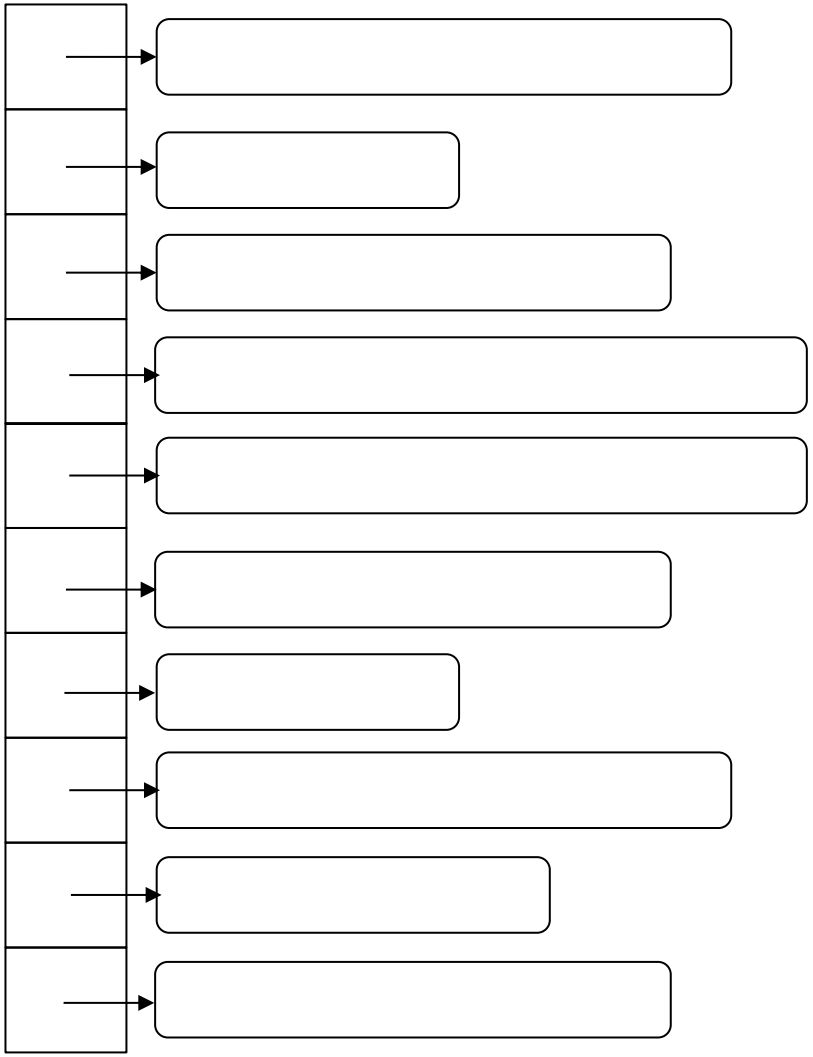
Κακή απόδοση

- Ο πίνακας έχει **μικρό** μέγεθος
  - νομοτελειακά όλα τα δοχεία θα έχουν πολλές εγγραφές
- Χρησιμοποιείται **κακή** συνάρτηση κατακερματισμού
  - κάποια δοχεία έχουν πολλές εγγραφές ενώ άλλα πολύ λίγες

=> κόστος αναζήτησης: αναζήτηση μέσα στα δοχεία

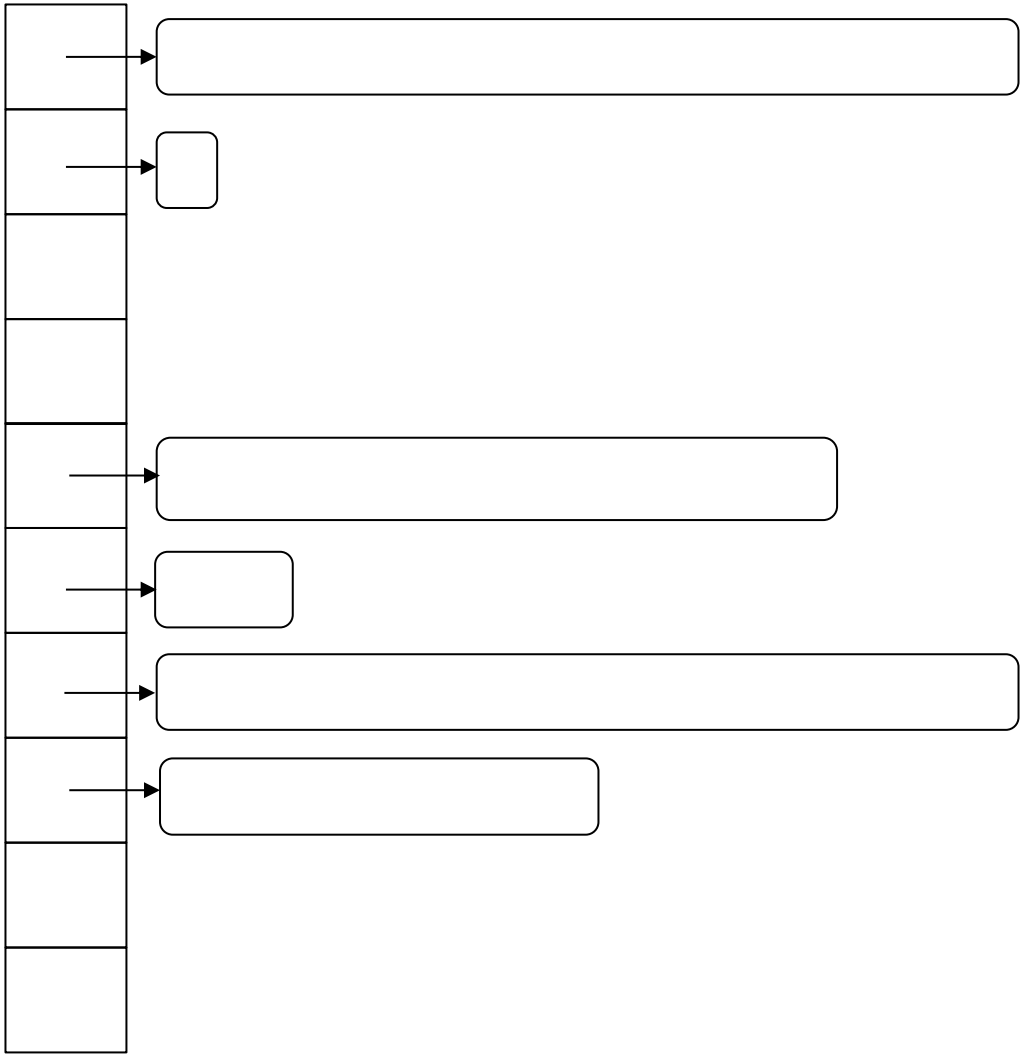
καλή κατανομή

σχετικά **μικρό** μέγεθος πίνακα



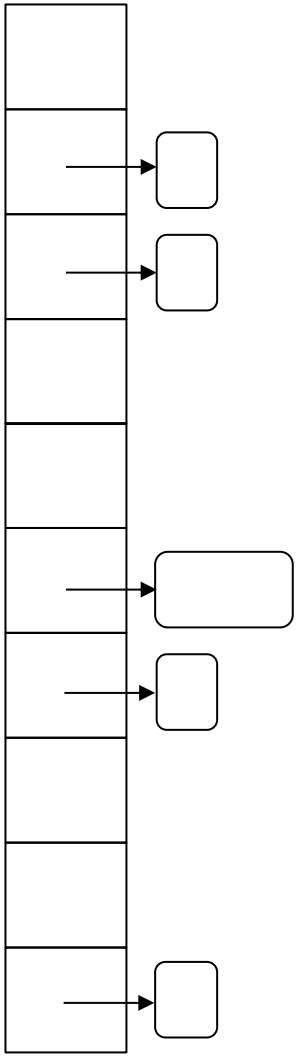
κακή κατανομή

σχετικά **επαρκές** μέγεθος πίνακα



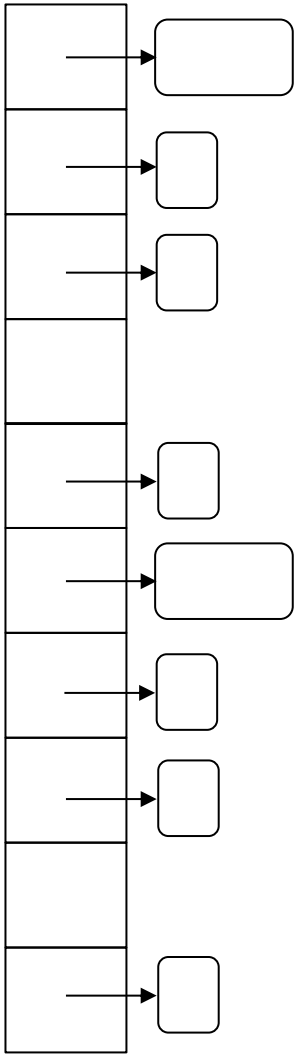
καλή κατανομή

σχετικά **μεγάλο** μέγεθος πίνακα



καλή κατανομή

καλό μέγεθος πίνακα



Μετρικές

- Μέγεθος του πίνακα (N)
- Αριθμός εισαγωγών (E)
- Μέγεθος δοχείων (B)

- **Load factor (LF):** E/N
  - $<< 1$ => ο πίνακας είναι υπο-φορτωμένος
  - > 1 => ο πίνακας είναι υπερ-φορτωμένος

- **Max/mean/average** B
  - $>> 1$ => σχετικά αργή αναζήτηση μέσα στα δοχεία
    - αναλόγως με την δομή που χρησιμοποιείται εσωτερικά
  - ~ 1 => γρήγορη αναζήτηση μέσα στα δοχεία

Τακτικές βελτίωσης απόδοσης

- Δυναμική αύξηση του μεγέθους του πίνακα
  - όταν $LF > 1$
  - για να επιτευχθεί μείωση του μεγέθους δοχείων
- Δυναμική μείωση του μεγέθους του πίνακα
  - όταν $LF << 1$
  - για να αποφευχθεί σπατάλη μνήμης
- Αλλαγή της συνάρτησης κατακερματισμού
  - όταν $\min/\max B >> 1$ και $LF < 1$
  - για να επιτευχθεί καλύτερη κατανομή ανάμεσα στα δοχεία

Ανακατακερματισμός (rehashing)

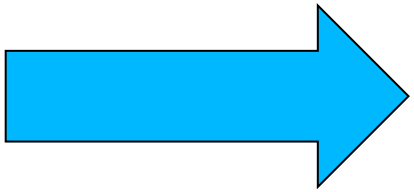
- Όταν αλλάξει το μέγεθος του πίνακα ή/και η συνάρτηση κατακερματισμού, πρέπει να γίνει **αναδιάταξη** των εγγραφών
- Αυτό ονομάζεται **ανακατακερματισμός** (rehashing)

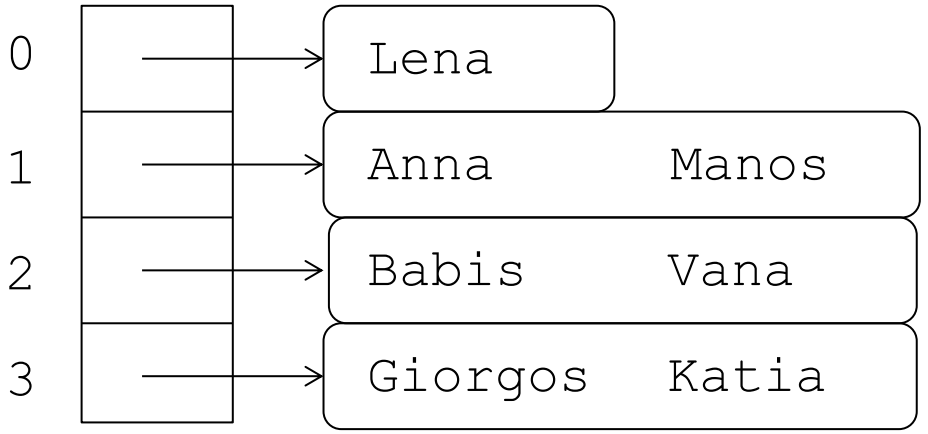
Απλή υλοποίηση

- Δημιουργία νέου πίνακα κατακερματισμού με άδεια δοχεία
- Αφαίρεση κάθε μιας εγγραφής από τον παλιό πίνακα και εισαγωγή της στον νέο πίνακα
  - μέχρι να αδειάσουν όλα τα δοχεία του παλιού πίνακα
- Καταστροφή του παλιού πίνακα

hash_function:

```
str[0] % 4
```

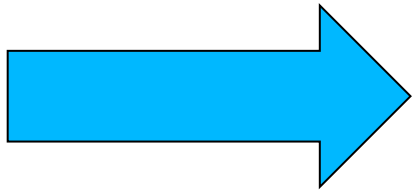


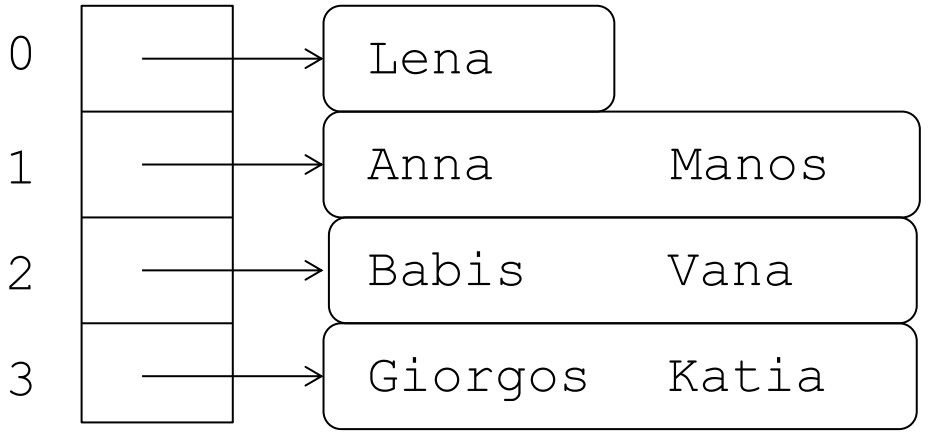


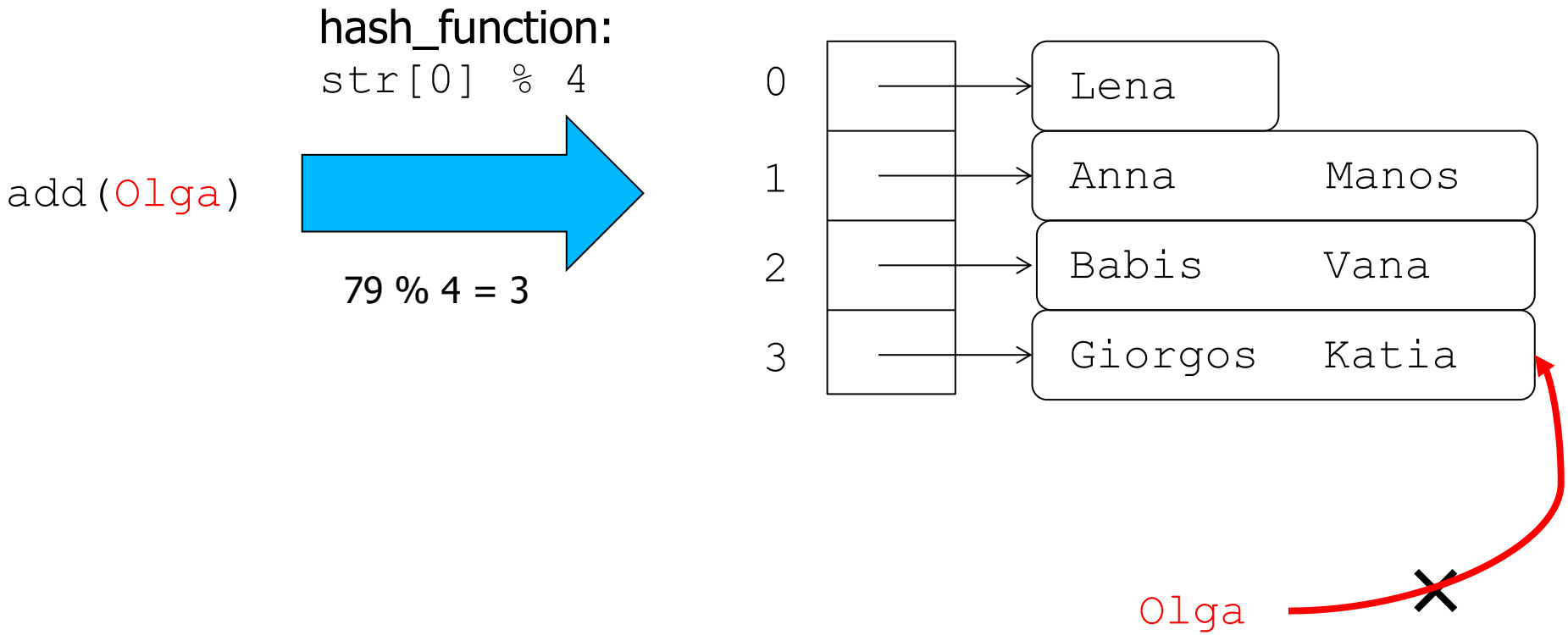
hash_function:

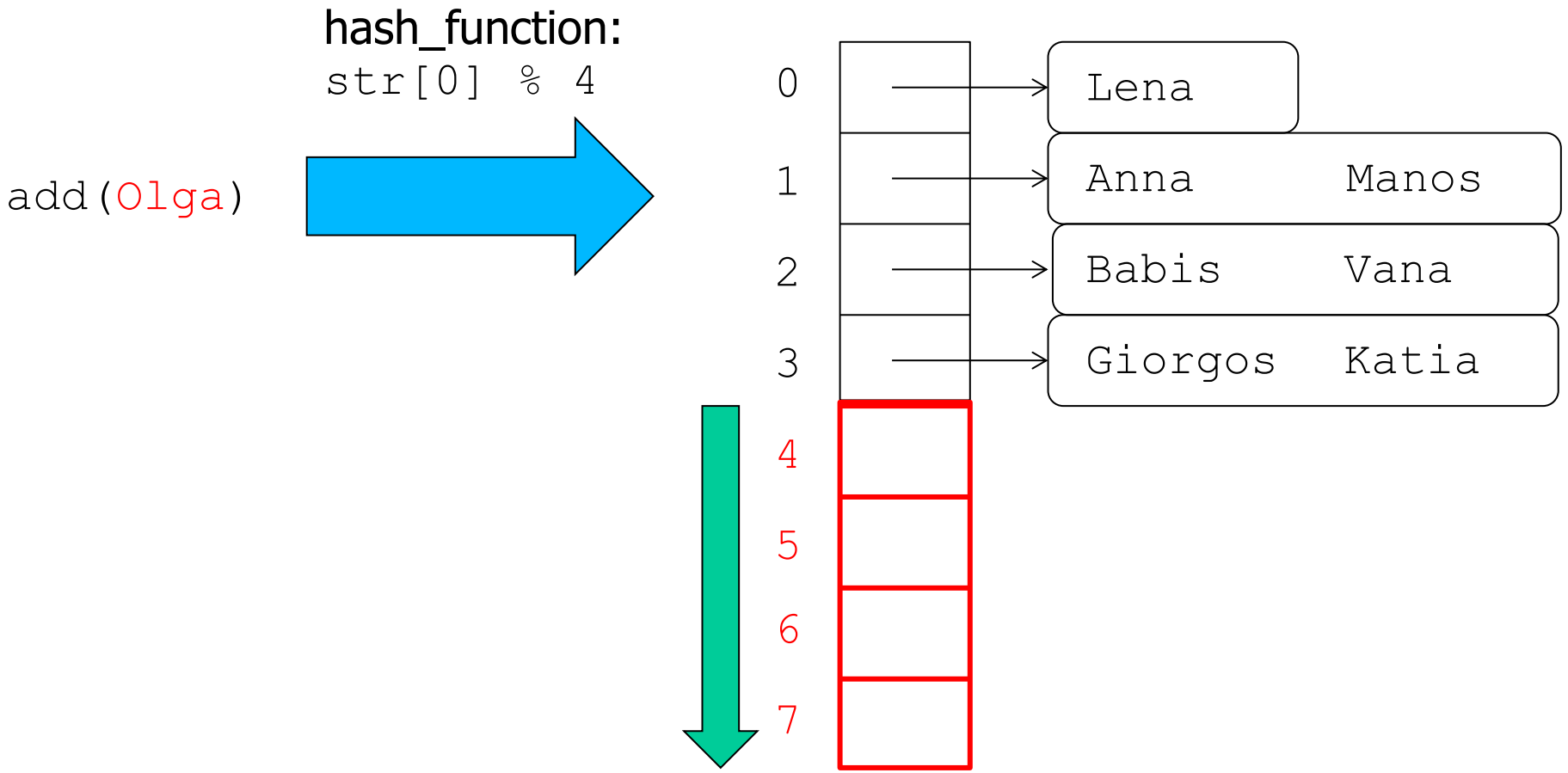
```
str[0] % 4
```

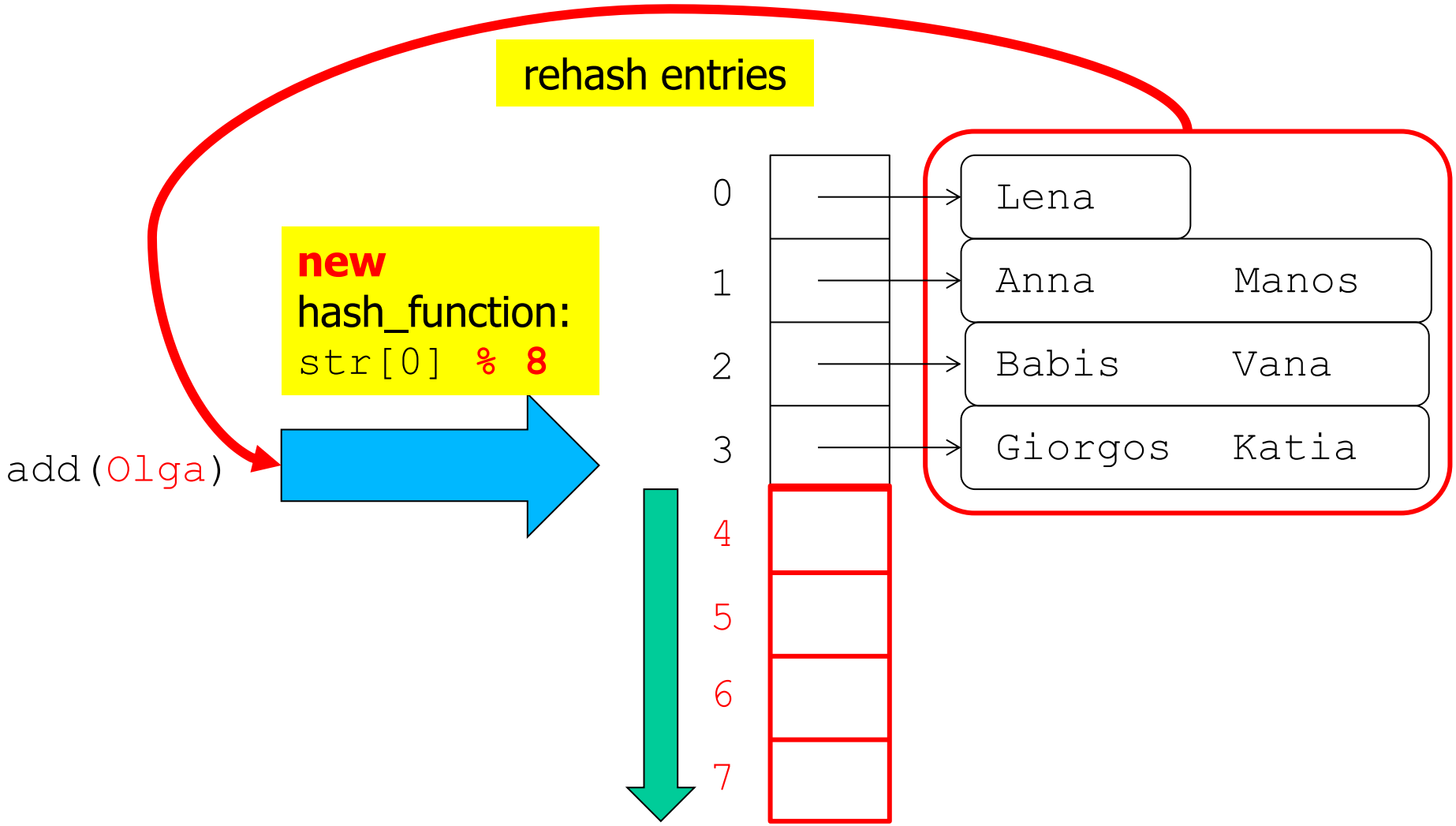
```
add(Olga)
```

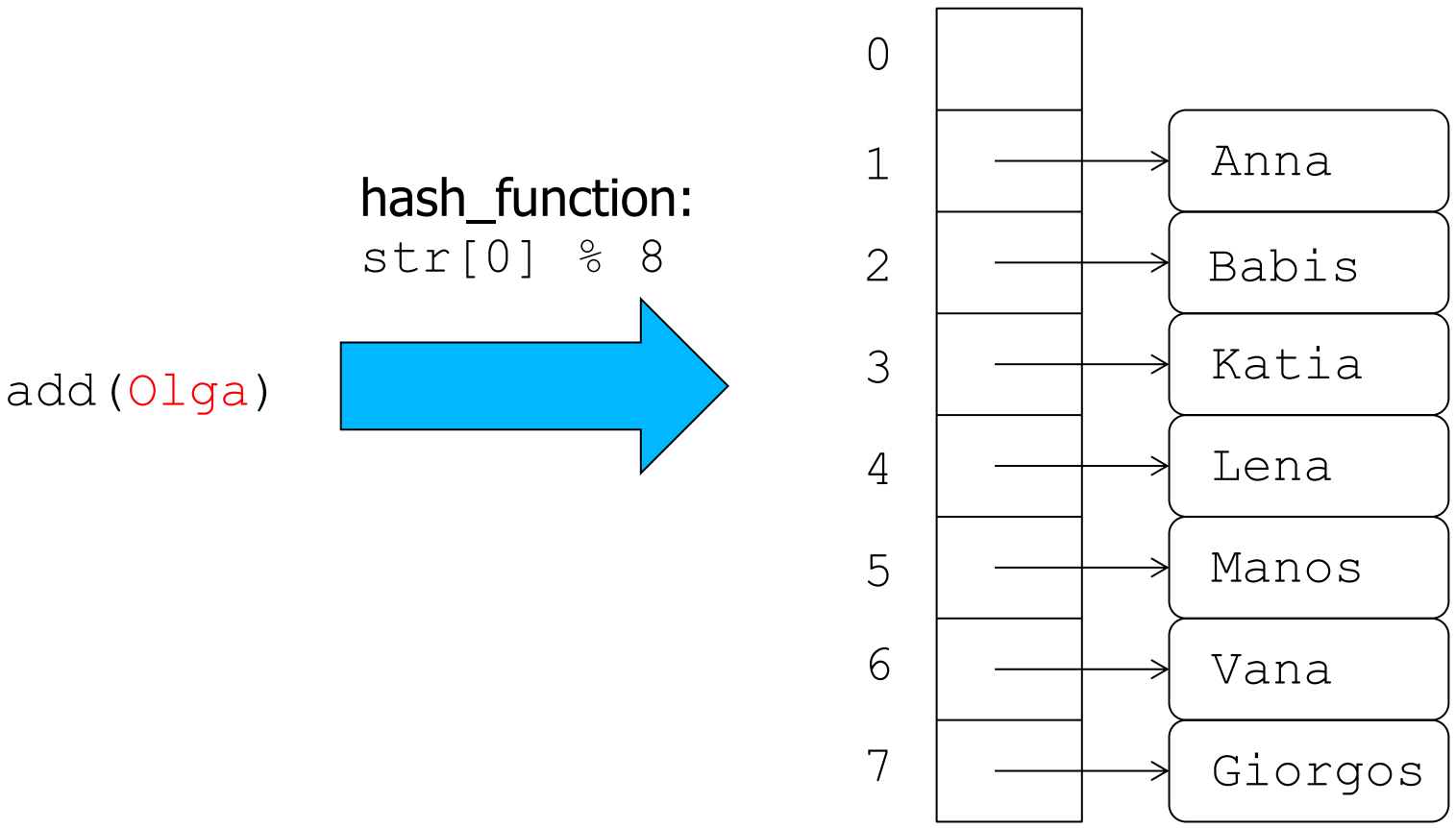


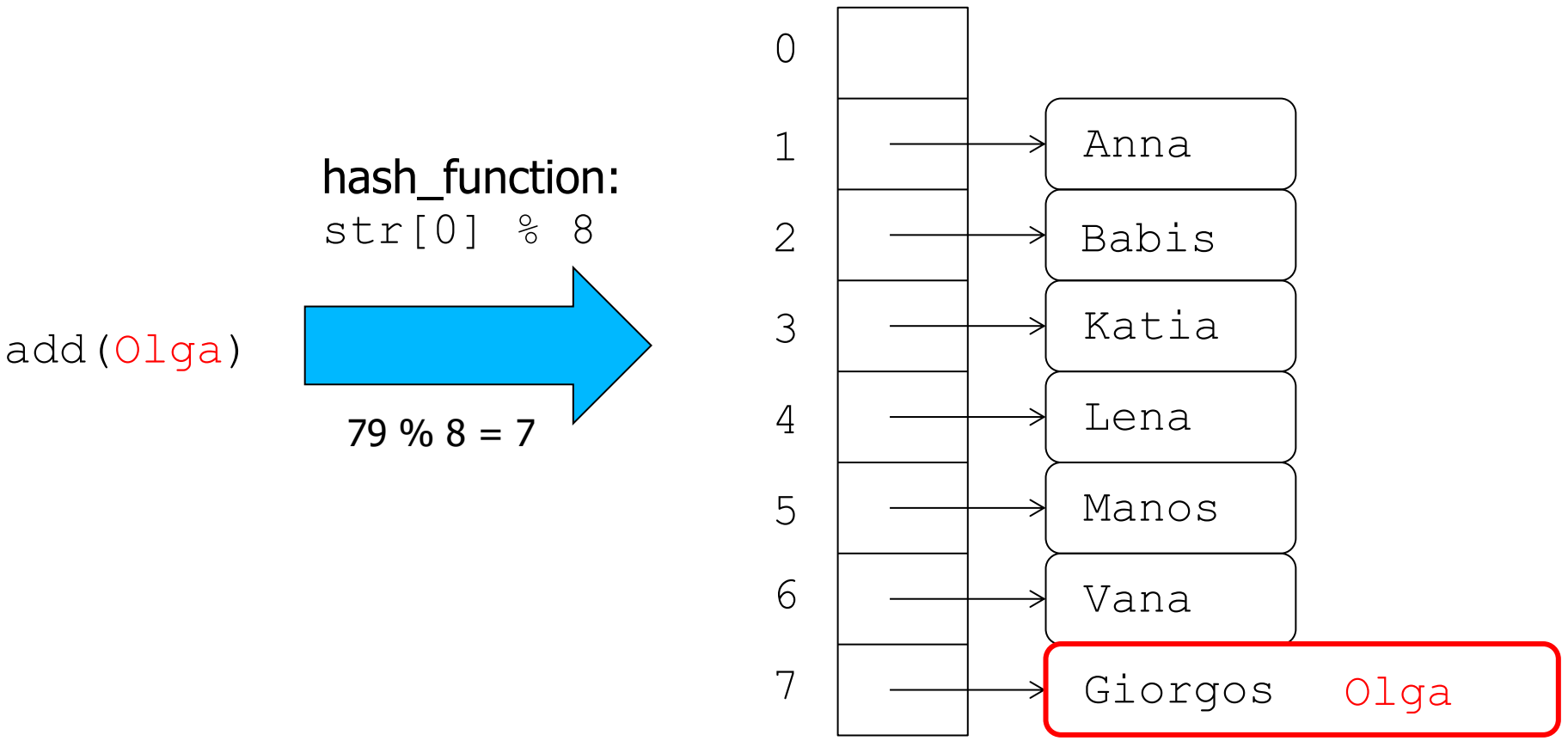


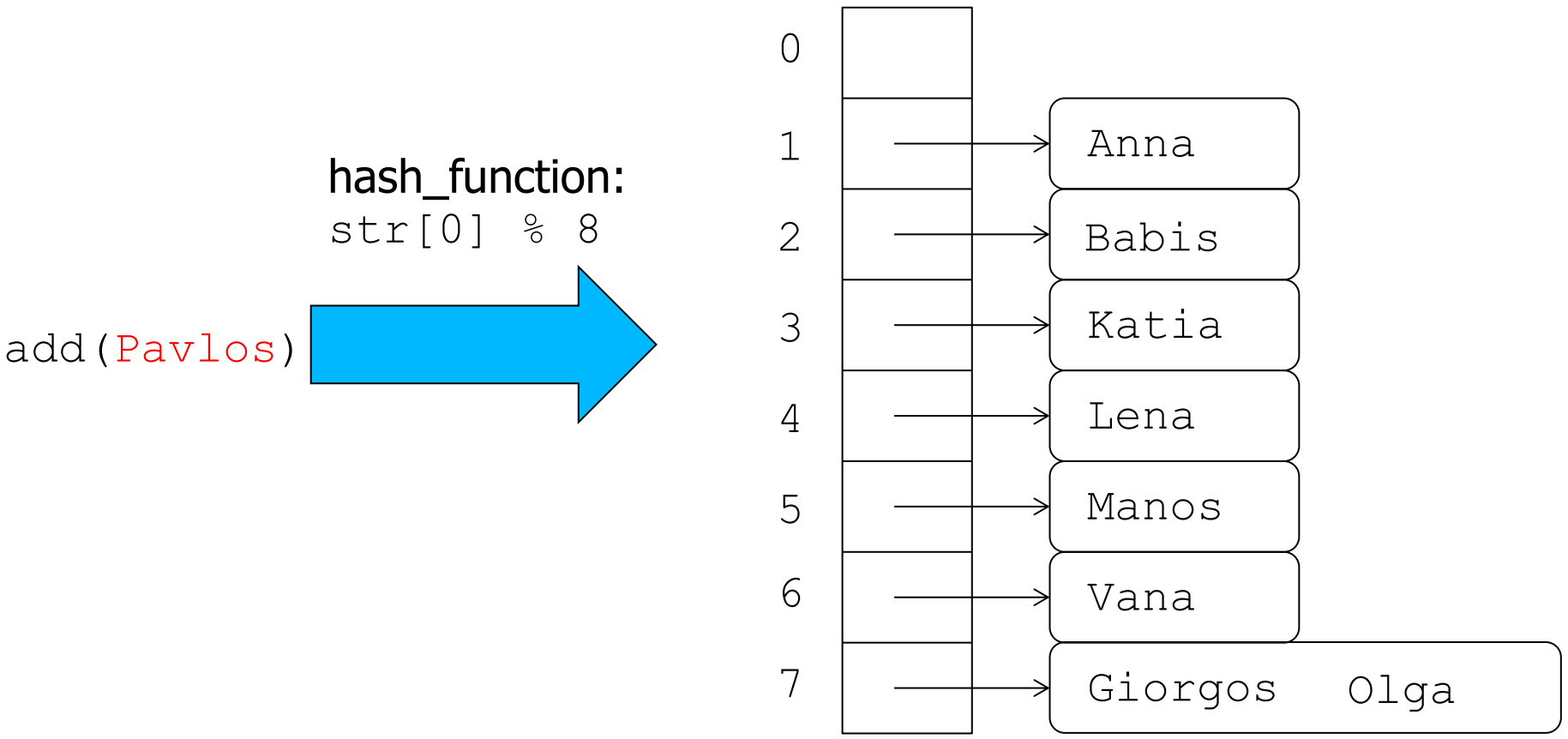

hash_function:
str[0] % 4
add(Olga)
79 % 4 = 3
0
1
2
3
Lena
Anna
Manos
Babis
Vana
Giorgos
Katia
Olga

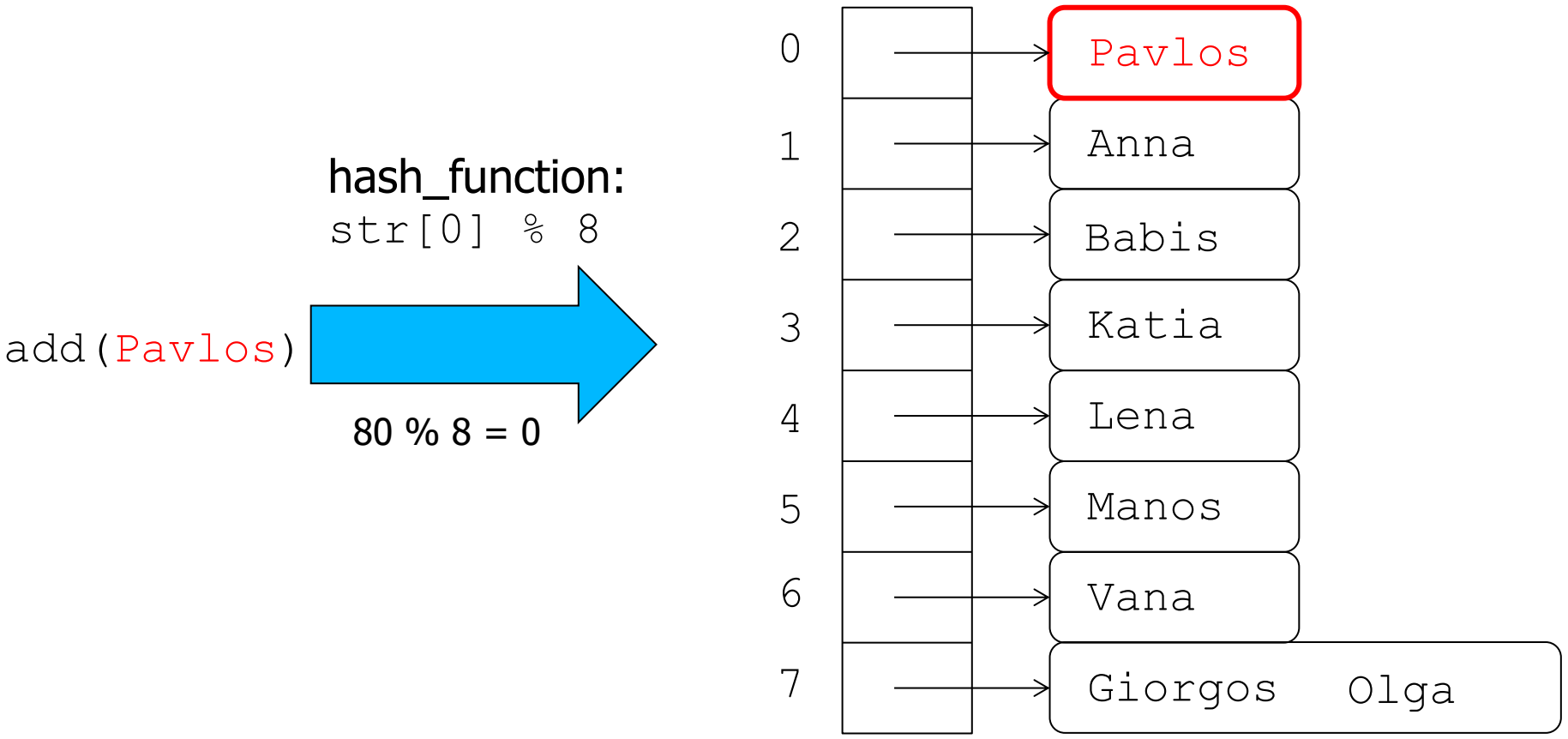

hash_function:
str[0] % 4
add(Olga)
0
1
2
3
4
5
6
7
Lena
Anna
Manos
Babis
Vana
Giorgos
Katia


rehash entries
new
hash_function:
str[0] % 8
add(Olga)
0
1
2
3
4
5
6
7
Lena
Anna
Manos
Babis
Vana
Giorgos
Katia


hash_function:
str[0] % 8
add(Olga)
0
1
2
3
4
5
6
7
Anna
Babis
Katia
Lena
Manos
Vana
Giorgos


hash_function:
str[0] % 8
add(Olga)
79 % 8 = 7
0
1
2
3
4
5
6
7
Anna
Babis
Katia
Lena
Manos
Vana
Giorgos
Olga

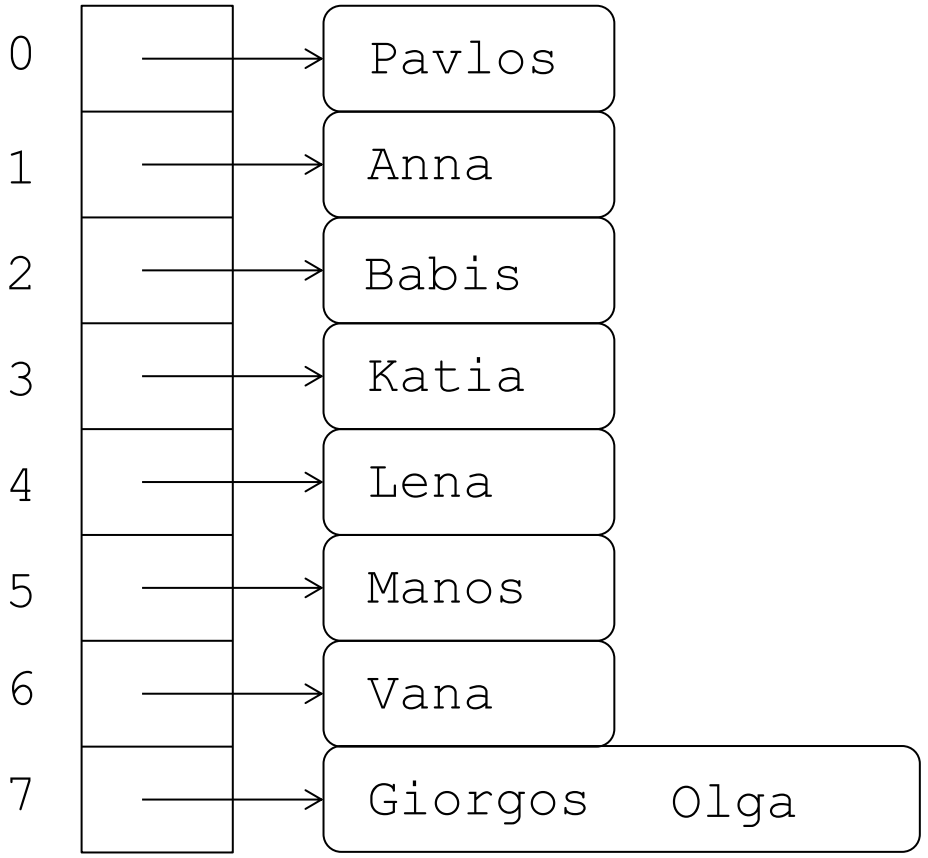

hash_function:
str[0] % 8
add(Pavlos)
0
1
2
3
4
5
6
7
Anna
Babis
Katia
Lena
Manos
Vana
Giorgos
Olga


hash_function:
str[0] % 8
add(Pavlos)
80 % 8 = 0
0
1
2
3
4
5
6
7
Pavlos
Anna
Babis
Katia
Lena
Manos
Vana
Giorgos
Olga

hash_function:

```
str[0] % 8
```

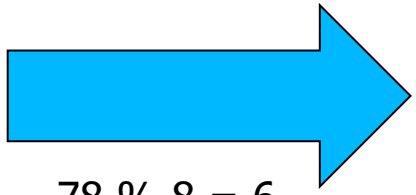
```
add(Nikos)
```



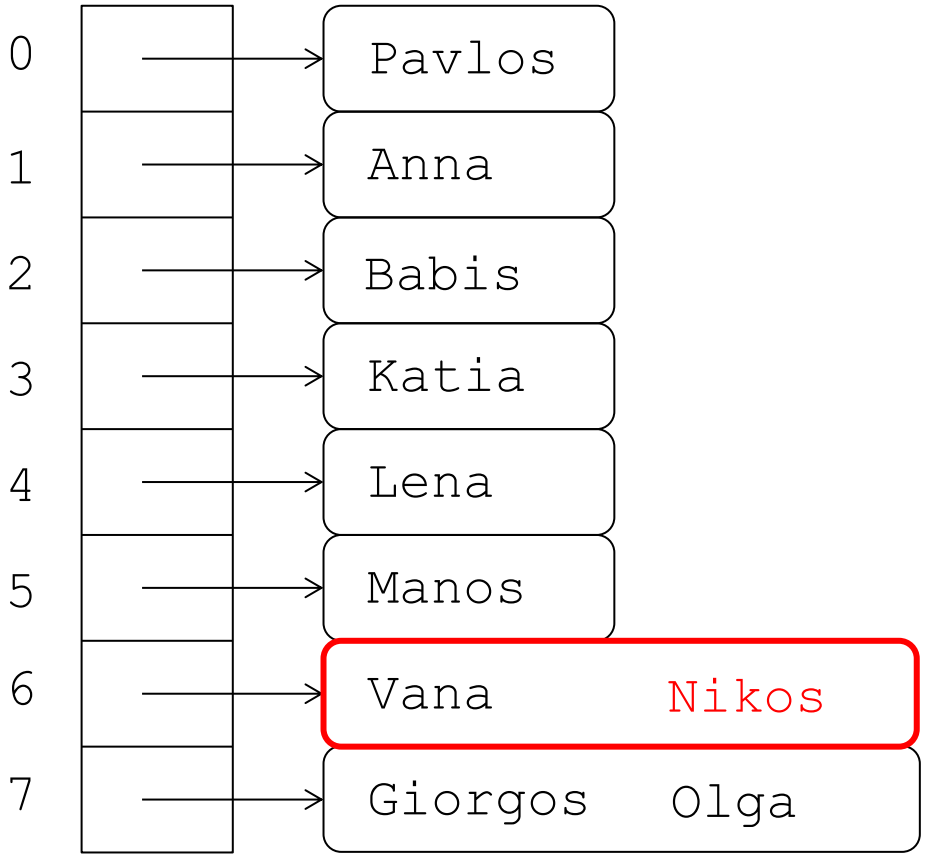
hash_function:

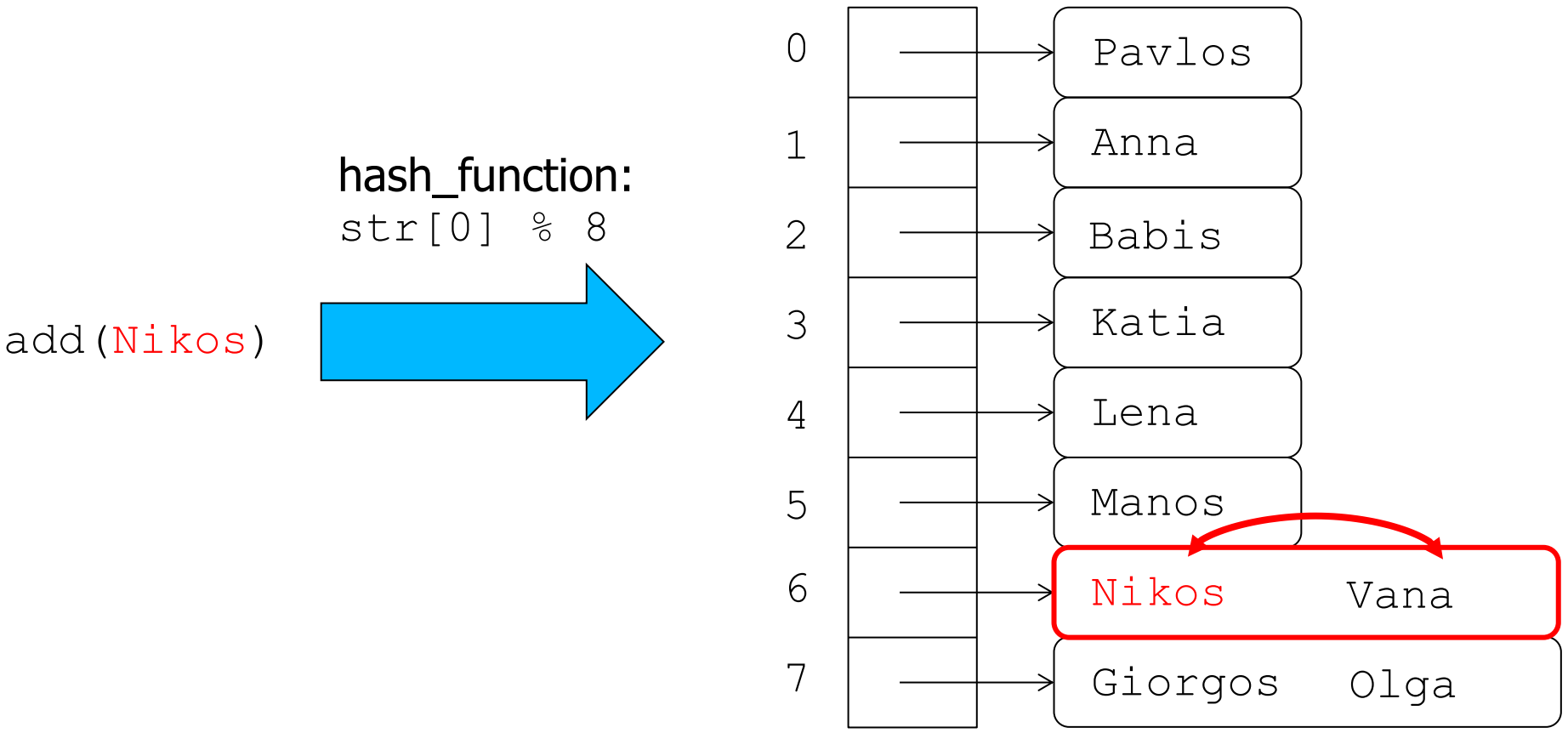
```
str[0] % 8
```

add(Nikos)



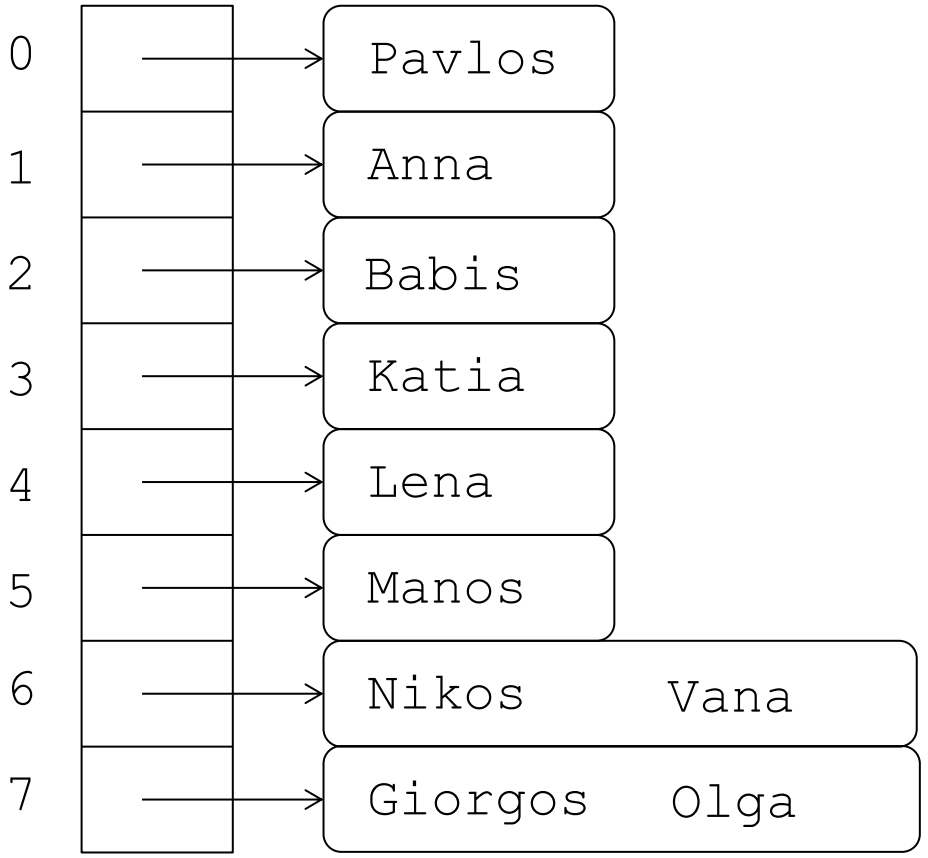
78 % 8 = 6




hash_function:
str[0] % 8
add(Nikos)
0
1
2
3
4
5
6
7
Pavlos
Anna
Babis
Katia
Lena
Manos
Nikos
Vana
Giorgos
Olga

hash_function:

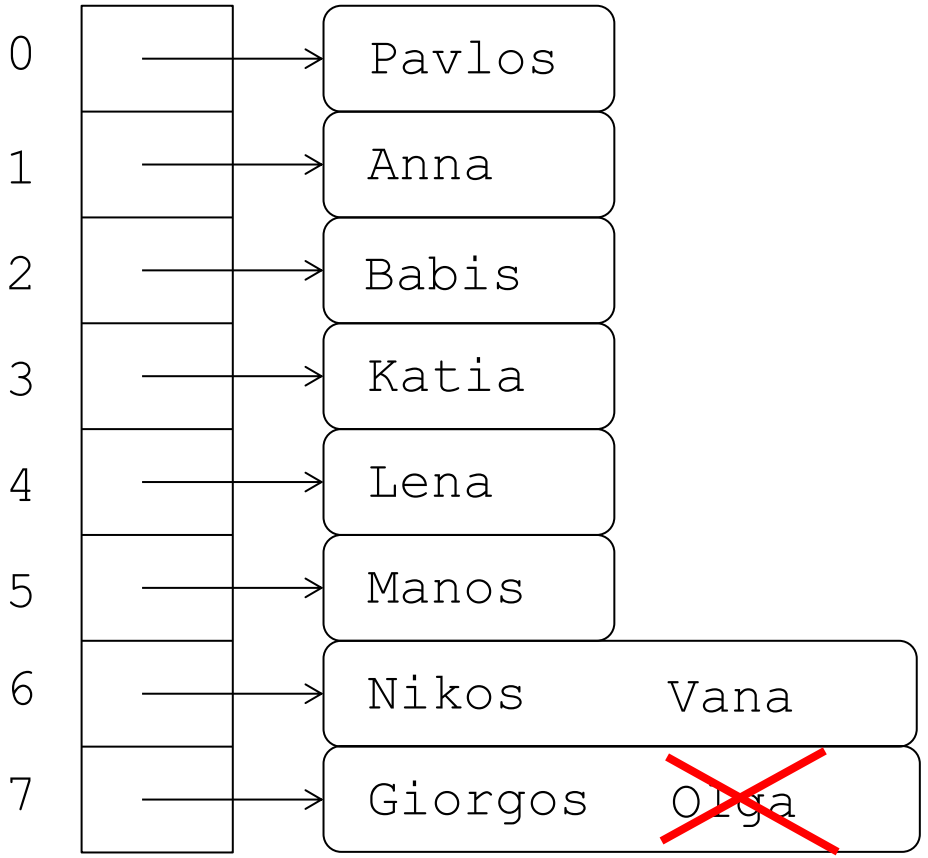
```
str[0] % 8
```



hash_function:

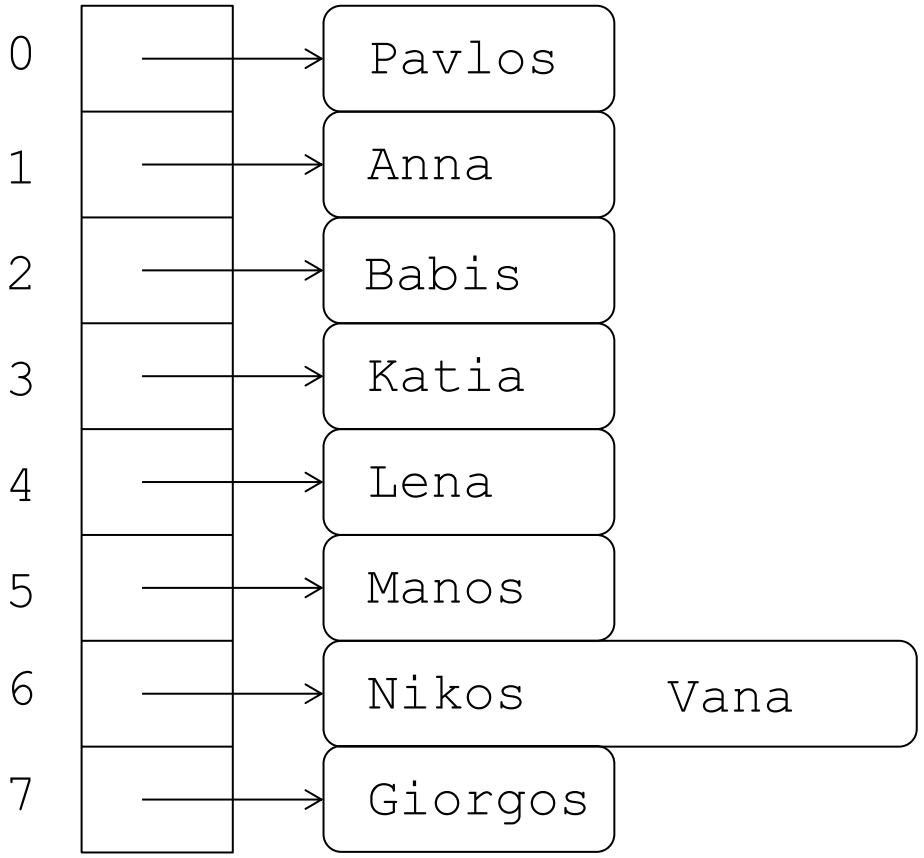
```
str[0] % 8
```

```
rmv(Olga)
```



hash_function:

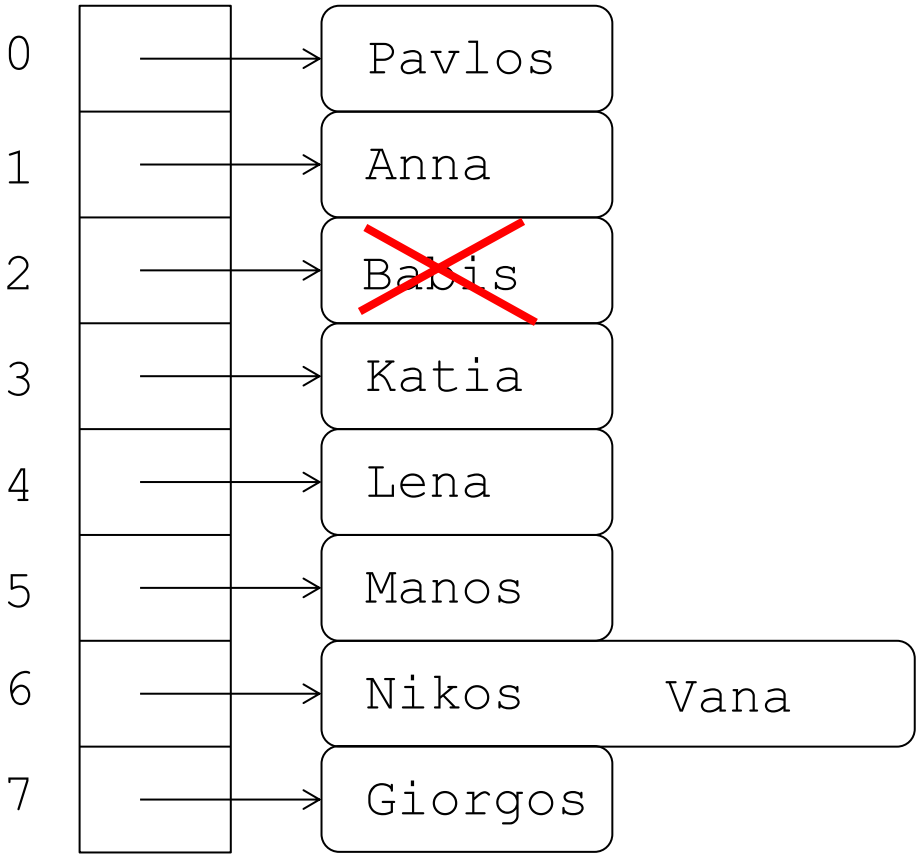
```
str[0] % 8
```



hash_function:

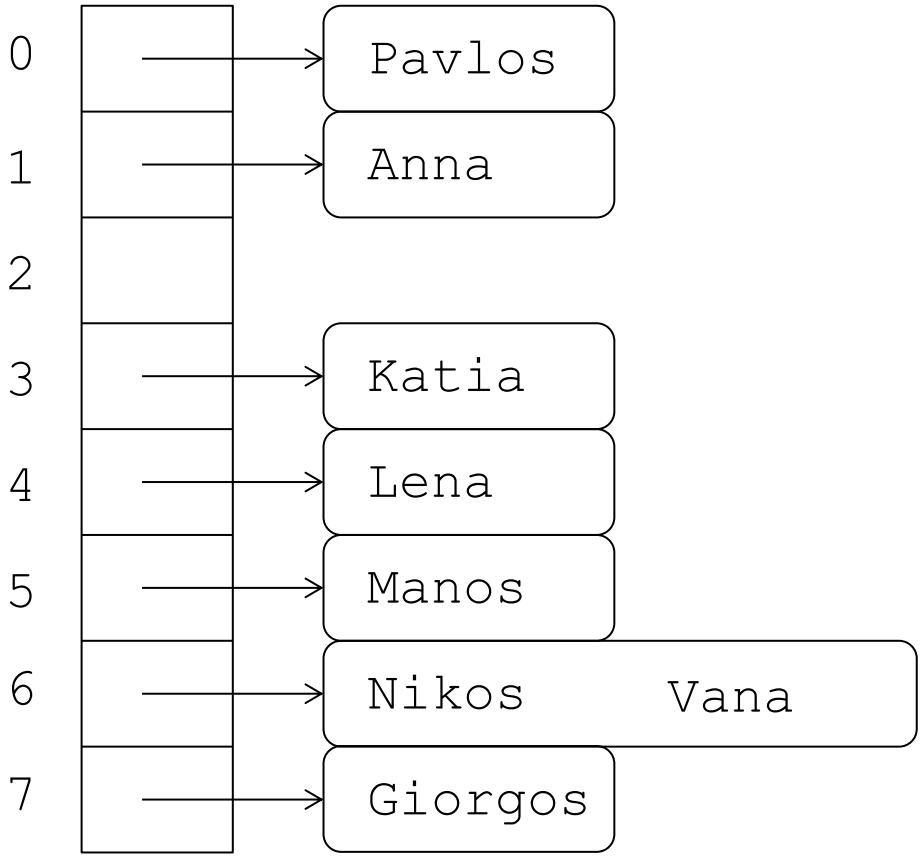
```
str[0] % 8
```

```
rmv(Basis)
```



hash_function:

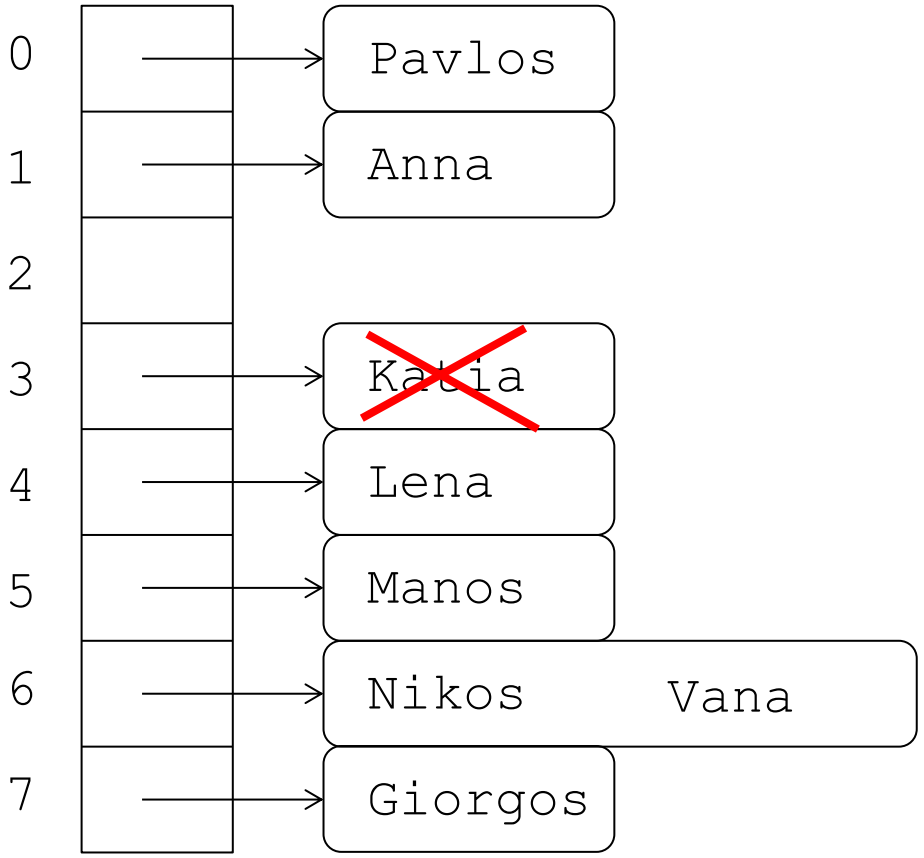
```
str[0] % 8
```



hash_function:

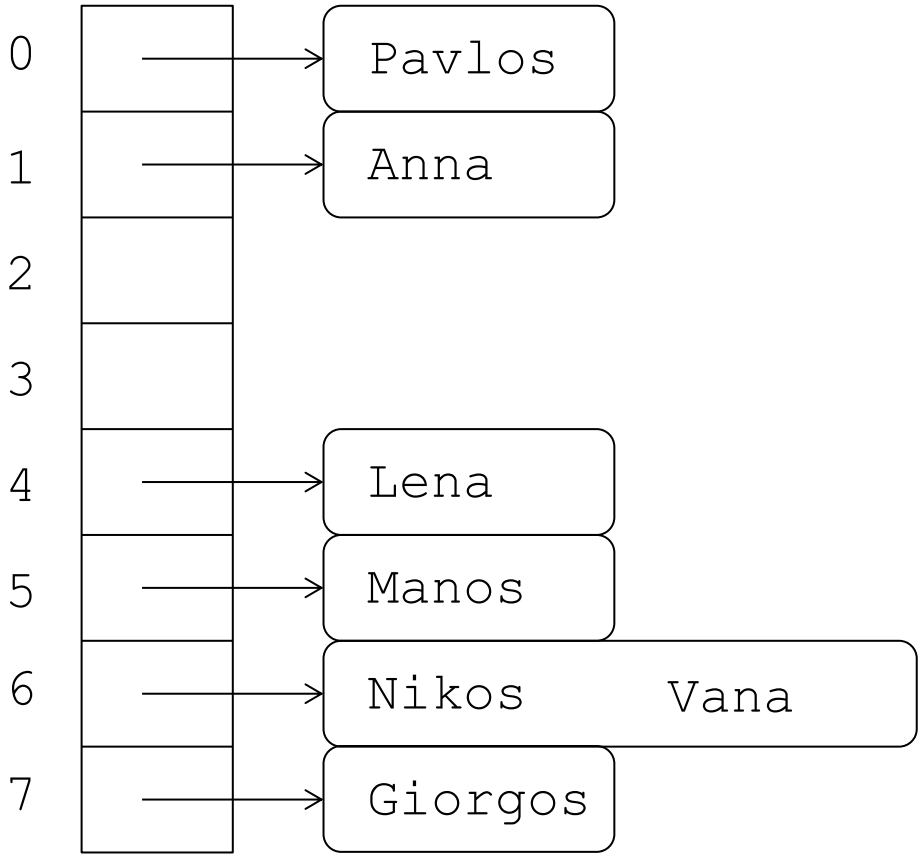
```
str[0] % 8
```

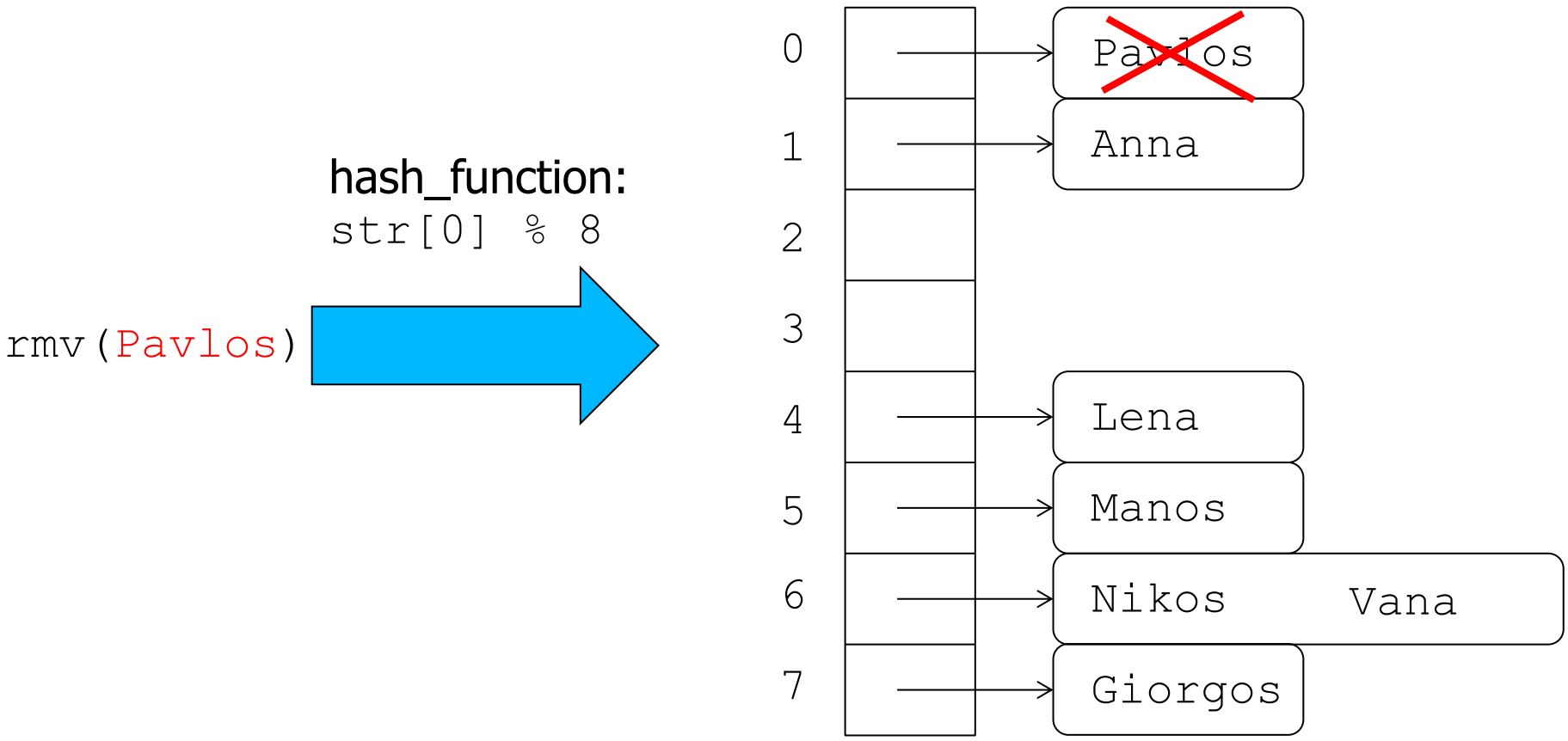
```
rmv(Katia)
```

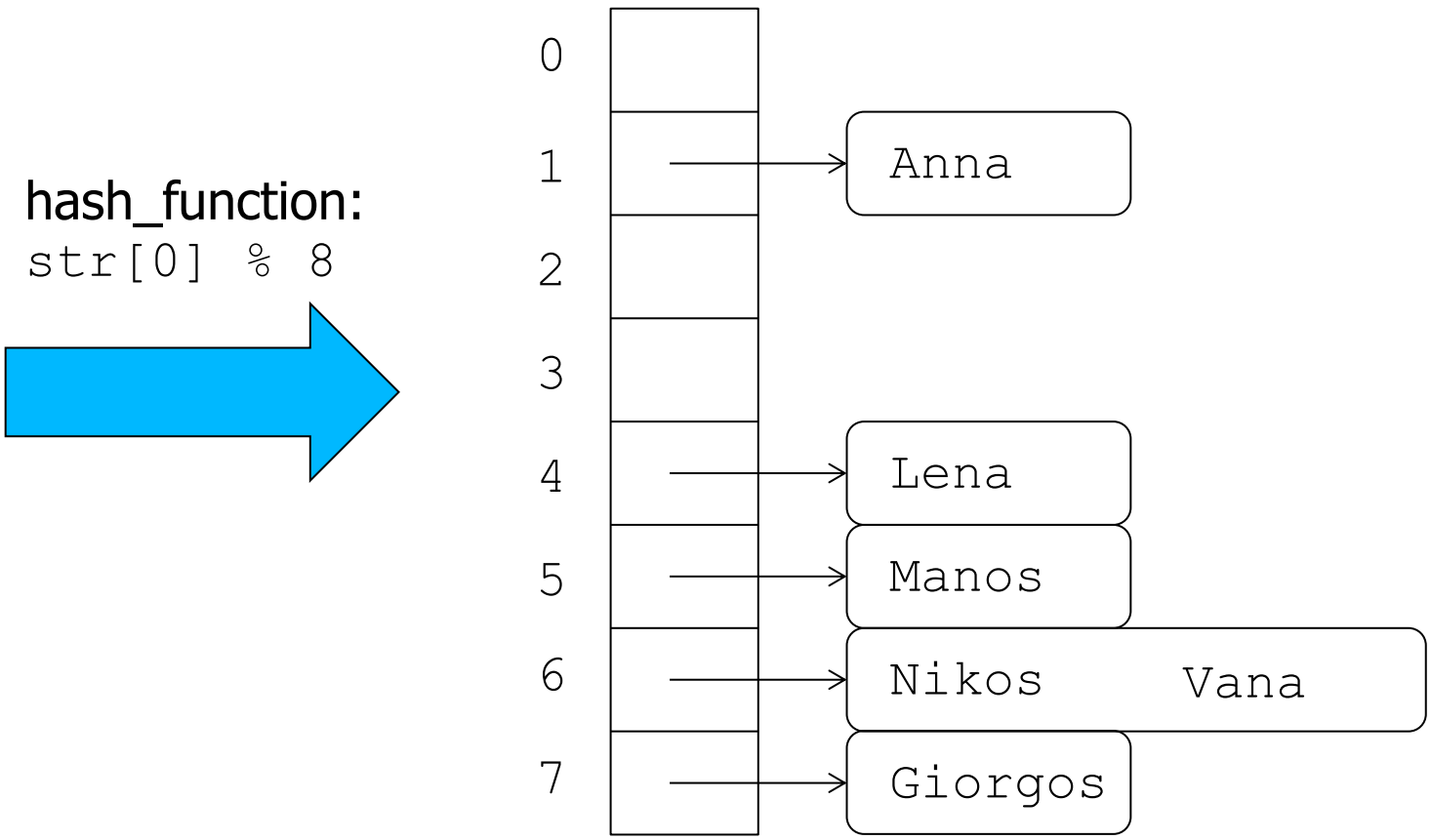


hash_function:

```
str[0] % 8
```

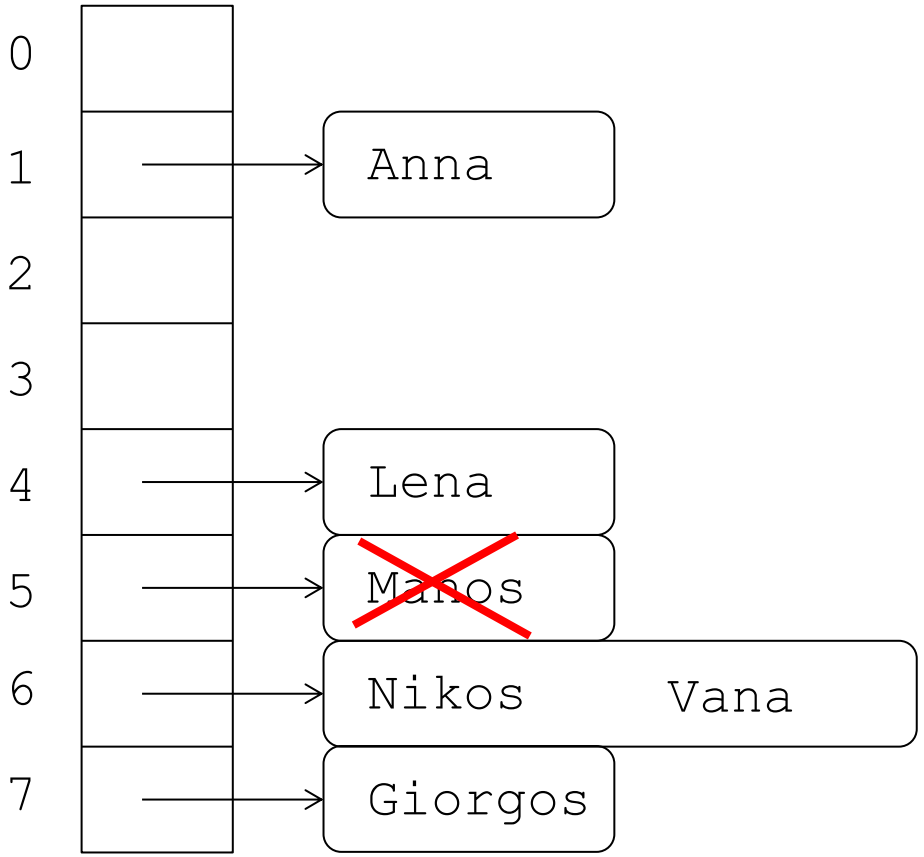



hash_function:
str[0] % 8
rmv(Pavlos)
0
1
2
3
4
5
6
7
Pavlos
Anna
Lena
Manos
Nikos
Vana
Giorgos


hash_function:
str[0] % 8
0
1
2
3
4
5
6
7
Anna
Lena
Manos
Nikos
Vana
Giorgos

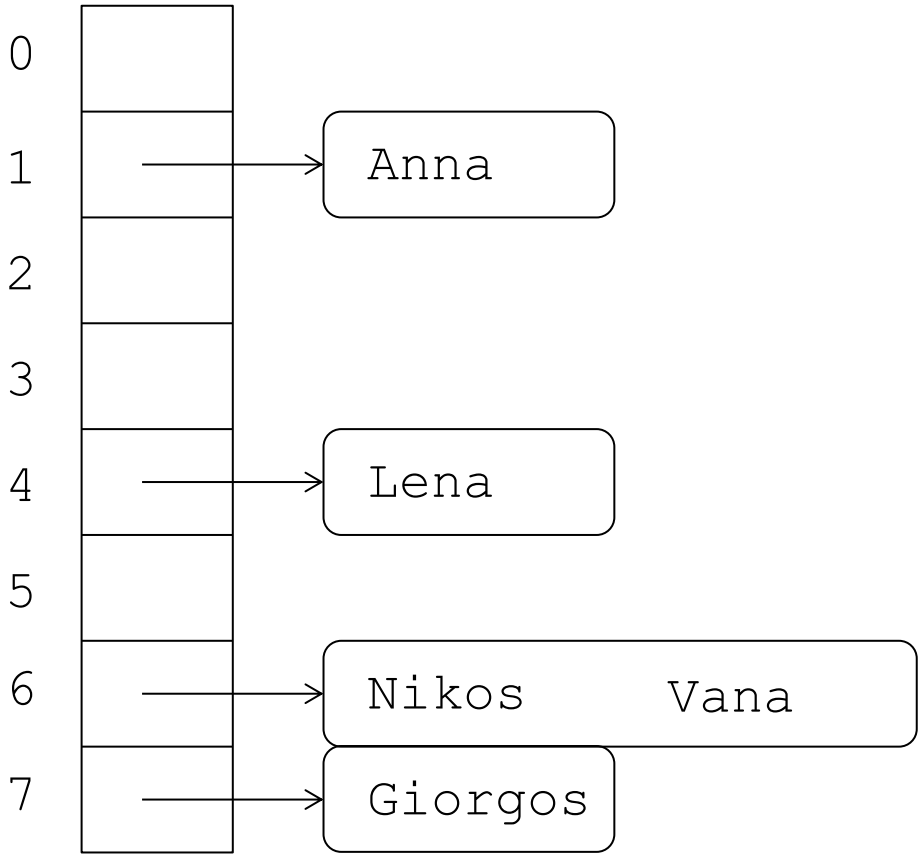
hash_function:
`str[0] % 8`

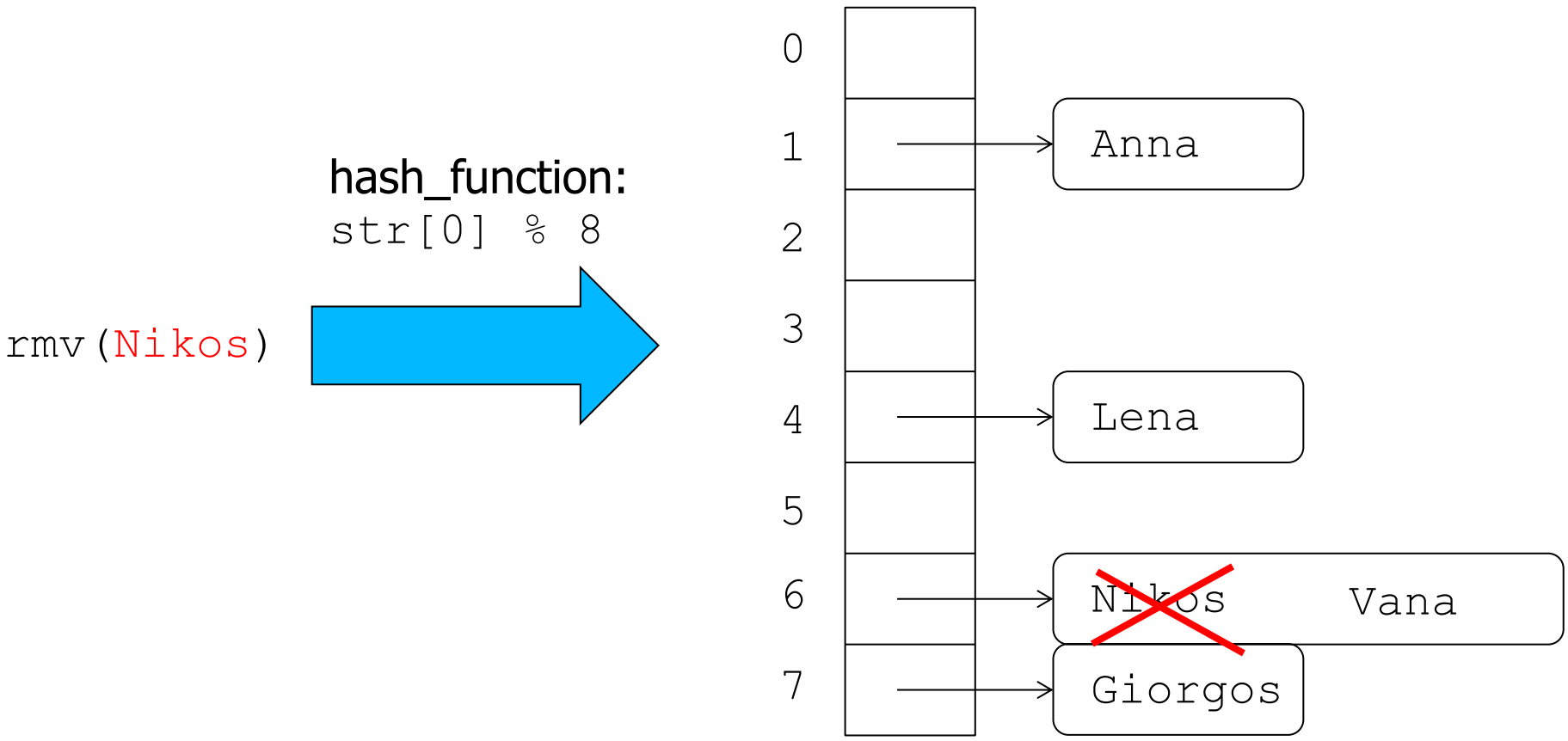
`rmv(Manos)`

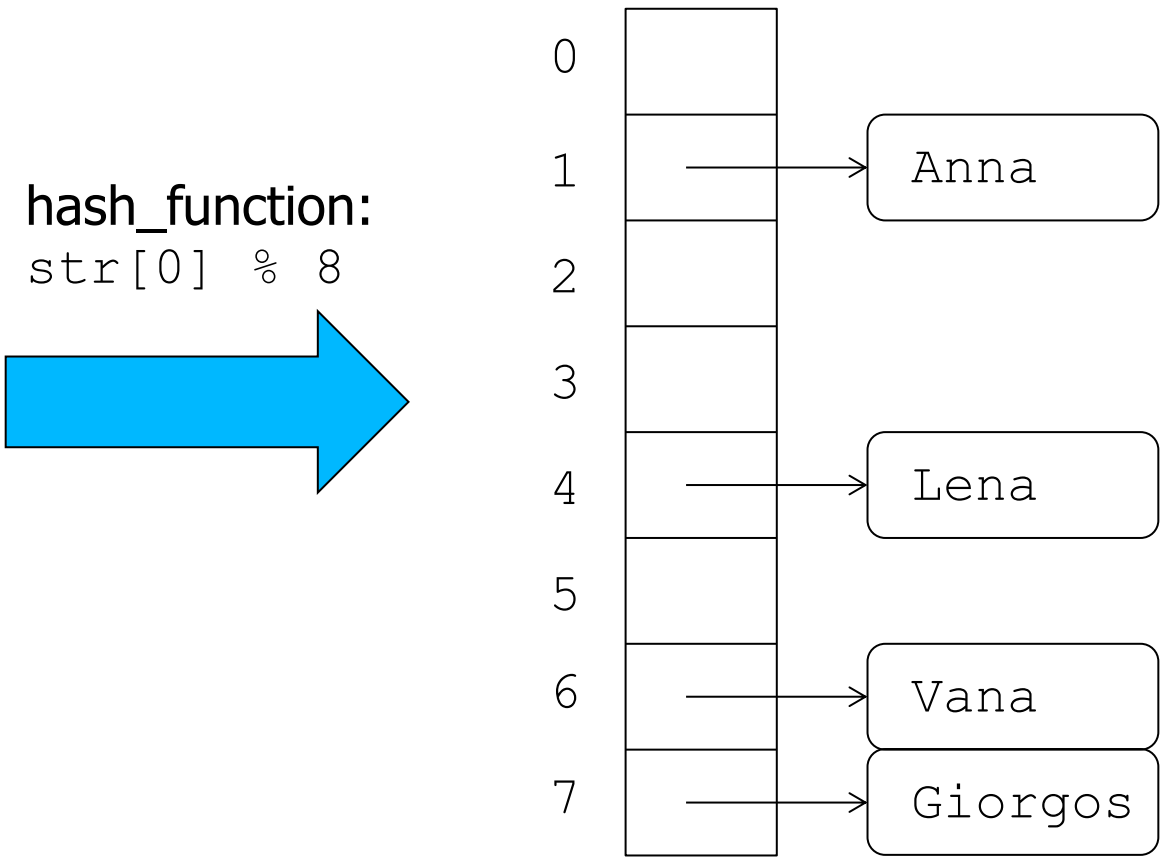


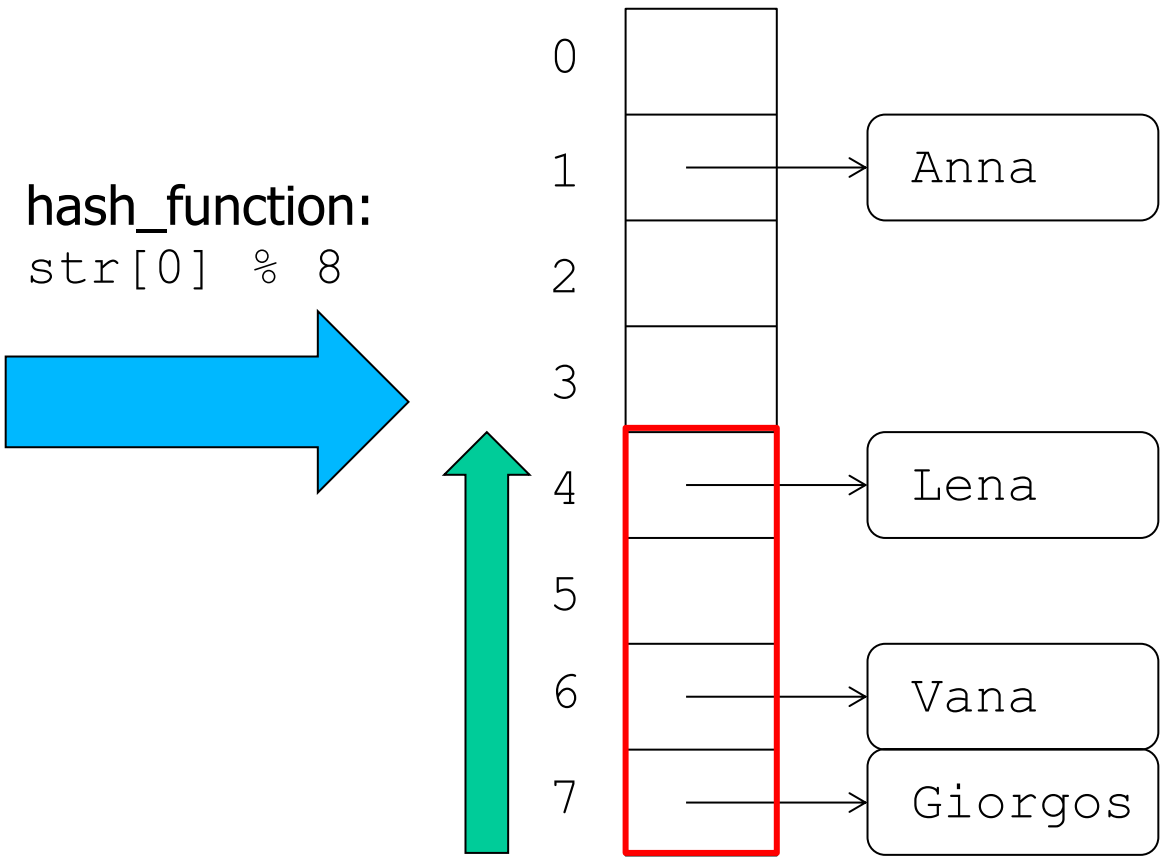
hash_function:

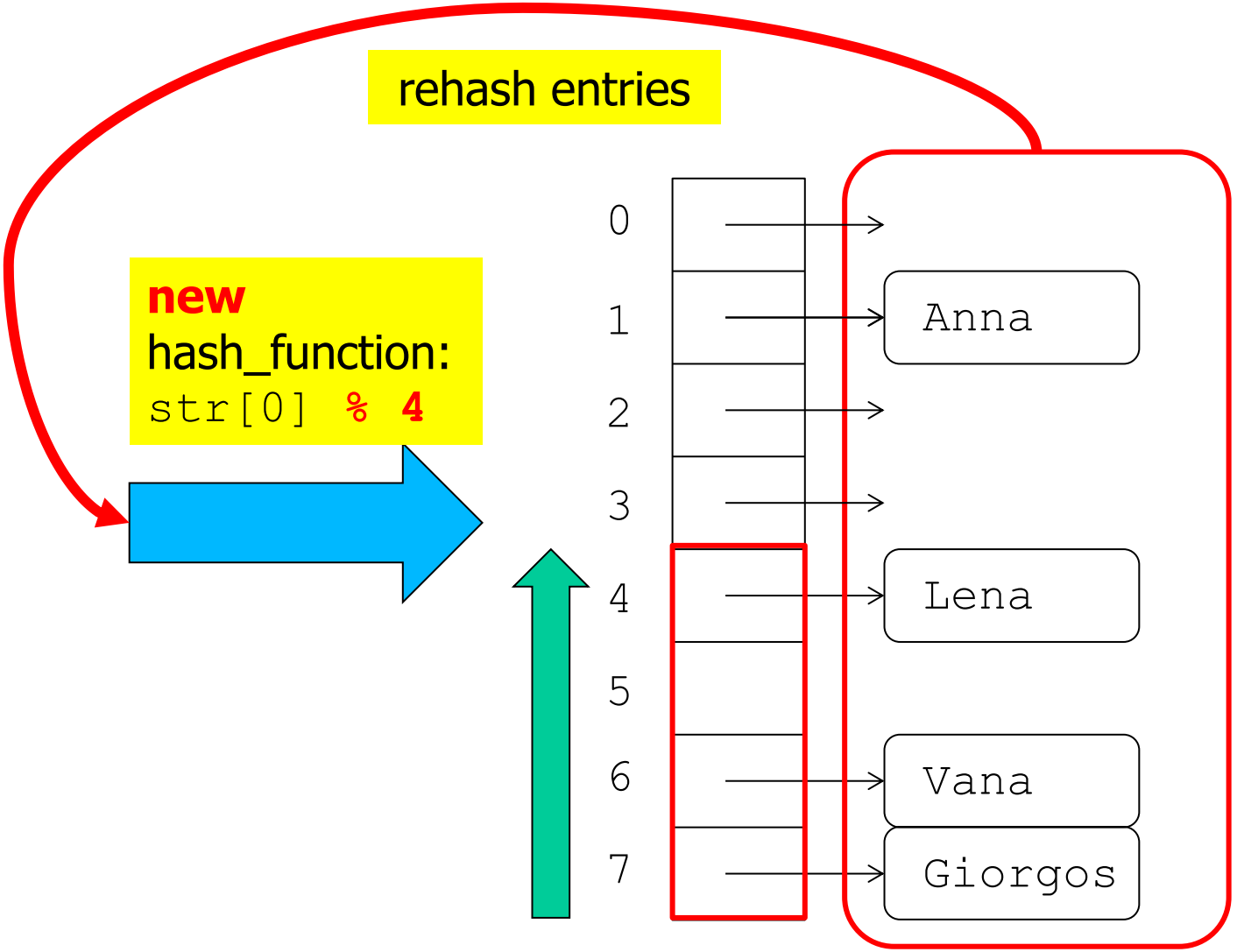
```
str[0] % 8
```

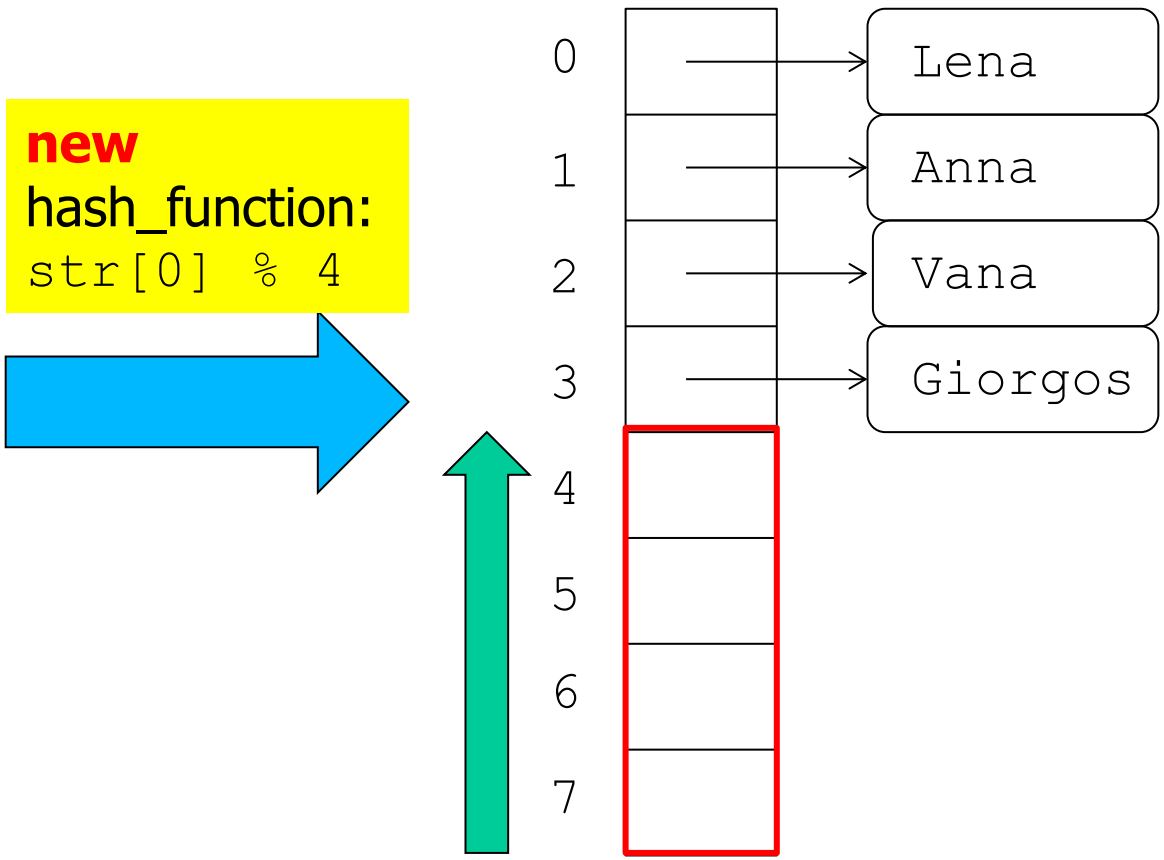


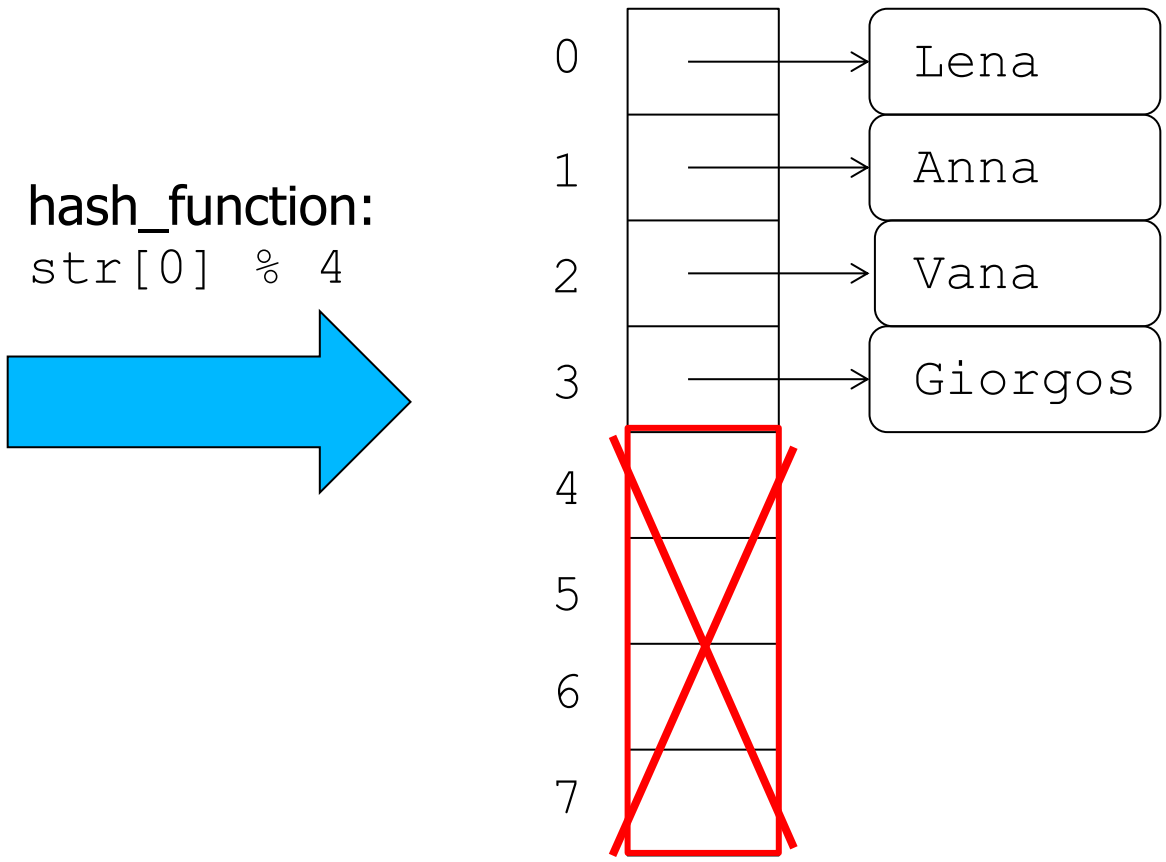

hash_function:
str[0] % 8
rmv(Nikos)
0
1
2
3
4
5
6
7
Anna
Lena
Nikos
Vana
Giorgos

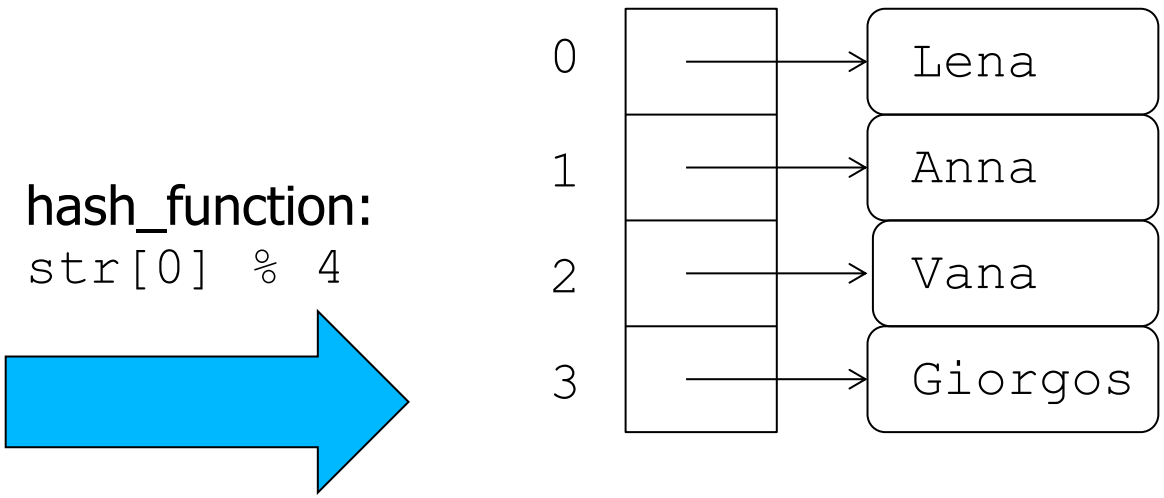

hash_function:
str[0] % 8
0
1
2
3
4
5
6
7
Anna
Lena
Vana
Giorgos


hash_function:
str[0] % 8
0
1
2
3
4
5
6
7
Anna
Lena
Vana
Giorgos


rehash entries
new
hash_function:
str[0] % 4
0
1
2
3
4
5
6
7
Anna
Lena
Vana
Giorgos


new
hash_function:
str[0] % 4
0
1
2
3
4
5
6
7
Lena
Anna
Vana
Giorgos


hash_function:
str[0] % 4
0
1
2
3
4
5
6
7
Lena
Anna
Vana
Giorgos


hash_function:
str[0] % 4
0
1
2
3
Lena
Anna
Vana
Giorgos