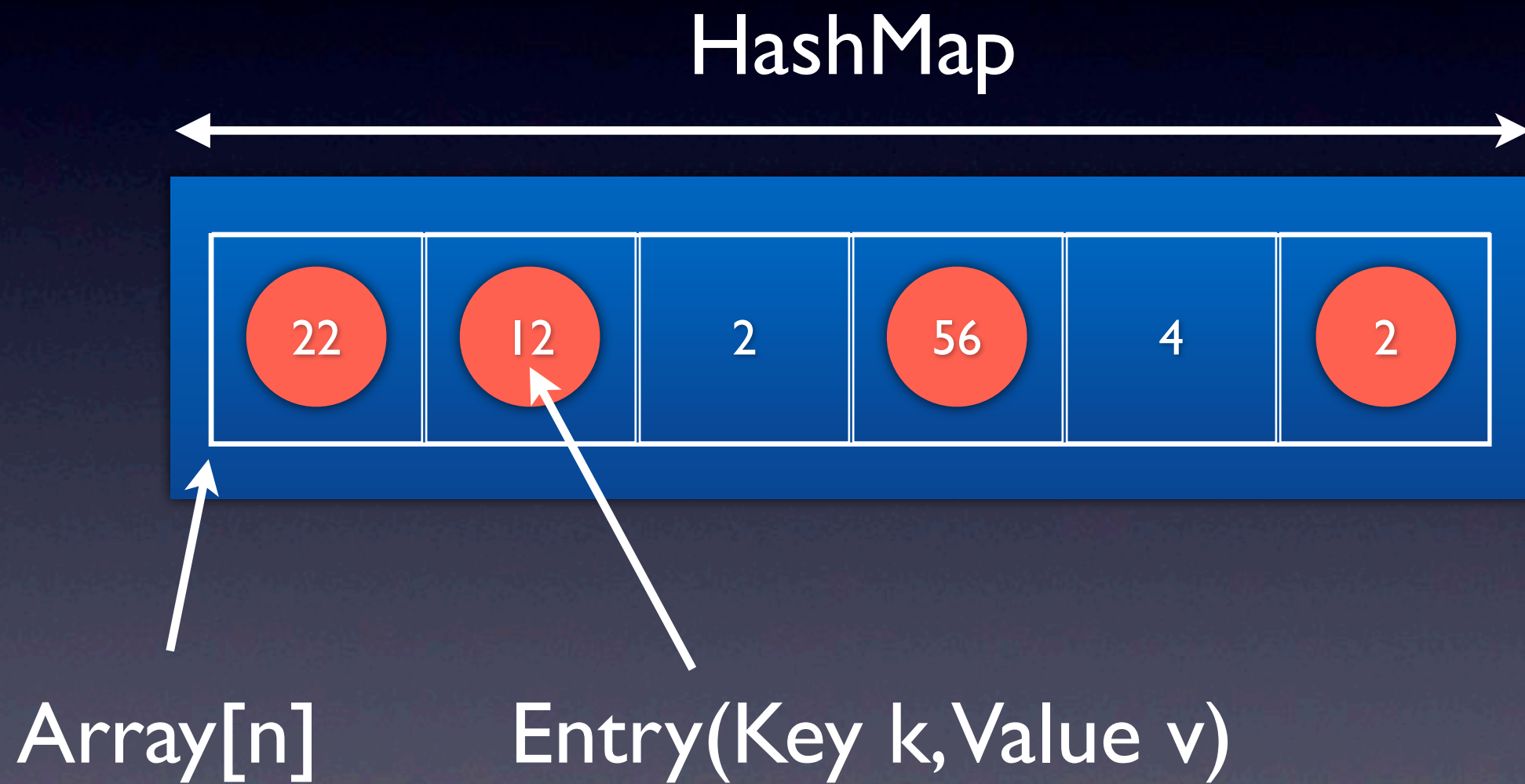


HashMap / Dictionary

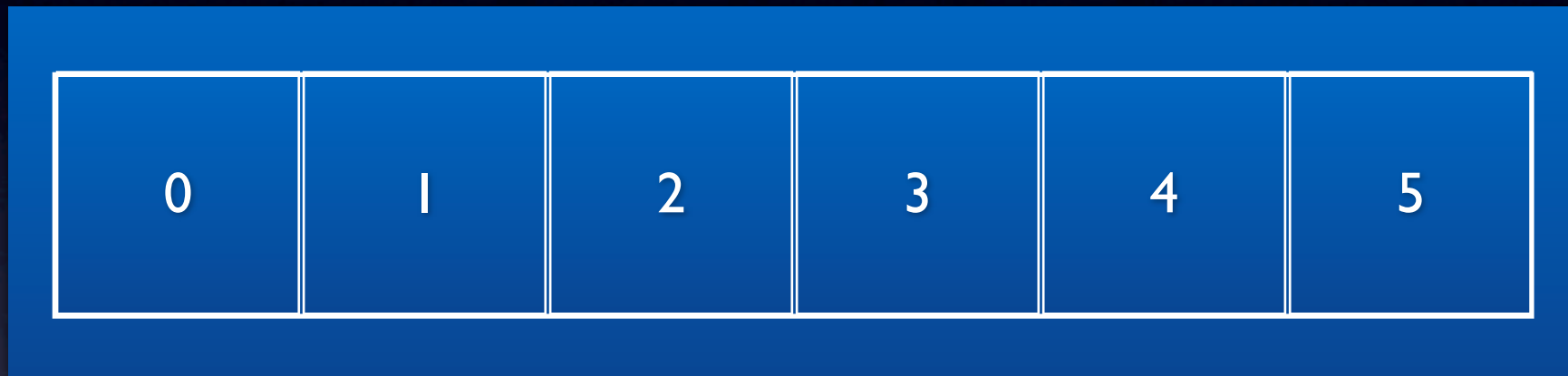
Aufbau



Motivation

- Effizienter Key-Value-Store
 - Schnelles Einfügen $\rightarrow O(1)$ (AVG)
 - Schnelle Suche $\rightarrow O(1)$ (AVG)

Einfügen eines Elements

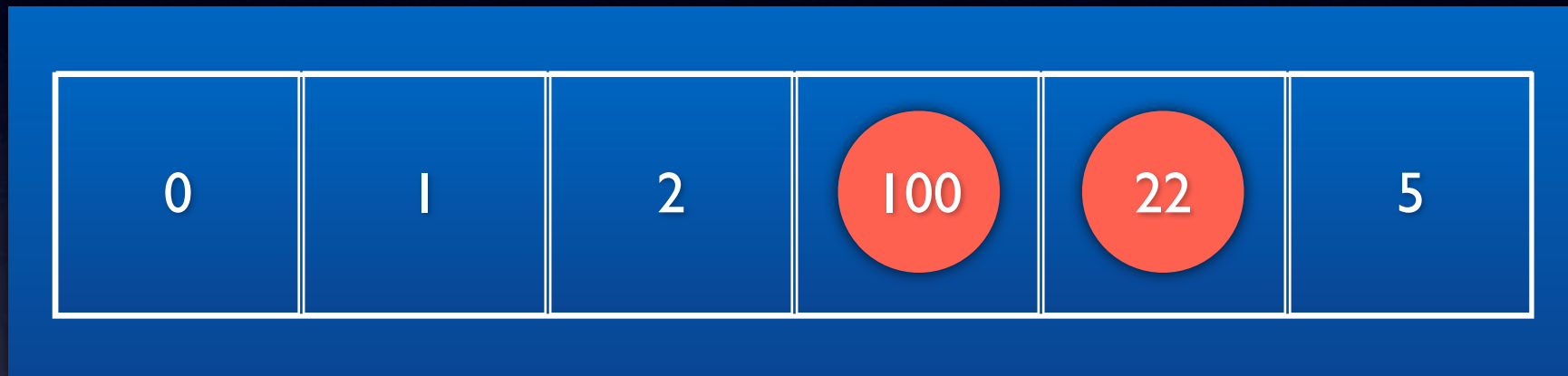


22 key: "test"
value: 22

100 key: "bla"
value: 100

```
public int hash(String s) {  
    return s.length;  
}
```


Einfügen eines Elements

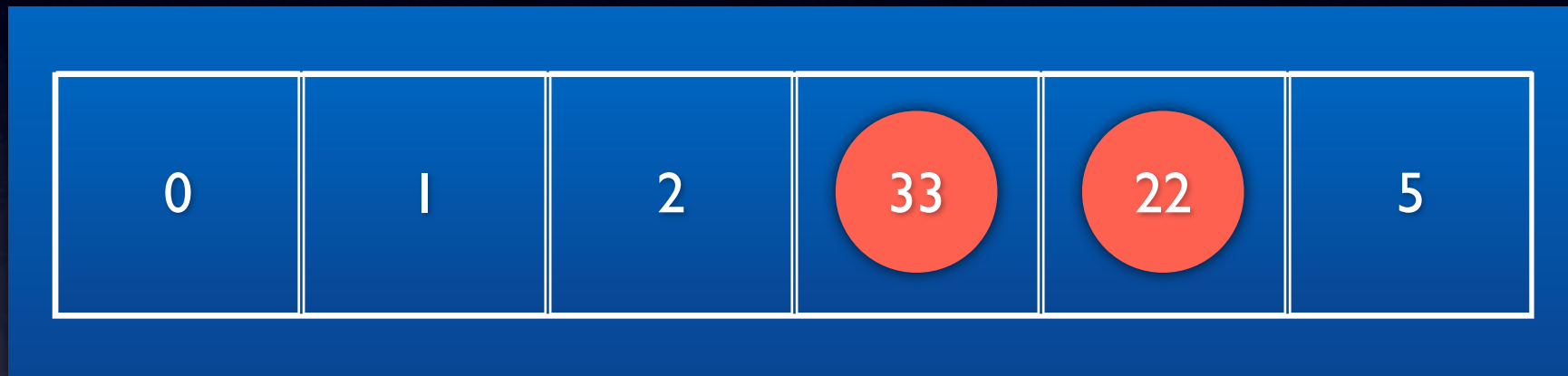


22 key: "test"
value: 22

100 key: "bla"
value: 100

```
public int hash(String s) {  
    return s.length;  
}
```

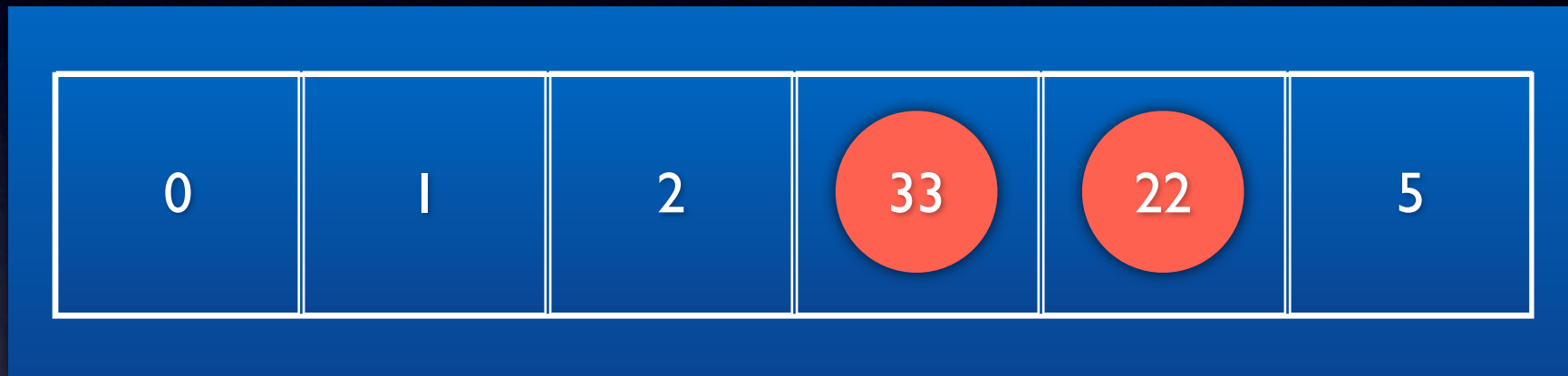
Problem: Hash Funktion



key: "einzulangerkey"
value: 12

```
public int hash(String s) {  
    return s.length;  
}
```

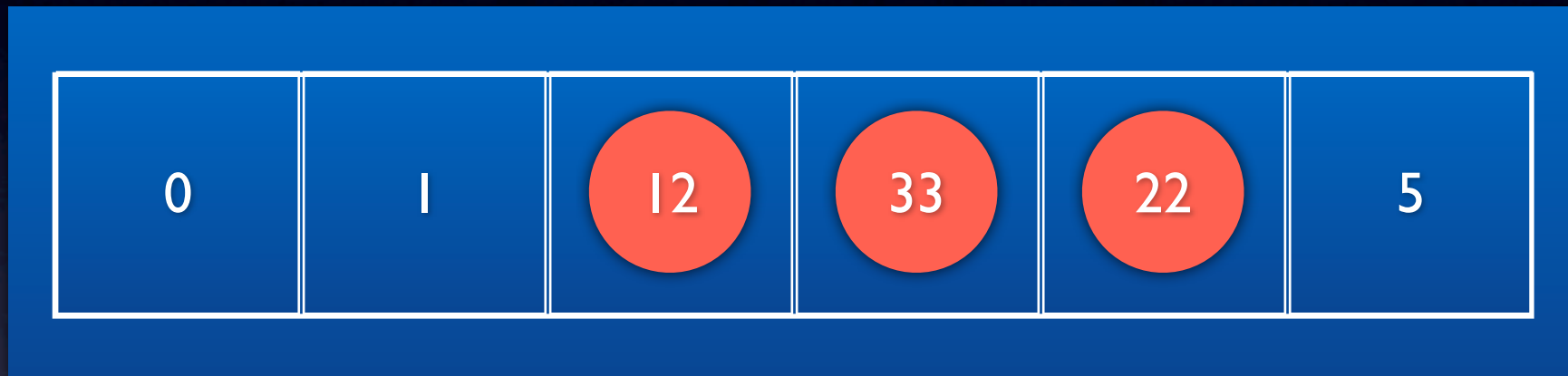
Lösung: Modulo



12 key: "einzulangerkey"
value: 12

```
int pos = Math.abs(key.hashCode()) % data.length;
```

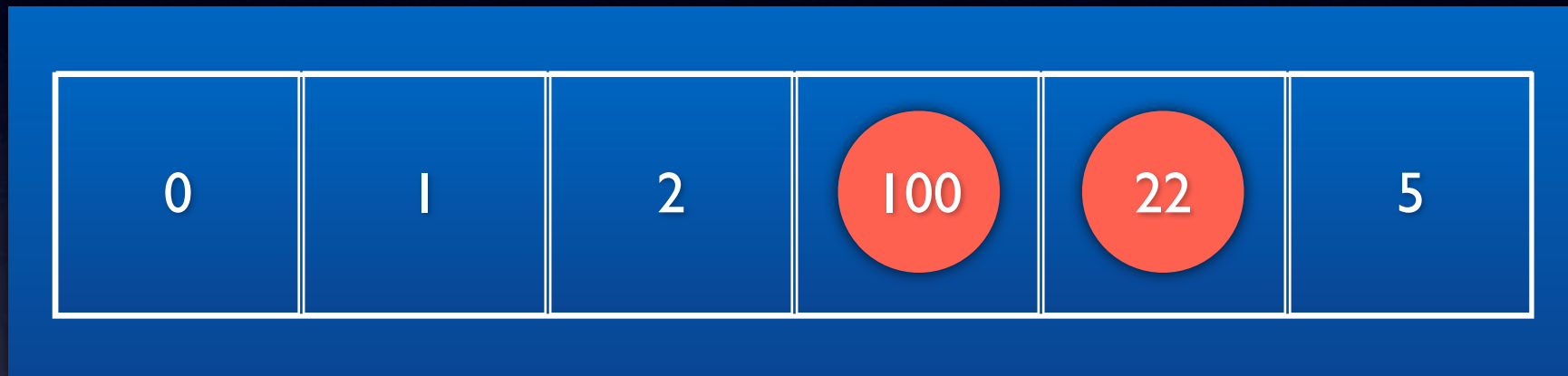
Lösung: Modulo



12 key: "einzulangerkey"
value: 12

```
int pos = Math.abs(key.hashCode()) % data.length;
```

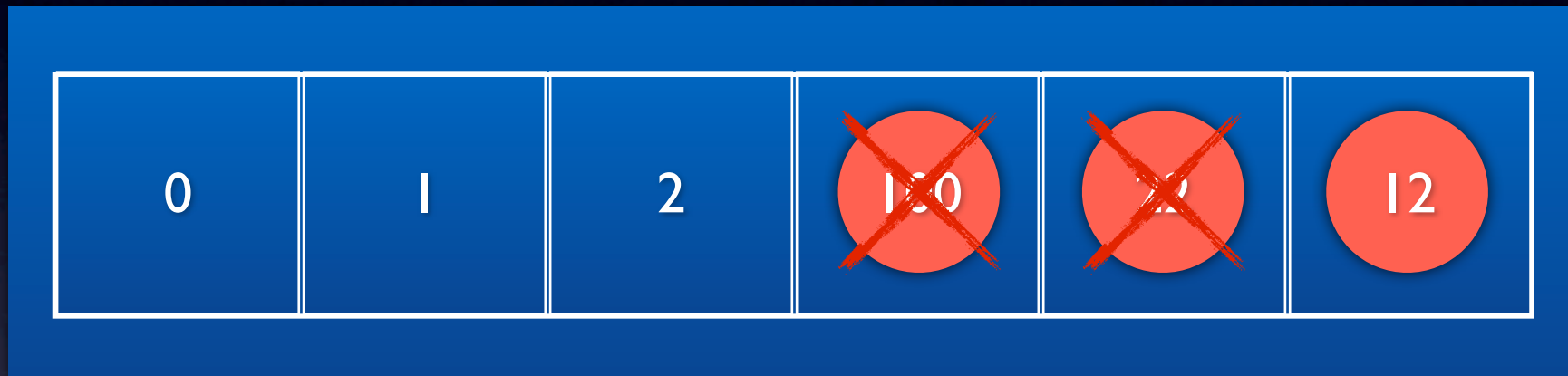

Problem: Kollision



key: "bob"
value: 12

```
public int hash(String s) {  
    return s.length;  
}
```

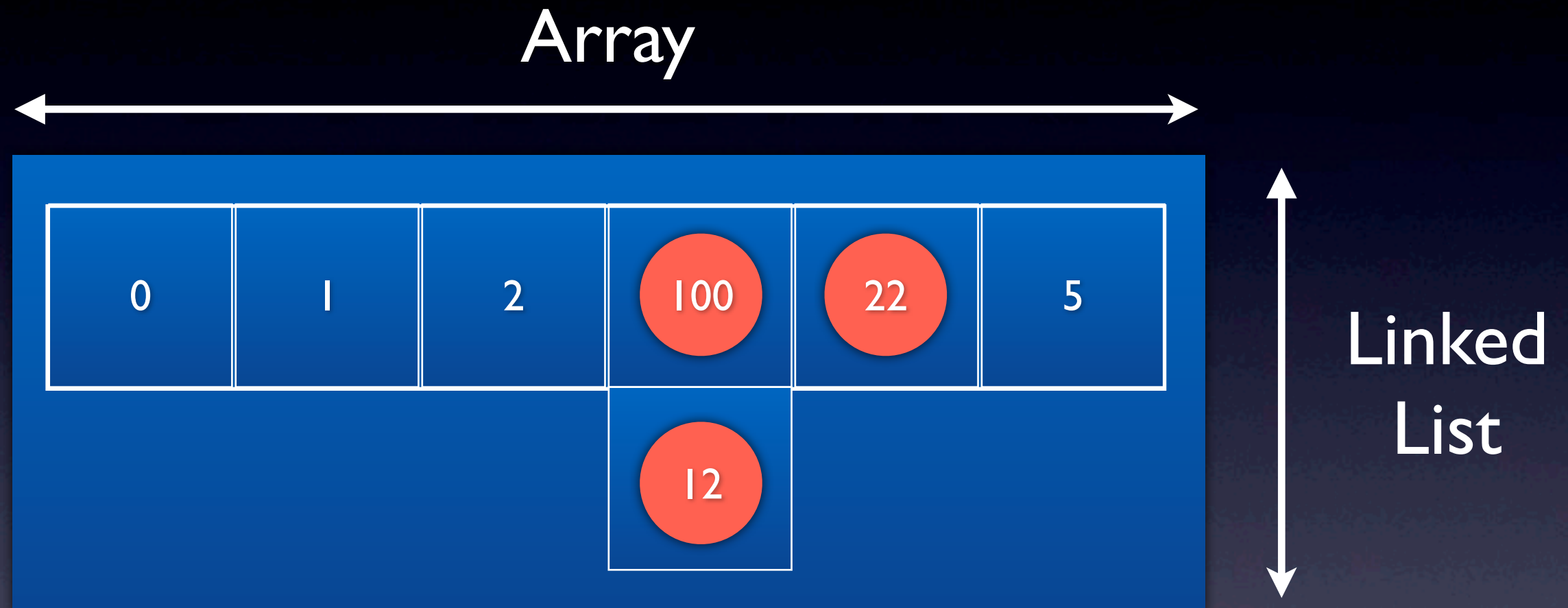
Lösung: verschieben



key: "bob"
value: 12

```
public int hash(String s) {  
    return s.length;  
}
```

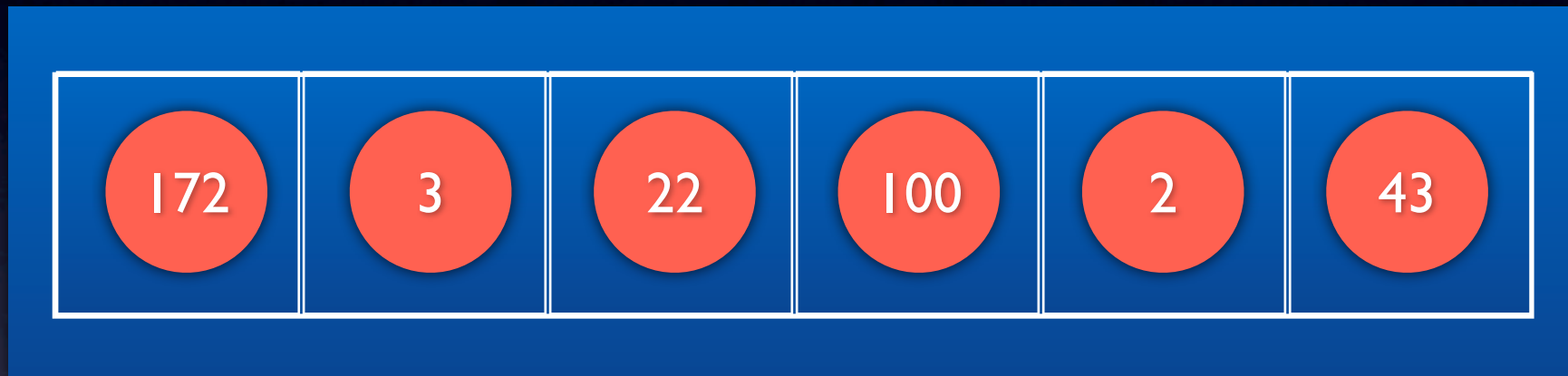
Lösung: verketten



12 key: "bob"
value: 12

```
public int hash(String s) {  
    return s.length;  
}
```

Problem: Array überfüllt



12 key: "bob"
value: 12

```
public int hash(String s) {  
    return s.length;  
}
```


Lösung: Array vergrößern

- Neues Array anlegen
 - ~ Doppelte Größe
- Alle Werte des alten Arrays übertragen
 - Für jeden Wert den Hash neu berechnen!

Beispiel

22

key: "testing"
value: 22

0	1	2	3	4	5
---	---	---	---	---	---

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

Beispiel

22 key: "testing"
value: 22

