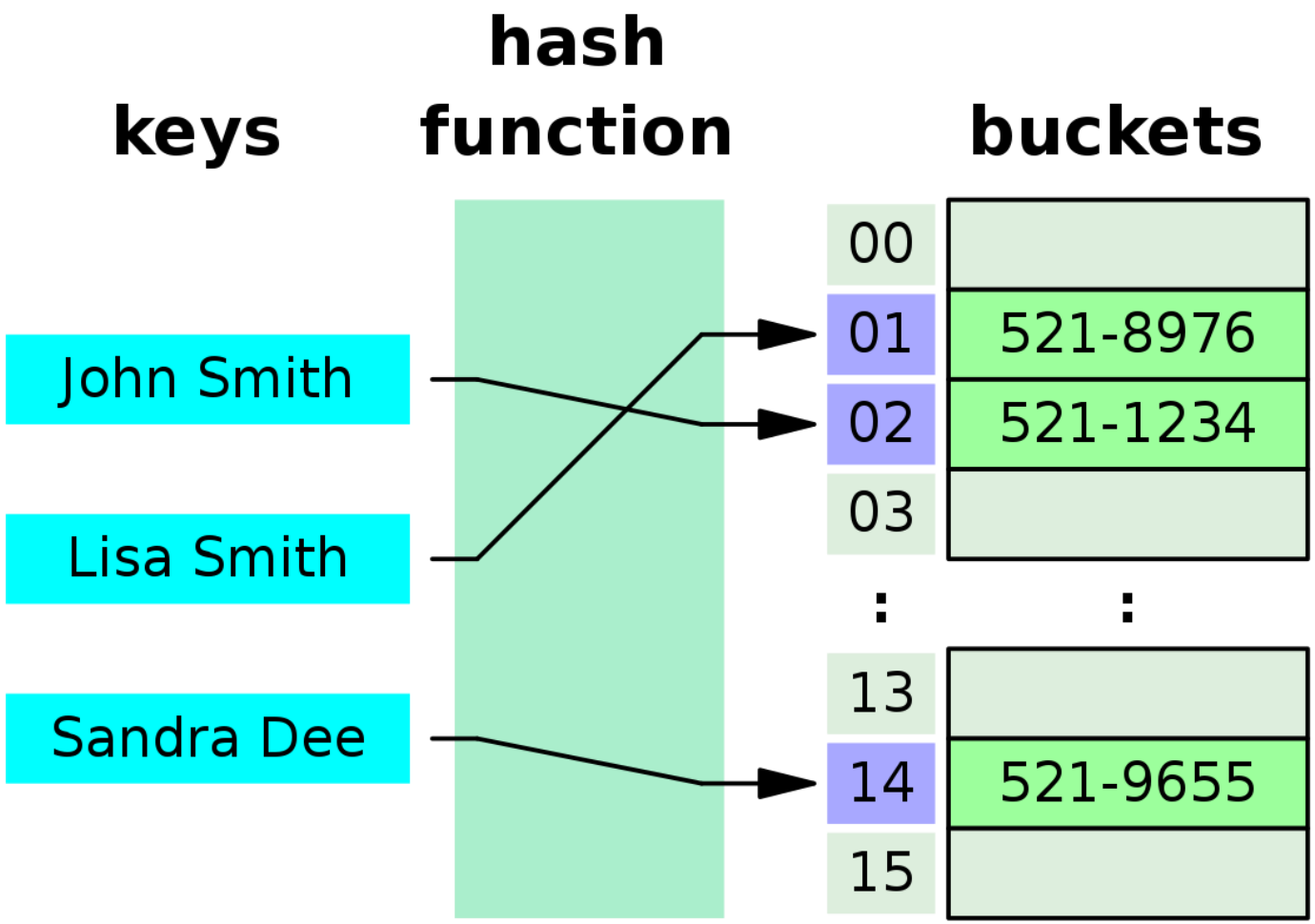
A Hash Table or Dictionary is a collection of key-value pairs

Keys are mapped to a bucket (typically an array index) through a **hash function**

In Java, the implementation of a Hash Table is called a **HashMap**



Key Value Pairs:

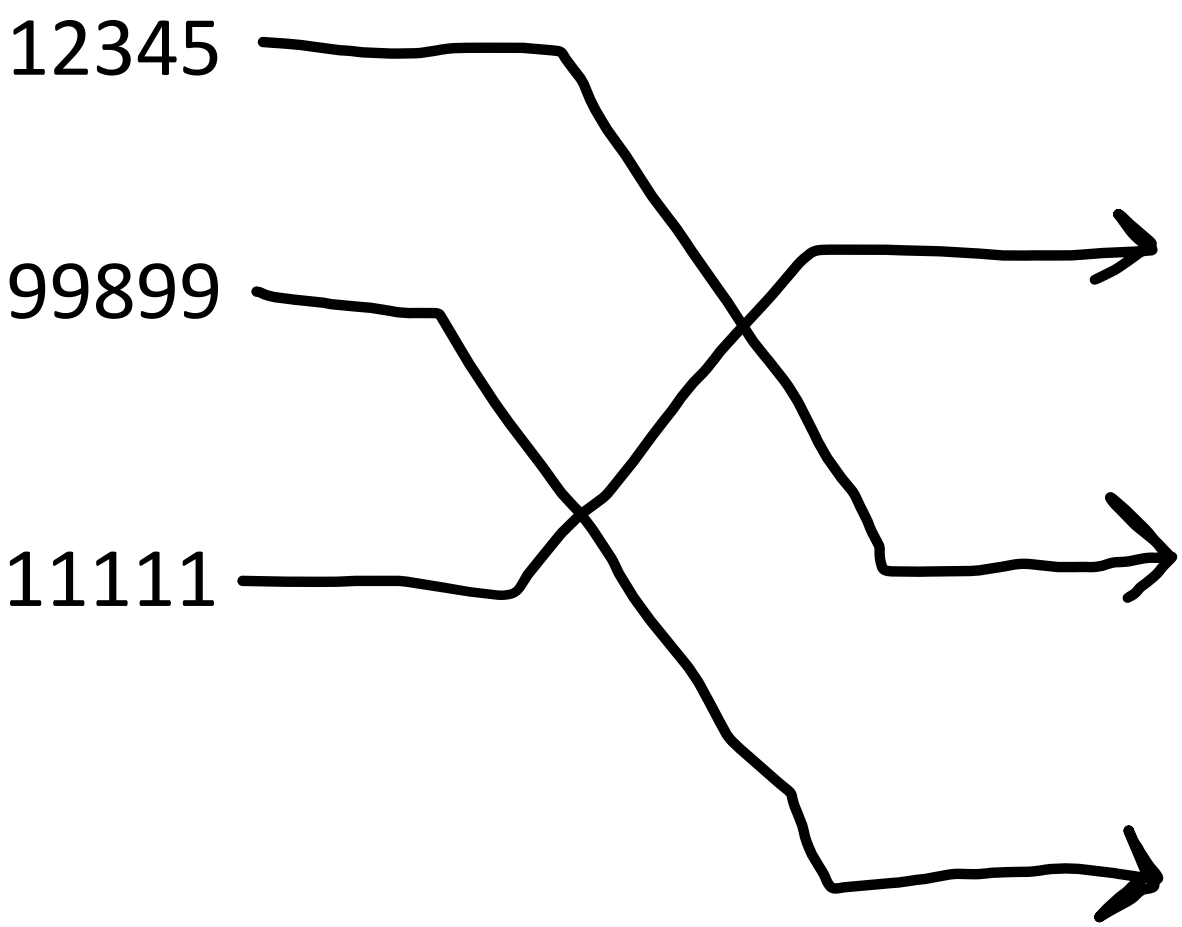
{ John Smith:521-1234, Lisa Smith: 521-8976, Sandra Dee: 521-9655 }

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

Hash Function

```
ID % 100
```



Database

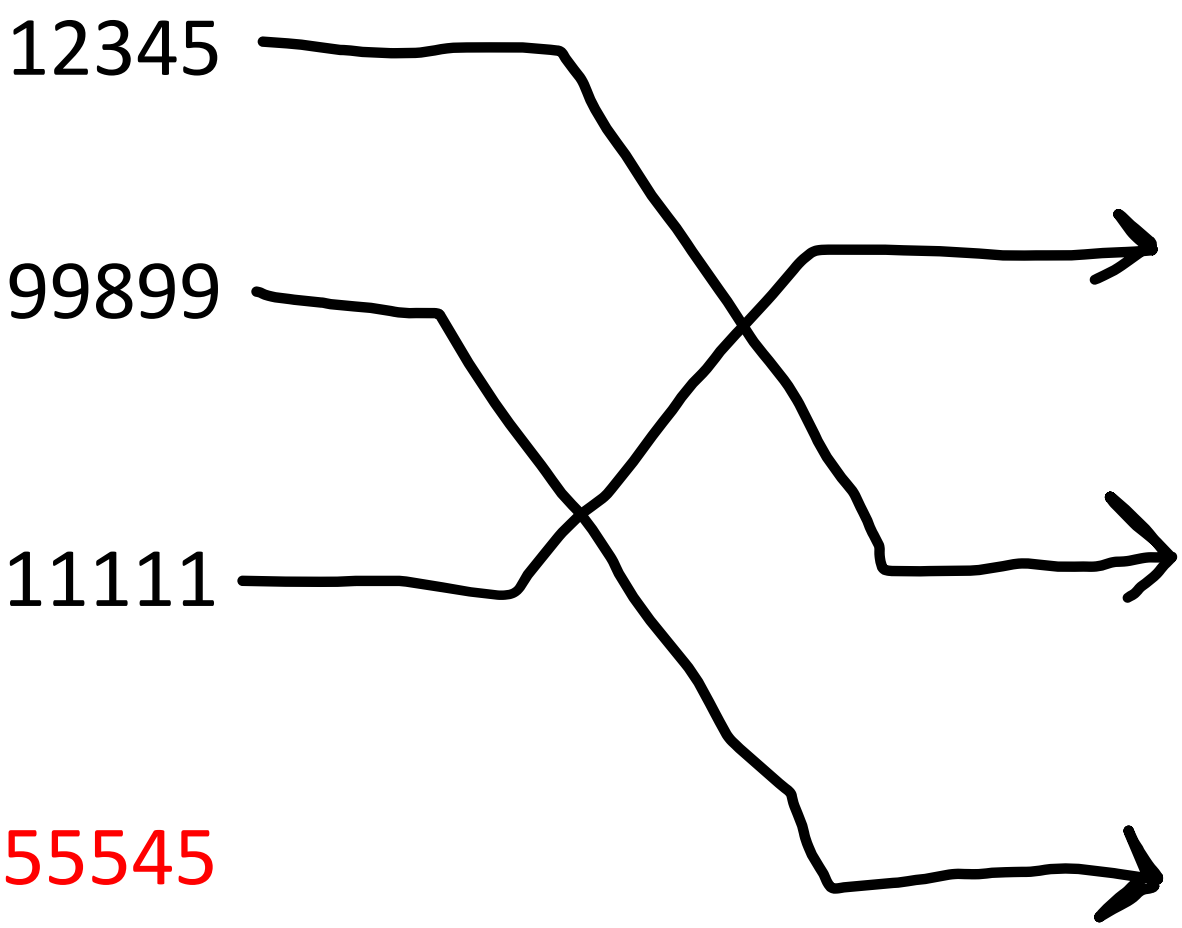
0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

Hash Function

```
ID % 100
```



Database

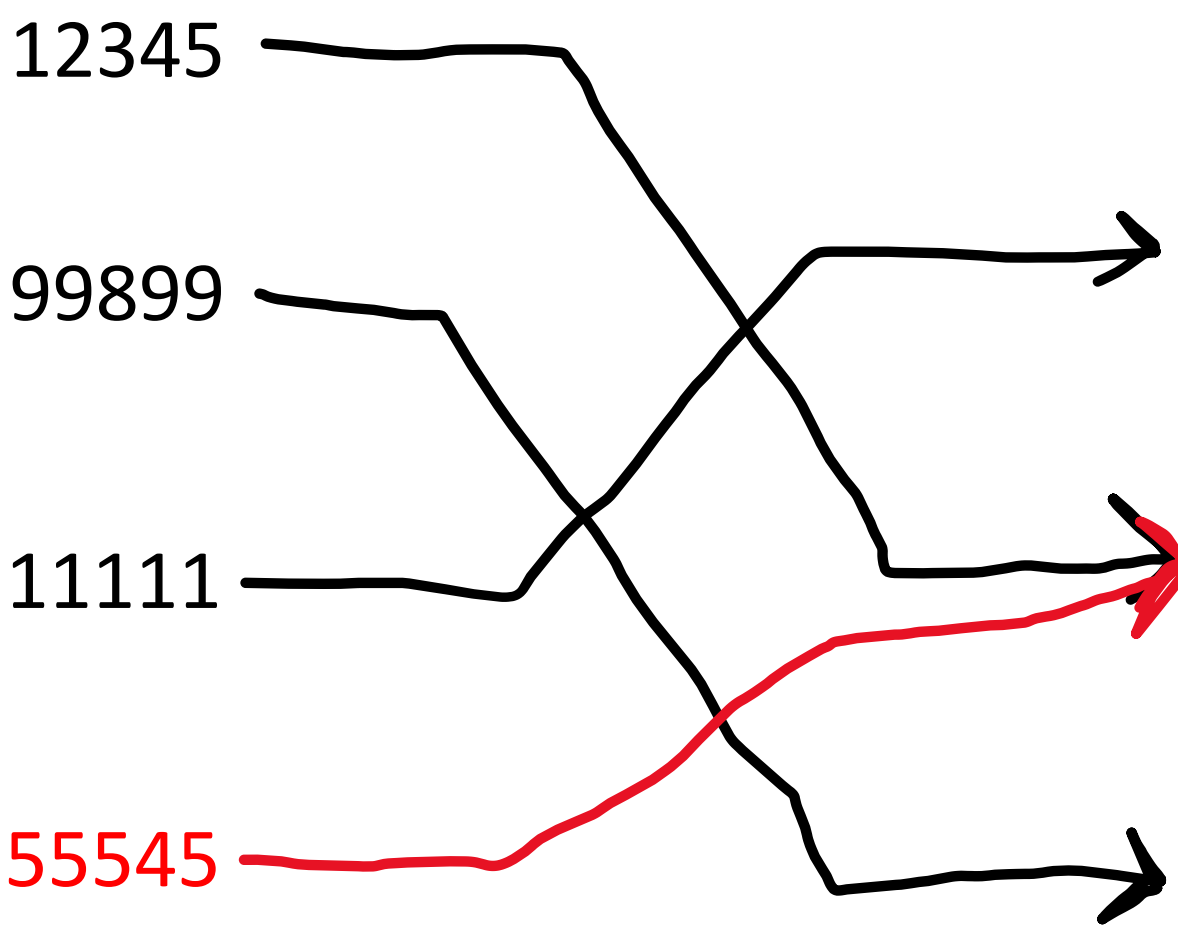
0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

Hash Function

```
ID % 100
```



Database

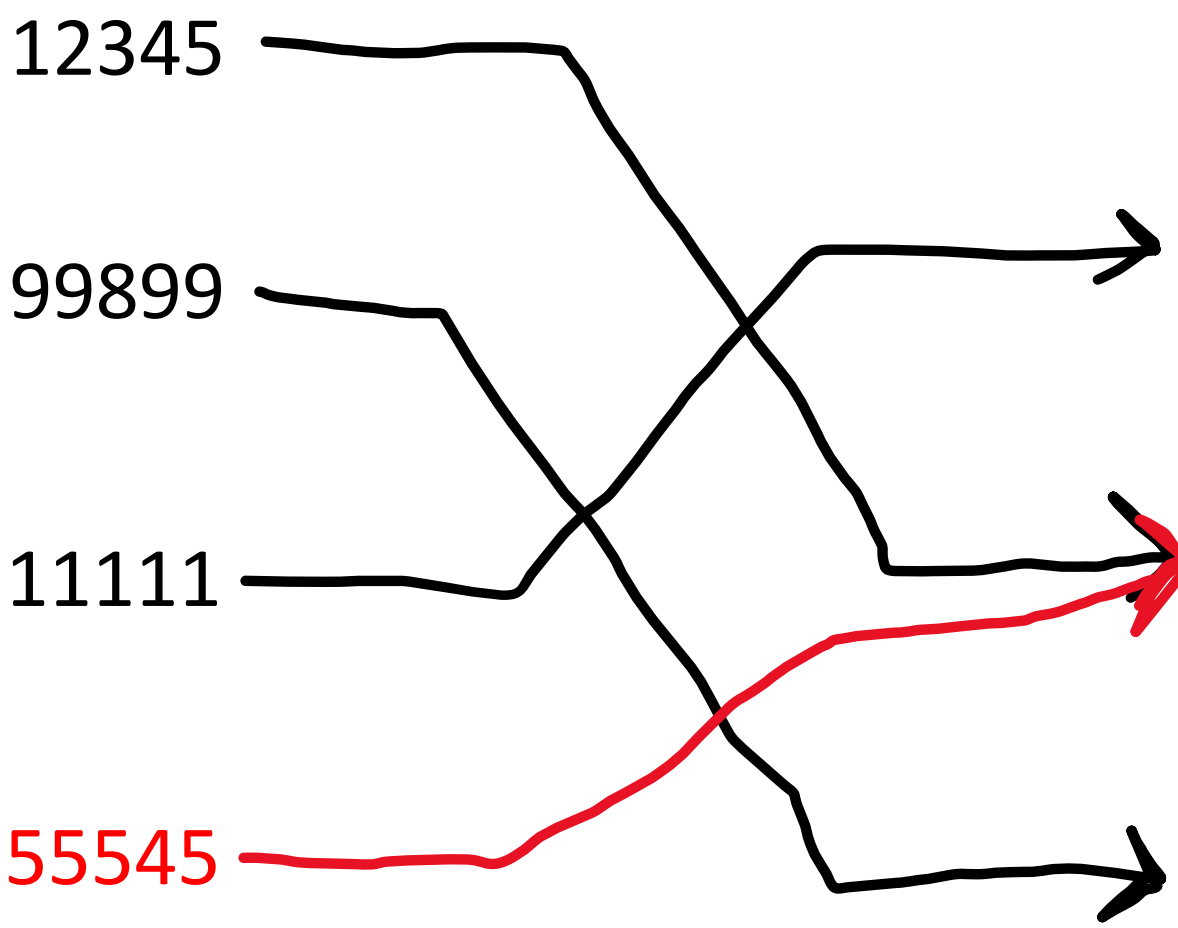
0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

Hash Function

```
ID % 100
```



Database

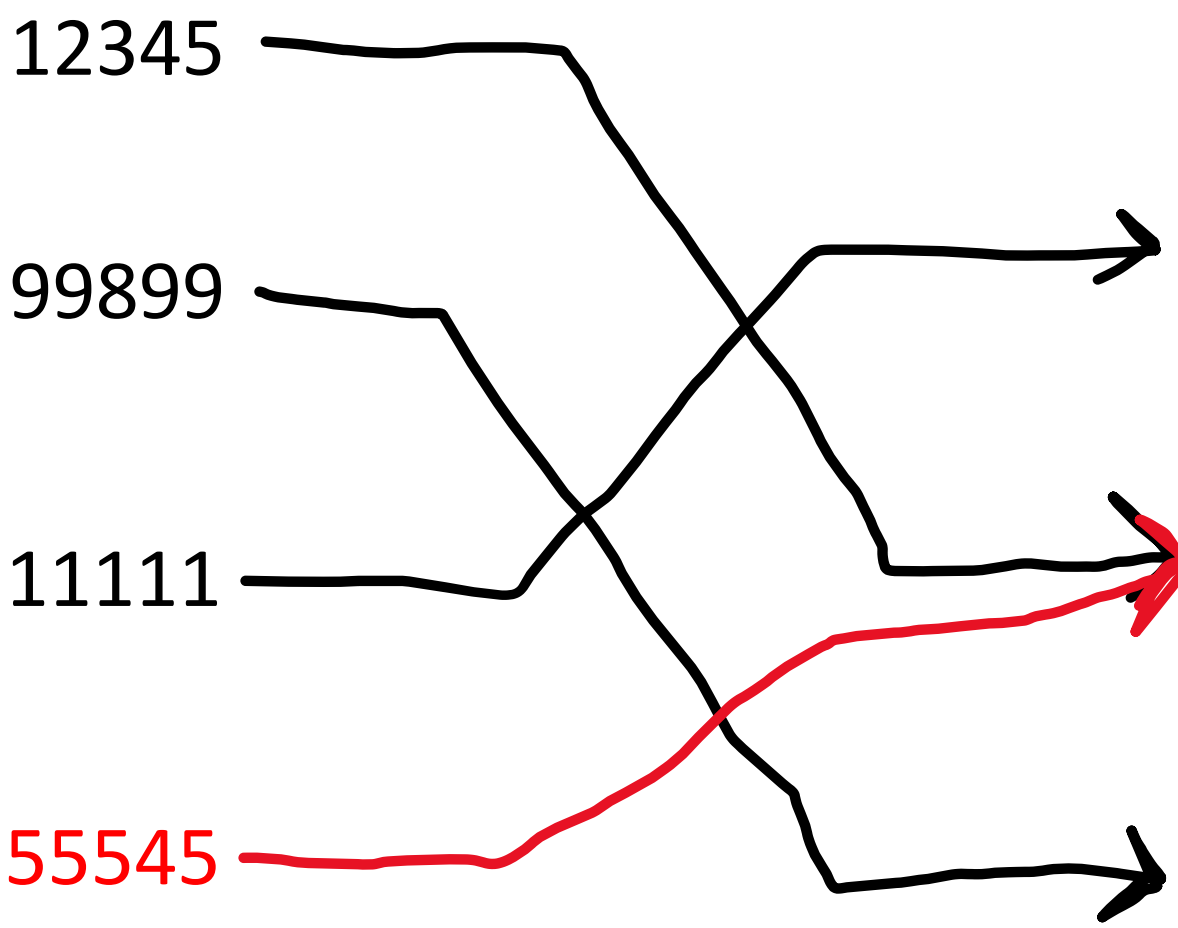
0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(New Student, 55545, Biology, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

Hash Function

`ID % 100`



Database

0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(New Student, 55545, Biology, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Keys should be unique, or you should have a way to hold multiple values at a bucket

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

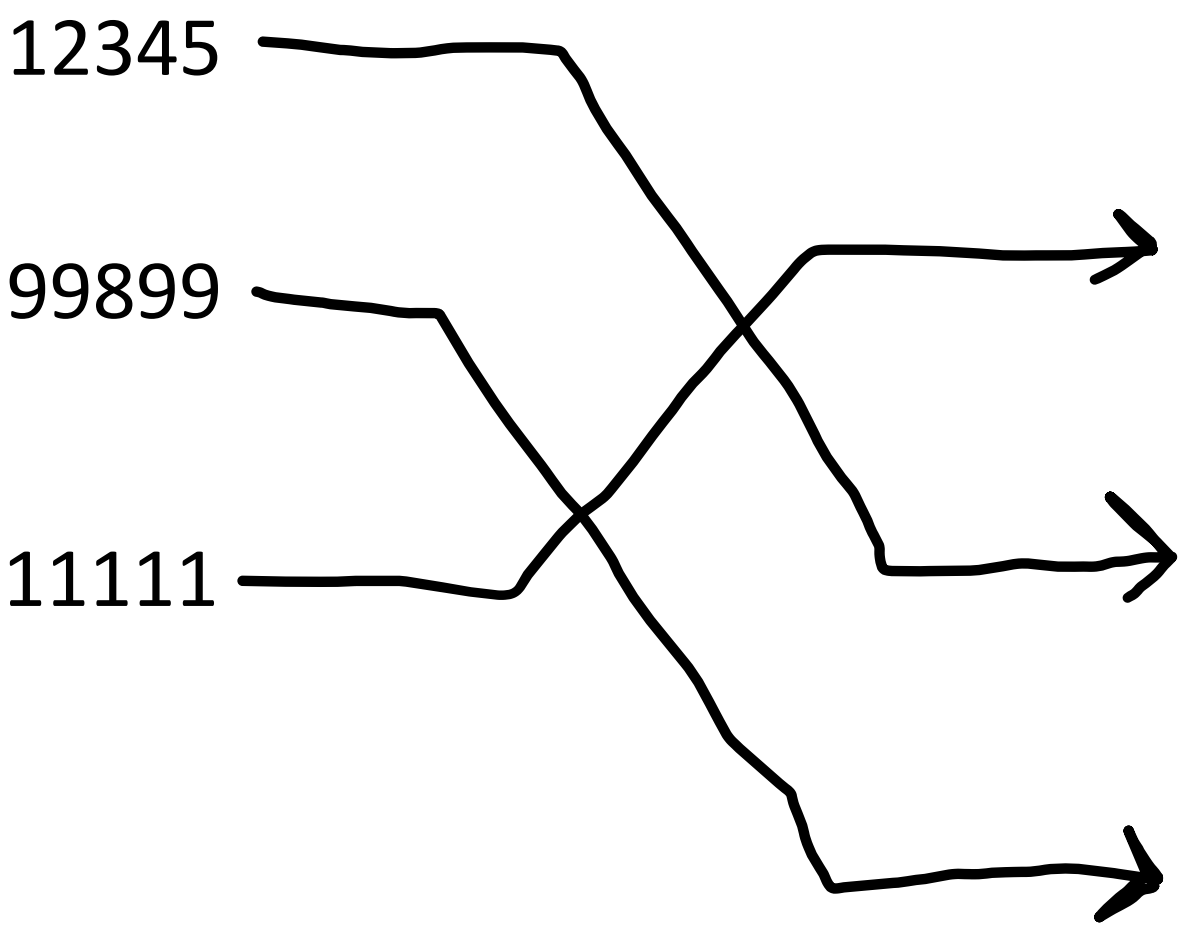
Student ID

Hash Function

`ID % 100`

In our hash function, we mod the Id by our **array size**

Database



0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

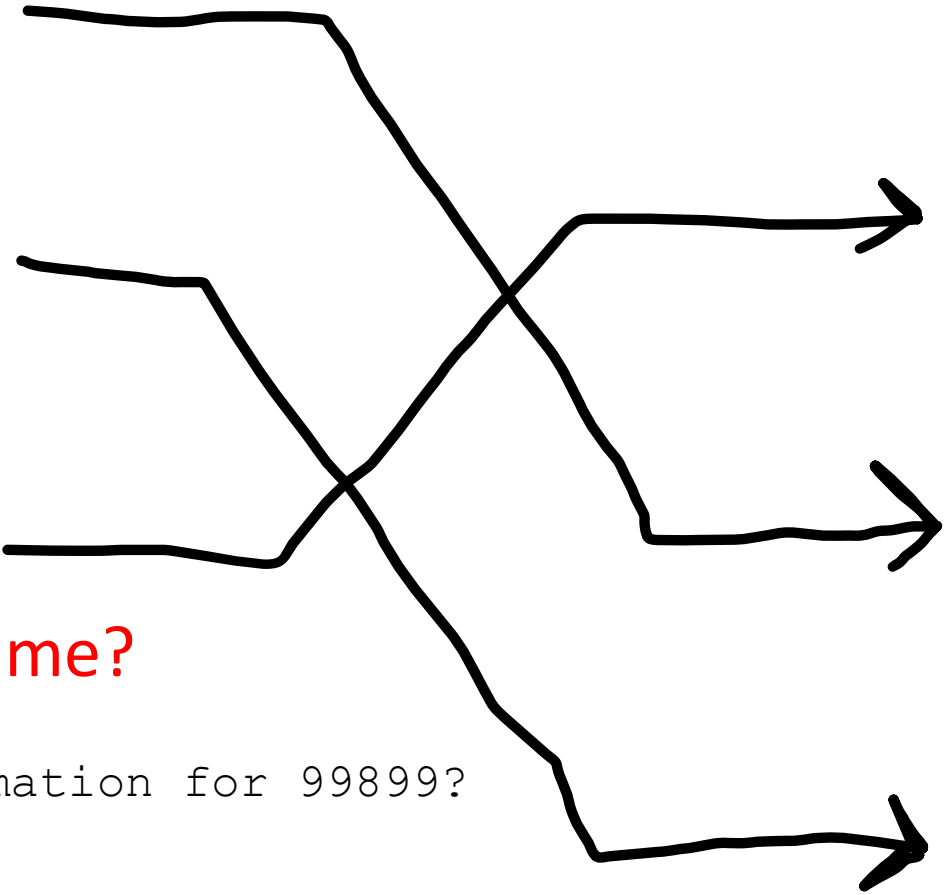
Hash Function

`ID % 100`

12345

99899

11111



Database	
0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Lookup time?

`Get information for 99899?`

O(1)

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

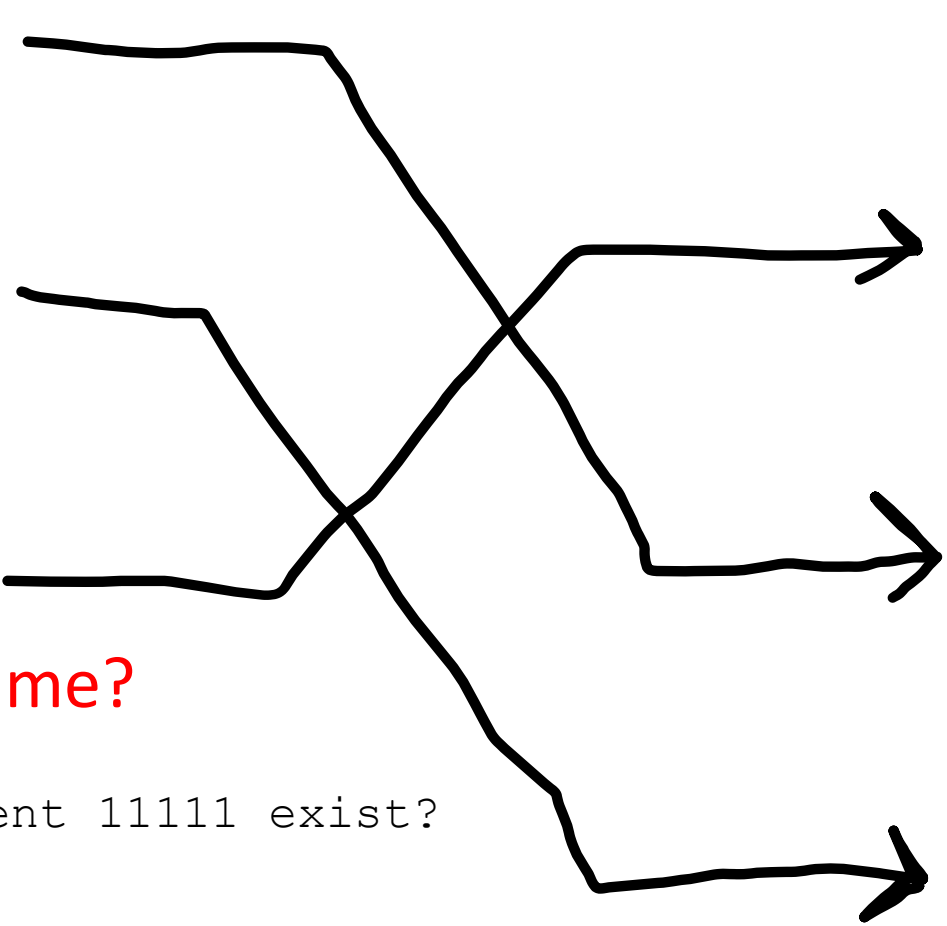
Hash Function

`ID % 100`

12345

99899

11111



Database	
0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Lookup time?

`Does student 11111 exist?`

O(1)

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

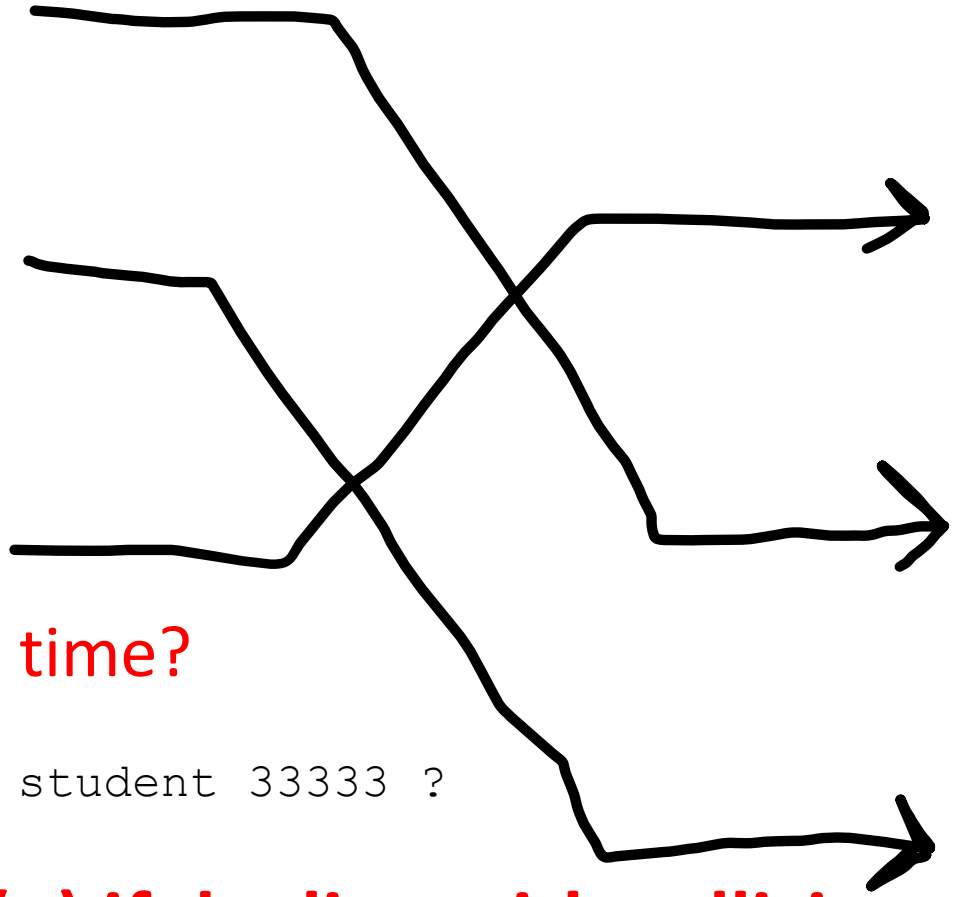
Hash Function

`ID % 100`

12345

99899

11111



Database

0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Insertion time?

`Add a new student 33333 ?`

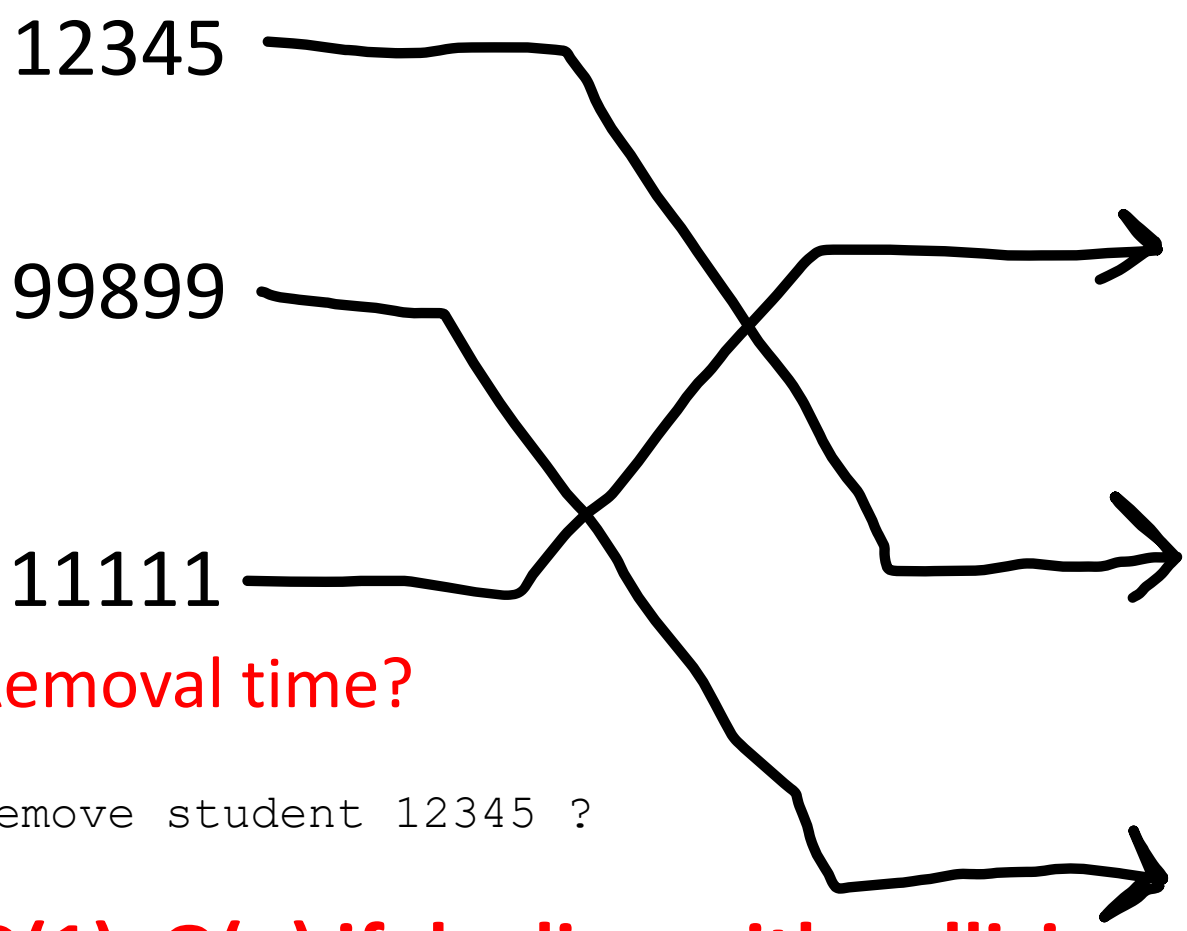
O(1), O(n) if dealing with collisions

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

Hash Function

`ID % 100`



Database

0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Removal time?

`Remove student 12345 ?`

O(1), O(n) if dealing with collisions

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

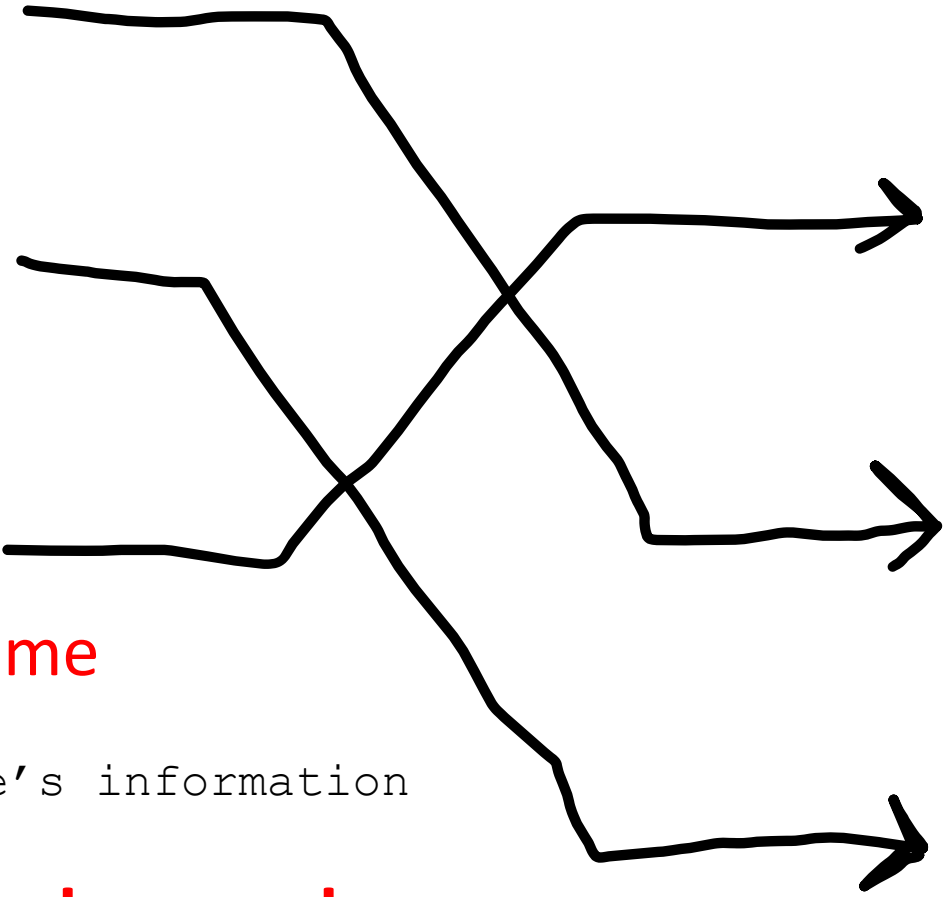
Hash Function

`ID % 100`

12345

99899

11111



Database

0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Lookup time

`Get me Joe's information`

O(n), n = | array |

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

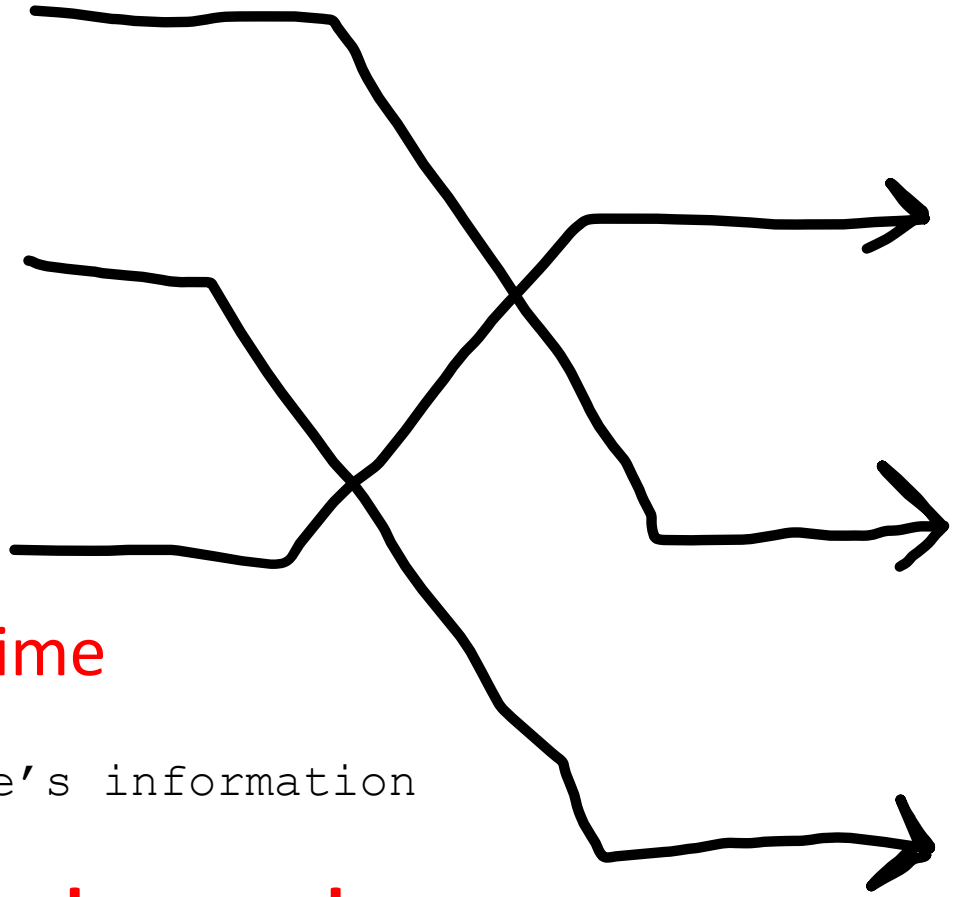
Hash Function

`ID % 100`

12345

99899

11111



Database	
0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

keySpace ArrayList = { 12345, 99899, 11111 }

Lookup time

`Get me Joe's information`

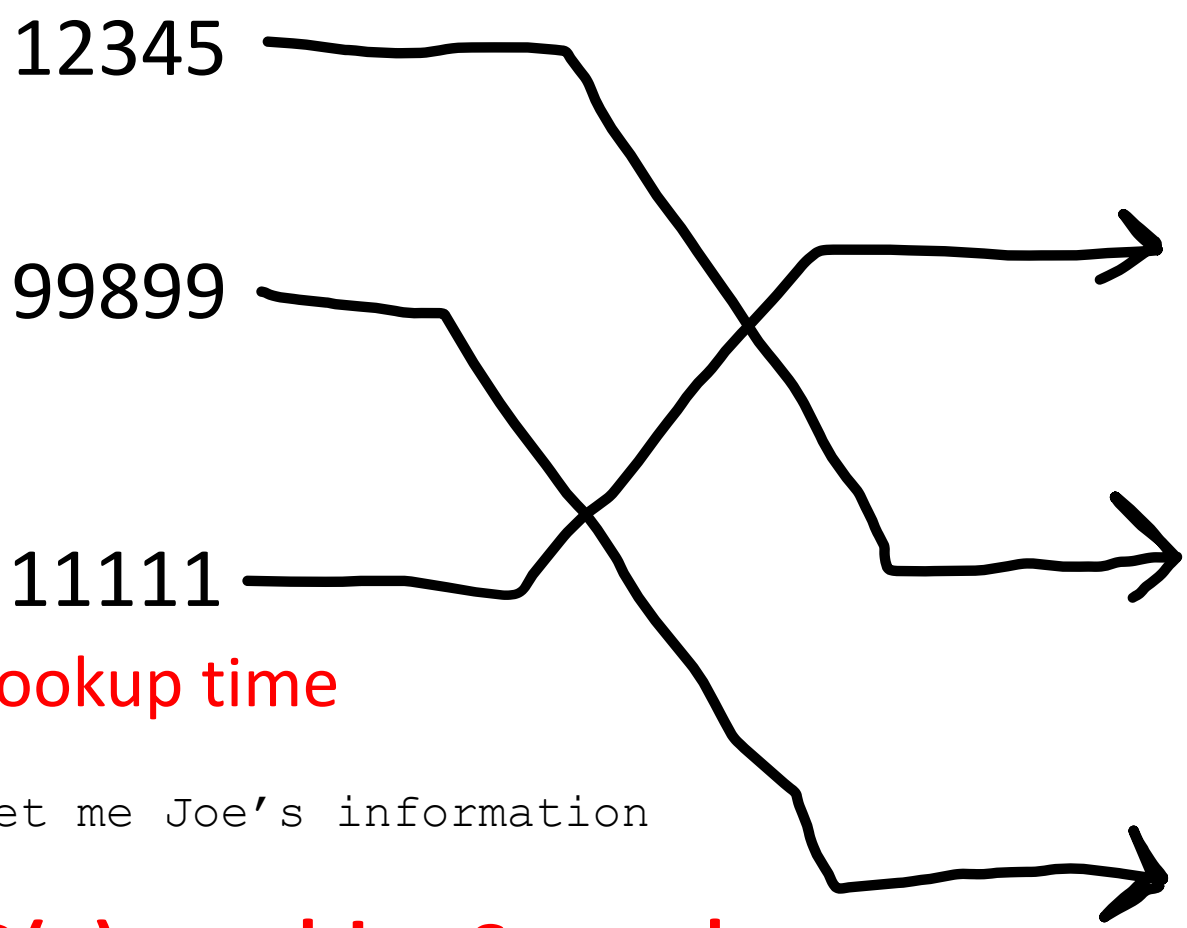
O(n), n = | array |

Yesterday, we built our own Hash Table Data Structure that mapped Student ID (Integer) to Student Objects

Student ID

Hash Function

`ID % 100`



Database

0	
1	
…	
…	
11	(Sean, 11111, History, 3.5)
…	
…	
…	
45	(Joe, 12345, Computer Science, 3.5)
…	
…	
…	
99	(Sally, 99899, Math, 3.5)

Lookup time

`Get me Joe's information`

O(n), n = | keySpace |

keySpace ArrayList = { 12345, 99899, 11111 }

In Java, we can import a Hash Table, called a **HashMap**

```
HashMap<String, String> capitalCities = new HashMap<String, String>();
```

Adding new Key Value Pairs (put)

```
capitalCities.put("England", "London");
capitalCities.put("Germany", "Berlin");
capitalCities.put("Norway", "Oslo");
capitalCities.put("USA", "Washington DC");
```

Retrieving a Value

```
capitalCities.get("England");
```

Removing a Value

```
capitalCities.remove("England");
```

Other Helpful Methods

- **keySet() → returns set of keys**
- **values() → returns set of values**
- containsKey()
- containsValue()
- replace()
- size()

A Java HashSet is an implementation of the Set interface that uses a Hash Map under the hood

A set is a collection with no duplicate elements

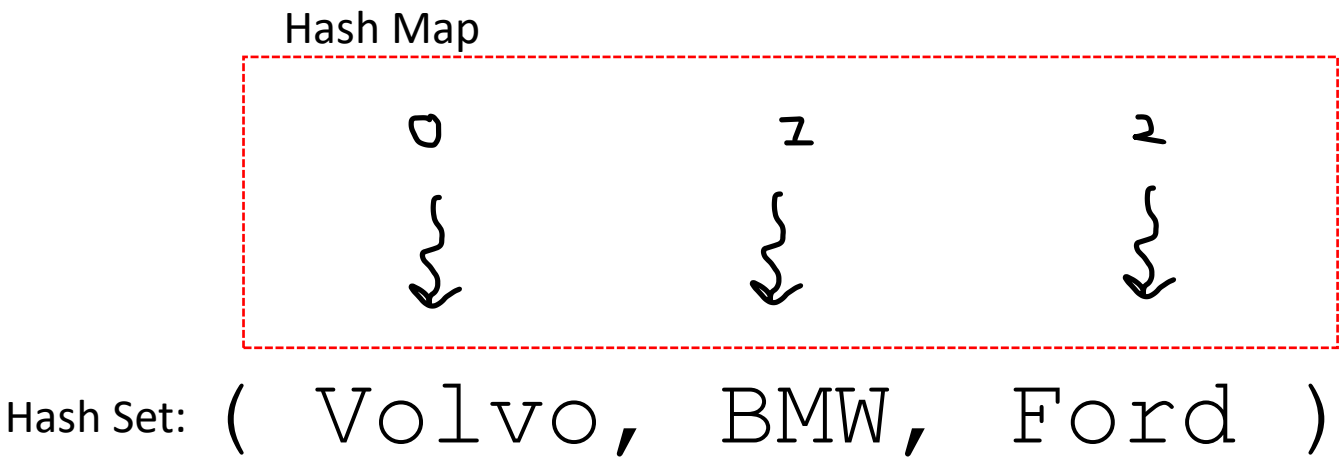
- You can think of this as a dynamic array, but without the ability to use indices

```
HashSet<String> cars = new HashSet<String>();

cars.add("Volvo");
cars.add("BMW");
cars.add("Ford");


cars.contains("Mazda");

cars.remove("Volvo");
```



Updating our Student Database Program:

Replace our array with a HashMap,

Update add, remove and print methods

Add method that will compute the number of CS majors, Math majors, History Majors, etc…

Add method that will compute which student(s) have a 4.0, 3.5, 3.1, 3.0, etc

Pirate Speak Translator